COMMERCENET
eCo Framework Project

eCo Architecture for Electronic Commerce Interoperability

This version:
1999-06-29T06:00-4:00

Chairman:
Murray Maloney (Muzmo Communication Inc.)

Editors:
Jeff Chen (3 Com)
Peter Kacandes (Sun Microsystems)
Dave Manning (UWI.Com)
Bart Meltzer (Commerce One)
Tom Rhodes (NIST)

Authors:
The Members of the eCo Working Group:

Jim Adkins (Netscape)
Terry Allen (Veo Systems Inc.)
Jon Bosak (Sun Microsystems)
Doug Bunting (Intuit)
David Burdett (Mondex International)
Jeff Chen (3Com)
Paul Curtin (NEC)
Jim Dills (eCommunity
Rik Drummond (ASC/X12)
Scott Hamilton (CommerceNet
Rodney Heisterberg
Denis Hill (ISO)
Dave Hollander (CommerceNet)
Ed Howe (Harbinger)
Puwei Huang (American Power Conversion)
Peter Kacandes (Sun)
Murray Maloney (Muzmo Communication Inc.)
Dave Manning (UWI.com)
John McCarthy (Berkeley National Lab)
Bart Meltzer (Commerce One)
Bob Miller (GEIS)
Takayuki Nakao (NTT)
Klaus-Dieter Naujok
Pat O'Sullivan (Intel)
Frank Olken (Berkeley National Lab)
Gerry Owens (EAN International)
Jean Pawluk (Compaq/Tandem)
Laird Popkin (News Information Service)
Kevin Poulter (Ontology.org)
ALN Reddy (Netscape)
Tom Rhodes, NIST
Jared Rodriguez (TRADE'ex E-Commerce Systems)
Thomas Schweinberger (IBM)
Steve Sklepowich (Microsoft)
Chris Smith (Royal Bank of Canada)
Howard Smith (Computer Sciences Corporation)
Hiroyuki Tanahashi (NTT)
Nagender Vedula (Microsoft)
Neil Webber (Vignette)
John Whitehead (Cisco)
Andy Whittaker (ANA)

To contact the eCo Working Group send e-mail to: eco-comments@lists.commerce.net

Table of Contents

1. Introduction

The simplest business models in the "Internet Economy" are straightforward extensions of those in the "bricks and mortar" world, but many of the most exciting ones are almost unimaginable in any environment but the Internet. Internet trading communities and marketplaces with aggregated and customized catalogs, intermediary services such as auctions and reverse auctions, and information brokerage and syndication networks are all examples of Internet business models that combine or interconnect offerings or services from multiple businesses.

The explosive and massively uncoordinated innovation in Internet technology and business models makes it an exciting time. But with so much creativity and innovation going on, an Internet business that wants to combine its services with others to create a virtual enterprise or community can be faced with a bewildering variety of implementations for electronic commerce (e-commerce) applications. This diversity is a growing barrier to interoperability that raises the costs implementing e-commerce, and limits the alternatives for companies who want to establish new Internet relationships.

"Interoperability" means the ability of separate systems to be linked together and then operate as if they were a single entity. For Internet commerce and business to business transactions, interoperability essentially means, "to be able to do business" without ad hoc and proprietary integrations, which are required when "everyone does it a different way." Without interoperability, fear of sunk costs can discourage experimentation in business relationships and result in lost opportunities. We are certainly not the first to recognize interoperability barriers as a downside of the exponential growth in Internet commerce. There are many efforts underway to facilitate interoperability in electronic commerce by developing specifications, usually based on XML, for the information most often exchanged between companies in a particular industry or sector. Notable initiatives include OBI, RosettaNet, IOTP, and ICE.

But while each new specification for a particular industry or marketplace is a step forward for that industry, each contributes to the interoperability problem as an inevitable result of their overlap in scope. Different industries need specialized vocabularies with unique terms and properties to describe their products and services, but some concepts and constructs needed in these "vertical" specifications apply to all business domains. Nevertheless, it has been easier for each new specification to "start from scratch" and invent everything, even basic definitions for a business, its location, and standard business documents like purchase orders and invoices that are surely needed by all industries. Likewise, the industry specifications often contain architectures for compound documents, message formats, and communication protocols that would ideally be common across industries.

Market forces will ultimately lead to some convergence and consolidation in electronic commerce architectures and implementations. But it is impossible to predict when that will happen, and in the meantime, more new specifications will be created. It may be that a single standard, or even a small set of them, will never emerge because of the inherent diversity in business models and practices throughout the world. Any prescriptive attempt to confine electronic commerce systems to standard architectural models would be futile. So we believe it is more realistic to aim for business to business interoperability and not standardization in electronic commerce architectures and protocols.

Thinking of interoperability as a set of levels makes it possible to create a specification for business to business interoperability. First of all, it isn’t necessary for businesses to agree on what they do or how they do it - they must simply agree on a common method of describing what they do. Furthermore, this method for describing electronic commerce systems doesn’t have to completely describe every aspect of every system. Instead, meta-data can be used to describe those system characteristics that are relevant to assessing the potential for integration. In other words, this method is only intended to provide a common basis for two parties to negotiate an understanding of how they will do business. Most of the details of how their systems are implemented "behind the scenes" are irrelevant, and never need to be disclosed. This allows businesses to remain free to differentiate themselves in the extent or quality of interoperability.

Furthermore, universal interoperability is not always necessary. For most companies, the decisions on which markets to enter are based on a myriad of complex trade-offs. Thus, interoperability should not be cast as an all-or-none issue. For one business to decide whether it can do business with another, it must answer a set of questions that define levels of interoperability. Obviously the most basic prerequisites for interoperability are being able to find a potential business partner and discover what services it offers. Only then can the first business recognize a potential trading partner and investigate further whether it can in fact do business with that partner. This investigation will involve studying the application interfaces provided and estimating the cost of any modifications or mappings required to use them. As part of this investigation, some of the questions that a business might ask are:

- What other businesses can I find?
- What services do they offer?
- What kinds of interactions do they expect?
- What protocols do they follow?
- Can our systems communicate?
- What application interfaces do they provide?
- Are our interfaces compatible?
- What information must we exchange?

In addition to the idea of interoperability levels, the idea of using Extensible Markup Language (XML) documents to describe the application interfaces to business services has also been fundamental to our thinking. Describing interfaces in terms of documents that describe properties increases the transparency of implementation, making interoperability easier to achieve and maintain.

We can call this idea a "document services architecture," but by any name it is fundamental to achieving the goal of creating virtual enterprises or communities by combining the services of different businesses. For example, a trading community (such as the community of businesses that unite to produce automobiles) can be described by the types of documents that are exchanged among participating trading partners. A trading community operator or "market maker" defines the "community standards" for business documents. Then buyers, suppliers, or other service providers, like shippers or payment acquirers, can participate if they can produce and consume those documents. How the documents are produced and what actions result when they are consumed are strictly up to the business. This elevates integration from the system level to the business level. It enables a business to present a clean and stable interface to its business partners despite changes in its internal technology implementation, organization, or processes.

1.1 The eCo Framework Project

The eCo Framework Project was chartered by CommerceNet in August, 1998, with Commerce One as the primary corporate sponsor. Participation by CommerceNet and Commerce One personnel was partly funded by the U.S. Department of Commerce (NIST) Advanced Technology Program award 70NANB7H3048 to Veo Systems (acquired by Commerce One), CommerceNet, BusinessBots, and Tesserae Information Systems (now renamed Cadabra). Murray Maloney of Muzmo Communication was hired by CommerceNet to serve as the Chairman of the eCo Framework project.

The eCo Working Group is an industry-neutral group consisting of experts from over 35 companies and organizations throughout the world. Because the eCo Working Group was chartered as part of this CommerceNet project, many of the participants belonged to CommerceNet member companies. The project benefited from the contributions of a number of experts who participated in developing important Internet and electronic commerce specifications, including CBL, OBI, OFX, IOTP, XML/EDI, RosettaNet, URN, HTML and a number of the W3C XML specifications. Participants include representatives from 3Com, American

Express, American Power Conversion, ASC/X12, Berkeley National Lab, Cisco Systems, Commerce One, Compaq, CSC, GEIS, Harbinger, Hewlett-Packard, IBM, OASIS, Intel, Intuit, ISO, Microsoft, Mondex International, Muzmo, NEC, Netscape, NIST, Novell, Ontology.org, Royal Bank of Canada, Sun Microsystems, UWI.Com, Vignette and others.

1.2 Status of this Document

This document is a Final Draft of the eCo Working Group. It is the result of collaborative work between members of the eCo Working Group and the editorial team assigned to produce the document. Comments should be sent by email to eco-comments@lists.commerce.net

1.3 Relationship to Other Specifications

There are direct references to other standards and specifications in the bibliography of this specification.

Because this work is intended to facilitate the interoperation of both existing and new e-commerce systems, the consideration of other open, Internet-based e-commerce standards was a key part of its formulation. Further, it is a design principle of this work to use existing standards as the building blocks for this specification wherever reasonable.

1.4 Design Principles

In developing this specification, the Working Group has focussed on business to business solutions, and has been guided by several fundamental design principles. These principles were chosen for their relevance to our stated objective, that is, providing an architecture for e-commerce interoperability. These principles are:

- Use Internet Standards and Specifications. The Working Group will favor open Internet standards and specifications over proprietary Internet solutions. In particular, XML-based languages and protocols will be used wherever possible to provide a common framework for data exchange.
- Use Existing or Emerging Standards. Where appropriate, the Working Group will use existing or emerging standards, such as XML and XML Schemas. This will minimize the number of additional protocols required to implement this specification and encourage our overall goal of interoperability.
- Promote Interoperability. The Working Group will strive to create an architecture that can describe all e-commerce systems. Since business is conducted differently in different markets and in different parts of the world, the Working Group expects there will be a wide variety of e-commerce protocols required to express the full breadth of business practice. By ensuring that each of these protocols can be described by a unifying framework, the Working Group believes that interoperability can be promoted and achieved.
- Allow for International Language Support. The Working Group will ensure that this specification will not impede the development of e-commerce applications in any known language or character set.
- Extensibility. Since it is impossible to foresee all the possible application environments for this technology, the Working Group will, as much as is feasible, ensure that this specification can be extended to accommodate unforeseen requirements. By doing so, the Working group hopes to ensure that its work can be of benefit to the widest possible audience.
- Simplicity. The Working Group will strive to keep this specification as simple as possible. By doing so, the Working group hopes to create a specification that is simple enough for businesses and solution developers of all sizes to implement.
- Achievable. The Working Group will describe methods and mechanisms that can be implemented with widely available technology.

1.5 Requirements

After studying the problem of creating an architecture that will enable the interoperation of a heterogeneous set of e-commerce systems, the Working Group has agreed upon the following design requirements for this specification:

- The Architecture needs to provide a common conceptual framework through which e-commerce systems can be viewed. In considering the problem of describing the nature of an e-commerce offering, it is clear that there is a need for a common conceptual framework that can be used to structure the description of the offering. Providing this basic architectural framework is the first step in establishing a basis for determining interoperability.
- The Architecture needs to enable the discovery of e-commerce systems on the Internet. To fully enable an open market for goods and services, there is a basic requirement that potential suppliers can be located in a reliable and efficient manner.
- The Architecture needs to provide a common method for definitively referencing the communication protocols used by an e-commerce system. Given that it is one of our fundamental design principles not to intentionally preclude any e-commerce standard from working with this Architecture, it is clear that we need some method of determining which of the many e-commerce protocols a system is using, while also providing or pointing to an explanation of that protocol.
- The Architecture needs to provide a mechanism for defining a set of common e-commerce semantics. One of the key challenges in achieving e-commerce interoperability is in establishing a common vocabulary of business terms that can be used to express key concepts. While the Architecture itself does not try to define a standard set of terminology, it is recognized that it needs to support a method of resolving business semantics. It is further understood that this method needs to be extensible so industry and market specific vocabularies can evolve as required.
- The Architecture needs to enable the combining of one businesses' offerings with that of another. In order to support many of the business models in practice today, the Architecture needs to allow businesses to combine their offerings into higher-value joint offerings.
- The Architecture needs to enable business processes that span a number of organizations. In order to support many of the business models in practice today, the Architecture needs to allow for the definition of business processes that span any number of organizations.
- The Architecture needs to provide a mechanism for exposing existing e-commerce systems in eCo compliant terms. In order to support existing e-commerce systems that were not designed with this specification in mind, there must be a mechanism for describing and providing information on interfacing with those systems.

2. eCo Architecture

The purpose of this specification is to:

- Define a conceptual architecture through which information on e-commerce systems can be communicated.
- Define a method of querying that information.
- Define the structure that will be used to return that information.

The eCo working group has determined that in order to promote business-to-business (B2B) interoperation between heterogeneous e-commerce systems on the Internet, there is a basic requirement that trading partners be able to:

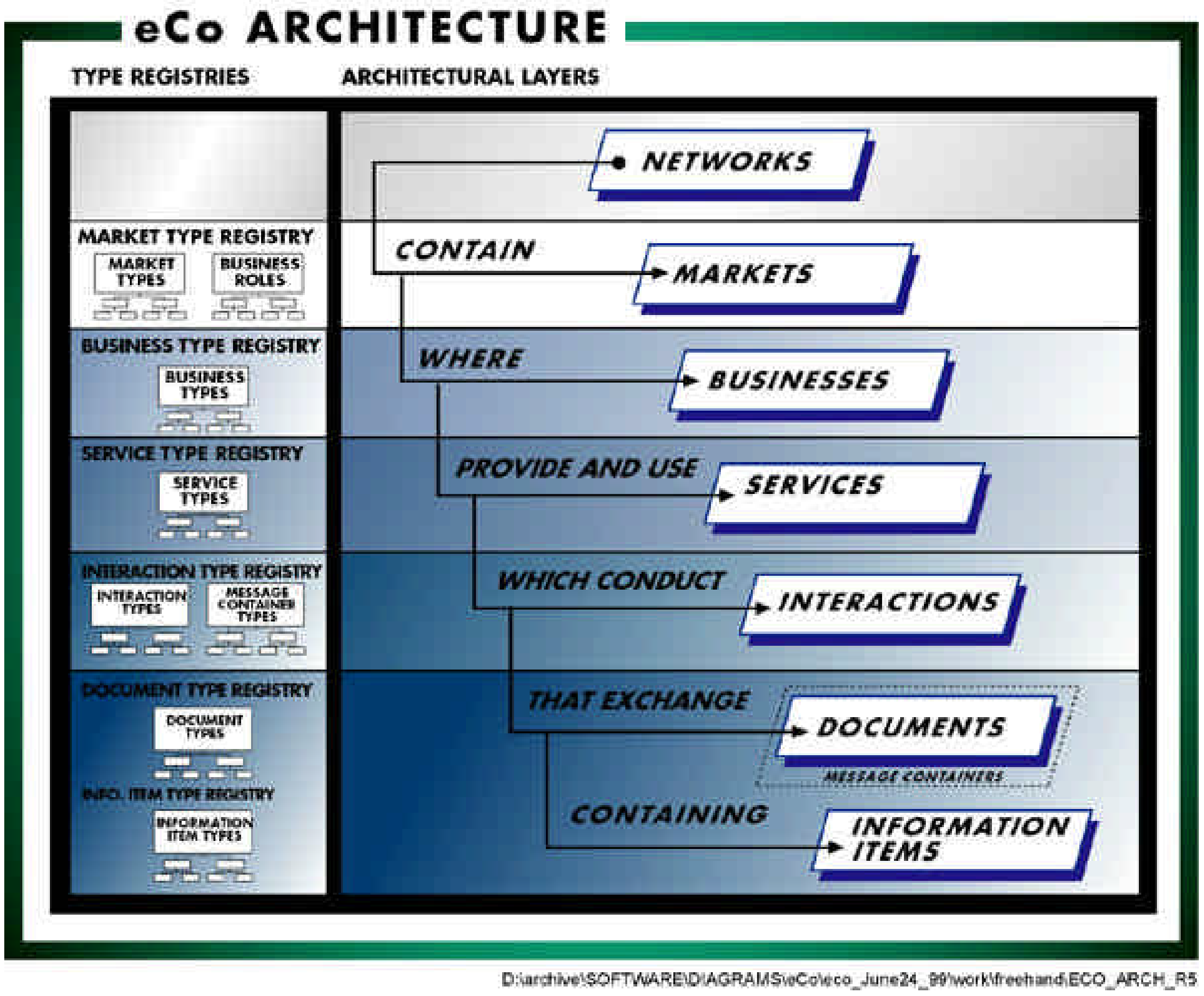
- discover other businesses on the Internet
- determine whether they want to do business and how they can participate within a market
- determine what Services are provided and consumed by other businesses
- determine the underlying interactions and the documents and information used by interactions
- determine if and how their e-commerce systems can communicate
- if necessary, determine what modifications need to be implemented to ensure interoperability between their systems
- if desired, establish communications through other channels than the Internet

With this in mind, the eCo Architecture has been developed primarily along business principals rather than technical ones. Through the conceptual framework defined by the eCo Architecture, businesses can define and publish (i.e. publically expose) meta-data descriptions on their e-commerce systems (i.e. it's "Properties"). The goal of this meta-data is to allow interested parties to gain an understanding of the various aspects of the e-commerce system being represented and thus provide the information needed to satisfy the above requirements and enable interoperability.

In order for this exchange of meta-data to occur between two interested parties, they need to share a common understanding of the basic components that make up an e-commerce environment, and how those components relate. Further, they need to have a common way of exposing information about of each of these components. It is with this requirement in mind that the eCo architectural model (the "Architecture") and its attendant Interfaces for accessing Properties of the e-commerce system were developed.

The eCo Architecture is not intended to represent any specific e-commerce system. On the contrary, it is expected that e-commerce system implementations will vary widely, and any attempt to confine or restrict them to a specific architectural model would be futile. Rather, this architecture is designed to represent those aspects of an e-commerce system which contribute to its interaction with a prospective trading partner. In other words, this architectural model is intended to provide a common basis for two parties to negotiate an understanding of how they will do business. The details of how their systems are implemented "behind the scenes" are up to them (e.g. internal business processes, data models, etc). However, business communities and marketplaces can be expected to adopt specific interfaces and protocols for conducting business.

The basic structure of the eCo Architecture is outlined in the following diagram:



2.1 Architectural Layers

As can be seen in the previous diagram, the eCo Architecture is a layered model representing an electronic commerce environment. Each layer relates to the next layer in a defined way, and information or properties (i.e. metadata) of each layer are described separately through various "Type Registries" associated with each layer of the Architecture. Each layer, and it's Registries, are intended to provide some aspect of information about the electronic commerce environment and enable an interested party (e.g. business) to obtain information at each layer to potentially use offerred services, or to join the marketplace and either provide new services or interoperate as a trading partner with other businesses in that marketplace.

In the eCo Architecture model, the top layer, "Networks," represents one or more physical networks (e.g. Internet) in which electronic commerce systems can exist. These "Networks" will contain various marketplaces, or "Markets," for providing or obtaining specific goods and services.

The network operator may choose to organize the network according to any principle appropriate to the purpose of that network. For example, the network could be an index of markets in a specific ontology, much as the popular search engines index web sites.

Each "Market" in the Network may be independent, and may have it's own rules, procedures, and protocols for participation within that Market. Markets themselves are made up of one or more businesses that are described by the "Businesses" layer of the eCo Architecture. For example, this layer might identify the type of business, it's location, web page, and other business-related information.

Any one business may participate in multiple markets. A computer manufacturer, for example, could participate in markets as varied as hardware, software, finance, consulting, service, and employment. In other words, any activities or functions within the business could participate in a market dedicated to satisfying the needs of those activities. Furthermore, these markets could be specific to one business or could be operated among multiple businesses. For example, a business might wish to make a market for all of the companies that participate in its supply chain. Alternatively, that same business could participate in third party supply chain markets for the specific vertical industry of that business.

The next layer of the eCo Architecture, the "Services" layer, enables each business to describe the types of business services offerred, their interfaces, and other information needed to use a particular service offerring. Examples could include catalog browsing, ordering products, making payment, checking order status, product life cycle, new product introduction and so on. The interface for each business service offered would be described by the "Services" layer of the Architecuture.

The types of services and their interfaces can vary among business providers, although a group of businesses or marketplace may adopt some common conventions. For example, in a specific vertical industry, businesses participating in that industry could agree on a common catalog update service, a product data exchange service, a failure analysis service and so on.

Each provided service may be composed of sub-services, which may also be described at the "Services" layer of the Architecture, that provide for various options of the overall service. An Original Equipment Manufacturer, for example, might have a product life cycle service that could be composed of sub services for each stage of the life cyle, such as new product introduction, sustaining, end-of-life, and so on. These services can continue to recurse until an atomic service is reached.

In addition to sub-services, a service may invoke other services in order to complete that service. These relationships among services and sub-services are described in the next layer of the eCo Architecture, the "Interactions" layer, and are normally hidden from the service-user. In one sense, this layer describes a "choreography" of interactions that may take place when a service is invoked. The exact sequence of events in this choreography may be pre-determined, or it may be event-driven by the user's selection of specific options offered by the service. This layer is used to describe the types of interactions behind each service, and the types of messages which are exchanged during each interaction. Each message or message container may contain one or more types of documents or information needed during that interaction. The document types exchanged in an interaction are described by the "Documents" layer of the eCo Architecture. Finally, the "Information Items" layer of the eCo Architecture describes the type of information items (e.g. data elements and attributes) comprising each type of document used by an interaction.

A business may define their own data elements and documents. A business could also use the data elements and documents that have been defined by a third party, or for a specific domain or standard. For example, businesses in the financial industry might choose to agree on a set of elements and documents needed for financial transactions. Businesses which use financial services offered by businesses in that industry might need to use those predefined data elements and documents. It is also possible that businesses might use a pre-defined base set of elements, extend those elements by adding new ones, and create new types of documents that may contain elements both from the base and the extended set.

The eCo Architecture further defines interfaces for querying or accessing the set of information or properties described at each Layer. By querying a Layer's published information and examining its property set, an interested party can obtain information about that Layer, and determine enough information about the Layer to inter-operate with it. By examining all the Layers implemented by a fully eCo compliant system, a prospective trading partner can determine all the necessary information needed to interact with that system.

Section 3 provides further details of each eCo Architecture Layer, and it's defined interfaces. eCo Architecture Compliance is discussed further in Section 4 of this document.

2.2 Type Registries

The eCo Architecture defines a set of Type Registries associated with each Layer of the Architecture that are used to establish type information describing the various document and element type components in an e-commerce system. Each Registry can provide information on one or more specific sets of types appropriate to that Registry. For example, the Business Registry provides a set of type definitions required by the Business Layer, the Interaction Registry provides a set of Interaction types and Message Container types, and so on.

Using Type Registries, it is possible to:

- Determine the equivalency of two types that are defined within the same Registry.
- Retrieve the definition of a type.
- Determine the relationships that exist between types (e.g. which types were derived from which other ones).

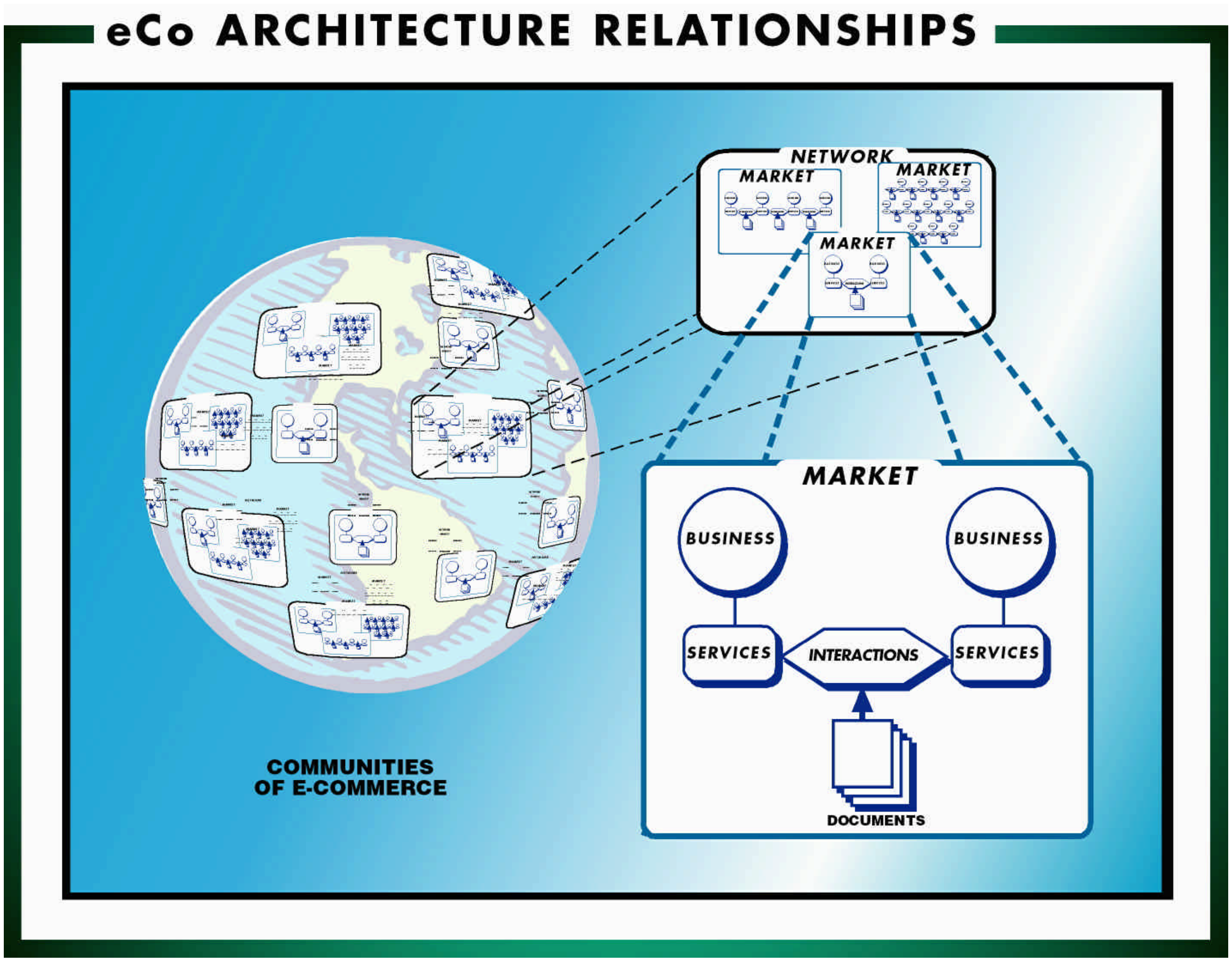
Each type listing contained within a Registry is structured as a hierarchy, allowing type definitions to be refined through the addition of sub-types. In this context, eCo supports single inheritance and simple trees. For example, a Market for automotive parts might be further refined by breaking it down into markets for engines, tires, and so on. If something is defined by a type in a Registry, it is also a member of that type's type lattice or hierarchy. For example, using the Market types from the previous example, a Market for engines would also be considered a Market for automotive parts.

Each Registry exposes a Published Interface in much the same way as the eCo Layers do. These Published Interfaces allow the Registry to be queried for information.

It is important to note that eCo confines the scope of its Registry Interfaces to the retrieval of type information. No attempt is made to define Registry administrative Interfaces as these functions do not directly contribute to the exposure of meta-data on e-commerce systems.

2.3 Relationships within the eCo Architecture

To better understand the relationships that exist between the Layers of the Architecture consider the following scenario:



In the previous diagram, two companies are represented as eCo Businesses (implementations of the eCo Business Layer). These Businesses exist together in an eCo Market that provides a venue for the type of e-commerce that they are conducting. The eCo Market is indexed within a Network of Markets so that it can be located on the Internet.

Each of these Businesses offers a set of e-commerce "Services" to each other. Each Service represents an interface to a business process. Example Services might be "Become a VAR", "Order some product", or "Review a catalogue of products." At the highest level, Businesses interact by using the Services of each other within the context of some business process.

A Service is composed of a set of document exchanges. We call these exchanges "Interactions". An Interaction occurs when a request (consisting of one or more Documents) is sent from one Business to another and a response (again consisting of one or more documents) is received.

The Documents that are exchanged as part of any Interaction consist of discrete elements of information. These "Information Elements" are the building blocks from which Documents are composed.

Using this basic model of Networks, Markets, Businesses, Services, Interactions, Documents, and Information Items, it is possible to describe the basic architecture of almost any e-commerce system. By then exposing meta-data on each of these components, it is possible to describe that system in sufficient detail to provide a basis for interoperability.

The usage of the Registries is considerably easier to understand. Each Layer in the Architecture (with the exception of the Network Layer) can type itself by referencing type definitions in one or more Registries. Markets can type themselves by referencing a Market Registry, Businesses can type themselves by referencing a Business Registry, and so on.

In the previous example, the Businesses can use a common Business Registry to identify their types within the Market, the types of the Services they provide can be identified by referencing a common Service Registry, and so on.

It is important to note that Registries do not record information on specific e-commerce implementations, or instances. Registries are only intended to provide a common source of type definitions that can be referenced by various e-commerce implementations.

3. eCo Layers and Registries

eCo market places feature many buyers and sellers seeking to make the most informed decisions. Sellers information will include countless products and content from multiple database types, constructs and libraries, integrated through trusted open registries. Registry schemas are built on sets of XML-based data elements. Sellers can register their products/services, authenticate themselves and their service policies. Buyers can compare products accurately. Buyers and sellers can register their preferences, profiles and authorization boundaries publically or privately. These registration capabilities will be integrated through server based tools that perform translation and semantic mapping services for business products services and processes.

In many cases, the buyers and sellers are already known to each other and the relationships between them are determined contractually. In this case, the eCo system can be used to capture the relationship and to implement the terms of the relationship on an on-going basis. For example, an OEM and a component supplier will usually have an agreement that specifies the conditions under which the supplier will deliver parts, in whatever quantities, within specific reponse times, at specified prices, and with certain quality characteristics. The OEM and supplier could implement this agreement as a set of eCo services. The OEM would use the service with specific parameters as determined by their current product needs, and the supplier would deliver according to the order placed by the OEM.

This section provides a detailed examination of each Layer and Registry in the Architecture. For each Layer or Registry examined, its Published Interface is provided, and the document types that its queries can return are listed.

3.1 Published Interfaces

3.1.1 Querying a Published Interface

The mechanism that is used to query a Published Interface is a simple URL based protocol. This method was chosen over other more sophisticated possibilities in order to allow companies to leverage their existing Web infrastructures to the greatest degree possible.

Each Published Interface is identified by a unique URI. This URI can be resolved to a "Base URL" for that Interface. By appending the desired query name to the end of the Interface's Base URL, that query can be performed. Any parameters required by the query are passed in the request URL using the same URL encoding technique used by HTML forms [RFC 2483].

Note that each Interface name is case sensitive, and the URL used should respect case accordingly.

Example #1: In order to execute the query " BusinessGetServices " on the Published Interface with a base URL of "http://www.commerce.net/eco" an interested party would execute the URL "http://www.commerce.net/eco/BusinessGetServices".

Example #2: If an interested party wanted to execute the query " BusinessGetServicesByType" on the same Published Interface used in the previous example and pass the parameter "Type" with a value of "urn:foo:12345-54321" the query would look like:
"http://www.commerce.net/eco/BusinessGetServicesByType?Type=urn:foo:12345-54321".

The response to any query is always in the form of an XML document that is formatted in accordance with a pre-defined Document Type Definition (DTD). This method was chosen to minimize the need for server-side scripting.

3.1.2 Extending a Published Interface

The Published Interface offered by a Layer can be extended to include queries over and above the base set defined by this document. These are called "Interface Extensions." In order for an interested party to discover what queries are available through a given Interface a "QueryInterface" query is defined at each Layer. This query returns an EcoInterfaceDefinition that enumerates each query supported by that Interface, its parameters, and a brief description of that query.

Using this mechanism, an interested party can discover any public Interface Extensions that are available. This mechanism also allows an interested party to determine which of the optional base set of queries and parameters an Interface supports.

In order to prevent namespace conflicts between queries defined in future versions of this specification and Interface Extensions, the names of all Interface Extension queries must begin with an underscore character ('_'). For example, you might extend the interface with a query called "_InteractionCancel", which would have a single parameter of "InteractionID."

3.1.3 Bootstrapping Interface Discovery

In order to allow a casual visitor to a Web site (such as a robot or a search engine) to determine if the site is host to any eCo Published Interfaces, a method of bootstrapping the discovery of eCo Interfaces has been defined.

In order to inform a casual visitor that a given Web site is host to an eCo Published Interface, the owner of that site may optionally place a document called "eco.xml" (case sensitive) at the root level of the site. This document should be structured in accordance with the EcoInterfaceList DTD.

Example: If a business that implemented the eCo Business Environment wanted to allow itself to be indexed as an eCo compliant business, it could place the following XML file at the root of its Web site:

```
<?xml version="1.0"?>
<EcoInterfaces xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.commerce.net/...</Identifier>
      <Creator>http://www.commerce.net/~bob</Creator> <Date>19990628</Date>
      <Version>1.0</Version>
      <TimeToLive>86400</TimeToLive>
      <Description>...</Description>
      <Label>...</Label>
   </Head>
   <Interface type="Business">
      <Identifier>http://www.mycorp.com/eco</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
      <Version>1.0</Version>
      <Date>19990628</Date>
      <TimeToLive>86400</TimeToLive>
      <Description>My business interface description.</Description>
      <Label>My Business</Label>
   </Interface>
   <Interface type="Service">
      <Identifier>http://www.commerce.net/eco/OrderService</Identifier>
      <Creator>http://www.commerce.net/~bob</Creator>
      <Version>1.0</Version>
      <Date>19990628</Date>
      <TimeToLive>86400</TimeToLive>
      <Description>My order service interface description.</Description>
      <Label>My Business' Order Service</Label>
   </Interface>
   <Interface type="Service">
      <Identifier>http://www.commerce.net/eco/CatalogService</Identifier>
      <Creator>http://www.commerce.net/~bob</Creator> ...
   </Interface>
</EcoInterfaces>
```

By doing so, the business could enable search engines to identify that its site was host to an eCo version 1.0 compliant Business who's Published Interface could be found at "http://www.commerce.net/eco". With this information, the search engine could then proceed to index the Business on its eCo Properties. If the identifier listed were to be a URN rather than a URL, the URN would need to be resolved before the Interface could be used.

While the above example describes a Business exposing its Published Interface, it is equally valid for this technique to be used to expose any other type of Published Interface in this manner. (For example, an eCo Market might expose its Published Interface using this technique so that it could be indexed by a robot.)

A description of the key elements and attributes in an "eco.xml" document can be found in section 3.4 Commonly Returned Document Types, under EcoInterfaceList.

3.2 The eCo Document Wrapper

Every XML document that is returned as a result of a query to a Published Interface must conform to the eCo document wrapper element architecture. This wrapper element architecture includes the "eCo" namespace declaration attributes and a set of elements that provide meta-data about the document. The following key elements and attributes make up this common document wrapper.

DocumentWrapper

Key Elements & Attributes:

`root` *(mandatory)* – This is the key element that serves as a and wrapper for any other eCo XML content. The eCo namespace is also defined here. The names assigned to the root elements of eCo document types vary. The root element must include namespace declaration(s).

```
xmlns="http://www.commerce.net/eCo"

xmlns:eCo="http://www.commerce.net/eCo"
```

`Head` *(mandatory)* – This is a container for the meta-data elements that follow. The Head will not change between versions of this specification.

`Identifier` *(mandatory)* – This element contains a URI that uniquely identifies the Published Interface of an implementation of an eCo Layer.

`Creator` *(optional)* – This is a URI for the person or organization responsible for creating the document. The URI can be to a business registry, or simply an email address using the mailto: scheme.

`Date` *(optional)* – The date that the document was made available in its present form. Date format is an 8-digit number in the form YYYYMMDD.

`Version` *(optional)* – The version of the eCo Architecture Specification that the document conforms to.

`TimeToLive` *(mandatory)* – This is the time in seconds that the document should be considered valid before being discarded. This time is measured from the time that the document is retrieved.

`Description` *(optional)* – This is a plain language description of the document.

`Label` *(optional)* - This is a string that defines the name by which the item is most commonly known.

Example:

```
<?xml version="1.0"?>
<myDocumentType xmlns = 'http://www.commerce.net/eco'>
<Head>
   <Identifier>http://www.commerce.net/...</Identifier>
   <Creator>http://www.commerce.net/~bob</Creator>
   <Date>19990628</Date>
   <Version>1.0</Version>
   <TimeToLive>86400</TimeToLive>
   <Description>My document's description.</Description>
   <Label>My Document</Label>
</Head>
   ...
</myDocumentType>
```

Thus, an eCo document is any document whose namespace resolves to the eCo URI and contains the eCo Head element.

As noted in Section 3.1.3, the Eco.xml file also uses this common document wrapper to communicate basic document meta-data.

3.3 Common Queries

As noted in the previous section, a Published Interface is provided for each Layer and Registry in the Architecture. In addition to the queries detailed below, there is a common set of queries that must be supported by all Published Interfaces. These shared queries are defined in this section.

`GetSupportedVersions` *(mandatory)*

Description:	This query returns a list of the versions of the eCo Specification that an Interface supports. This query and the structure of the EcoVersionsList will remain unchanged in future versions of this specification. If a given Interface wishes to support other mutually incompatible versions of this specification it can provide and advertise a second Interface to its systems.
Call:	*Base URL* / GetSupportedVersions
Parameters:	None
Return:	EcoVersionsList

`QueryInterface` *(mandatory)*

Description:	This query returns a list of the queries that have been implemented in this Interface. If a query is not listed, it has not been implemented.
Call:	*Base URL* / QueryInterface
Parameters:	None
Return:	EcoInterfaceDefinition

3.4 Commonly Returned Document Types

As noted above, each query made to a Layer's Published Interface returns a document. In addition to the document types discussed in each of the Layers outlined below, there is a set of commonly used document types that are not specific to any single Layer of the Architecture. These shared document types are described in this section.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

EcoVersionsList

Description:	This document is used to represent the list of versions of this specification supported by an Interface. The format of this document will remain unchanged in future versions of this specification.
DTD:	`EcoVersionsList.dtd`
Key Elements & Attributes:	**`Version`** *(mandatory)* - This is a string that identifies a version of this specification supported by this Interface. Versions are listed in "major.minor" form. (For example "1.0".)

Example:

```
<?xml version="1.0"?>
<EcoVersionsList xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
      <Version>1.0</Version>
      <Version>2.0</Version>
         ...
      <Version>n.m</Version>
</EcoVersionsList>
```

EcoInterfaceList

Description:	This document is used to represent a list of URIs to Published Interfaces.
DTD:	`EcoInterfaceList.dtd`
Key Elements & Attributes:	**`Interface`** *(mandatory)* - The Interface element contains the Head element with its Identifier URI to uniquely identify the Published Interface to an implementation of an eCo Layer. The type of the Interface is specified as an attribute to this element. **`Label`** *(mandatory)* - This is a string that defines the name by which the Published Interface is most commonly known. Complete name information for the Interface's underlying Layer or Registry can be derived from querying the Interface itself. For example, to find the name of a Network given the Published Interface for that Network, call NetworkGetProperties.

Example:

```
<?xml version="1.0"?>
<EcoInterfaceList xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
         ...
   </Head>
   <Interface type="Registry">
      <Head>
         <Identifier>http://www.whoever.com/...</Identifier>
```

```
            <Creator>http://www.whoever.com/~bob</Creator>
               ...
            <Label>http://www.whoever.com/...</Label>
         </Head>
      </Interface>
      <Interface type="Market">
         <Head>
            <Identifier>http://www.whoever.com/...</Identifier>
            <Creator>http://www.whoever.com/~bob</Creator> ...
            <Label>http://www.whoever.com/...</Label>
         </Head>
      </Interface>
      <Interface type="Business">
         <Head>...</Head>
      </Interface>
      <Interface type="Service">
         <Head>...</Head>
      </Interface>
      <Interface type="Interaction">
         <Head>...</Head>
      </Interface>
      <Interface type="Document">
         <Head>...</Head>
      </Interface>
      <Interface type="DataElement">
         <Head>...</Head>
      </Interface>
   </EcoInterfaceList>
```

EcoInterfaceDefinition

Description: This document is used to provide a definition of a Published Interface. It contains a listing of all the queries supported by an Interface as well as the parameters supported by those queries. Such a listing is necessary to help an interested party determine which of the optional queries and parameters a particular Interface is supporting. This document also provides a way to extend the list of queries supported by an Interface beyond those defined by this specification.

DTD: `EcoInterfaceDefinition.dtd`

Key Elements & Attributes: **QueryDefinition** *(mandatory)* - The QueryDefinition element contains the name the query, an *(optional)* plain-text description of the query and a list of its parameters in the form of ParameterDefinition elements (see below).

ParameterDefinition *(optional)* - The ParameterDefinition element contains the name of the parameter, a brief plain-text description of the parameter and tag indicating if the parameter is "optional" or "mandatory".

Example:

```
<?xml version="1.0"?>
<EcoInterfaceDefinition name="myInterface" type="..."
            xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
         ...
   </Head>
   <QueryDefinition name="myQuery">
      <Description>My interface description.</Description>
         <ParameterDefinition name="Param1" required="true">
            <Description>Param1 description.</Description>
         </ParameterDefinition>
         <ParameterDefinition name="Param2" required="false">
            <Description>Param2 description.</Description>
         </ParameterDefinition>
      </QueryDefinition>
</EcoInterfaceDefinition>
```

EcoTypeList

Description: This document is used to represent a list of references in a type registry.

DTD: `EcoTypeList.dtd`

Key Elements & Attributes: **`Type Reference`** *(optional)* - This attribute contains a URI that uniquely identifies a specific type definition within a registry of types. Using this URI, information on that type can be obtained and type equivalence can be determined.

`Type Registry` *(optional)* - This attribute contains a URI that identifies the registry being used.

Example:

```
<?xml version="1.0"?>
<EcoTypeList xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Type reference="type1" registry="...">
   <Type reference="type2" registry="...">
      ...
   <Type reference="typeN" registry="...">
</EcoTypeList>
```

EcoBoolean

Description: This document is used to return a true or false answer to a query. It is a simple wrapper around a boolean value.

DTD: `EcoBoolean.dtd`

Key Elements & Attributes: **`BooleanValue`** *(mandatory)* - The content of his element is the value being returned. It is an EcoBoolean datatype value of "true" or "false", and is case sensitive.

Example:

```
<?xml version="1.0"?>
<EcoBoolean xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
```

```
         <Creator>http://www.whoever.com/~bob</Creator> ...
      </Head>
      <BooleanValue>true</BooleanValue>
   </EcoBoolean>
```

3.5 Architectural Layer Details

3.5.1 The Commerce Network Layer

The Commerce Network Layer provides a way to type and locate markets or communities of commerce on a network. The Layer consists of a set of Interfaces that can be used to find Markets and relate those Markets to a set of defined Market types.

3.5.1.1 Published Interface

The Published Interface to a Network defines a set of queries that can be requested of that Network. The following standard queries are defined:

`NetworkGetProperties` *(mandatory)*

Description:	This query returns a set of properties that describe the Network.
Call:	Base URL / NetworkGetProperties
Parameters:	none
Return:	NetworkPropertySheet

`NetworkGetMarkets` *(optional)*

Description:	This query returns a list of Markets that participate in this Network. This list contains the identifying URI for each Market. By using its URI, a description of a Market can be obtained and its Published Interface can be queried.
Call:	*Base URL* / NetworkGetMarkets
Parameters:	None
Return:	EcoInterfaceList

`NetworkGetMarketTypes` *(optional)*

Description:	This query returns a list of Market types that participate in this Network. This list contains the URI for each Market type participating in the Network. Using these URIs, information on each type can be obtained and type equivalence can be established.
Call:	*Base URL* / NetworkGetMarketTypes
Parameters	None
Return:	EcoTypeList

`NetworkGetMarketsByType` *(optional)*

Description:	This query returns a list of the Markets that this Network has to offer filtered by type. A list of all the Market types participating in this Network can be obtained by calling NetworkGetMarketTypes.
Call:	*Base URL* / NetworkGetMarketsByType?Type=URI
Parameters:	**`Type`** *(mandatory)* - This is the unique identifier of the Market type in question. A list of all the Market types that participate in this Network can be obtained by calling NetworkGetMarketTypes.
Return:	EcoInterfaceList

3.5.1.2 Return Documents

The following document types are returned in response to queries to a Network's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

Note: The Commerce Network Layer provides a high-level grouping of Markets on a network. It does not establish any mandates regarding lower levels in the eCo Architecture (Businesses, Services, Interactions, Documents, or Data Elements.)

The eCo DTDs for these documents are detailed in Appendix A of this document.

`NetworkPropertySheet`

Description: This document is used to describe an eCo Network. It is returned in response to a NetworkGetProperties query in a Network’s Published Interface.

DTD: NetworkPropertySheet.dtd

Key Elements & Attributes:

`NetworkPropertySheetIdentifier` *(mandatory)* - This is the URI that uniquely identifies the Network. Using this identifier, the Published Interface to the Network can be queried.

`Name` *(mandatory)* – This item contains the Networks' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Network can have multiple names.

`Description` *(mandatory)* - This is a plain language description of the Network.

`MarketRegistryLocation` *(optional)* - This is the URI that can be used to query the registry used to define Market types in this Network. By providing this Property, a Network is establishing a mandate that its Markets provide type information using this registry.

`Maker` *(mandatory)* - This is a list of the URIs that uniquely identifies the Network Makers for this Network. The Network Makers are the businesses that are sponsoring the Network. Each Network Maker within the Network is defined as an eCo Business. More information on each Network Maker can be derived from its URI by querying its Published Interface.

Operator *(mandatory)* - This is a URI that uniquely identify the Network Operator for this Network. The Network Operator is the business responsible for maintaining the Network on the Internet, and handles all technical issues. The Network Operator is defined as an eCo Business within the Network. More information on the Network Operator can be derived by querying its Published Interface.

RegistrationService *(optional)* - This is a URI that uniquely identifies the eCo Service through which new Markets can join the Network. It is expected (but not necessary) that this Service is provided by the Network Operator. For other administrative Services defined in this Market, the Market Operator's Published Interface should be examined. By querying the Network Operator, more information on this service can be derived.

TermsAndConditions *(optional)* - This is a URL to a document that specifies the terms and conditions of the Network.

NetworkSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network itself. Using this Interface, Network specific data can be appended without compromising the structure of this document. It is envisioned that this ability might be used by a Network to describe itself in ways that are only meaningful in the context of its constituent Markets.

Credentials *(optional)* - This is a list of structured items that define certifications that have been granted to this Network. Each certification consists of a digitally signed block containing a plain language description, the date that the certification expires (date format is an 8-digit number in the form YYYYMMDD), an optional pointer into a registry of certification types, and a reference to the issuing authority. Note that while a signing mechanism has not yet been specified, the final specification from the W3C XML Signature Working Group is expected to be used.The DTD for the digitally signed block is currently incomplete pending the outcome of the W3C XML Signature Working Group.

```
Example:        <?xml version="1.0"?>
                <NetworkPropertySheet xmlns = 'http://www.commerce.net/eco'>
                   <Head>
                      <Identifier>http://www.whoever.com/...</Identifier>
                      <Creator>http://www.whoever.com/~bob</Creator>
                      <Date>19990628</Date>
                      <TimeToLive>19990831</TimeToLive>
                      <Description>...</Description>
                      <Label>...</Label>
                   </Head>
                      <Operator>http://www.whoever.com/...</Operator>
                         <Maker>...</Maker>
                         <Maker>...</Maker>
                            ...
                         <RegistrationService>...</RegistrationService>
                         <TermsAndConditions>...</TermsAndConditions>
                         <MarketRegistryLocation>
                            ...
                         </MarketRegistryLocation>
                         <Credentials>
                            <foo:yyy>...</foo:yyy>
                         </Credentials>
                         <NetworkSpecificProperties>
                            <foo:xxx>...</foo:xxx>
                         </NetworkSpecificProperties>
                   </NetworkPropertySheet>
                </Eco>
```

3.5.2 The eCo Market Layer

The eCo Market Layer allows a set of businesses to group together through a common portal or access point. By participating in an eCo Market Environment, a business can offer its goods or services in a shared context with other related businesses The eCo Market Layer is conceptually similar to the physical markets in which businesses often group together.

Each Market is governed by a set of terms and conditions for participation. These will most likely dictate a common set of protocols and business processes that are required to insure interoperability within the Market. The degree to which a Market defines and enforces these common terms and conditions is left to the discretion of its managers.

Markets define an abstract concept of a "Role". Roles define a conceptual grouping of Services that a business can provide in the Market. All eCo Markets must use two default Roles - "Market Maker" and "Market Operator". The Market Maker Role is played by those businesses that are responsible for the inception and high-level administration of the Market. There may be any number of Market Makers in a Market. Market Operators provide the day-to-day operation of a Market. Market Operators typically provide the administrative Services required to run the Market. Each Market only has one Market Operator.

Through the use of a Registry, Markets can define new types of Roles to meet their own requirements.

3.5.2.1 Published Interface

The Published Interface to a Market defines a set of queries that can be requested of that Market. The following standard queries are defined:

`MarketGetProperties` *(mandatory)*

Description:	This query returns a set of properties that describe the Market.
Call:	*Base URL* / MarketGetProperties
Parameters:	none
Return:	MarketPropertySheet

`MarketGetBusinesses` *(optional)*

Description:	This query returns a list of Businesses that participate in this Market. This list contains the identifying URI for each Business. By using its URI, a description of a Business can be obtained and its Published Interface can be queried.
Call:	*Base URL* / MarketGetBusinesses
Parameters:	None
Return:	EcoInterfaceList

`MarketGetBusinessTypes` *(optional)*

Description:	This query returns a list of Business types that participate in this Market. This list contains the URI for each Business type participating in the Market. Using these URIs, information on each type can be obtained and type equivalence can be established.
Call:	*Base URL* / MarketGetBusinessTypes
Parameters:	None
Return:	EcoTypeList

`MarketGetBusinessesByType` *(optional)*

Description:	This query returns a list of the Businesses that this Market has to offer filtered by type. A list of all the Business types participating in this Market can be obtained by calling MarketGetBusinessTypes.
Call:	*Base URL* / MarketGetBusinessesByType?Type=URI
Parameters:	**Type** (mandatory) - This is the unique identifier of the Business type in question. A list of all the Business types that participate in this Market can be obtained by calling MarketGetBusinessTypes.
Return:	EcoInterfaceList

`MarketGetBusinessRoles` *(optional)*

Description:	This query returns a list of Business Roles that are used in this Market. This list contains the URI for each Business Role used in the Market. Using these URIs, information on each Role can be obtained and type equivalence can be established. All Markets that use this mechanism must define at least two roles - Market Maker and Market Operator.
Call:	*Base URL* / MarketGetBusinessRoles
Parameters:	None
Return:	EcoTypeList

`MarketGetBusinessesByRole` *(optional)*

Description:	This query returns a list of the Businesses that this Market has to offer filtered by Role. A list of all the Business Roles used in this Market can be obtained by calling MarketGetBusinessRoles.
Call:	*Base URL* / MarketGetBusinessesByRole?Role=URI
Parameters:	**`Role`** *(mandatory)* - This is the unique identifier of the Business Role in question. A list of all the Business Roles that are used in this Market can be obtained by calling MarketGetBusinessRoles.
Return:	EcoInterfaceList

`MarketGetNetworks` *(optional)*

Description:	This query returns a list of the Networks that this Market is listed in. This list is advisory, and will contain only Network Interfaces. If there is a conflict between the list returned by this call and the list of Markets provided by the Network, the Network's list is definitive. (A Network has the final say on its membership.)
Call:	*Base URL* / MarketGetNetworks
Parameters:	None
Return:	EcoInterfaceList

3.5.2.2 Return Documents

The following document types are returned in response to queries to a Market's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

MarketPropertySheet

Description:	This document is used to describe an eCo Market. It is returned in response to the MarketGetProperties query in a Market's Published Interface.
DTD:	`MarketPropertySheet.dtd`
Key Elements & Attributes:	**MarketPropertySheet Identifier** (mandatory) - This is the URI that uniquely identifies the Market. Using this identifier, the Published Interface to the Market can be queried.

Name *(mandatory)* – This item contains the Markets' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Market can have multiple names.

Description *(mandatory)* - This is a plain language description of the Market.

Type *(optional)* - This is a list of compound elements that contain URIs that indicate this Market's place within a Registry of Market types and the URIs to the relevent Registries. Using these URIs, an interested party can automatically determine if this Market is of a given type and retrieve information on that type.

BusinessRegistryLocation (optional) - This is the URI that can be used to query the registry that defines Business types in this Market. By providing this Property, a Market is establishing a mandate that its Businesses provide type information using this registry.

ServiceRegistryLocation (optional) - This is the URI that can be used to query the registry that defines Service types in this Market. By providing this property, a Market is establishing a mandate that its Services provide type information using this registry.

InteractionRegistryLocation (optional) - This is the URI that can be used to query the registry that defines Interaction types in this Market. By providing this Property, a Market is establishing a mandate that its Interactions provide type information using this registry.

DocumentRegistryLocation (optional) - This is the URI that can be used to query the registry that defines Document types in this Market. By providing this Property, a Market is establishing a mandate that its Documents provide type information using this registry.

DataElementRegistryLocation (optional) - This is the URI that can be used to query the registry that defines Data Element types in this Market. By providing this Property, a Market is establishing a mandate that its Data Elements provide type information using this registry.

Maker (mandatory) - This is a list of URIs that uniquely identify the Market Makers for this Market. The Market Makers are the businesses that are sponsoring the market, and drive the creation of the Internet presence of that Market. Each Market Maker within the Market is defined as an eCo Business. More information on each Market Maker can be derived from its URI by querying its Published Interface.

Operator (mandatory) - This is a URI that uniquely identifies the Market Operator for this Market. The Market Operator is the business responsible for maintaining the Market on the Internet, and handles all technical issues. The Market Operator is defined as an eCo Business within the Market. More information on the Market Operator can be derived from its URI by querying its Published Interface.

`RegistrationService` (optional) - This is a URI that uniquely identifies the Service through which new Businesses can join the Market. This Service is provided by the Market Operator within the Market. By querying the Market Operator, more information on this service can be derived.

`ProcessDefinitionLanguage` (optional) - This is a string that can be used to name a business process definition language to be used in this Market.

`ProcessDefinition` (optional) - This is URI to a Process Definition used within this Market. This element may be repeated any number of times within this document. These process definitions are given in the language specified in the "ProcessDefinitionLanguage" above. Participants in Market-level processes can be specified as specific Businesses, Businesses of a given Type or Businesses that fill a certain Role.

`TermsAndConditions` (optional) - This is a URL to a document that specifies the terms and conditions of the Market. Any legal information that pertains to the use of this Market should also be presented here.

`NetworkSpecificProperties` (optional) - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Market operates. Using this Interface, Network specific data can be appended to the Market definition without compromising the structure of this document.

`MarketSpecificProperties` (optional) - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Market itself or the Markets which relate to this Market. Using this Interface, Market specific data can to be appended without compromising the structure of this document. It is envisioned that this ability might be used by a Market to describe itself in ways that are only meaningful in the context of its related Markets.

`Credentials` (optional) - This is a list of structured items that define certifications that have been granted to this Market. Each certification consists of a digitally signed block containing a plain language description, the date that the certification expires (date format is an 8-digit number in the form YYYYMMDD), an optional pointer into a registry of certification types, and a reference to the issuing authority. Note that while a signing mechanism has not yet been specified, the final specification from the W3C XML Signature Working Group is expected to be used.The DTD for the digitally signed block is currently incomplete pending the outcome of the W3C XML Signature Working Group.

Example:

```
<?xml version="1.0"?>
<MarketPropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Name>...</Name>
```

```
      <Type>...</Type>
      <Maker>...</Maker>
      <Maker>...</Maker>
      <Operator>...</Operator>
      <BusinessRegistryLocation>...</BusinessRegistryLocation>
      <ServiceRegistryLocation>...</ServiceRegistryLocation>
      <InteractionRegistryLocation>...</InteractionRegistryLocation>
      <DocumentRegistryLocation>...</DocumentRegistryLocation>
      <DataElementRegistryLocation>...</DataElementRegistryLocation>
      <RegistrationService>...</RegistrationService>
      <TermsAndConditions>...</TermsAndConditions>
      <Credentials><foo:yyy>...</foo:yyy></Credentials>
      <ProcessDefinitionLanguage>...</ProcessDefinitionLanguage >
      <ProcessDefinition>...</ ProcessDefinition>
      <ProcessDefinition>...</ ProcessDefinition>
      <MarketSpecificProperties>
         <foo:xxx>...</foo:xxx>
      </MarketSpecificProperties>
   </MarketPropertySheet>
```

3.5.3 The eCo Business Layer (Required)

The eCo Business Layer encompasses that part of an e-commerce system that represents a single entity in a trading relationship, its formal identification and the services it offers or consumes. While the name "eCo Business Layer" is used to label this concept, this Layer can represent a business, a single individual, or a business unit within a larger organization.

3.5.3.1 Published Interface

The Published Interface to a Business defines a set of queries that can be requested of that Business. The following standard queries are defined:

BusinessGetProperties *(mandatory)*

Description:	This query returns a set of properties that describe the Business.
Call:	*Base URL* / BusinessGetProperties
Parameters:	None
Return:	BusinessPropertySheet

BusinessGetServices *(optional)*

Description:	This query returns a list of Services that this Business has to offer. This list contains the identifying URI for each Service. By using its URI, the base URL for the Service can be obtained and its Published Interface can be queried. It should be noted that it is acceptable for a business to return an empty list of Services in response to this query.
Call:	*Base URL* / BusinessGetServices
Parameters:	None

Return:	EcoInterfaceList

BusinessGetServiceTypes *(optional)*

Description:	This query returns a list of the types of Services that this Business has to offer. This list contains the URI for each Service type offered by the Business. Using these URIs, information on each type can be obtained and type equivalence can be established.
Call:	*Base URL*/ BusinessGetServiceTypes
Parameters:	None
Return:	EcoTypeList

BusinessGetServicesByType *(optional)*

Description:	This query returns a list of the URIs to those Services that this Business has to offer that match a specific type. The Service types that a Business supports can be obtained by calling BusinessGetServiceTypes.
Call:	*Base URL*/BusinessGetServicesByType?Type=URI
Parameters:	**Type** *(mandatory)* - This is the unique identifier of the Service type in question. The list of allthe Service types that are provided by a Business can be obtained by calling BusinessGetServiceTypes.
Return:	EcoInterfaceList

BusinessGetMarkets *(optional)*

Description:	This query returns a list of the Markets that this Business is listed in. This list is advisory, and will contain only Market Interfaces. If there is a conflict between the list returned by this call and the list of Businesses provided by the Market, the Market's list is definitive. (A Market has the final say on its membership.)
Call:	*Base URL*/ BusinessGetMarkets
Parameters:	None
Return:	EcoInterfaceList

3.5.3.2 Return Documents

The following document types are returned in response to queries to a Business's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

`BusinessPropertySheet`

Description: This document is used to describe an eCo Business. It is returned in response to the BusinessGetProperties query in a Business's Published Interface.

DTD: `BusinessPropertySheet.dtd`

Key Elements & Attributes:

`BusinessPropertySheetIdentifier` *(mandatory)* - This is the URI that uniquely identifies the Business. Using this identifier, the Published Interface to the Business can be queried.

`Description` (mandatory) - This is a plain language description of the Business.

`Type` *(optional)* - This is a list of URIs that indicate this Business' place within a registry of Business types and the URIs to the relevant Registries. Using these URIs, an interested party can automatically determine if this Business is of a given type and retrieve information on that type.

`ServiceRegistryLocation` *(optional)* - This is the URI that can be used to query the Service Registry used by this Business. If this Business is part of a larger Market, it is likely that the Market will mandate a specific registry to be used for Service type definitions.

`InteractionRegistryLocation` *(optional)* - This is the URI that can be used to query the Interaction Registry used by this Business. If this Business is part of a larger Market, it is likely that the Market will mandate a specific registry to be used for Interaction type definitions.

`DocumentRegistryLocation` *(optional)* - This is the URI that can be used to query the Document Registry used by this Business. If this Business is part of a larger Market, it is likely that the Market will mandate a specific registry to be used for Document type definitions.

`DataElementRegistryLocation` *(optional)* - This is the URI that can be used to query the Data Element Registry used by this Business. If this Business is part of a larger Market, it is likely that the Market will mandate a specific registry to be used for Data Element type definitions.

`Name` *(mandatory)* – This item contains the Business' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Business can have multiple names.

`GeneralAddress` *(mandatory)* - This is structured item that contains the Business' full postal address. It is to this address that all physical correspondence should be sent.

`GeneralPhone` *(mandatory)* - This is a phone number that can be used to contact the business.

`GeneralEmail` *(optional)* - This is a general delivery email address that can be used to contact the business.

`TechnicalAddress` *(optional)* - This is the physical address that can be used to direct questions pertaining to the technical operation of the business' e-commerce systems. This is structured item that contains the Business' full postal address. If this address is not supplied, it is assumed that questions about the business' e-commerce systems can be sent to the General Address.

`TechnicalEmail` *(optional)* - This is an email address that can be used to direct questions pertaining to the technical operation of the business' e-commerce systems. If this address is not supplied, it is assumed that questions about the business' e-commerce systems can be sent to the GeneralEmail address.

`TechnicalPhone` *(optional)* - This is a phone number that can be used to directly contact the administrator of the business' e-commerce systems. If this phone number is not supplied, it is assumed that information pertaining to the business' e-commerce systems can be obtained using the General Phone Number.

`ProcessDefinitionLanguage` *(optional)* - This is a string that can be used to name a business process definition language to be used by this Business. If this Business is part of a larger Market, it is likely that the Market will mandate a specific process definition language if one is used.

`ProcessDefinition` *(optional)* - This is a URI to a Process Definition to describe the Choreography of this Service. This element can be repeated any number of if necessary to represent multiple discontinuous paths through this Service.

`TermsAndConditions` *(optional)* - This is a pointer to a document that contains information on the terms and conditions under which this Business trades. Any legal information that this Business wishes to communicate to its potential trading partners should be presented here.

`NetworkSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Business operates. Using this Interface, Network specific data can be appended to the Businesss definition without compromising the structure of this document.

`MarketSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Markets within which this Business operates. Using this Interface, Market specific data can be appended to the Business definition without compromising the structure of this document. It is envisioned that this ability might be used by Business to describe itself in ways that are only meaningful in a specific market context.

`BusinessSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Business that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Business itself. This allows the Business to extend its own Properties without compromising the structure of this document. It is envisioned that this ability might be used by a Business to describe specific attributes or conditions to a known set of trading partners.

`Credentials` *(optional)* - This is a list of structured items that define certifications that have been granted to this Business. Each certification consists of a digitally signed block containing a plain language description, the date that the certification expires (date format is an 8-digit number in the form YYYYMMDD), an optional pointer into a registry of certification types, and a reference to the issuing authority. Note that while a signing mechanism has not yet been specified, the final specification from the W3C XML Signature Working Group is expected to be used.The DTD for the digitally signed block is currently incomplete pending the outcome of the W3C XML Signature Working Group.

Example:

```
<?xml version="1.0"?>
<BusinessPropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Name>...</Name>
   <Type>...</Type>
      ...
   <GeneralAddress>...</GeneralAddress>
   <GeneralEmail>...</GeneralEmail>
   <GeneralPhone>...<GeneralPhone>
   <TechnicalAddress>...</TechnicalAddress>
   <TechnicalEmail>...</TechnicalEmail>
   <TechnicalPhone>...<TechnicalPhone>
   <ServiceRegistryLocation>...</ServiceRegistryLocation>
   <BusinessRegistryLocation>...</BusinessRegistryLocation>
   <DocumentRegistryLocation>...</DocumentRegistryLocation>
   <DataElementRegistryLocation>...</DataElementRegistryLocation>
   <TermsAndConditions>...</TermsAndConditions>
   <Credentials>....</Credentials>
   <ProcessDefinitionLanguage>
      ...
   </ProcessDefinitionLanguage>
   <ProcessDefinition>...</ProcessDefinition>
   <MarketSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </MarketSpecificProperties>
   <BusinessSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </BusinessSpecificProperties>
</BusinessPropertySheet>
```

3.5.4 The eCo Service Layer

This Layer represents that part of an e-commerce system that is responsible for providing meta-data about business services and the exchange of commercial documents.

Services are interfaces to a business process. Each Service offered by a Business provides the ability for a trading partner to interact with that Business in some way.

A Business or "Service Provider" offers a Service to "Service Consumers". Any Business can be both a provider and a consumer of Services. Example Services offered by a company that makes widgets might be:

- "Examine my catalogue of widgets."
- "Buy a widget."

- "Submit engineering change order."
- "Become a VAR."
- "Find the cheapest price on this item and then apply for a loan to pay based on my credit rating and ability to establish a long term relationship."
- "Initiate manufacturing corrective action."

Services create their value by grouping Interactions and other Services together. The logic that governs the order of execution of the Interactions and Services contained within a Service is called the Service Choreography. The definition of a Service Choreography can be explicitly communicated using a formal Business Process Definition Language or it can be implicitly communicated through the natural flow of the Service.

In some cases, eCo Services, Interactions, Documents, and Information Items will be fully defined in eCo compliant terms, and in others an eCo Service might be used to "wrap" an e-commerce offering that is defined externally using a human or machine readable specification. For example, an eCo Service might "wrap" a traditional EDI system defined in accordance with a human readable specification. By examining the Interface, potential partners can be made aware of offerings or get the specification. Regardless of the method used, a potential trading partner must be able to fully determine the protocols required to use a Service by examining the information available through the Service Environment. Furthermore, it is a requirement that a trading partner should be able to automatically determine if a Service's interoperability requirements have changed prior to using the Service.

It should be noted, if a Service explicitly uses one or more Sub-Services then a Service Consumer must also query those Sub-Services, according to the Choreography, to fully determine the interoperability requirements of the enclosing Service. It is also possible to "mask" a sub-service. In essence, a masked sub-service is not exposed to a Service Consumer, but is used by a Service that a Consumer might call. This can be useful if you are calling on Services offered by another business, but do not need to expose those Services to the Consumer (for example, if you are arranging payment with a bank as part of the Service, but do not want the Consumer to have access to those details).

3.5.4.1 Published Interface

The Published Interface to a Service defines a set of queries that can be requested of that Service. The following standard queries are defined:

`ServiceGetProperties` *(mandatory)*

Description:	This query returns a set of Properties that describe the Service.
Call:	*Base URL* / ServiceGetProperties
Parameters:	None
Return:	ServicePropertySheet

`ServiceGetInteractions` *(optional)*

Description:	This query returns the list of all the Interactions that make up this Service. Because a Service's Choreography is a complex map, it is not guaranteed that the Interactions are in the order in which they will be encountered when the Service is used.
Call:	*Base URL* / ServiceGetAllInteractions

Parameters: None

Return: EcoInterfaceList

ServiceGetInteractionTypes *(mandatory if ServiceGetInteractionsByType is implemented)*

Description: This query returns a list of the types of Interactions that this Service uses. This list contains the URI for each Interaction type.

Call: *Base URL* / ServiceGetInteractionTypes

Parameters: None

Return: EcoTypeList

ServiceGetInteractionsByType *(optional)*

Description: This query returns a list of the URIs to those Interactions that compose this Service and match a specific type. The Interaction types that a Service contains can be obtained by calling ServiceGetInteractionTypes (which must also be implemented).

Call: *Base URL* / ServiceGetInteractionsByType?Type=URI

Parameters: **Type** *(mandatory)* - This is the unique identifier of the Interaction type in question. The list of all the Interaction types that a Service contains can be obtained by calling ServiceGetInteractionTypes.

Return: EcoInterfaceList

ServiceGetSubServices *(optional)*

Description: This query returns a list of URIs to the Published Interfaces of the Sub-Services used by this Service. This allows a Service Consumer to query those Sub-Services for their Property Sets. It should also be noted that not all Services called from this Service need to be exposed in this manner. Services can be "masked" as long as the requirements for interoperation with the masked Services are incorporated into this Service's properties. Through a succession a queries, a chain of ownership of Services can be established. As noted above, a Service Consumer should examine each of a Service's Sub-Services to fully determine the enclosing Service's interoperability requirements.

Call: *Base URL* / ServiceGetSubServices

Parameters: None

Return: EcoInterfaceList

ServiceGetSubServiceTypes *(optional)*

Description: This query returns a list of the types of Sub-Services that this Service uses. This list contains the URI for each Sub-Services type.

Call: *Base URL* / ServiceGetSubServiceTypes

Parameters: None

Return: EcoTypeList

`ServiceGetSubServicesByType` *(optional)*

Description: This query returns a list of the URIs to those Sub-Services that are used by this Service and match a specific type. The Sub-Service types that a Service contains can be obtained by calling ServiceGetSubServiceTypes.

Call: *Base URL*/ ServiceGetSubServicesByType?Type=URI

Parameters: **`Type`** *(mandatory)* - This is the unique identifier of the Sub-Service type in question. The list of all the Sub-Service types that a Service contains can be obtained by calling ServiceGetSubServiceTypes.

Return: EcoInterfaceList

`ServiceGetFirstStep` *(optional)*

Description: This query returns the first step in a Service's Choreography. It is important to realize that the Interface returned could be the Published Interface to an Interaction or to another Service. The type of the Interface returned can be determined by examining the EcoInterfaceList. With this information, the Service consumer can then act as appropriate to examine the next step of the Service.

This method of gaining an understanding of Service Choreography is intended as a simple alternative to using a formal process definition language.

Call: *Base URL*/ServiceGetFirstStep

Parameters: None

Return: EcoInterfaceList

`ServiceGetNextStep` *(optional)*

Description: This query returns the next step (or possible steps) within a Service according to the callers current position within the Service Choreography. An empty list is returned if there is no next step (that is, if the end of the Service is reached). It is important to realize that the Interface returned could be the Published Interface to an Interaction or to another Service. The type of the Interface returned can be determined by examining the EcoInterfaceList.

This method of gaining an understanding of Service Choreography is intended as a simple alternative to using a formal process definition language.

Call: *Base URL*/ ServiceGetNextStep?StepID=URI

Parameters: **`StepID`** *(mandatory)* - This is the URI to a previous Step in the Service's Choreography.

Return: EcoInterfaceList

ServiceGetPreviousStep *(optional)*

Description:	This query returns the previous step (or possible steps) within a Service's Choreography. By returning a previous Interaction, a Service is implying that it is reasonable to "fall back" and execute that Interaction. If it is not reasonable to perform the previous Interaction, an empty Interface list should be returned. It is important to realize that the Interface returned could be the Published Interface to an Interaction or to another Service. The type of the Interface returned can be determined by examining the EcoInterfaceList. This method of gaining an understanding of Service Choreography is intended as a simple alternative to using a formal process definition language.
Call:	*Base URL* / ServiceGetPreviousStep?StepID=URI
Parameters:	**StepID** *(mandatory)* - This is the URI to a previous step in the Service's Choreography.
Return:	EcoInterfaceList

ServiceGetBusiness *(optional)*

Description:	This query returns a list of the Businesses that offer this Service. This list is advisory, and will contain only Business Interfaces. If there is a conflict between the list returned by this call and the lists of Services and Sub-Services provided by a Business, the Business's list is definitive. (A Business has the final say on which Services it is actually using.)
Call:	*Base URL* / ServiceGetBusiness
Parameters:	None
Return:	EcoInterfaceList

3.5.4.2 Return Documents

The following document types are returned in response to queries to a Service's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

ServicePropertySheet

Description:	This document is used to describe an eCo Service. It is returned in response to a query to the ServiceGetProperties query in a Service's Published Interface.
DTD:	`ServicePropertySheet.dtd`
Key Elements & Attributes:	**Identifier** *(mandatory)* - This is the URI that uniquely identifies the Service. Using this identifier, the Published Interface to the Service can be queried.

Scope *(mandatory)* - This attribute must be one of the following three values: "published", "obscured" or "private". A Service with a "published" scope is intended for use directly by a Service Consumer. Published Services are initial entry points into business processes and should be listed in the Business's Property Set. Services marked as being "obscured " are not intended for direct use by a Service Consumer, but can be exposed as a Sub-Service within a published Service. Private Services are not intended for exposure to the Service Consumer. To use a Private Service, it must be called as a sub-service of another Service. The Private Service will be "masked" (that is, it will not be exposed to the Service Consumer), while the enclosing Service will be available to the Consumer. Scope can be useful as a means of filtering Services.

Description *(mandatory)* - This is a plain language description of the Service.

Name *(mandatory)* – This item contains the Services' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Service can have multiple names.

Type *(optional)* - This is a list of URIs that indicate this Service's place within a registry of Service types and the URIs to the relevent Registries. Using these URIs, an interested party can automatically determine if this Service is of a given type and retrieve information on that type.

InteractionRegistryLocation *(optional)* - This is the URI that can be used to query the Interaction Registry used by this Service.

DocumentRegistryLocation *(optional)* - This is the URI that can be used to query the Document Registry used by this Service.

DataElementRegistryLocation *(optional)* - This is the URI that can be used to query the Data Element Registry used by this Service.

Specification *(mandatory if the Service is not fully defined in eCo compliant terms)* This property contains a URI to a plain language specification of this Service and a plain text description of that specification. If this Service is not defined in eCo compliant terms, the URI specified by this Property can provide the documentation that a potential Consumer requires to use this Service. This allows a Business to "wrap" an existing e-commerce protocol as an eCo Service without having to try to define it completely in terms of eCo Interactions, Documents and Information Items. In this situation, this Property must point to a specification that defines the protocols required to use this service in sufficient detail that they may be completely implemented by an interested Service Consumer.

ServicePreConditions *(optional)* - This Property can be used to define any technical or administrative preconditions to the use of this Service. These preconditions are defined in plain language.

TermsAndConditions *(optional)* - This is a URL to a document that specifies any legal terms and conditions that surround the use of this Service. Any legal information that pertains to the use of this Service should also be presented here.

NetworkSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Service operates. Using this Interface, Network specific data can be appended to the Service definition without compromising the structure of this document.

`BusinessSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Business that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Business that defined this Service. This allows the owning Business to extend the Service's Properties without compromising the structure of this document.

`MarketSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Markets within which this Service is used. This allows market specific data to be appended to the Service definition without compromising the structure of this document.

`ProcessDefinitionLanguage` *(optional)* - This is a string that can be used to name a process definition language used to describe the flow of Interactions and Sub-Services within this Service (for example, PSL).

`Credentials` *(optional)* - This is a list of structured items that define certifications that have been granted to this Service. Each certification consists of a digitally signed block containing a plain language description, the date that the certification expires (date format is an 8-digit number in the form YYYYMMDD), an optional pointer into a registry of certification types, and a reference to the issuing authority. Note that while a signing mechanism has not yet been specified, the final specification from the W3C XML Signature Working Group is expected to be used.The DTD for the digitally signed block is currently incomplete pending the outcome of the W3C XML Signature Working Group.

Example:

```
<?xml version="1.0"?>
<ServicePropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Name>...</Name>
   <Type>...</Type>
   <InteractionRegistryLocation>...</InteractionRegistryLocation>
   <DocumentRegistryLocation>...</DocumentRegistryLocation>
   <DataElementRegistryLocation>...</DataElementRegistryLocation>
   <TermsAndConditions>...</TermsAndConditions>
   <Credentials>....</Credentials>
   <ProcessDefinitionLanguage>
      ...
   </ProcessDefinitionLanguage>
   <ProcessDefinition>...</ProcessDefinition>
   <MarketSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </MarketSpecificProperties>
   <BusinessSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </BusinessSpecificProperties>
<</ServicePropertySheet>
```

3.5.5 The eCo Interaction Layer

Interactions represent the atomic blocks from which Services are built. Each Interaction consists of a request and response pair. Both the request and response can contain one or more Documents.

The Interactions in a Service may or may not require adherence to a formal Process Definition.

The specific Interactions within a Service are going to vary widely depending on the domain, purpose, function and quality of service that is offered by the service provider. Often, well known interaction specifications will be used within particular domains. Examples of these domains include IOTP and Rosetta Net.

3.5.5.1 Published Interface

The Published Interface to a Interaction defines a set of queries that can be requested of that Interaction. The following standard queries are defined:

`InteractionGetProperties` *(mandatory)*

Description:	This query returns a set of properties that describe an Interaction.
Call:	*Base URL* / InteractionGetProperties
Parameters:	None
Return:	InteractionPropertySheet

`InteractionGetServices` *(optional)*

Description:	This query returns a list of the Services that use this Interaction. This list is advisory, and will contain only Service Interfaces. If there is a conflict between the list returned by this call and the list of Interacrtions provided by a Service, the Service's list is definitive. (A Service has the final say on which Interactions it uses.)
Call:	*Base URL* / InteractionGetServices
Parameters:	None
Return:	EcoInterfaceList

3.5.5.2 Return Documents

The following document types are returned in response to queries to an Interaction's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

InteractionPropertySheet

Description:	This document is used to describe an eCo Interaction. It is returned in response to a query to the InteractionGetProperties query in the Interaction's Published Interface.
DTD:	`InteractionPropertySheet.dtd`
Key Elements & Attributes:	**Identifier** *(mandatory)* - This is the URI that uniquely identifies the Interaction. Using this identifier, the Interaction's Published Interface can be queried for further information.

Name *(mandatory)* – This item contains the Interactions' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Interaction can have multiple names.

Description *(mandatory)* - This is a plain language description of the Interaction.

Type *(optional)* - This is a list of URIs that indicate this Interaction's place within a registry of Interaction types and the URIs to the relevant Registries. Using these URIs, an interested party can automatically determine if this Interaction is of a given type and retrieve information on that type. The type of the Interaction specifies the protocol being used to exchange the Interaction's documents.

InteractionMessageContainerType *(optional)* - This is a list of URIs that indicate the type of Message Container protocol within a registry of Interaction types and the URIs to the relevant Registries. Using these URIs, an interested party can automatically determine if this Interaction is of a given type and retrieve information on that type.

This property is intended for use where the Message Container is independent of the Interaction's type (specified above.)

DocumentRegistryLocation *(optional)* - This is the URI that can be used to query the Document Registry used by this Interaction.

DataElementRegistryLocation *(optional)* - This is the URI that can be used to query the Data Element Registry used by this Interaction.

Specification *(mandatory if the Interaction is not fully defined in eCo compliant terms)* - This property contains a URI to a plain language specification of this Interaction.

InputDocument *(mandatory)* - This property is a list of URIs that uniquely identifies the possible input documents for this Interaction.

OutputDocument *(mandatory)* - This property is a list of URIs that uniquely identifies the possible valid output documents for this Interaction.

ErrorDocument *(mandatory)* - This property is a list of URIs that uniquely identifies the possible error documents for this Interaction.

Execution *(optional)* - If appropriate to the Interaction's type, This Property gives the URI that can be used to run the Interaction. In many Interaction scenarios, this URI will be used as a location to submit the input document (InputDocument) to. As a response to this submission a document of a type defined in either OutputDocument or ErrorDocument is expected.

NetworkSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Interaction operates. Using this Interface, Network specific data can be appended to the Interaction definition without compromising the structure of this document.

BusinessSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Business that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Business that defined this Interaction. This allows the owning Business to extend the Interaction's Properties without compromising the structure of this document.

MarketSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Markets within which this Interaction is used. This allows market specific data to be appended to the Interaction definition without compromising the structure of this document.

Example:

```
<?xml version="1.0"?>
<InteractionPropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Name>...</Name>
   <Type>...</Type>
   <Location>...</Location>
   <Execution>...</Execution>
   <InputDocument>...</InputDocument>
      ...
   <OutputDocument>...</OutputDocument>
      ...
   <ErrorDocument>...</ErrorDocument>
      ...
   <DocumentRegistryLocation>....</DocumentRegistryLocation>
   <DataElementRegistryLocation>...</DataElementRegistryLocation>
      ...
   <MarketSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </MarketSpecificProperties>
   <BusinessSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </BusinessSpecificProperties>
</InteractionPropertySheet>
```

3.5.6 The eCo Document Layer

Used as information item containers, Documents represent a physical encapsulation of information items. Documents are in the form of XML (with DTD's) and XML Schemas (DTD's as instances with other information) according to the W3C recommendations.

3.5.6.1 Published Interface

The Published Interface to a Document defines a set of queries that can be requested of that Document. The following standard queries are defined:

DocumentGetProperties *(mandatory)*

Description:	This query returns a set of properties that describe a Document.
Call:	*Base URL* / DocumentGetProperties
Parameters:	none
Return:	DocumentPropertySheet

DocumentGetDataElements *(optional)*

Description:	This query returns a list of Data Elements that comprise this Document. This list contains the identifying URI for each Data Element. By using its URI, a description of the Data Element can be obtained and its Published Interface can be queried.
Call:	*Base URL* / DocumentGetDataElements
Parameters:	None
Return:	EcoInterfaceList

DocumentGetDataElementTypes *(optional)*

Description:	This query returns a list of Data Element types that comprise this Document. This list contains the URI for each Data Element type in this Document. Using these URIs, information on each type can be obtained and type equivalence can be established.
Call:	*Base URL* / DocumentGetDataElementTypes
Parameters:	None
Return:	EcoTypeList

DocumentGetDataElementsByType *(optional)*

Description:	This query returns a list of the Data Elements in this Document filtered by type. A list of all the Data Element types in this Document can be obtained by calling DocumentGetDataElementTypes.
Call:	*Base URL* / DocumentGetDataElementsByType?Type=URI

Parameters:	**Type** *(mandatory)* - This is the unique identifier of the Data Element type in question. A list of all the Data Element types that participate in this Market can be obtained by calling DocumentGetDataElementTypes.
Return:	EcoInterfaceList

DocumentGetInteractions *(optional)*

Description:	This query returns a list of the Interactions that use this Document. This list is advisory, and will contain only Interaction Interfaces. If there is a conflict between the list returned by this call and the list of Documents used by an Interaction, the Interaction's list is definitive. (An Interaction has the final say on which Documents it uses.)
Call:	Base URL/DocumentGetInteractions
Parameters:	None
Return:	EcoInterfaceList

3.5.6.2 Return Documents

The following document types are returned in response to queries to a Document's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

DocumentPropertySheet

Description:	This document is used to describe an eCo Document used within an Interaction. It is returned in response to the DocumentGetProperties query in a Document's Published Interface.
DTD:	`DocumentPropertySheet.dtd`
Key Elements & Attributes:	**Identifier** *(mandatory)* - This is the URI that uniquely identifies the Interaction Document. Using this identifier, further information on the Document can be obtained.
	Name *(mandatory)* – This item contains the Documents' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Document can have multiple names.
	Description *(mandatory)* - This is a plain language description of the Document.
	Type *(optional)* - This is a list of URIs that indicate this Document's place within a registry of Document types and the URIs to the relevant Registries. Using these URIs, an interested party can automatically determine if this Document is of a given type and retrieve information on that type.

DataElementRegistryLocation *(optional)* - This is the URI that can be used to query the Data Element Registry used by this Document.

Specification *(mandatory if DocumentType is undefined)* - This property contains a URI to a DTD or Schema for this Document.

NetworkSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Document is used. Using this Interface, Network specific data can be appended to the Document definition without compromising the structure of this document.

BusinessSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Business that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Business that defined this Document. This allows the owning Business to extend the Document's properties without compromising the structure of this document.

MarketSpecificProperties *(optional)* - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Markets within which this Document is used. This allows market specific data to be appended to the Document definition without compromising the structure of this Document.

Example:

```
<?xml version="1.0"?>
<DocumentPropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
   <Creator>http://www.whoever.com/~bob</Creator>
      ...
   </Head>
   <Name>...</Name>
   <Type>...</Type>
   <DataElementRegistryLocation>...</DataElementRegistryLocation>
   <MarketSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </MarketSpecificProperties>
   <BusinessSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </BusinessSpecificProperties>
</DocumentPropertySheet>
```

3.5.7 The eCo Data Element Layer

As the markup elements within a Document, Data Elements serve to encapsulate data (or code) to be used in a particular Interaction. The information items will be concretely defined according to the XML specification for syntax and organized into the eCo Semantic Recommendation (XML) in order to have an approach to document architecture that promotes interoperability.

Documents can be created using a flat set of information items with no additional intermediate structure added. For example, a document may contain the information items:

- Name
- Address
- Phone
- Email

Another document with the same information items may group them into a structure, for example, Contact Information, which includes:

- Name
- Address
- Phone
- Email

The element groupings (or structures) will be kept in a registry along with the documents that use them. Through the registry, documents and document structures can be understood.

Further discussion of this can be found in the Registry section of the specification.

3.5.7.1 Published Interface

The Published Interface to a Data Element defines a set of queries that can be requested of that Data Element. The following standard queries are defined:

`DataElementGetProperties` *(mandatory)*

Description:	This query returns a set of properties that describe an Interaction.
Call:	*Base URL*/ DataElementGetProperties
Parameters:	none
Return:	DataElementPropertySheet

`DataElementGetSubDataElement` *(optional)*

Description:	This query returns a list of URIs to the Published Interfaces of the Sub-DataElements contained in this Data Element.
Call:	*Base URL*/ DataElementGetSubDataElements
Parameters:	None
Return:	EcoInterfaceList

`DataElementGetSubDataElementTypes` *(optional)*

Description:	This query returns a list of the types of Sub-DataElements that this Data Element uses. This list contains the URI for each Sub-DataElement type.
Call:	*Base URL*/ DataElementGetSubDataElementTypes
Parameters:	None

Return: EcoTypeList

DataElementGetSubDataElementsByType *(optional)*

Description: This query returns a list of the URIs to those Sub-Data Elements that are contained by this Data Element and match a specific type. The Sub-Data Element types that a Data Element contains can be obtained by calling DataElementGetSubDataElementTypes.

Call: *Base URL*/ DataElementGetSubDataElementsByType?Type=URI

Parameters: **Type** *(mandatory)* - This is the unique identifier of the Sub-Data Element type in question. The list of all the Sub-Data Element types that a Data Element contains can be obtained by calling DataElementGetSubDataElementTypes.

Return: EcoInterfaceList

DataElementGetDocuments *(optional)*

Description: This query returns a list of the Documents that use this Data Element. This list is advisory, and will contain only Document Interfaces. If there is a conflict between the list returned by this call and the list of Data Elements provided by a Document, the Document's list is definitive. (A Document has the final say on which Data Elements it uses.)

Call: *Base URL*/DataElementGetDocuments

Parameters: None

Return: EcoInterfaceList

3.5.7.2 Return Documents

The following document types are returned in response to queries to a Data Element's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

DataElementPropertySheet

Description: This document is used to describe an eCo Data Element used within a Document. It is returned in response to a query to the DataElementGetProperties query in a Data Element's Published Interface.

DTD: `DataElementPropertySheet.dtd`

Key Elements & Attributes: **Identifier** *(mandatory)* - This is the URI that uniquely identifies The Data Element. Using this identifier, further information on Attributes: the Data Element can be obtained.

Name *(mandatory)* – This item contains the DataElements' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Data Element can have multiple names.

`Description` *(mandatory)* - This is a plain language description of the Data Element.

`Type` *(optional)* - This is a list of URIs that indicate this Data Element's place within a registry of Data Element types and the URIs to the relevant Registries. Using these URIs, an interested party can automatically determine if this Data Element is of a given type and retrieve information on that type.

`Specification` *(mandatory if DocumentType is undefined)* - This property contains a URI to a DTD or Schema for this Document.

`NetworkSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Network that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Network within which this Data Element is used. Using this Interface, Network specific data can be appended to the Data Element definition without compromising the structure of this document.

`BusinessSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Business that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Business that defined this Data Element. This allows the owning Business to extend the Data Elements's properties without compromising the structure of this document.

`MarketSpecificProperties` *(optional)* - This is a list of URI pairs. The first URI identifies the Market that the properties are specific to, and the second URI identifies a related set of Properties. The structure of the Properties is defined by the Markets within which this Data Element is used. This allows market specific data to be appended to the Data Element definition without compromising the structure of this document.

Example:

```
<?xml version="1.0"?>
<DataElementPropertySheet xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator> ...
   </Head>
   <Name>...</Name>
   <Type>...</Type>
   <MarketSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </MarketSpecificProperties>
   <BusinessSpecificProperties>
      <foo:xxx>...</foo:xxx>
   </BusinessSpecificProperties>
</DataElementPropertySheet>
```

3.6 Type Registry Details

3.6.1 Common Registry Queries and Documents

Type Registries list types, as opposed to instances. For example, a Type Registry might define a Service Type of "purchase steel". This Service Type might then have specific instances such as "purchase steel from ABC Ltd.", "purchase steel from Acme Corp.", and so on.

Each Type Registry entry is identified by a unique URI. By comparing these identifying URIs, Type equivalency can be determined. If the URIs used to type two objects are the same, then those object are of the same type.

The following queries are available in all Type Registry Interfaces. They are defined here to prevent repetition in the Interface definitions. It should also be noted that the common queries defined in the common interface section are also available in all Registry Interfaces. As with eCo Layers, the information is returned in XML format.

3.6.1.1 Published Interface

The Published Interface to a Type Registry defines a set of queries that can be requested of that Registry. The following standard queries are defined:

`RegistryGetProperties` *(mandatory)*

Description:	This query returns a set of properties that describes this Type Registry (not a particular type definition in this Registry.)
Call:	*Base URL*/ RegistryGetProperties
Parameters:	none
Return:	RegistryPropertySheet

`RegistryGetChildren` *(mandatory)*

Description:	This query returns a list of the types that have been directly derived from this type (its children.) If a type is a child of another type, then it is a refinement of that type (a sub-type). Because of this relationship, a sub-type is also considered to be an instance of it's super-types. For example, a manufacturer of carburetors might be considered to be a sub-type of engine part manufacturer. In this case, the carburetor manufacturer can also be considered to be an engine parts manufacturer.
Call:	*Base URL*/ RegistryGetChildren?Node=URI
Parameters:	Node (mandatory) - This is the unique identifier of the node being queried.
Return:	RegistryNodeList

`RegistryGetParent` *(mandatory)*

Description:	This query returns the ID of the type that this type has been derived from (its parent.) This call will only ever return one node in the returned RegistryNodeList. If a type is a

parent of another type, then it is a generalization on that type (a super-type). Because of this relationship, a sub-type is also considered to be an instance of it's super-types. For example, a manufacturer of bolts might be considered to be a sub-type of hardware manufacturer. In this case, the bolt manufacturer can also be considered to be a hardware manufacturer.

Call:	*Base URL*/ RegistryGetParent?Node=URI
Parameters:	**Node** *(mandatory)* - This is the unique identifier of the node being queried.
Return:	RegistryNodeID

RegistryGetSiblings *(mandatory)*

Description:	This query returns a list of the types that have been directly derived from the same type as this type (its siblings.)
Call:	*Base URL*/ RegistryGetSiblings?Node=URI
Parameters:	**Node** *(mandatory)* - This is the unique identifier of the node being queried.
Return:	RegistryNodeList

RegistryGetAllDescendants *(mandatory)*

Description:	This query returns a list off all the descendants that a Type Node has (those nodes that are below it in the type hierarchy). This query can be used to retrieve all the type definitions in a particular Registry by calling it with the root node of that Registry. It can also be used to get all the types that are derived from a particular type. This query might also be used to answer the example question: "What are all the types of plastic suppliers defined in this Registry?"
Call:	*Base URL*/ RegistryGetAllDescendants?Node=URI
Parameters:	**Node** *(mandatory)* - This is the unique identifier of the node being queried.
Return:	RegistryNodeList

RegistryIsDescendant *(mandatory)*

Description:	This query returns a true response if the given type "Node1" is a descendant of the type identified by "Node2". This query is useful for determining if a given type is derived from another one. This query might be used to answer the example question: "Is a men's shoe manufacturer a manufacturer?"
Call:	*Base URL*/ RegistryIsDescendant?Node1=URI&Node2=URI
Parameters:	**Node1** *(mandatory)* - This is the unique identifier of the node being queried.
	Node2 *(mandatory)* - This is the unique identifier of the possible parent node.
Return:	EcoBoolean

RegistryGetNodeDefinition *(mandatory)*

Description: This query returns the type definition for the given node.

Call: *Base URL* / RegistryGetNodeDefinition?Node=URI

Parameters: **Node** *(mandatory)* - This is the unique identifier of the node being queried.

Return: NodeTypeDefinition

Note that URIs can be compared to determine equivalency. If the the URIs being compared are the same, then they are both the same type.

3.6.1.2 Return Documents

The following document types are returned in response to queries to a Registry's Interface.

As with all XML documents defined in this specification, the portions of these documents that contain plain language descriptions may be made available in several languages. The language to be used in any given instance will be determined by HTTP content negotiation.

The eCo DTDs for these documents are detailed in Appendix A of this document.

RegistryNodeList

Description: This document is used to return a list of zero, one or more nodes in a Registry's type lattice. This list differs from an eCoTypeList in that the base URI for the Registry is not included with each node.

DTD: `RegistryNodeList.dtd`

Key Elements & Attributes: **Node** *(mandatory)* - This is a URI that uniquely identifies a node in a Registry's type lattice. There may be any number of Node elements in a RegistryNodeList.

Example:

```
<?xml version="1.0"?>
<Eco xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
      ...
   </Head>
   <Node>http://www.reg.org/types/market/steel/pipe</Node>
   <Node>http://www.reg.org/types/market/toys/games</Node>
</RegistryNodeList>
```

RegistryNodeID

Description: This document is used to return a single node in a Registry's type lattice. Note that the base URI for the Registry is not included with the Node.

DTD: `RegistryNodeID.dtd`

Key Elements & Attributes: Node (mandatory) - This is a URI that uniquely identifies a node in a Registry's type lattice. There can only be a single Node element in a RegistryNodeID.

Example:

```
<?xml version="1.0"?>
<RegistryNodeID xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
      ...
   </Head>
   <Node>http://www.reg.org/types/market/steel/pipe</Node>
</RegistryNodeID>
```

`RegistryPropertySheet`

Description: This document is used to describe an eCo Registry. It is returned in response to a query to the RegistryGetProperties query in a Registry's Published Interface.

DTD: `RegistryPropertySheet.dtd`

Key Elements & Attributes: **`Identifier`** *(mandatory)* - This is the URI that uniquely identifies the Registry. Using this identifier, the Published Interface to the Registry can be queried.

`Operator` *(mandatory)* - This is a URI that uniquely identifies the Registry Operator for this Registry. The Registry Operator is the business responsible for maintaining the Registry on the Internet, and handles all technical issues. The Registry Operator is defined as an eCo Business. More information on the Registry Operator can be derived from its URI by querying its Published Interface.

`Type` *(mandatory)* - This identifies the type of Registry described by the Published Interface. Possible values for this element are MarketRegistry, BusinessRegistry, InteractionRegistry, ServiceRegistry, DocumentInformationItemRegistry.

`Name` *(mandatory)* – This item contains the Registrys' name, which can be identified as a Legal name, Trade name, Common name, Former name or Alias. This element can also be identified as Other to indicate that it is any type of Name. A Registry can have multiple names.

`RegistryDocumentTypeDefinitionLanguage` *(mandatory)* - This element contains the MIME type for the document type definition language (DTD or schema language) being used by this Registry. All document type definition in this Registry will be defined in this language. In the unlikely event that a Registry wishes to support more than one such language, a separate Interface must be created for that Registry.

`RegistrySpecificProperties` *(optional)* - This is a URI to a related document whose structure is defined by the Registry itself. This allows the Registry to extend its own properties without compromising the structure of this document. It is envisioned that this ability might be used by a Registry to describe specific attributes or conditions to a known set of users.

`Credentials` *(optional)* - This is a list of structured items that define certifications that have been granted to this Registry. Each certification consists of a digitally signed block containing a plain language description, the date that the certification expires (date format is an 8-digit number in the form YYYYMMDD), an optional pointer into a registry of certification types, and a reference to the issuing authority. Note that while a signing mechanism has not yet been specified, the final specification from the W3C XML Signature Working Group is expected to be used.The DTD for the digitally signed block is currently incomplete pending the outcome of the W3C XML Signature Working Group.

Example:

```
<?xml version="1.0"?>
<RegistryPropertySheet xmlns = 'http://www.commerce.net/eco'
            type  = 'Business'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
      ...
   </Head>
   <Name>...</Name>
   <Operator>http://www.anybody.com</Operator>
   <Credentials>
      <foo:yyy>...</foo:yyy>
   </Credentials>
   <RegistryDocumentTypeDefinitionLanguage>
      ...
   </RegistryDocumentTypeDefinitionLanguage>
   <RegistrySpecificProperties>
      <foo:xxx>...</foo:xxx>
   </RegistrySpecificProperties>
</RegistryPropertySheet>
```

NodeTypeDefinition

Description: This document is used to describe the properties of a node in a Registry of types. It provides some simple meta-data on the type including any associated DTD's or Schema.

DTD: `NodeTypeDefinition.dtd`

Key Elements & Attributes:

NodeTypeDefinitionIdentifier *(mandatory)* - This attribute contains the URI that uniquely identifies this Node in the type Registry. Using this identifier the issuing Registry can be queried for information on this type Node.

NodeTypeDefinitionRegisteringBusiness *(optional)* - This attribute contains the URI representing the Published Interface of the Business responsible for registering this Type Node in this Registry. Using this URI the Business's Published Interface can be queried and information on that Business can be obtained.

Description *(mandatory)* - This is a plain language description of the type represented by this Type Node.

DocumentTypeDefinition *(see explanation)* - This element contains a URI to a document type definition (a DTD or schema) that describes the document or element type in question. This information is mandatory if this Type Node is describing a Document type or an Information Item type. This element can be omitted when describing other types.

Specification *(optional)* - This element contains a URI to an external plain-language specification for this type. This element is most often used when this Type Node is describing an Interaction Type or a Message Container Type although it can be used whenever it is appropriate to provide a source of extensive written documentation to define a type.

`RegistrySpecificProperties` *(optional)* - This is a list of URIs to a related set of Documents whose structure is defined by the Registry within which this Document is used. This allows Registry specific data to be appended to this document without compromising the structure of this Document.

Example:

```
<?xml version="1.0"?>
<NodeTypeDefinition xmlns = 'http://www.commerce.net/eco'>
   <Head>
      <Identifier>http://www.whoever.com/...</Identifier>
      <Creator>http://www.whoever.com/~bob</Creator>
      ...
   </Head>
   <Type>...</Type>
   <Specification>...</Specification>
   <RegistrySpecificProperties >
      <foo:xxx>...</foo:xxx>
   </RegistrySpecificProperties >
</NodeTypeDefinition>
```

3.6.2 Market Registry

The Market Registry contains the information needed to type eCo Markets. As such, the Market Registry is most often used when multiple markets are grouped together in an eCo Network although stand-alone Markets can still type themselves using a common Market Registry.

3.6.2.1 Published Interface

The Published Interface to a Market Registry defines a set of queries that can be requested of that Registry. The following standard queries are defined:

`MarketRegistryGetMarketTypeRoot` *(mandatory)*

Description: This query returns the root node for the tree of Market types. Using this node, information on a specific Market type can be obtained.

Call: *Base URL* / MarketRegistryGetMarketTypeRoot

Parameters: none

Return: RegistryNodeID

`MarketRegistryGetBusinessRolesRoot` *(optional)*

Description: This query returns the root node for the tree of Business Roles. Using this node, information on a specific Buisness Role type can be obtained. Each Market Registry that implements this query must define at least two Roles in its hierarchy of Business Roles: Market Maker and Market Operator. By adding other types to this hierarchy,

	other business Roles can be defined for use within a Market.
Call:	*Base URL* / MarketRegistryGetBusinessRolesRoot
Parameters:	none
Return:	RegistryNodeID

3.6.3 Business Registry

The Business Registry contains the information needed to type eCo Businesses. As such, the Business Registry is most often used when multiple businesses are grouped together in an eCo Market although stand-alone Businesses can still type themselves using a common Business Registry.

3.6.3.1 Published Interface

The Published Interface to a Business Registry defines a set of queries that can be requested of that Registry. The following standard queries are defined:

BusinessRegistryGetBusinessTypeRoot *(mandatory)*

Description:	This query returns the root node for the tree of Business types. Using this node, information on a specific Business type can be obtained.
Call:	*Base URL* / BusinessRegistryGetBusinessTypeRoot
Parameters:	none
Return:	RegistryNodeID

3.6.4 Interaction Registry

The Interaction Registry contains the information needed to type eCo Interactions. The Interaction Registry contains two trees of type information: one that is used to type Interaction types (the protocol used to conduct the document exchange) and one that is used to type Message Containers (encoding that may be used on the documents being exchanged.)

3.6.4.1 Published Interface

The Published Interface to an Interaction Registry defines a set of queries that can be requested of that Registry. The following standard queries are defined:

InteractionRegistryGetInteractionTypeRoot *(mandatory)*

Description:	This query returns the root node for the tree of Interaction types. Using this node, information on a specific Interaction type can be obtained.
Call:	*Base URL* / InteractionRegistryGetInteractionTypeRoot
Parameters:	none
Return:	RegistryNodeID

`InteractionRegistryMessageComtainerTypeRoot` *(mandatory)*

Description:	This query returns the root node for the tree of Message Container types. Using this node, information on a specific Message Container type can be obtained.
Call:	*Base URL*/ InteractionRegistryMessageComtainerTypeRoot
Parameters:	none
Return:	RegistryNodeID

3.6.5 Service Registry

The Service Registry contains the information needed to type eCo Services.

3.6.5.1 Published Interface

The Published Interface to a Service Registry defines a set of queries that can be requested of that Registry. The following standard query is defined:

`ServiceRegistryGetServiceTypeRoot` *(mandatory)*

Description:	This query returns the root node for the tree of Service types. Using this node, information on a specific Service type can be obtained.
Call:	*Base URL*/ ServiceRegistryGetServiceTypeRoot
Parameters:	none
Return:	RegistryNodeID

3.6.6 Document and Information Item Registry

The Document and Information Item Registry contains the type information for the documents defined within Interactions. This Registry contains two sets of type definitions - one for the documents themselves, and one for the information items that are used within the documents.

3.6.6.1 Published Interface

The Published Interface to a Document and Information Item Registry defines a set of queries that can be requested of that Registry. The following standard queries are defined:

`DocumentRegistryGetDocumentTypeRoot` *(mandatory)*

Description:	This query returns the root node for the tree of Document types. Using this node, information on a specific Document type can be obtained.
Call:	*Base URL*/ DocumentRegistryGetDocumentTypeRoot
Parameters:	none
Return:	RegistryNodeID

DocumentRegistryGetInformationItemTypeRoot *(mandatory)*

Description:	This query returns the root node for the tree of Information Item types. Using this node, information on a specific Information Item type can be obtained.
Call:	*Base URL* / DocumentRegistryGetInformationItemTypeRoot
Parameters:	none
Return:	RegistryNodeID

4. Compliance

Compliance requires that all published queries work correctly. However, all available queries do not need to be published.

Because the eCo Architecture encompasses a broad expanse of territory, it is unrealistic to expect that its complete implementation will be appropriate in all circumstances. In order to reach their interoperability goals, different business environments will demand greater and lesser degrees of compliance with this specification.

In an effort to accommodate these different requirements, this specification allows for a variety of compliance scenarios. Each compliance scenario provides a different degree of interoperability with other eCo compliant systems. Of course, the more of this specification that a company or institution chooses to implement, the greater the level of interoperability it will derive.

4.1 Rules for compliance

- All implementations of this specification must implement the eCo Business Layer. The implementation or use of the other Layers and Registries in the Architecture is optional. For more information, see The Business Layer (Section 3.5.3.)
- When implementing a Layer or Registry, all of the queries marked "mandatory" must be implemented. This includes the mandatory common queries defined in Sections 3.3 and 3.6.1.
- When implementing a query, all parameters marked "mandatory" must be implemented and the specified document must be returned.
- When returning a document from a query, all the elements and attributes marked "mandatory" must be returned in all successful cases.
- The eCo Document Wrapper must be used to enclose any documents returned as the result of a successful Query to an eCo Interface. For more information, see The eCo Document Wrapper (Section 3.2.)
- The document (EcoInterfaceDefinition), returned as the result of a QueryInterface query, must accurately reflect the queries implemented by its Interface. For more information about the QueryInterface query, see Common Queries (Section 3.3.)
- The use of the "eCo.xml" file to advertise the existence of one or more interfaces on a Web site is optional. For more information about the "eCo.xml" file, see Bootstrapping Interface Discovery (Section 3.1.3.)
- Any Interface or Document extensions must be made using the methods set forth in this document.
- All implementations of an eCo Layer must provide access to its Published Interface using the HTTP or HTTPS protocols. Other protocols may also be implemented and published as a separate Interface.

4.2 Examples of Compliance Scenarios

The following examples are intended to illustrate a few of the possible compliance scenarios that are possible:

4.2.1 A Minimal eCo Implementation

In order to be minimally compliant with this specification, a business would have to implement the Business Layer of the Architecture in accordance with the rules defined in Section 3.5.3. This would involve:

- Implementing the mandatory queries defined for the Business Layer of the Architecture (GetSupportedVersions, QueryInterface, and BusinessGetProperties.)
- Creating the documents returned by those queries (including the eCo Document Wrapper) and the mandatory elements and attributes they define.

While this may sound complicated, becoming minimally compliant is a very simple process. The Business Layer Interface can be implemented on a corporate Web site without doing any custom programming. Each of the mandatory interfaces defined in the Business Layer can be implemented as a simple HTTP document retrieval. The XML documents returned by the queries can be created using any text editor and posted to a Web site. The URLs for the various queries can then be mapped to the appropriate response documents.

The business has the option of creating an "eco.xml" file, which can be posted on its Web site to allow robots and search engines to index its Business Properties.

By becoming compliant with the Business Level of the Architecture, a business can make its identity and some basic information about itself known to prospective business partners in a structured and well-defined way. The Business can also choose to type itself using a common Registry of Business types. By doing so, it further empowers prospective trading partners to locate it effectively.

4.2.2 A Little More Complexity

By choosing to be compliant with the Service Layer of the Architecture, a business can expose information about the Services that it provides and the e-commerce protocols (if any) that are being used to implement them. In this compliance scenario, a business would implement both the Business and Services Layers in accordance with the rules defined in Section 3.5. This would involve:

- Implementing the Business layer of the Architecture as outlined in the previous section.
- Implementing the mandatory queries defined for the Service Layer of the Architecture. (GetSupportedVersions, QueryInterface, and ServiceGetProperties.)
- Create the documents returned by those queries (including the eCo Document Wrapper) and the mandatory elements and attributes they define.

By implementing a few more simple queries (once again, no coding is required) a business can allow its prospective trading partners to locate it based on its service offerings. Catalogues of service offerings can be built to help locate companies that offer certain types of services or services offered through certain protocols.

This level of compliance allows a business to describe the services it offers without detailing the interactions involved or the documents that are exchanged. Because of this, it is ideally suited to "wrapping" existing e-commerce initiatives without necessitating any extensive changes.

4.2.3 A Complete Compliance Scenario

By fully defining itself in eCo compliant terms, a business can describe its e-commerce system in sufficient detail that a prospective trading partner can definitively determine its interoperability requirements. Using the eCo Architecture, a Business can:

- Describe its self to prospective trading partners.
- Describe the Services that it offers.
- Describe the document exchanges that it uses to transact those Services.
- Situate its self in one or more e-commerce Markets that are in turn contained within one or more e-commerce Networks.
- Take part in shared business processes that span multiple enterprises.
- Incorporate Services provided by other Businesses into its own Service offerings.
- Extend its Properties and Interfaces to describe its unique value proposition in the context of its competitive environment.

In order to be compliant with all the levels in this specification, the queries and return documents at each level must be implemented in accordance with the rules in Section 4.1.

4.2.4 Defining an eCo Registry

Any business or organization can define a Registry to provide type definitions to a trading community. In order to create a registry, a business would be required to implement the Business Layer (to represent the creating business) and the Registry in question. Both of these objects must be implemented in accordance with the rules defined in Section 4.1.

Once again, the business has the option of creating an "eCo.xml" file, which can be posted on its Web site to allow robots and search engines to index the Registry and Business that created it.

5. Business Models (Business Scenarios to Demonstrate Architecture)

- This section is still in development.

6. Implementation issues

6.1 Error Conditions/handling

Because Queries to Published interfaces are made using existing Web protocols (primarily HTTP or HTTPS), errors generated as the result of an eCo Query are reported using the methods these protocols provide. While all of the status codes defined by these protocols can be used, the following are worthy of special mention:

200 - OK
This code indicates a successful query. If this code is returned, the response should be as defined in this specification. Implementers of this specification should take special care not to return this code unless the query was successful.

404 - Not Found
If this code is returned, the requested Query is not supported on this Published Interface.

500 - Internal Server Error
If this code is returned, the Query could not be executed.

If a protocol other than HTTP or HTTPS (for example, SMTP) is used to provide an eCo Interface, then the above errors must be supported in a manner that is suitable to the protocol being used.

6.2 Security

This specification mandates the use of established Web protocols in order to formulate its Queries. By doing so, it enables implementers to take advantage of any of the security systems that they define. These mechanisms include:

- basic-authentication
- SSL 2.0
- SSL 3.0
- Other security protocols that become accepted.

While use of these security mechanisms is optional, it is expected that implementers will them to restrict access to their Interfaces and encrypt their Query responses to suits their business needs.

- This section is still in development.

Appendix A: eCo Defined in XML DTDs

A.1 Commonly Used Content Models

These are some commonly used content models that appear throughout the following DTDs. They are explained here for facilitate understanding of the sections that follow.

- When PCDATA is the specified content model of an element, the same element may use an *e-dtype* attribute to indicate a more specific datatype for the element content. For example:

```
<!ELEMENT MyElement  (#PCDATA )>
<!ATTLIST MyElement  e-dtype NMTOKEN  #FIXED 'uri' >
```

  In this case, the datatype of *MyElement* is considered to be 'uri'

- When an attribute associated with an element has a declared type of CDATA or NMTOKEN, then an a-dtype attribute can be used to indicate a more specific datatype. For example:

```
<!ATTLIST MyElement
              xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
              xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
              a-dtype   NMTOKENS  'xmlns     uri
                                   xmlns:eCo uri' >
```

  In this case, the datatype of both *xmlns* and *xmlns:eCo* is 'uri'.

These specific conventions are not standard or even widely practiced, however they are based on a standardized technique known as *architectural forms*. While there is no expectation that an *architectural forms processor* is required, or even desirable, to support the eCo Framework, the desirability of such a datatype mapping mechanism is clear. The application of these conventions facilitates conversion of a DTD into an XML Schema Language through available tools.

A.2 Common Elements

The following elements are used throughout these DTDs. They are defined here for convenience.

```
<!ELEMENT Head  ((Identifier, Creator),
                 (Date, Version, TimeToLive),
                 (Description?, Label?) )>

<!ELEMENT Identifier  (#PCDATA )>
<!ATTLIST Identifier  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Creator (#PCDATA)>
<!ATTLIST Creator e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Date  (#PCDATA )>
<!ATTLIST Date  e-dtype NMTOKEN  #FIXED 'dateTime.tz' >
<!ELEMENT Version  (#PCDATA )>
<!ATTLIST Version  e-dtype NMTOKEN  #FIXED 'decimal' ><!ELEMENT TimeToLive
(#PCDATA )>
<!ATTLIST TimeToLive  e-dtype NMTOKEN  #FIXED 'dateTime.tz' >
<!ELEMENT Description ANY >
<!ELEMENT Label ANY >

<!ELEMENT Maker (#PCDATA)>
<!ATTLIST Maker e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Operator (#PCDATA)>
<!ATTLIST Operator e-dtype NMTOKEN  #FIXED 'uri' >

<!--  Name is a key that may contain string content -->
<!ELEMENT Name  (#PCDATA )*>
<!ATTLIST Name  type  (Legal | Trade | Alias | Former | Common | Other )  'Other'
>

<!ELEMENT Type ANY>
<!ATTLIST Type  reference NMTOKEN   #REQUIRED
                registry  CDATA     #REQUIRED
                a-dtype   NMTOKENS  'registry uri' >

<!ELEMENT Value ANY >

<!ELEMENT BooleanValue (#PCDATA) >
<!ATTLIST BooleanValue e-dtype NMTOKEN  #FIXED 'boolean' >

<!ELEMENT Specification (SpecificationIdentifier, SpecificationDescription?)>
<!ELEMENT SpecificationIdentifier  (#PCDATA )>
<!ATTLIST SpecificationIdentifier  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT SpecificationDescription ANY >

<!ELEMENT RegistrationService (#PCDATA )>
<!ATTLIST RegistrationService  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT TermsAndConditions (#PCDATA )>
<!ATTLIST TermsAndConditions  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT Credentials (Certification+)>
<!ELEMENT Certification (Description, Signature, IssuingAuthority,
CertificateType?, Expiry)>
<!ELEMENT Signature UNDEFINED >
<!ELEMENT IssuingAuthority (#PCDATA )>
```

```
<!ATTLIST IssuingAuthority e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT CertificateType (#PCDATA )>
<!ATTLIST CertificateType e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Expiry (#PCDATA )>
<!ATTLIST Expiry  e-dtype NMTOKEN  #FIXED 'dateTime.tz' >

<!ELEMENT NetworkRegistryLocation  (#PCDATA )>
<!ATTLIST NetworkRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT MarketRegistryLocation  (#PCDATA )>
<!ATTLIST MarketRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT BusinessRegistryLocation  (#PCDATA )>
<!ATTLIST BusinessRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT ServiceRegistryLocation  (#PCDATA )>
<!ATTLIST ServiceRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT InteractionRegistryLocation  (#PCDATA )>
<!ATTLIST InteractionRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT DocumentRegistryLocation  (#PCDATA )>
<!ATTLIST DocumentRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT DataElementRegistryLocation  (#PCDATA )>
<!ATTLIST DataElementRegistryLocation  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT NetworkSpecificProperties  (NetworkIdentifier, PropertyIdentifier)>
<!ELEMENT NetworkIdentifier  (#PCDATA )>
<!ATTLIST NetworkIdentifier  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT MarketSpecificProperties  (MarketIdentifier, PropertyIdentifier)>
<!ELEMENT MarketIdentifier  (#PCDATA )>
<!ATTLIST MarketIdentifier  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT BusinessSpecificProperties  (BusinessIdentifier, PropertyIdentifier)>
<!ELEMENT BusinessIdentifier  (#PCDATA )>
<!ATTLIST BusinessIdentifier  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT PropertyIdentifier  (#PCDATA )>
<!ATTLIST PropertyIdentifier  e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT RegistrySpecificProperties  (#PCDATA )>
<!ATTLIST RegistrySpecificProperties  e-dtype NMTOKEN  #FIXED 'uri' >

<!-- Process Description Language key -->
<!ELEMENT ProcessDefinitionLanguage (#PCDATA )*>
<!ATTLIST ProcessDefinitionLanguage e-dtype NMTOKEN  #FIXED 'uri' >

<!ELEMENT ProcessDefinition (#PCDATA )*>

<!ELEMENT Interface  (Head)>
<!ATTLIST Interface  type (Network | Market | Business | Service | Interaction
                           | Document | DataElement  | MarketRegistry |
BusinessRegistry
                           | ServiceRegistry | InteractionRegistry |
DocumentRegistry
                           | DataElementRegistry)  #REQUIRED >
```

A.3 eCo Interface Discovery with eCo.xml

The eCo.xml document should be structured in accordance with the EcoInterfaceList DTD.

A.4 Namespace Declaration and eCo Document Wrapper

The following DTD defines an eCo element that is used as a wrapper for any other eCo XML content. The eCo namespace is also defined here.

A.4.1 DocumentWrapper.dtd

```
<!ELEMENT Eco (Head)>
<!ATTLIST Eco xmlns     CDATA   FIXED "http://www.commerce.net/eco"
              xmlns:Eco CDATA   FIXED "http://www.commerce.net/eco" >
```

A.5 Commonly Returned Document Types

A.5.1 EcoVersionsList.dtd

```
<!ELEMENT EcoVersionsList  (Head, Version+)>
<!ATTLIST EcoVersionsList
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.5.2 EcoInterfaceList.dtd

```
<!ELEMENT EcoInterfaceList  (Head, Interface*)>
<!ATTLIST EcoInterfaceList
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.5.3 EcoInterfaceDefinition.dtd

```
<!ELEMENT EcoInterfaceDefinition  (Head, QueryDefinition+)>
<!ATTLIST EcoInterfaceDefinition  name NMTOKEN  #REQUIRED
                                  type NMTOKEN  #REQUIRED
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >

<!ELEMENT QueryDefinition  (Description, ParameterDefinition*)>
<!ATTLIST QueryDefinition  name NMTOKEN  #REQUIRED >
<!ELEMENT ParameterDefinition  (Description)>
<!ATTLIST ParameterDefinition  name     NMTOKEN   #REQUIRED
                               required CDATA     'true'
                               a-dtype  NMTOKENS  'required boolean' >
```

A.5.4 EcoTypeList.dtd

```
<!ELEMENT EcoTypeList  (Head, Type+)>
<!ATTLIST EcoTypeList
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.5.5 EcoBoolean.dtd

```
<!ELEMENT EcoBoolean  (Head, BooleanValue)>
<!ATTLIST EcoBoolean
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.6 The Network Layer

A.6.1 NetworkPropertySheet.dtd

```
<!ELEMENT NetworkPropertySheet  (Head, (Name+,  Maker+, Operator),
(MarketRegistryLocation?), (RegistrationService?, TermsAndConditions?,
Credentials?), NetworkSpecificProperties? )>
<!ATTLIST NetworkPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.7 The Market Layer

A.7.1 MarketPropertySheet.dtd

```
<!ELEMENT MarketPropertySheet   (Head, (Name+, Type*, Maker+, Operator),
                                (MarketRegistryLocation?,
BusinessRegistryLocation?,
                                 ServiceRegistryLocation?,
DocumentRegistryLocation?,
                                 DataElementRegistryLocation?),
                                (RegistrationService?, TermsAndConditions?,
Credentials?),
                                (ProcessDefinitionLanguage?,
ProcessDefinition*),NetworkSpecificProperties*,
                                MarketSpecificProperties* )>
<!ATTLIST MarketPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.8 The Business Layer

A.8.1 BusinessPropertySheet.dtd

```
<!ELEMENT BusinessPropertySheet   (Head, (Name+, Type*),
                                  (GeneralAddress, GeneralPhone, GeneralEmail),
                                  (TechnicalAddress?, TechnicalPhone?,
TechnicalEmail?),
                                  (BusinessRegistryLocation?,
ServiceRegistryLocation?,
                                  DocumentRegistryLocation?,
DataElementRegistryLocation?),
                                  (TermsAndConditions?, Credentials?),
                                  (ProcessDefinitionLanguage?,
ProcessDefinition*),
                                  NetworkSpecificProperties*,
                                  MarketSpecificProperties*,
BusinessSpecificProperties* )>
<!ATTLIST BusinessPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >

<!ELEMENT GeneralAddress  (#PCDATA )>
<!ELEMENT GeneralPhone  (#PCDATA )>
<!ELEMENT GeneralEmail  (#PCDATA )>

<!ELEMENT TechnicalAddress  (#PCDATA )>
<!ELEMENT TechnicalPhone  (#PCDATA )>
<!ELEMENT TechnicalEmail  (#PCDATA )>
```

A.9 The Service Layer

A.9.1 ServicePropertySheet.dtd

```
<!ELEMENT ServicePropertySheet   (Head, (Name+, Type?, Specification?),
                                 (InteractionRegistryLocation?,
DocumentRegistryLocation?,
                                 DataElementRegistryLocation?),
                                 (TermsAndConditions?, Credentials?),
                                 (ProcessDefinitionLanguage?),
                                 NetworkSpecificProperties*,
                                 MarketSpecificProperties*,
BusinessSpecificProperties*,
                                 ServicePreConditions?)>
<!ATTLIST ServicePropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               scope     (published | protected | obscured )  'published'
               a-dtype   NMTOKENS  'xmlns     uri
```

```
                                   xmlns:eCo uri' >

<!ELEMENT ServicePreConditions  (#PCDATA )>
```

A.10 The Interaction Layer

A.10.1 InteractionPropertySheet.dtd

```
<!ELEMENT InteractionPropertySheet  (Head, (Name+, Type?, Specification?),
                                    (Execution?,
InteractionMessageContainerType?),
                                    (InputDocument+, OutputDocument+,
ErrorDocument+),
                                    (DocumentRegistryLocation?,
DataElementRegistryLocation?),
                                    NetworkSpecificProperties*,
                                    MarketSpecificProperties*,
BusinessSpecificProperties* )>
<!ATTLIST InteractionPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               scope     (published | protected | obscured )  'published'
               a-dtype   NMTOKENS  'xmlns     uri
                                   xmlns:eCo uri' >

<!ELEMENT InteractionMessageContainerType  (#PCDATA )>
<!ATTLIST InteractionMessageContainerType  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT Execution  (#PCDATA )>
<!ATTLIST Execution  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT InputDocument  (#PCDATA )>
<!ATTLIST InputDocument  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT OutputDocument  (#PCDATA )>
<!ATTLIST OutputDocument  e-dtype NMTOKEN  #FIXED 'uri' >
<!ELEMENT ErrorDocument  (#PCDATA )>
<!ATTLIST ErrorDocument  e-dtype NMTOKEN  #FIXED 'uri' >
```

A.11 The Document Layer

A.11.1 Document Property Sheet.dtd

```
<!ELEMENT DocumentPropertySheet  (Head, (Name+, Type, Specification?),
                                 (DataElementRegistryLocation*),
                                 NetworkSpecificProperties*,
                                 MarketSpecificProperties*,
BusinessSpecificProperties* )>
<!ATTLIST DocumentPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                   xmlns:eCo uri' >
```

A.12 The Data Element Layer

A.12.1 DataElementPropertySheet.dtd

```
<!ELEMENT DataElementPropertySheet  (Head, (Name+, Type?, Specification?),
                                     NetworkSpecificProperties*,
                                     MarketSpecificProperties*,
BusinessSpecificProperties* )>
<!ATTLIST DataElementPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               scope     (published | protected | obscured )  'published'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.13 The Registry Environment

A.13.1 RegistryNodeList.dtd

```
<!ELEMENT RegistryNodeList  (Head, Node*)>
<!ATTLIST RegistryNodeList
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >

<!ELEMENT Node  (#PCDATA )>
<!ATTLIST Node  e-dtype NMTOKEN  #FIXED 'uri' >
```

A.13.2 RegistryNodeID.dtd

```
<!ELEMENT RegistryNodeID  (Head, Node )>
<!ATTLIST RegistryNodeID
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

A.13.3 RegistryPropertySheet.dtd

```
<!ELEMENT RegistryPropertySheet  (Head, (Name, Operator, Credentials?),
                                  RegistryDocumentTypeDefinitionLanguage,
                                  RegistrySpecificProperties? )>
<!ATTLIST RegistryPropertySheet
               type (Market | Business | Service | Interaction
               | Document | DataElement )  #REQUIRED
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >

<!ELEMENT RegistryDocumentTypeDefinitionLanguage  (#PCDATA )>
```

```
<!ATTLIST RegistryDocumentTypeDefinitionLanguage  e-dtype NMTOKEN  #FIXED 'uri' >
```

A.13.4 NodeTypeDefinition.dtd

```
<!ELEMENT NodeTypeDefinition  (Head, (Type*, Specification?),
RegistrySpecificProperties? )>
<!ATTLIST NodeTypeDefinition
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eco'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eco'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

Appendix B: eCo Defined in XML Schema

B.1 Commonly Used Content Models

B.2 Common Elements

The following elements are used throughout these DTDs. They are defined here for convenience.

```
<?xml version="1.0"?>
<!DOCTYPE schema PUBLIC "-//W3C//DTD XSDL 19990506//EN" "WD-xsdl.dtd">

<schema name="Common.xsd"
 xmlns="http://www.w3.org/1999/05/06-xmlschema-1/structures.xsd" model="closed">

  <elementType name="Maker">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="Operator">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <comment>Name is a key that may contain string content</comment>
  <elementType name="Name">
    <datatypeRef name="string"> </datatypeRef>
    <attrDecl name="type">
      <datatypeRef name="NMTOKEN">
        <enumeration>
          <literal>Legal</literal>
          <literal>Trade</literal>
          <literal>Alias</literal>
          <literal>Former</literal>
          <literal>Other</literal>
        </enumeration>
      </datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="Type">
    <any/>
    <attrDecl name="reference" required="true">
      <datatypeRef name="NMTOKEN"> </datatypeRef>
    </attrDecl>
    <attrDecl name="registry" required="true">
      <datatypeRef name="uri"> </datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="Value">
    <any/>
  </elementType>

  <elementType name="BooleanValue">
    <datatypeRef name="boolean"> </datatypeRef>
  </elementType>

  <elementType name="Specification">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="RegistrationService">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="TermsAndConditions">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="Credentials">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="NetworkRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="MarketRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="BusinessRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="ServiceRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>
  <elementType name="InteractionRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="DocumentRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="DataElementRegistryLocation">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="NetworkSpecificProperties">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="MarketSpecificProperties">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="BusinessSpecificProperties">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="RegistrySpecificProperties">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <comment> Process Description Language key </comment>
  <elementType name="ProcessDefinitionLanguage">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="ProcessDefinition">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

</schema>
```

B.3 eCo Interface Discovery with eCo.xml

B.3.1 eCo.dtd

```
  <elementType name="EcoInterfaces">
    <refines><archetypeRef name="eCo"/></refines>
    <elementTypeRef name="Interface" minOccur="1" maxOccur="*"/>
  </elementType>

  <elementType name="Interface">
    <sequence>
      <elementTypeRef name="Head"/>
    </sequence>
    <attrDecl name="type" required="true">
      <datatypeRef name="NMTOKEN">
        <enumeration>
          <literal>Network</literal>
          <literal>Market</literal>
          <literal>Business</literal>
          <literal>Service</literal>
          <literal>Interaction</literal>
          <literal>Document</literal>
          <literal>DataElement</literal>
          <literal>MarketRegistry</literal>
          <literal>BusinessRegistry</literal>
          <literal>ServiceRegistry</literal>
          <literal>InteractionRegistry</literal>
          <literal>DocumentRegistry</literal>
          <literal>DataElementRegistry</literal>
        </enumeration>
      </datatypeRef>
    </attrDecl>
  </elementType>
```

B.4 Namespace Declaration and eCo Document Wrapper

The following DTD defines an eCo element that is used as a wrapper for any other eCo XML content. The eCo namespace is also defined here.

B.4.1 DocumentWrapper.dtd

An archetype is an XML Schema feature that enables definition of non-terminals that may be reused in the definition of element types and even other archetypes. In the eCo document archetype, the Head element and CommerceNet eCo namespace are required, and the archetype is open. Simply, all eCo documents have at least this element and these attributes. As you will see in other examples,

```
  <archetype name="eCo" model="open">
    <elementTypeRef name="Head"/>
    <attrDecl name="xmlns" required="true">
      <datatypeRef name="uri">
        <default>http://www.commerce.net/eco</default> </datatypeRef>
    </attrDecl>
    <attrDecl name="xmlns:eCo" required="true">
      <datatypeRef name="uri">
        <default>http://www.commerce.net/eco</default> </datatypeRef>
    </attrDecl>
  </archetype>

  <elementType name="Head">
    <sequence>
      <sequence>
        <elementTypeRef name="Identifier"/>
        <elementTypeRef name="Creator"/>
      </sequence>
      <sequence>
        <elementTypeRef name="Date"/>
        <elementTypeRef name="Version"/>
        <elementTypeRef name="TimeToLive"/>
      </sequence>
      <sequence>
        <elementTypeRef name="Description"/>
        <elementTypeRef name="Label"/>
      </sequence>
    </sequence>
  </elementType>

  <elementType name="Identifier">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="Creator">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="Date">
    <datatypeRef name="dateTime"> </datatypeRef>
  </elementType>

  <elementType name="Version">
    <datatypeRef name="decimal"> </datatypeRef>
  </elementType>

  <elementType name="TimeToLive">
    <datatypeRef name="dateTime"> </datatypeRef>
  </elementType>

  <elementType name="Description">
    <any/>
  </elementType>

  <elementType name="Label">
    <any/>
  </elementType>
```

B.5 Commonly Returned Document Types

B.5.1 EcoVersionsList.dtd

```
  <elementType name="EcoVersionsList">
    <refines><archetypeRef name="eCo"/></refines>
    <elementTypeRef name="Version" minOccur="1" maxOccur="*"/>
  </elementType>
```

B.5.2 EcoInterfaceList.dtd

```
  <elementType name="EcoInterfaceList">
```

```
    <refines><archetypeRef name="eCo"/></refines>
    <elementTypeRef name="Interface" minOccur="1" maxOccur="*"/>
  </elementType>
```

B.5.3 EcoInterfaceDefinition.dtd

```
  <elementType name="EcoInterfaceDefinition">
    <refines><archetypeRef name="eCo"/></refines>
      <elementTypeRef name="QueryDefinition" minOccur="1" maxOccur="*"/>
    <attrDecl name="name" required="true">
      <datatypeRef name="NMTOKEN"> </datatypeRef>
    </attrDecl>
    <attrDecl name="type" required="true">
      <datatypeRef name="NMTOKEN"> </datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="QueryDefinition">
    <sequence>
      <elementTypeRef name="Description"/>
      <elementTypeRef name="ParameterDefinition" minOccur="0" maxOccur="*"/>
    </sequence>
    <attrDecl name="name" required="true">
      <datatypeRef name="NMTOKEN"> </datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="ParameterDefinition">
    <sequence>
      <elementTypeRef name="Description"/>
    </sequence>
    <attrDecl name="name" required="true">
      <datatypeRef name="NMTOKEN"> </datatypeRef>
    </attrDecl>
    <attrDecl name="required">
      <datatypeRef name="boolean">
        <default>true</default> </datatypeRef>
    </attrDecl>
  </elementType>
```

B.5.4 EcoTypeList.dtd

```
  <elementType name="EcoTypeList">
    <refines><archetypeRef name="eCo"/></refines>
      <elementTypeRef name="Type" minOccur="1" maxOccur="*"/>
  </elementType>
```

B.5.5 EcoBoolean.dtd

```
  <elementType name="EcoBoolean">
    <refines><archetypeRef name="eCo"/></refines>
    <elementTypeRef name="BooleanValue"/>
  </elementType>
```

B.6 The Network Layer

B.6.1 NetworkPropertySheet.dtd

```
  <elementType name="NetworkPropertySheet">
    <refines><archetypeRef name="eCo" schemaAbbrev="eCo"/></refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Maker" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Operator"/>
      </sequence>
      <sequence>
        <elementTypeRef name="MarketRegistryLocation" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="RegistrationService" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="TermsAndConditions" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="Credentials" minOccur="0" maxOccur="1"/>
      </sequence>
      <elementTypeRef name="NetworkSpecificProperties" minOccur="0"
maxOccur="1"/>
    </sequence>
  </elementType>
```

B.7 The Market Layer

B.7.1 MarketPropertySheet.dtd

```
  <elementType name="MarketPropertySheet">
    <refines><archetypeRef name="eCo" schemaAbbrev="eCo"/></refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type" minOccur="0" maxOccur="*"/>
        <elementTypeRef name="Maker" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Operator"/>
      </sequence>
      <sequence>
        <elementTypeRef name="MarketRegistryLocation" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="BusinessRegistryLocation" minOccur="0"
maxOccur="1"/>
        <elementTypeRef name="ServiceRegistryLocation" minOccur="0"
maxOccur="1"/>
        <elementTypeRef name="DocumentRegistryLocation" minOccur="0"
maxOccur="1"/>
        <elementTypeRef name="DataElementRegistryLocation" minOccur="0"
maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="RegistrationService" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="TermsAndConditions" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="Credentials" minOccur="0" maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="ProcessDefinitionLanguage" minOccur="0"
maxOccur="1"/>
        <elementTypeRef name="ProcessDefinition" minOccur="0" maxOccur="*"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0"
       maxOccur="1"/>
    </sequence>
  </elementType>
```

B.8 The Business Layer

B.8.1 BusinessPropertySheet.dtd

```
  <elementType name="BusinessPropertySheet">
    <refines>
      <archetypeRef name="eCo" schemaAbbrev="eCo"/>
    </refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type" minOccur="0" maxOccur="*"/>
      </sequence>
      <sequence>
        <elementTypeRef name="GeneralAddress"/>
        <elementTypeRef name="GeneralPhone"/>
        <elementTypeRef name="GeneralEmail"/>
      </sequence>
      <sequence>
        <elementTypeRef name="TechnicalAddress"
         minOccur="0" maxOccur="1"/>
        <elementTypeRef name="TechnicalPhone" minOccur="0"maxOccur="1"/>
        <elementTypeRef name="TechnicalEmail" minOccur="0"maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="BusinessRegistryLocation"
minOccur="0"maxOccur="1"/>
        <elementTypeRef name="ServiceRegistryLocation" minOccur="0"maxOccur="1"/>
        <elementTypeRef name="DocumentRegistryLocation"
minOccur="0"maxOccur="1"/>
        <elementTypeRef name="DataElementRegistryLocation"
         minOccur="0" maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="TermsAndConditions" minOccur="0"maxOccur="1"/>
        <elementTypeRef name="Credentials" minOccur="0"maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="ProcessDefinitionLanguage"
minOccur="0"maxOccur="1"/>
        <elementTypeRef name="ProcessDefinition" minOccur="0"maxOccur="*"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0"
       maxOccur="1"/>
      <elementTypeRef name="BusinessSpecificProperties" minOccur="0"
       maxOccur="1"/>
    </sequence>

  </elementType>

  <elementType name="GeneralAddress">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

  <elementType name="GeneralPhone">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

  <elementType name="GeneralEmail">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

  <elementType name="TechnicalAddress">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

  <elementType name="TechnicalPhone">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>

  <elementType name="TechnicalEmail">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>
```

B.9 The Service Layer

B.9.1 ServicePropertySheet.dtd

```
  <elementType name="ServicePropertySheet">
    <refines>
      <archetypeRef name="eCo" schemaAbbrev="eCo"/>
    </refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="Specification" minOccur="0"maxOccur="1"/>
      </sequence>
      <sequence>
```

```
        <elementTypeRef name="InteractionRegistryLocation"
         minOccur="0" maxOccur="1"/>
        <elementTypeRef name="DocumentRegistryLocation"
minOccur="0"maxOccur="1"/>
        <elementTypeRef name="DataElementRegistryLocation"
         minOccur="0" maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="TermsAndConditions" minOccur="0"maxOccur="1"/>
        <elementTypeRef name="Credentials" minOccur="0"maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="ProcessDefinitionLanguage"
minOccur="0"maxOccur="1"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0" maxOccur="1"/>
      <elementTypeRef name="BusinessSpecificProperties" minOccur="0"
maxOccur="1"/>
      <elementTypeRef name="ServicePreConditions" minOccur="0" maxOccur="1"/>
    </sequence>
    <attrDecl name="scope">
      <datatypeRef name="scope"></datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="ServicePreConditions">
    <datatypeRef name="string"> </datatypeRef>
  </elementType>
```

B.10 The Interaction Layer

B.10.1 InteractionPropertySheet.dtd

```
  <elementType name="InteractionPropertySheet">
    <refines>
      <archetypeRef name="eCo" schemaAbbrev="eCo"/>
    </refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="Specification" minOccur="0"maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="Execution" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="InteractionMessageContainerType"
         minOccur="0" maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="InputDocument" minOccur="1"maxOccur="*"/>
        <elementTypeRef name="OutputDocument" minOccur="1"maxOccur="*"/>
        <elementTypeRef name="ErrorDocument" minOccur="1"maxOccur="*"/>
      </sequence>
      <sequence>
        <elementTypeRef name="DocumentRegistryLocation"
minOccur="0"maxOccur="1"/>
        <elementTypeRef name="DataElementRegistryLocation" minOccur="0"
maxOccur="1"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0" maxOccur="1"/>
      <elementTypeRef name="BusinessSpecificProperties" minOccur="0"
maxOccur="1"/>
    </sequence>
    <attrDecl name="scope">
      <datatypeRef name="scope"></datatypeRef>
    </attrDecl>
  </elementType>

  <elementType name="InteractionMessageContainerType">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="Execution">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="InputDocument">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="OutputDocument">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="ErrorDocument">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>
```

B.11 The Document Layer

B.11.1 Document Property Sheet.dtd

```
  <elementType name="DocumentPropertySheet">
    <refines>
      <archetypeRef name="eCo" schemaAbbrev="eCo"/>
    </refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type"/>
        <elementTypeRef name="Specification" minOccur="0" maxOccur="1"/>
      </sequence>
      <sequence>
        <elementTypeRef name="DataElementRegistryLocation" minOccur="0"
```

```
maxOccur="*"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0" maxOccur="1"/>
      <elementTypeRef name="BusinessSpecificProperties" minOccur="0"
maxOccur="1"/>
    </sequence>
  </elementType>
```

B.12 The Data Element Layer

B.12.1 DataElementPropertySheet.dtd

```
  <elementType name="DataElementPropertySheet">
    <refines>
      <archetypeRef name="eCo" schemaAbbrev="eCo"/>
    </refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" minOccur="1" maxOccur="*"/>
        <elementTypeRef name="Type" minOccur="0" maxOccur="1"/>
        <elementTypeRef name="Specification" minOccur="0"maxOccur="1"/>
      </sequence>
      <elementTypeRef name="MarketSpecificProperties" minOccur="0" maxOccur="1"/>
      <elementTypeRef name="BusinessSpecificProperties" minOccur="0"
maxOccur="1"/>
    </sequence>
    <attrDecl name="scope">
      <datatypeRef name="scope"></datatypeRef>
    </attrDecl>
  </elementType>
```

B.13 The Registry Environment

B.13.1 RegistryNodeList.dtd

```
    <elementType name="RegistryNodeList">
     <refines><archetypeRef name="eCo"/></refines>
     <elementTypeRef name="Node" minOccur="1" maxOccur="*"/>
    </elementType>

    <elementType name="Node">
     <datatypeRef name="uri"> </datatypeRef>
    </elementType>
```

B.13.2 RegistryNodeID.dtd

```
  <elementType name="RegistryNodeID">
```

```
    <refines><archetypeRef name="eCo"/></refines>
    <elementTypeRef name="Node"/>
  </elementType>
```

B.13.3 RegistryPropertySheet.dtd

```
  <elementType name="RegistryPropertySheet">
    <refines><archetypeRef name="eCo"/></refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Name" schemaAbbrev="."/>
        <elementTypeRef name="Operator" schemaAbbrev="."/>
        <elementTypeRef name="Credentials" minOccur="0"
         maxOccur="1"/>
      </sequence>
      <elementTypeRef name="RegistryDocumentTypeDefinitionLanguage"/>
      <elementTypeRef name="RegistrySpecificProperties" minOccur="0"
       maxOccur="1"/>
    </sequence>
    <attrDecl name="type" required="true">
      <datatypeRef name="NMTOKEN">
        <enumeration>
          <literal>Market</literal>
          <literal>Business</literal>
          <literal>Service</literal>
          <literal>Interaction</literal>
          <literal>Document</literal>
          <literal>DataElement</literal>
        </enumeration>
      </datatypeRef>
    </attrDecl>
  </elementType>
```

B.13.4 NodeTypeDefinition.dtd

```
  <elementType name="RegistryDocumentTypeDefinitionLanguage">
    <datatypeRef name="uri"> </datatypeRef>
  </elementType>

  <elementType name="NodeTypeDefinition">
    <refines><archetypeRef name="eCo"/></refines>
    <sequence>
      <sequence>
        <elementTypeRef name="Type" minOccur="0" maxOccur="*"/>
        <elementTypeRef name="Specification" minOccur="0"
         maxOccur="1"/>
      </sequence>
      <elementTypeRef name="RegistrySpecificProperties" minOccur="0"
       maxOccur="1"/>
    </sequence>

</elementType
```

Appendix C: eCo Defined in XDR Schemas

C.1 Commonly Used Content Models

When the dt:type attribute is specified on any element, that element's datatype is given by that attribute's value.

C.2 Common Elements

The following elements are used throughout these DTDs. They are defined here for convenience.

```
<ElementType name="Head" content="eltOnly" order="seq">
    <group order="seq">
        <element type="Identifier"/>
        <element type="Creator"/>
    </group>
    <group order="seq">
        <element type="Date"/>
        <element type="Version"/>
        <element type="TimeToLive"/>
    </group>
    <group order="seq">
        <element type="Description" minOccurs="0" maxOccurs="1"/>
        <element type="Label" minOccurs="0" maxOccurs="1"/>
    </group>
</ElementType>

<ElementType name="Identifier" content="textOnly" dt:type="uri"/>
<ElementType name="Creator" content="textOnly" dt:type="uri"/>
<ElementType name="Date" content="textOnly" dt:type="dateTime.tz"/>
<ElementType name="Version" content="textOnly" dt:type="decimal"/>
<ElementType name="TimeToLive" content="textOnly" dt:type="dateTime.tz"/>
<ElementType name="Description" content="mixed" model="open"/>
<ElementType name="Label" content="mixed" model="open"/>
```

```
    <ElementType name="Maker" content="textOnly" dt:type="uri"/>
    <ElementType name="Operator" content="textOnly" dt:type="uri"/>

    <ElementType name="Name" content="textOnly">
        <AttributeType name="type" dt:type="enumeration"
                         dt:values="Legal Trade Alias Former Other"
default="Other"/>
        <attribute type="type"/>
    </ElementType>

    <ElementType name="Type" content="mixed" model="open">
        <AttributeType name="reference" dt:type="NMTOKEN" required="yes"/>
        <AttributeType name="registry" dt:type="uri" required="yes"/>
        <attribute type="reference"/>
        <attribute type="registry"/>
    </ElementType>

    <ElementType name="Value" content="mixed" model="open"/>
    <ElementType name="BooleanValue" content="textOnly" dt:type="boolean"/>

    <ElementType name="Specification" content="textOnly" dt:type="uri"/>
    <ElementType name="RegistrationService" content="textOnly" dt:type="uri"/>
    <ElementType name="TermsAndConditions" content="textOnly" dt:type="uri"/>
    <ElementType name="Credentials" content="textOnly" dt:type="uri"/>

    <ElementType name="NetworkRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="MarketRegistryLocation" content="textOnly" dt:type="uri"/>
    <ElementType name="BusinessRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="ServiceRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="InteractionRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="DocumentRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="DataElementRegistryLocation" content="textOnly"
dt:type="uri"/>
    <ElementType name="NetworkSpecificProperties" content="textOnly"
dt:type="uri"/>
    <ElementType name="MarketSpecificProperties" content="textOnly"
dt:type="uri"/>
    <ElementType name="BusinessSpecificProperties" content="textOnly"
dt:type="uri"/>
    <ElementType name="RegistrySpecificProperties" content="textOnly"
dt:type="uri"/>
    <ElementType name="ProcessDefinitionLanguage" content="textOnly"
dt:type="uri">/
    <ElementType name="ProcessDefinition" content="textOnly"/>
```

C.3 eCo Interface Discovery with eCo.xml

C.3.1 eCo.xdr

```
    <ElementType name="EcoInterfaces" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="Interface" minOccurs="1" maxOccurs="*"/>
    </ElementType>

    <ElementType name="Interface" content="eltOnly" order="seq">
        <AttributeType name="type" dt:type="enumeration"
                         dt:values="Network Market Business Service Interaction
                            Document DataElement MarketRegistry BusinessRegistry
                            ServiceRegistry InteractionRegistry DocumentRegistry
                            DataElementRegistry" required="yes"/>
        <attribute type="type"/>
        <element type="common:Head"/>
    </ElementType>
```

C.4 Namespace Declaration and eCo Document Wrapper

The following DTD defines an eCo element that is used as a wrapper for any other eCo XML content. The eCo namespace is also defined here.

C.4.1 DocumentWrapper.xdr

```
<Schema name="Eco.xdr"
     xmlns="urn:schemas-microsoft-com:xml-data"
     xmlns:dt="urn:schemas-microsoft-com:datatypes"
     xmlns:common="http://www.eco.commerce.net/Schemas/common.xdr">

     <AttributeType name="xmlns" dt:type="uri"
default="http://www.commerce.net/eCo"
                    required="yes"/>
     <AttributeType name="xmlns:eCo" dt:type="uri"
default="http://www.commerce.net/eCo"
                   required="yes"/>

    <ElementType name="Eco" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
    </ElementType>
```

C.5 Commonly Returned Document Types

C.5.1 EcoVersionsList.xdr

```
    <ElementType name="EcoVersionsList" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="common:Version" minOccurs="1" maxOccurs="*"/>
    </ElementType>
```

C.5.2 EcoInterfaceList.xdr

```
    <ElementType name="EcoInterfaceList" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="Interface" minOccurs="1" maxOccurs="*"/>
    </ElementType>
```

C.5.3 EcoInterfaceDefinition.xdr

```
    <ElementType name="EcoInterfaceDefinition" content="eltOnly" order="seq">
        <AttributeType name="name" dt:type="NMTOKEN" required="yes"/>
        <AttributeType name="type" dt:type="NMTOKEN" required="yes"/>
        <attribute type="name"/>
        <attribute type="type"/>
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="QueryDefinition" minOccurs="1" maxOccurs="*"/>
    </ElementType>

    <ElementType name="QueryDefinition" content="eltOnly" order="seq">
        <AttributeType name="name" dt:type="NMTOKEN" required="yes"/>
        <attribute type="name"/>
        <element type="common:Description"/>
        <element type="ParameterDefinition" minOccurs="0" maxOccurs="*"/>
    </ElementType>

    <ElementType name="ParameterDefinition" content="eltOnly" order="seq">
        <AttributeType name="name" dt:type="NMTOKEN" required="yes"/>
        <AttributeType name="required" dt:type="boolean" default="true"/>
        <attribute type="name"/>
        <attribute type="required"/>
        <element type="common:Description"/>
    </ElementType>
```

C.5.4 EcoTypeList.xdr

```
    <ElementType name="EcoTypeList" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="common:Type" minOccurs="1" maxOccurs="*"/>
    </ElementType>
```

C.5.5 EcoBoolean.xdr

```
    <ElementType name="EcoBoolean" content="eltOnly" order="seq">
        <attribute type="xmlns"/>
        <attribute type="xmlns:eCo"/>
        <element type="common:Head"/>
        <element type="common:BooleanValue"/>
    </ElementType>
```

C.6 The Network Layer

C.6.1 NetworkPropertySheet.xdr

```
<!ELEMENT NetworkPropertySheet  (Head, (Name+,  Maker+, Operator),
(MarketRegistryLocation?), (RegistrationService?, TermsAndConditions?,
Credentials?), NetworkSpecificProperties? )>
<!ATTLIST NetworkPropertySheet
               xmlns     CDATA     #FIXED 'http://www.commerce.net/eCo'
               xmlns:eCo CDATA     #FIXED 'http://www.commerce.net/eCo'
               a-dtype   NMTOKENS  'xmlns     uri
                                    xmlns:eCo uri' >
```

C.7 The Market Layer

C.7.1 MarketPropertySheet.xdr

```
    <ElementType name="MarketPropertySheet" content="eltOnly" order="seq">
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Name" minOccurs="1" maxOccurs="*"/>
          <element type="common:Type" minOccurs="0" maxOccurs="*"/>
          <element type="common:Maker" minOccurs="1" maxOccurs="*"/>
          <element type="common:Operator"/>
       </group>
       <group order="seq">
          <element type="common:MarketRegistryLocation" minOccurs="0"
                  maxOccurs="1"/>
          <element type="common:BusinessRegistryLocation" minOccurs="0"
                  maxOccurs="1"/>
          <element type="common:ServiceRegistryLocation" minOccurs="0"
                  maxOccurs="1"/>
          <element type="common:DocumentRegistryLocation" minOccurs="0"
                  maxOccurs="1"/>
          <element type="common:DataElementRegistryLocation" minOccurs="0"
                  maxOccurs="1"/>
       </group>
       <group order="seq">
          <element type="common:RegistrationService" minOccurs="0"
                  maxOccurs="1"/>
```

```
         <element type="common:TermsAndConditions" minOccurs="0" maxOccurs="1"/>
         <element type="common:Credentials" minOccurs="0" maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:ProcessDefinitionLanguage" minOccurs="0"
                 maxOccurs="1"/>
         <element type="common:ProcessDefinition" minOccurs="0" maxOccurs="*"/>
      </group>
      <element type="common:MarketSpecificProperties" minOccurs="0"
                 maxOccurs="1"/>
   </ElementType>
```

C.8 The Business Layer

C.8.1 BusinessPropertySheet.xdr

```
   <ElementType name="BusinessPropertySheet" content="eltOnly" order="seq">
      <attribute type="xmlns"/>
      <attribute type="xmlns:eCo"/>
      <element type="common:Head"/>
      <group order="seq">
         <element type="common:Name" minOccurs="1" maxOccurs="*"/>
         <element type="common:Type" minOccurs="0" maxOccurs="*"/>
      </group>
      <group order="seq">
         <element type="GeneralAddress"/>
         <element type="GeneralPhone"/>
         <element type="GeneralEmail"/>
      </group>
      <group order="seq">
         <element type="TechnicalAddress" minOccurs="0" maxOccurs="1"/>
         <element type="TechnicalPhone" minOccurs="0" maxOccurs="1"/>
         <element type="TechnicalEmail" minOccurs="0" maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:BusinessRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
         <element type="common:ServiceRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
         <element type="common:DocumentRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
         <element type="common:DataElementRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:TermsAndConditions" minOccurs="0" maxOccurs="1"/>
         <element type="common:Credentials" minOccurs="0" maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:ProcessDefinitionLanguage" minOccurs="0"
                 maxOccurs="1"/>
         <element type="common:ProcessDefinition" minOccurs="0" maxOccurs="*"/>
      </group>
      <element type="common:MarketSpecificProperties" minOccurs="0"
               maxOccurs="1"/>
      <element type="common:BusinessSpecificProperties" minOccurs="0"
               maxOccurs="1"/>
   </ElementType>

   <ElementType name="GeneralAddress" content="textOnly"/>
   <ElementType name="GeneralPhone" content="textOnly"/>
   <ElementType name="GeneralEmail" content="textOnly"/>
   <ElementType name="TechnicalAddress" content="textOnly"/>
   <ElementType name="TechnicalPhone" content="textOnly"/>
   <ElementType name="TechnicalEmail" content="textOnly"/>
```

C.9 The Service Layer

C.9.1 ServicePropertySheet.xdr

```
   <ElementType name="ServicePropertySheet" content="eltOnly" order="seq">
      <AttributeType name="scope" dt:type="enumeration" dt:values="published
               protected obscured" default="published"/>
      <attribute type="xmlns"/>
      <attribute type="xmlns:eCo"/>
      <attribute type="scope"/>
      <element type="common:Head"/>
      <group order="seq">
         <element type="common:Name" minOccurs="1" maxOccurs="*"/>
         <element type="common:Type" minOccurs="0" maxOccurs="1"/>
         <element type="common:Specification" minOccurs="0" maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:InteractionRegistryLocation" minOccurs="0"
               maxOccurs="1"/>
         <element type="common:DocumentRegistryLocation" minOccurs="0"
               maxOccurs="1"/>
         <element type="common:DataElementRegistryLocation" minOccurs="0"
               maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:TermsAndConditions" minOccurs="0" maxOccurs="1"/>
         <element type="common:Credentials" minOccurs="0" maxOccurs="1"/>
      </group>
      <group order="seq">
         <element type="common:ProcessDefinitionLanguage" minOccurs="0"
               maxOccurs="1"/>
      </group>
      <element type="common:MarketSpecificProperties" minOccurs="0"
               maxOccurs="1"/>
      <element type="common:BusinessSpecificProperties" minOccurs="0"
               maxOccurs="1"/>
      <element type="ServicePreConditions" minOccurs="0" maxOccurs="1"/>
```

```
    </ElementType>

    <ElementType name="ServicePreConditions" content="textOnly"/>
```

C.10 The Interaction Layer

C.10.1 InteractionPropertySheet.xdr

```
    <ElementType name="InteractionPropertySheet" content="eltOnly" order="seq">
       <AttributeType name="scope" dt:type="enumeration" dt:values="published
                 protected obscured" default="published"/>
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <attribute type="scope"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Name" minOccurs="1" maxOccurs="*"/>
          <element type="common:Type" minOccurs="0" maxOccurs="1"/>
          <element type="common:Specification" minOccurs="0" maxOccurs="1"/>
       </group>
       <group order="seq">
          <element type="Location" minOccurs="0" maxOccurs="1"/>
          <element type="Execution" minOccurs="0" maxOccurs="1"/>
          <element type="InteractionMessageContainerType" minOccurs="0"
                 maxOccurs="1"/>
       </group>
       <group order="seq">
          <element type="InputDocument" minOccurs="1" maxOccurs="*"/>
          <element type="OutputDocument" minOccurs="1" maxOccurs="*"/>
          <element type="ErrorDocument" minOccurs="1" maxOccurs="*"/>
       </group>
       <group order="seq">
          <element type="common:DocumentRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
          <element type="common:DataElementRegistryLocation" minOccurs="0"
                 maxOccurs="1"/>
       </group>
       <element type="common:MarketSpecificProperties" minOccurs="0"
                 maxOccurs="1"/>
       <element type="common:BusinessSpecificProperties" minOccurs="0"
                 maxOccurs="1"/>
    </ElementType>

    <ElementType name="Location" content="textOnly" dt:type="uri"/>
    <ElementType name="InteractionMessageContainerType" content="textOnly"
                 dt:type="uri"/>
    <ElementType name="Execution" content="textOnly" dt:type="uri"/>
    <ElementType name="InputDocument" content="textOnly" dt:type="uri"/>
    <ElementType name="OutputDocument" content="textOnly" dt:type="uri"/>
    <ElementType name="ErrorDocument" content="textOnly" dt:type="uri"/>
```

C.11 The Document Layer

C.11.1 Document Property Sheet.xdr

```
    <ElementType name="DocumentPropertySheet" content="eltOnly" order="seq">
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Name" minOccurs="1" maxOccurs="*"/>
          <element type="common:Type"/>
          <element type="common:Specification" minOccurs="0" maxOccurs="1"/>
       </group>
       <group order="seq">
          <element type="common:DataElementRegistryLocation" minOccurs="0"
                  maxOccurs="*"/>
       </group>
       <element type="common:MarketSpecificProperties" minOccurs="0"
                  maxOccurs="1"/>
       <element type="common:BusinessSpecificProperties" minOccurs="0"
                  maxOccurs="1"/>
    </ElementType>
```

C.12 The Data Element Layer

C.12.1 DataElementPropertySheet.xdr

```
    <ElementType name="DataElementPropertySheet" content="eltOnly" order="seq">
       <AttributeType name="scope" dt:type="enumeration" dt:values="published
protected obscured" default="published"/>
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <attribute type="scope"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Name" minOccurs="1" maxOccurs="*"/>
          <element type="common:Type" minOccurs="0" maxOccurs="1"/>
          <element type="common:Specification" minOccurs="0" maxOccurs="1"/>
       </group>
       <element type="common:MarketSpecificProperties" minOccurs="0"
                  maxOccurs="1"/>
       <element type="common:BusinessSpecificProperties" minOccurs="0"
                  maxOccurs="1"/>
    </ElementType>
```

C.13 The Registry Environment

C.13.1 RegistryNodeList.xdr

```
    <ElementType name="RegistryNodeList" content="eltOnly" order="seq">
       <description> A.13 The Registry Environment </description>
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <element type="Node" minOccurs="1" maxOccurs="*"/>
    </ElementType>

    <ElementType name="Node" content="textOnly" dt:type="uri"/>
```

C.13.2 RegistryNodeID.xdr

```
    <ElementType name="RegistryNodeID" content="eltOnly" order="seq">
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <element type="Node"/>
    </ElementType>
```

C.13.3 RegistryPropertySheet.xdr

```
    <ElementType name="RegistryPropertySheet" content="eltOnly" order="seq">
       <AttributeType name="type" dt:type="enumeration"
                      dt:values="Market Business Service Interaction Document
DataElement"
                      required="yes"/>
       <attribute type="type"/>
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Name"/>
          <element type="common:Operator"/>
          <element type="common:Credentials" minOccurs="0" maxOccurs="1"/>
       </group>
       <element type="RegistryDocumentTypeDefinitionLanguage"/>
       <element type="common:RegistrySpecificProperties" minOccurs="0"
maxOccurs="1"/>
    </ElementType>

    <ElementType name="RegistryDocumentTypeDefinitionLanguage" content="textOnly"
                 dt:type="uri"/>
```

C.13.4 NodeTypeDefinition.xdr

```
    <ElementType name="NodeTypeDefinition" content="eltOnly" order="seq">
       <attribute type="xmlns"/>
       <attribute type="xmlns:eCo"/>
       <element type="common:Head"/>
       <group order="seq">
          <element type="common:Type" minOccurs="0" maxOccurs="*"/>
```

```
            <element type="common:Specification" minOccurs="0" maxOccurs="1"/>
         </group>
         <element type="common:RegistrySpecificProperties" minOccurs="0"
                       maxOccurs="1"/>
      </ElementType>
```

Bibliography

[DATETIME]
Date and Time Formats, Misha Wolf and Charles Wicksteed. W3C, 15 September 1997.
See http://www.w3.org/TR/NOTE-datetime-970915

[ICE]
Information and Content Exchange (ICE) Protocol, Neil Webber, et al. W3C, 26 October 1998.
See http://www.w3.org/TR/NOTE-ice

[IOTP]
Internet Open Trading Protocol (IOTP) 1.0, David Burdett, et al. IETF, 28 February 1999.
See
http://www.ietf.cnri.reston.va.us/internet-drafts/draft-ietf-trade-iotp-v1.0-protocol-03.txt

[ISO-31]
ISO 31 -- Quantities and units. International Organization for Standardization

[ISO-639]
ISO 639:1988 -- Codes for the representation of names of languages. International Organization for Standardization

[ISO-3166]
ISO 3166:1993 Countries. International Organization for Standardization

[ISO-8601]
ISO 8601 -- Date and Time. International Organization for Standardization

[ISO-10646]
ISO 10646 -- Information Technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane, ISO/IEC 10646-1:1993. The current specification also takes into consideration the first five amendments to ISO/IEC 10646-1:1993.

[ISO-11404]
ISO 11404 -- Information Technology -- Programming Languages, their environments and system software interfaces -- Language-independent datatypes, ISO/IEC 11404:1996(E).

[OBI]
Open Buying on the Internet (OBI) 2.0, OBI Consortium. March 1999. See
http://www.openbuy.org/obi/specs/

[OFX]
Open Financial Exchange (OFX), CheckFree, Intuit, Microsoft. 1997. See
http://www.ofx.net/ofx/default.asp

[RFC-1808]
RFC 1808, Relative Uniform Resource Locators. Internet Engineering Task Force.
See http://ds.internic.net/rfc/rfc1808.txt

[RFC 2483]
RFC 2483, ... Internet Engineering Task Force. See http://ds.internic.net/rfc/rfc2483.txt

[RosettaNet]
See http://www.rosettanet.org/

[URI]
ID, Uniform Resource Identifiers (URI): Generic Syntax and Semantics See
http://www.ics.uci.edu/pub/ietf/uri/draft-fielding-uri-syntax-01.txt

[URL]
RFC 1738, Uniform Resource Locators (URL). Internet Engineering Task Force.
See http://ds.internic.net/rfc/rfc1738.txt

[URN]
RFC 2141, URN Syntax. Internet Engineering Task Force. See
http://ds.internic.net/rfc/rfc2141.txt

[XML]
Extensible Markup Language (XML) 1.0, Tim Bray, et al. W3C, 10 February 1998.
See http://www.w3.org/TR/REC-xml

Glossary

ANSI (American National Standards Institute)
: Officially designated organization for promoting national U.S. standards and specifications to ISO/IEC, JTC1, and Information Technology. Now called NCITS(pronounced "Insights").

Application
: One or more software programs cooperating to perform some intended task(s). This term usually refers to end-user programs. Architecture. See eCo Architecture.

Bootstrapping
: To start-up a system or initiate a course of action. This specification defines a document called Eco.xml that assists users in initiating transactions with an eCo compliant business. See also Eco.xml.

Business
: See eCo Business Layer

Business Model
: A model that provides views of a business's products, services, information flows, projected benefits, and sources of revenue. This model also identifies the business actors and their roles. See also, Market Model.

Business Participant
: An organization or person using or providing business services within a marketplace.

Business Process
: A choreographed set of human and/or computer activities intended to accomplish some business objective.

Business Registry
: A hierarchical database that provides a set of type definitions for the eCo Business Layer. The Business Registry contains the information needed to classify eCo Businesses. As such, the Business Registry is most often used when multiple businesses are grouped together in an eCo Market although stand-alone Businesses can still classify themselves using a common Business Registry.

Business Type
: The category that a Business is classified in according to the definitions in the Business Registry. See also Type.

Choreography
: The logic that governs the order of execution of activities such as, Services, Interactions, or events. The choreography is intended to produce a desired result or outcome, for example, the Choreography of Interactions used by an eCo Service may vary depending on the options selected by the user.

Client
: Software that requests and processes information from a Server, for example a Web browser such as Netscape Navigator or Internet Explorer.

Commerce Network Layer
: A set of Published Interfaces that can be used to find Markets and relate those Markets to a set of defined Market types.

CommerceNet
: Founded in 1994, a global non-profit organization with over 500 members, whose mission is to promote and advance interoperable electronic commerce to support emerging communities of commerce. (http://www.commerce.net/)

Compliance
: To act in accordance with this specification. In order to reach their interoperability goals, different business environments must implement this specification to varying degrees. The more of this

specification that a company or institution chooses to implement, the greater the level of interoperability it will derive.
All implementations of an eCo Layer must provide access to its Published Interface using the HTTP or HTTPS protocols. Other protocols may also be implemented and published as a separate Interface.

Content

Data (i.e. data, text, images, audio, video, etc) and/or software (e.g. CGI) on the web providing information and/or services to an end-user or another software application; content may be simple or complex, fine-grained or coarse-grained.

Data Element

See eCo Data Element Layer

Document

See eCo Document Layer

Document and Information Item Registry

A hierarchical database that provides a set of type definitions for the Documents defined within Interactions. This Registry contains two sets of type definitions - one for the Documents themselves, and one for the Data Elements that are used within the Documents. These type definitions are used to classify Documents and the Data Elements within those Documents.

Document Type

The category that a Document is classified in according to the definitions in the Document and Information Item Registry. See also Type.

Document Type Definition

A grammar referenced by an XML document that describes what tags and attributes are valid, and the valid structure or context in which the tags and attributes are valid. (see also XML and Schema):

Document Wrapper

See eCo Document Wrapper.

DTD

See Document Type Definition.

eCo

An abbreviation used by CommerceNet to denote electronic commerce.

eCo Architecture

An architecture specification developed by the eCo Working Group that describes a seven-layer model for building open and interoperable, electronic commerce business communities on the web. The eCo Architecture may be implemented in parts depending on business needs or software requirements.

eCo Architecture Layers

unctional groupings of services and data enabling interoperable communications and interactions between eCo. Each Layer in the eCo architecture exposes a query-able interface and static set of properties it uses to describe itself. By querying a Layer's "Published Interface" and by examining its property set, an interested party can determine enough information about the Layer to inter-operate with it.

eCo Business Layer

The eCo Business Layer encompasses that part of an e-commerce system that represents a single entity in a trading relationship, its formal identification and the services it offers or consumes. This must be implemented in order for a system to be considered eCo compliant.

eCo Data Element Layer

As the atomic markup elements within a Document, Data Elements serve to encapsulate data (or code) to be used in a particular Interaction. The Data elements, or Information Items will be concretely defined according to the XML specification for syntax and organized into the Business Semantics Library (XML) in order to have an approach to document architecture that promotes interoperability.

eCo Document Layer

A package of information having style, content, and structure.

eCo Document Wrapper

A simple set of meta-data about the document, its usage and intended to be the opening set of tags and closing tag of all documents defined in eCo terms. This information surrounds every XML document that is returned as a result of a query to a Published Interface.

eCo Interaction Layer

The atomic blocks from which Services are built. An Interaction consists of a request and a response. The request and the response can consist of zero or more Documents.

eCo Market Layer

A common portal or access point through which Businesses can group together.

eCo Service Layer

This Layer represents that part of an e-commerce system that is responsible for providing meta-data about business services and the exchange of commercial documents. Services are interfaces to a business process. Each Service offered by a Business provides the ability for a trading partner to interact with that Business in some way.

Eco.xml

An XML document that is located in the root directory of a business's Web site. This document informs visitors that the Web site is host to an eCo Published Interface.

e-commerce

See electronic commerce.

electronic commerce

Buying, selling, and conducting other business operations on the Web (also electronic business, procurement, trading, etc).

HTML - Hypertext Markup Language)

A specific set of markup tags (i.e. markup language) used to define web content (i.e. web documents) and enable web pages to link to other pages.

HTTP - Hyper-Text Transfer Protocol

An application-level protocol that web servers use to communicate with web browsers. An HTTP exchange consists of a request from the web browser and a response from a web server.

ICE - Information and Content Exchange Format and Protocol (formerly Internet Content Exchange) -

Submitted to the W3C, the ICE protocol defines the roles and responsibilities of (content) syndicates and subscribers, defines the format and method of content exchange, and provides support for management and control of syndication relationships. (http://www.w3.org/TR/NOTE-ice.html)

Information Item

See eCo Data Element Layer

Interaction

see eCo Interaction Layer

Interaction Registry

A hierarchical database that provides a set of type definitions for the eCo Interaction Layer. The Interaction Registry contains the information needed to classify eCo Interactions and Message Containers.

Interaction Type

The category that an Interaction is classified in according to the definitions in the Interaction Registry. See also Type.

Interface

See Published Interface.

Interoperability

The ability of separate systems to be linked together and then operate as if they were a single entity.

Layer

See eCo Architectural Layer.

Market

See eCo Market Layer.

Market Maker –

The Business that is responsible for the creation of a Market.

Market Model

A business model and marketing strategy of the involved business actors.

Market Operator

The business responsible for providing the day-to-day operation and administration of a Market.

Market Registry

A hierarchical database that provides a set of type definitions for the eCo Market Layer. The Market Registry contains the information needed to classify eCo Markets.

Market Type

The category that a Market is classified in according to the definitions in the Market Registry. See also Type.

Meta-data –

Data used to describe other data.

Methodology

A collection of methods (i.e. taxonomy) used to solve a problem; usually allows for incrementally solving a large and/or complex problem in a systematic manner.

Methods

Approaches or techniques for how to solve a problem.

NCITS (National Committee for Information Technology Standards - pronounced "Insights")

Develops national standards. This committee's technical experts participate on behalf of the United States in the international standards activities of ISO/IEC JTC 1, Information Technology. NCITS's mission is to produce market-driven, voluntary consensus standards in the areas of: multimedia (MPEG/JPEG), intercommunication among computing devices and information systems (including the Information Infrastructure, SCSI-2 interfaces, Geographic Information Systems), storage media (hard drives, removable cartridges), database including SQL3), security, and programming languages (such as C++).

Network

See Commerce Network Layer

NIST

National Institute of Standards and Technology.

OBI

Open Buying on the Internet.

Project

A planned and scheduled effort conducted by people or organizations to accomplish a stated goal or objective and produce tangible results (i.e. deliverables) within available resources (e.g. people, time, money, materials, etc).

Property

An identifiable attribute or characteristic describing an e-commerce system.

Property set

A set of meta-data about an e-commerce system.

Protocol

A defined convention for exchanging messages between two or more communicating parties.

Published Interface

A collection of Queries for interacting with eCo Architectural Layers and Registries.

Query

A method defined in an eCo Layer's Published...

RosettaNet

Founded in 1998, RosettaNet is an independent, self-funded, non-profit consortium dedicated to the development and deployment of standard electronic commerce interfaces to align the processes between IT supply chain partners on a global basis. RosettaNet and CommerceNet are both members of each other's organization. (http://www.rosettanet.org)

RSA

A public-key encryption technology invented by Rivest, Shamir, and Adelman of RSA Data Security, Inc. The RSA algorithm has become a de facto standard for strong encryption over the Internet. It is used by many software products, including Netscape Navigator, Microsoft Internet Explorer, Quicken, Lotus Notes, and other products. The U.S. government has restricted exporting RSA technology to foreign countries.

Schema

Schemas define the characteristics of classes of objects. Several XML-based schema languages (i.e. a vocabulary for defining schemas) for defining and documenting object classes have been submitted for consideration to the W3C Schema Working Group (see DTD, SOX, XDR, XML-Data and XML Schema). Schema languages may be used for classes which are strictly syntactic (for example, XML), or those which indicate concepts and relations among concepts (as used in relational databases, KR graphs and RDF). The former are called "syntactic schemas;" the latter "conceptual schemas.

Server

Software (and hardware) that can process and/or request information from a client (e.g. HTTP server, FTP server, mail server, print server, name server, etc).

Service

See eCo Service Layer.

Service Choreography

The logic that controls the order of execution of Interactions and Sub-Services within a Service.

Service Consumer

A consumer of a Business's Services.

Service Description

A human readable description of the Service provided.

Service Identifier

A URI that uniquely identifies a Service.

Service Provider

A business providing services.

Service Registry

A hierarchical database that provides a set of type definitions for the eCo Service Layer. The Service Registry contains the information needed to classify eCo Services.

Service Type

The category that a Service is classified in according to the definitions in the Service Registry. See also Type.

SET

Secure Electronic Transaction protocol: a means for authenticating credit card purchases on the Net using digital signatures. Transaction information is encrypted using 1024 bit RSA encryption.

SOX - Schema for Object-oriented XML:

Submitted to the W3C Schema Working Group; SOX provides an alternative to XML DTDs for modeling markup relationships to enable more efficient software development processes for distributed applications. SOX also provides basic intrinsic data types, an extensible data typing mechanism, content model and attribute interface inheritance, a powerful namespace mechanism, and embedded documentation. As compared to XML DTDs, SOX dramatically decreases the complexity of supporting interoperation among heterogenous applications by facilitating software mapping of XML data structures, expressing domain abstractions and common relationships directly and explicitly, enabling reuse at the document design and the application programming levels, and supporting the generation of common application components. (http://www.w3.org/Submission/1998/15/) (see also DCD, XDR, XML-Data and XML Schema.)

Sub-service

Subordinate Services used by a Service in providing some function.

Systems

Organized collections of interacting and connected components cooperating to accomplish a purpose.

Taxonomy

Related terms arranged as a hierarchy or graph according to some classification criteria.

Type

A classification designation used to identify an entity satisfying a given set of criteria or properties. A category that describes an e-commerce component. For example, a Market for automotive parts.

Type Definition

Specifies a category of e-commerce components and provides information about those components. Each type listing contained within a Type Registry is structured as a hierarchy, allowing type definitions to be refined through the addition of sub-types. For example, a Market for automotive parts might be further refined by breaking it down into Markets for engines, tires, and so on.

Type Registry

A hierarchical database that provides information on one or more specific sets of types. The Market, Business, Service, Interaction, and Document and Information Item Registries are all Type Registries.

URI - Universal Resource Identifier

Short strings that identify resources in the Web (e.g. documents, images, downloadable files, services, electronic mailboxes, and other resources). A URI can be further classified as a locator (e.g. URL), a name (e.g. URN), or both. The generic term for all types of names and addresses that refer to resources on the World Wide Web, for example, a URL is a type of URI.

URL - Universal Resource Locator

A type of URI that identifies the specific means of accessing (e.g. by location or network address) a document or other resource on the web.

URN - Universal Resource Name:

A unique name assigned to a web resource or object; URN refers to the subset of URI's that are required to remain globally unique and persistent even when the resource ceases to exist or becomes unavailable. A URN differs from a URL in that it's primary purpose is persistent labeling of a resource with an identifier (under development by IETF).

W3C

World Wide Web Consortium

Wrapper

See eCo Document Wrapper.

XDR - XML Data Reduced

One of several XML schema language specifications submitted to the W3C Schema Working Group for consideration as a W3C standard (see also DCD, SOX and XML Schema). Used in Microsoft's BizTalkTM Framework product.

XML - Extensible Markup Language:

A meta-language for describing document markup languages. A W3C specification published as an enhanced subset of SGML. It allows users to define and/or modify web-document markup tags (i.e. meta-data tags); share them with others; define the document structure (e.g. DTD); link-to and load numerous pages simultaneously; create bi-directional links; and create markup languages (i.e. tag sets and DTDs) for specific applications. XML was designed to deliver structured content over the web.

XML Namespace

XML namespaces provide a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by URI references. The primary use of such names in XML documents is to enable identification of logical structures in documents by software modules such as query processors, stylesheet-driven rendering engines, and schema-driven validators.

XML Vocabularies

XML-defined languages or "tag-sets" for describing document structure and content for a specific application. Examples include: Hypertext Markup Language (HTML), Mathematical Markup Language (MathML), Information and Content Exchange (ICE) Protocol, Channel Definition Format (CDF, used in the Microsoft IE4.0 browser), etc, etc.

XML-Data

Submitted to the W3C Schema Working Group; an XML vocabulary for describing schemas; that is, for defining and documenting object classes. It can be used for classes which as strictly syntactic (for example, XML) or those which indicate concepts and relations among concepts (as used in relational databases, KR graphs and RDF). The former are called "syntactic schemas;" the latter "conceptual schemas." (see also DCD, SOX, and XDR) (http://www.w3.org/TR/1998/NOTE-XML-data-0105/)

Errata

To: CommerceNet

From: eCo Framework Project Working Group

We, the Architecture Team Editors, are pleased to submit the final revision of the eCo Architecture Specification. We are delivering the specification as a series of linked HTML documents, including some graphics.

In addition to the specification, we have included in this letter a list of errata that we feel should be addressed in future work on the eCo Architecture. We also strongly suggest that this specification be made available in both it's current format (divided into parts), and as a single document that can be downloaded.

The Architecture Team Editors

Errata

Digital Signatures

All use of digital signatures will depend on the results of the W3C Signed XML Working Group. Once this Working Group has produced a specification, we recommend that the eCo Architecture Specification be modified to include the appropriate information. Particular areas that will need to be addressed include the Credentials and the Terms and Conditions elements that appear in some property sheets.

Open Marketplaces vs. Controlled Communities (section 3.0)

The eCo Architecture is useful for two purposes that have very distinct characteristics. One is to implement open, interoperabable commerce networks and marketplaces where businesses can find each other, establish relationships and interact together electronically over the Internet. The other is within a controlled community of commerce where the participants are all known prior to entry and the interoperability challenge is in making the interactions between business systems interoperable. Even though these two scenarios are quite different from a business perspective, they can each be supported by the eCo Architecture.

The major difference is in how the top three (or four) layers of the architecture are implemented and used. In both scenarios, the bottom layers are meant to achieve interoperability between the electronic interactions. On the one hand, the network, market, business and service information helps to facilitate business action in the discovery of trading partners and participation in networks and markets. The controlled community views the top layers of the architecture as a way to organize and manage information about who, where and how they are interacting with trading partners.

The Elements of the Head

The Head contains both a version and a date element. The version element is used elsewhere in the Specification to identify the version of a document. However, if we used the version element for this purpose in the Head, it would conflict with version elements that also appeared elsewhere in certain documents. Therefore, we decided to rely on the date element as a time stamp that could be used to identify the document version.

The version element was initially removed, but was later re-inserted as a way to identify which version of the eCo Specification the document complied with. Since this must always be determined, we have decided that the Head must remain unchanged between versions of the eCo Specification.

However, these decision were made late in the process, and a better solution may be found if the issues are revisited.

Market Layer Specification (section 3.5.2.2)

There is no way to use an alternate Market Specification at the Market Layer. This has been done for very specific reasons:

- The essence of the specification is about diverse systems interacting in a marketplace or electronic business community setting (open or closed) and for this to be true, the Market Layer must be specified in eco terms only.
- There are three main entry points to eco that make it unique and distinct from other architectures: the network layer, the market layer and the business layer which provide a business view of interoperability that is often left out of technical specifications.
- There are no well know internet marketplace or community standards, so in order for eco to satify the interoperability requirements, it was forced to develop it's own. There must be only one market specification for the reasons described, but eco would not have defined it if one existed that could be used.

The reason that other specifications can be used below the Business Layer (and within the Market Layer for things like the PSL) is that systems and standards are already in place at those levels making the challenge for interoperability one of communicating meta data about those systems and making them work together during interactions. The challenge at the Business, Market and Network Layers is communicating meta data that can facilitate business action.

There is a way to extend the Market Layer specification to incorporate specific qualities of a Market Layer implementation. For this reason, all of the layers in the architecture are extensible while still preserving a base level of interoperability.

eCo Architecture Compliance (section 4)

Compliance is discussed in section 4, but should be considered a work in progress. Other scenarios, such as partial compliance at different levels, should be developed and discussed in future work on the specification.

Business Models and Implementation Issues (sections 5 and 6)

These sections have been included as placeholders for future work we feel should be carried out.

Bibliography and Glossary

Both of these sections should be expanded as part of any future revisions.