electronic business XML (ebXML) Technical Architecture Specification

Draft v 0.6.5

This version:
http://www.ebxml.org/working/project_teams/technical_arch/

***Latest version*:**
http://www.ebxml.org/…

Previous version:
http://wwwebxml.org/…..

Team Leader:
Anders Grangard (EDIFrance) anders.grangard@edifrance.org

Editor:
Duane Nickull (XML Global) duane@xmlglobal.com

Abstract

The ebXML Technical Architecture Specification represents the work of the ebXML Technical Architecture Project Team. It defines the ebXML infrastructure and is based on the ebXML Requirements Specification document created by the ebXML Requirements Project Team

Status of this document

This document is in Draft status, It has not been approved by the Technical Architecture Project Team.

Comments on this Draft Specification are requested by May 26, 2000.

This document has been produced as part of the ebXML Technical Architecture effort. The goal of the Technical Architecture Team are discussed on the team's web page – http://www.ebxml.org/project_teams/architecture.htm

A list of current ebXML Technical Specifications and other technical documents can be found at http://www.ebxml.org/specindex.htm.

Public discussion on ebXML architecture takes place on the mailing list ebXML-architecture@oasis-open.org.

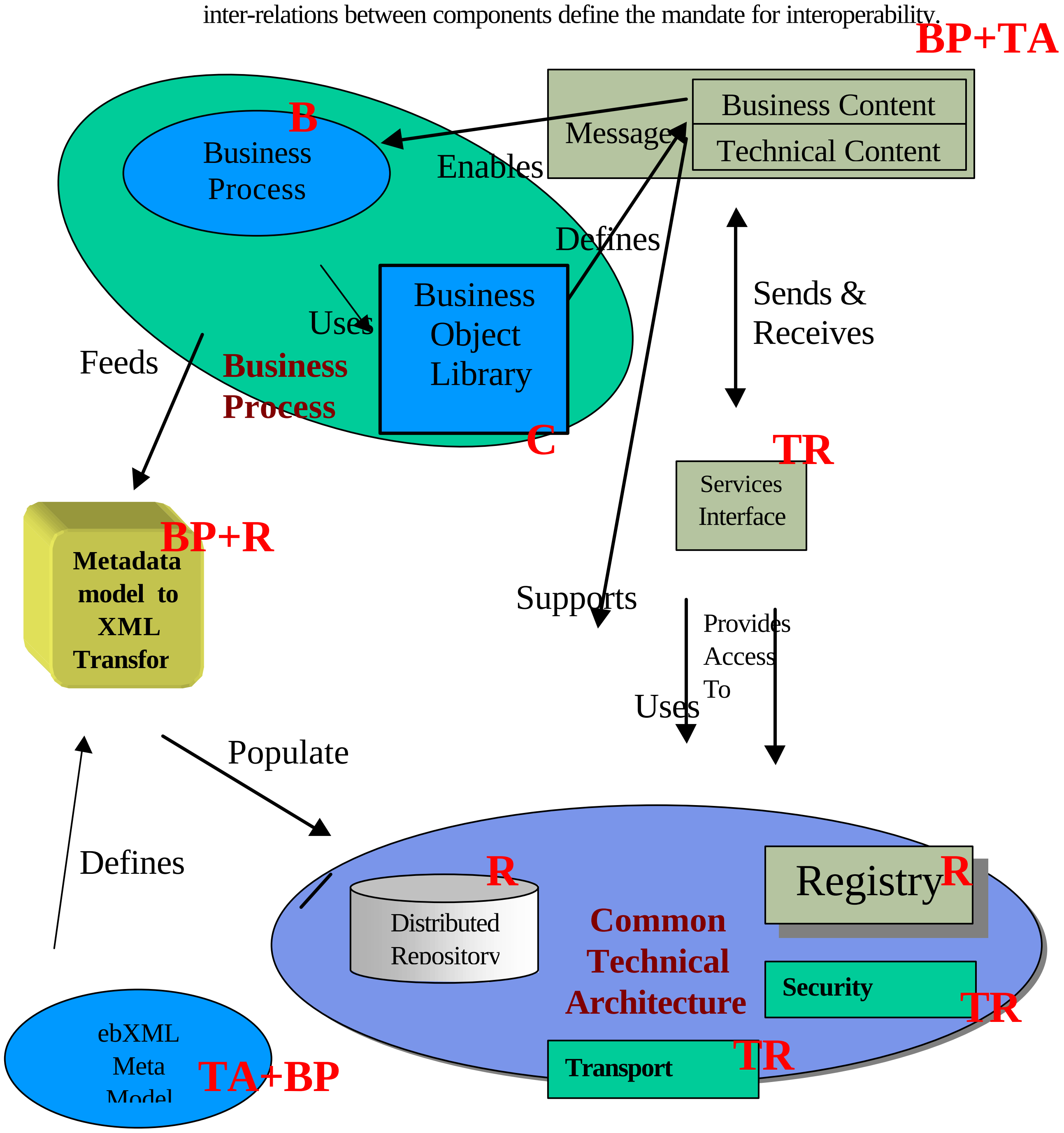
Please report errors in this document to the editor duane@xmlglobal.com or ebxml-architecture@oasis-open.org

TABLE of CONTENTS

1. Introduction

The ebXML (electronic business Extensible Markup Language) is a United Nations CEFACT/OASIS sponsored initiative. The ebXML has a mandate to create a single global electronic market. The ebXML mission is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties. XML is the Extensible Markup Language, a meta-language used to define vocabularies for marking up and providing structure to electronic data.

Figure 1.0 (below) Represents an abstraction of business and technological components that are within the scope of the ebXML Technical Architecture. The inter-relations between components define the mandate for interoperability.



Legend for Figure 1.0:

The red characters represent the ebXML project team primarily responsible for each area.

Projects Teams:

TA = Technical Architecture
BP = Business Process Methodology
TRP = Transport, Routing & Packaging
R2 = Registry and Repository
C2 = Core Components

The method for modelling objects and processes and the subsequent deriving of XML syntax in a consistent manner must be achieved by the component labelled "**Metadata model to XMLTransformation".** The method will be defined by the Business Process Methodology Project Team.

1.1 General ebXML Principles

The finished ebXML specification shall adhere to the original mandate and provide technical specifications that:

a. Enable simple, easy and ubiquitous use of XML for electronic business.

b. Provide a global, industry neutral open/interoperable infrastructure to facilitate business-to-business electronic interactions.

c. Meet the real needs of businesses in terms of legal, functional and equitable architecture.

d. Specify an easy on ramp and quick learning curve for companies in developing nations. Provide companies in developing nations an easy on ramp and quick learning curve.

e. Coalesce the structure and content components of divergent XML vocabularies.

f. Support timely interaction between trading partners using divergent electronic standards.

g. Avoid imposing unreasonable financial, technical and software requirements on those wishing to participate within the ebXML infrastructure, particularly smaller companies.

h. Facilitate multi-lingual (multi-byte character set) support.

i. Shall not preclude an easy transition path from existing EDI standards. This is to acknowledge the needs of trading partners who have substantial investments in existing systems.

j. Facilitate a scalable transition path which allows trading partners to gradually adopt loosely-coupled components from the ebXML infrastructure.

1.2 General Deliverables of ebXML

a. When ebXML is completed, the first specification shall be expressed as version 1.0.

b. This document is based on a set of requirements outlined in the ebXML Requirements Specification. That document is available at http://www.ebxml.org.

c. The ebXML 1.0 specification shall be expressed as a series of design rules and technical recommendations. It may not necessarily be in the form of one document. There are five major specification documents which shall collectively form the final specification:

    i) **Technical Architecture Specification** - contains an overview of the technical infrastructure that comprises ebXML and itemizes the design rules and guidelines.

    ii) **Repository and Registry Specification** - includes functional specification and technical design, interfaces, services.

    iii) **Transport, Routing and Packaging Specification** - addresses transport of ebXML messages, the means of security employed and the physical construction of the messaging used within the scope of the ebXML system.

    iv) **Business Process Modelling Specification** - the business process meta-model and the recommended methodology for using it.

    v) **Core Components Specification** - The set of ebXML core components, or the prescribed methodology for deriving them.

Appendices

vi) **Global ebXML definition dictionary** - A glossary of terms used within the ebXML infrastructure.

1.3 Scope Definition

a) All messages that are presenting themselves as ebXML messages, and claim to be compliant with the ebXML specification shall be in full compliance with the set of ebXML specifications.

b) A message that is conformant to the ebXML specifications but does not claim to be an ebXML compliant message is not considered within scope.

c) "Claiming to be ebXML compliant" needs further definition in the full set of ebXML specifications but is generally understood to be the by-product of using a set of ebXML specific elements and metadata (DTD's and possibly schemas in the future).

1.4 Special Considerations

a. The needs of SMEs (Small and Medium Enterprises) and participants using multi-byte character sets shall be addressed.

b. The ebXML should make use of the 20+ years experience gained from EDI particularly the ability to define real world business processes in electronic semantics.

c. The ebXML shall be an extensible architecture to meet future needs, not a rigid, set "standard" which limits the functionality of trading partners.

d. The ebXML shall make it easy and affordable to participate in the new global paradigm of e-business.

1.5 Goals of ebXML Technical Architecture

Definitions:

For this section only:

"Business Process" is a wide term used to describe any transaction between two trading partners.

a. Providing an architecture for integration of business processes among ad-hoc or established independent business partners by electronic means.

b. Reducing the need for collaborative business partners to have individual and expensive prior agreement on how to integrate business processes.

c. Providing a high-level business-centric view of distributed e-business processes.

d. Supporting and representing business processes independent of the technical solution.

e. Specifying and supporting a library of common, standard inter-business processes.

f. Allowing business processes and enabling technologies to evolve independently.

g. Integrating with new and legacy systems throughout the enterprise.

h. Leveraging existing technologies and standards

1.6 Conventions for Identifying ebXML Specifications Versions

a. Each ebXML technical specification shall contain a version identifier to uniquely identify it.

b. Standard conventions of major.minor as defined by ISO [Ed. Note - *INSERT STANDARD HERE*] shall be applied (e.g., version="1.17").

c. The first complete version of each technical specification shall be version 1.0.

1.7 Forwards Compatibility

a. Each subsequent new version of any ebXML specification shall be able to accommodate ebXML interactions which are performed consequently to a previous specification version.

b. New specification version interactions need to be recognized by previous specification versions architectural components. This means that an older architectural component shall be able to ascertain that the other component is of a newer version, but is not required to determine anything else.

c. The ebXML shall inherit the forwards and backwards compatibility built into the W3C's XML 1.0 specification.

2. ebXML System Architecture

This document contains a high level abstraction of the ebXML Technical Architecture. Section 2 specifically deals with the functional requirements for registries and repositories. The Registry and Repository Specification, available at http://www.ebxml.org/, contains much more detailed information and design rules for registries and repositories.

2.1 Definitions

The following definitions apply to Section 2 only:

Repository Object - an object, whether it be a business process or an actual data element, that is stored in a repository. Repository objects can be atomic or compound structures of atomic data elements.

Steward/Owner - a generic descriptor of a custodian who owns a repository object. A steward can be different from the submitting organization.

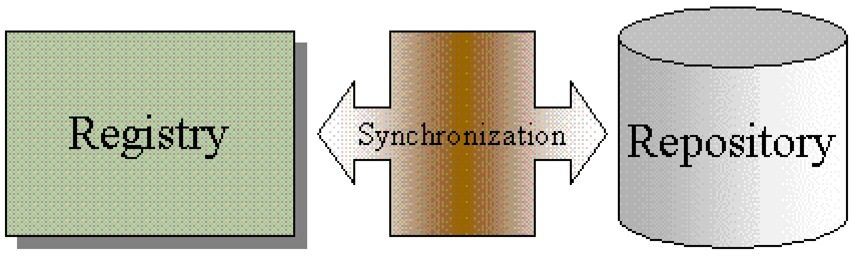
Administrator - (same as Repository Authority as defined by the OASIS Registry Repository group) the entity (person, organization, company) that operates a registry or repository. This designation is necessary to assign security responsibilities.

Data element - (not to be confused with an XML data element) the same as a repository object (see above). A data element is presumed to be atomic in structure unless specifically named as a "Compound Data Element."

Business Process - is a wide term used to describe any transaction between two trading partners.

2.2 Registries and Repositories

a. At the heart of the ebXML infrastructure is a powerful system of distributed registries and repositories.



b. The synchronization aspect refers to the symbiotic publish and subscribe relationship between the registry and the repository.

2.2.1 Registry Definition

a. A registry is a mechanism whereby relevant repository objects and metadata about them can be registered such that a pointer to their location, and all their metadata, can be retrieved as the result of a query.

b. A registry shall also be able to track and recognize stewards of repository objects and the repository objects themselves (see "2.6 Registry and Repository Security").

c. A registry shall incorporate a mechanism for querying a repository or a cache of a repository's index via an API (see "2.15 ebXML Business Process Search Requirements").

2.2.2 Repository Definition

a. A location or a set of distributed locations where repository objects pointed at by the registry reside and from which they can be retrieved by conventional means (e.g., http or ftp), perhaps with additional authentication/permission layers.

b. Repositories shall be distributed. The ebXML registry and repository architecture shall support distribution.

c. Examples of Repository Authorities can be top level organizations, verticals or SMEs.

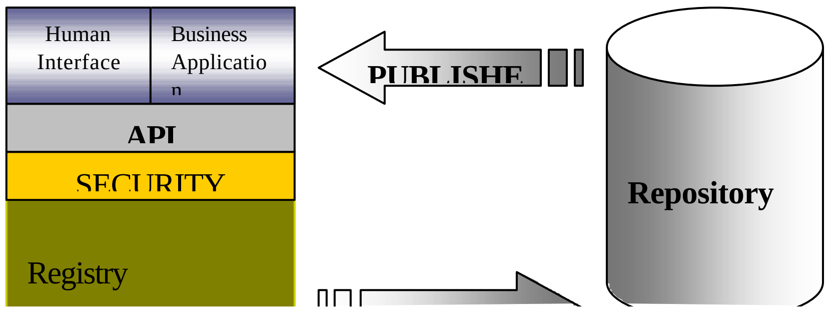
d. Registries and repositories shall both be capable of handling multi-byte character sets in the form of Unicode UTF-8 or UTF-16 encoded XML data.

2.2.3 Interactions

a. A registry shall have an interface to allow stewards to submit or update their repository objects.

b. A registry shall provide an interface for applications (i.e., API) to query its associated repository for repository objects.

Figure 2.1 - Interactions within a Registry and Repository System



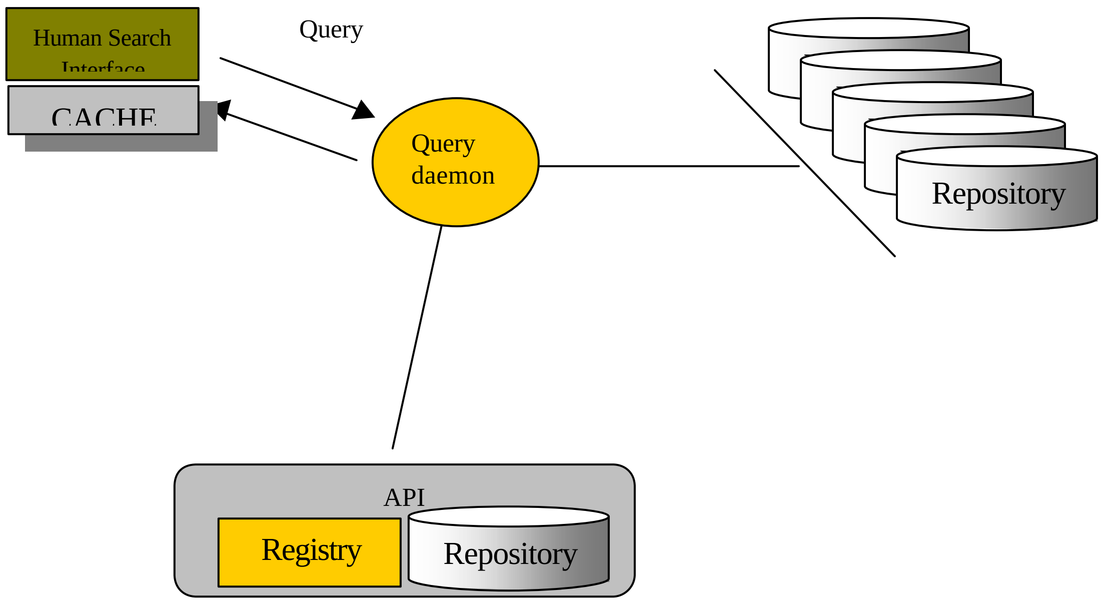
c. This API (Application Programming Interface) may also provide further functionality like allowing RPCs (Remote Procedure Calls) to modify repository objects (with appropriate permissions). An example could be an XML syntax message delivered to the API via the business application interface repository which results in a query being performed against the repository.

d. A registry shall provide an interface for humans (possibly CGI based) to manually query for repository objects.

e. A repository and registry shall synchronize their contents with one another in a symbiotic "publish/subscribe" relationship.

f. Rules 'd' and 'e' (above) shall specifically be able to scale across multiple repositories.

Figure 2.2 - How a Query mechanism shall scale (query) across multiple Repositories



[NOTES -
The incoming XML tells the XML search engine server to perform a “query” or “insert” into the repository/registry. A return string of XML is generated and delivered back to the client via HTTP.}

2.2.4 Registry and Repository Security

a. Appropriate security protocols shall be deployed to offer authentication, protection and non-repudiation functionality to a repository and registry. In instances where a repository or registry is deployed in a web accessible environment, a suitable security protocol shall be employed to ensure that system corruption or breaches are avoided. Specifically, the O/S shall be safeguarded against all reasonably foreseeable threats.

b. At the time an object is first created in a repository, or updated, the authentication shall be able to identify the Submitting Organization and ascertain whether or not that entity has the authority to perform the requested actions.

c. The registry and repository shall track the necessary information provided by the transport layer to archive information for purposes of non-repudiation.

d. The Repository Authority (RA) shall make known what security procedures are being followed.

e. The administrator of the registry or repository is responsible for ensuring the security protocol is fully operational and shall take all reasonable steps to ensure a fully secure service.

f. The Registry and Repository Specification shall clearly state the minimum requirements for any party interested in becoming a RA.

2.2.5 Registry/Repository Acceptance Policy

a. A submission to a registry/repository involves three stages of data element classification. The repository objects can be classified as:

   Development view:
   - Submissions - newly submitted items that need to be examined
   - Work in progress - a stage whereby the RA examines and suggests further refining of the repository objects.

   Run time view:
   - Standardized Objects - Repository objects which are now expressed in XML

syntax and available to parties other than the RA and the SA (submitting authority - also called the steward). All ebXML documents shall have a DTD associated with them or, at a later stage, a schema.

b. A system shall exist for a process of acceptance of artefacts within each new submission to a repository.

c. Repository objects which are classified as standardized objects shall be tested to ensure that it is in XML syntax.

2.2.6 Repository Objects

[NOTE - Add definitions and picture]

a. The structure of the common business object library shall work with the business process model. These common business objects are transformed into repository objects using transformation rules, thus (indirectly) defining the business content of messages.

b. A component library also facilitates the development of software components.

c. All repository objects have a steward associated with them. The steward is the custodian (owner) of the repository object--whether it be a data element or a business process.

2.2.7 Defining Processes

a. Authors shall utilize UML models, such as use case, class diagram and activity modelling, for business processes. A mechanism for deriving XML syntax from UML models in a consistent manner is a requirement.

b. The defining of business methods is the starting point. Once a business method or process in a repository is completed (i.e., standardized object, it needs to be expressed in XML syntax.

c. The process of identifying business processes shall be presented to SMEs (small to medium sized enterprises) in a user friendly and intuitive manner. The concerns of SME integration shall be addressed in the way in which they are assimilated into the ebXML infrastructure.

2.2.8 Defining Objects

a. Objects can generally be defined in two smaller sub-categories:

- Business Process: those which represent business transaction (submitting an invoice, sending a purchase order, etc.)

- Data Elements - Data elements are atomic elements such as those of a database that are data standard compliant. An example of a data element could be a first name or a postal code.

  Data Element Structures (Patterns. Also called Compound Data Elements (?) - Data structures composed of several smaller atomic data elements or even other compound Data elements.

b. Submitting Organizations shall have the ability to suggest the decomposition of a data element into two or more atomic data elements.

c. At the stage a new submission is received, it shall be examined by the Repository Authority who shall have a mandate to avoid duplication of atomic data elements.

d. The Core Components Project Team shall draft guidelines for aiding submitting organizations to define what constitutes an atomic data element.

[NOTE - Item c. represents a potentially costly and risky process]

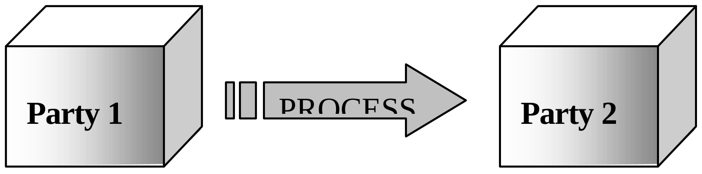
e. Data elements are atomic in their structure by definition. Data element definitions shall be such that no unilateral interpretations could result in a data element being sub-defined into smaller data elements.

f. Compound data elements are data elements that are comprised of two or more atomic or other compound data elements.

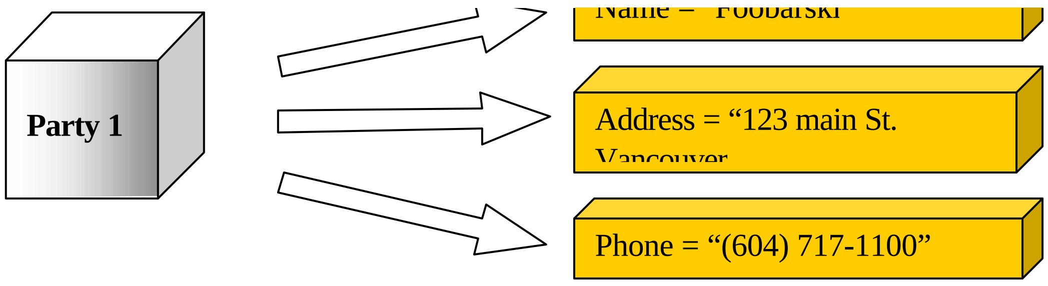
2.3 Modelling Data Elements from Business Objects

a. The registry and repository model for storing data elements predicates a necessity for extracting those data elements from UML models.

Figure 2.2 an example of a simple transaction may have two



b. This over-simplified model shows a basic transaction template.



c. Each data element when presented as a repository object in a business process shall be identified and classified until it is atomic.

d. An object shall contain at least one byte of information (8 bits).

Figure 2.2.1 In the above example, Party 1 has attributes which can be extracted and labelled as data elements (repository objects)

e. Party 1 has a NAME, ADDRESS and PHONE NUMBER
NAME = Foobarski Programming, Inc.
ADDRESS = 123 Main Street, Vancouver, BC Canada, V3J-3E1
PHONE NUMBER = (604) 717-1100

f. The first data element cannot be broken down into smaller, simpler objects however, the second and third can

g. The second and third are referred to as a "compound data elements," data elements that are comprised of two or more data elements (repository objects).

h. The ADDRESS data element can be broken down as follows:

STREET ADDRESS = 123 main Street
CITY = Vancouver
PROVINCE = BC
ZIP = V3J-3E1

The PHONE can be broken down into

PREFIX = (604)
PHONE = 717-1100

Further decomposition of the telephone number is possible but was not explored in this example.

i. To further provide a syntax neutral approach to identify data elements, once a data element has been identified, it can receive a unique identification.

j. Imagine in the following example, we take the data elements:

Identifier	**Description**	**Value**
PARTYNAME	NAME	Foobarski Programming, Inc.
MEE002	STREET ADDRESS	123Main Street
VILLE	CITY	Vancouver
THDX3	Province	BC
ZIP	ZIP	V3J-3E1
AREACODE	PREFIX	(604)
FONE	PHONE	717-1100

k. Each data element in a repository shall have a unique code. Additional fields in the repository shall contain human readable information about the data element (repository object).

IDENTIFIER	**DESCRIPTION**	**EXAMPLE**	**DATA**
PARTYNAME	COMPANY NAME	FooBar, Inc.	CDATA

l. The guidelines for syntax and semantic composition of the Unique Key for each Data Element shall be the responsibility of the Registry and Repository Project Team. Each Unique Identifier shall have the following functionality:

1. It must be used for one and only one repository data element
2. It is to be used by the Query mechanism to locate the repository data element.

m. A similar description file will be used to build compound data elements:

IDENTIFIER	**DESCRIPTION**	**EXAMPLE**	**DATA**
MY-ADDRESS S_IDENTIFIER	A complete address made of compound data elements	123 Main St. Vancouver, BC Canada, V3J3E1	MEE002, VILLE, THDX3, etc.

2.3.1 Addressing Registries and Repositories

a. A repository shall be addressable as a URL in order to allow HTTP as the default transportation protocol.

***Figure 2.2.2** - Example of a Repository URL*

http://www.foorepository.org/

b. Each data element shall be uniquely addressable via the Internet by using a combination of XPointer/Xlink and a URL.

Figure 2.2.3 - *Example of a Specific path to a Data Element*

http://www.foorepository.org/cgiToGetElements.pm?DataElement=THE_UNIQUE CODE

c. Assuming that the cgi_to_get_elements.pm file is an executable script which takes the two URL encoded arguments:

Data_element=THE_UNIQUE_CODE

and returns a value, we now can address the unique object from anywhere on the Internet.

2.4 Using the Data Elements in DTDs

a. Data element repository references may be placed into a DTD as a combination of Xpointer and a URL. The reference is there to provide either a machine or human with semantic information about what the character data in the corresponding XML files represents

XML FILE:

```
<?xml version=”1.0”?>
<doctype SYSTEM CompanyName.dtd>
<Companyname>
    <name>Anders’ Swedish Pleasure Palace</name>
</Companyname>
```

DTD FILE:

```
<?xml version=”1.0”?>
<!DOCTYPE CompanyName [
<!ELEMENT CompanyName(name)>
<!ELEMENT name (#CDATA)>
<!ATTLIST name type CDATA #FIXED
‘http://www.foorepository.com/CgiToGetElement.cgi?
DataElement=ME00001’
```

b. The preceding example allows the element to be associated with a unique data element in a repository.

c. The method for EbXML XML document instance references to specific data elements within a repository must be able to be accomplished by using either DTD's or Schemas (when schemas become eligible for inclusion within the EbXML Architecture). The exact syntax for using a schema to accomplish the functionality of the DTD file example (above) must adhere to the same functionality.

d. For the purposes of forwards compatibility with Schemas as described in (c) above, the term "DTD" shall be interchangeable with Schema for the remainder of this section.

e. A party who receives such an XML file, with a repository lookup definition in the accompanying DTD, can perform the repository lookup on the data element to find out what it is.

f. A Submitting Organization may use the process defined by the RA to have a namespace-unique data element placed into a repository.

[Known issue - Are the unique data elements to be defined through a Namespace?]

2.5 Retrieving Objects and Processes

a. Query and retrieval mechanisms shall be employed to access objects in a repository/registry.

~~b.~~ An ebXML search mechanism should provide a web search interface for humans to manually locate data (particularly important for SMEs) and an API for applications to rapidly access stored data.

c. A caching mechanism is recommended to increase the efficiency of a query mechanism.

2.6 ebXML Business Process Search Requirements

a. In instances where owners of XML documents wish to make the contents of those documents available to the public and potential business partners, a search mechanism shall be employed to provide a searchable index for discovery purposes. This section refers explicitly to XML documents that are not part of a repository.

b. The XML syntax data will be addressable at a set, standard relative location to a company's URL (domain). A likely syntax for addressing those documents will be:
   http://www.theCompanyDomain.com/ebXML/*

[Known issue - We will examine the robots exclusion standard to determine if we may use the robots.txt file to consistently reference this information. Duane]

c. The XML documents referenced at that location can contain information about:

   i) The nature and specific identification credentials for that business.

   ii) The appropriate URI and access protocols for the company.

   iii) The nature of the business processes that the company allows web interactions for.

   iv) Other information that a participating company may deem relevant to broadcast to potential business partners.

3. Transportation, Routing and Packaging

This section defines the architecture for the ebXML technical architecture. Section 3 specifically deals with the function requirements for transport, routing and packaging components of the ebXML infrastructure. The Transport, Routing and Packaging Specifications are available at http://www.ebxml.org/.

3.1 Section Definitions

The following definitions apply to Section 3 only:

[Known issues - We'll synchronize these defs with the ebXML common glossary when available.}

Message – any data that is being transported within the ebXML infrastructure.

Message Scope Definition – any message that claims to be an ebXML compliant message is considered to be within the scope of the ebXML infrastructure and subject to the technical restrictions thereof.

Participating Device – a generic description of any endpoint for an ebXML message or any device which claims to be able to handle the functionality of the ebXML message handling constraints.

Transaction – a single or sequence of messages that are sent and received by participating devices.

3.2 Transportation Protocol Requirements Architecture

a. A default, fallback transportation protocol shall be supported by all participating devices. That fallback protocol will be HTTP (Hyper Text Transport Protocol).

b. The fallback transportation protocol may be superseded by another transportation protocol in instances where:

    i) All participating devices in a specific transaction support the higher transportation protocol.

    ii) It is determined that there is no possible future outcome to the transaction which would result in another device being required to participate and that other device may not be able to support the transportation protocol superset.

    iii) A mechanism shall be developed whereby participating devices can, through a process of discovery, find out which additional protocols they both support and whether or not any further interactions are required by another participating device which may need to support that supplemental protocol.

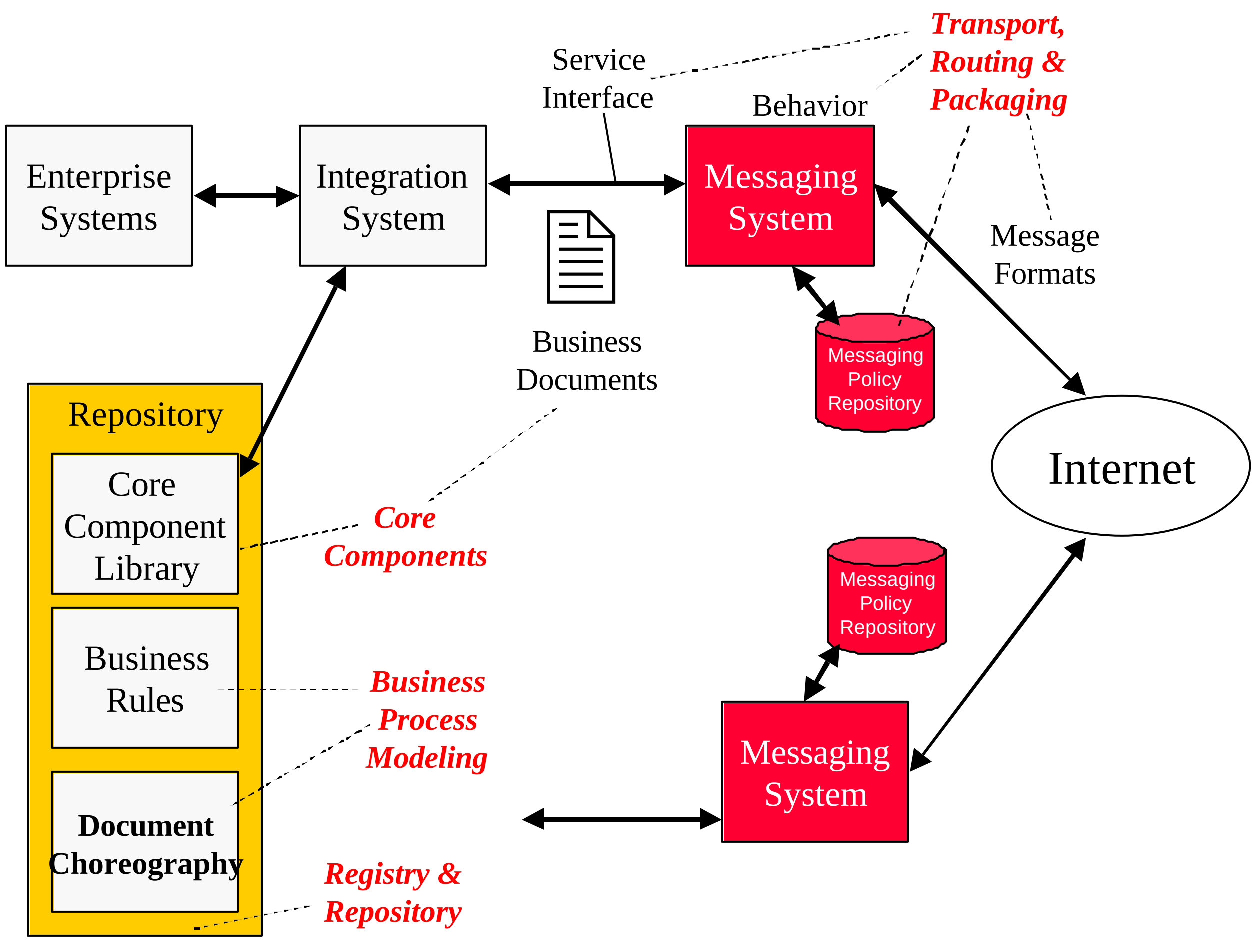
c. The transportation protocol shall be operating system independent.

d. The transportation protocol shall be independent of any security protocol.

e. Security protocols shall be present to meet the needs of the ebXML infrastructure.

Figure *.* - shows an overview of the transport, routing and packaging.

[NOTE - Update this picture]



3.3 Routing of ebXML Messages

a. The ebXML messages shall include a header component which shall contain routing and archiving information. The header shall be enclosed with an XML header <element> and is not to be confused with an HTTP header.

b. The specific names for the elements in the header shall be specified by the Transport, Routing and Packaging Group.

c. The routing information shall adequately and uniquely identify all of the parties that participate in the particular message set exchange to a transaction.

d. The routing header information shall contain versioning to enable an audit trail of each message.

3.4 ebXML Message Components

a. A message consists of an outer **transport envelope**, such as HTTP or SMTP, that wraps and a transport independent **message envelope**, which consists of a **header** with one or more header parts inside and a **body** that is the real payload of the message.

b. The ebXML messages business transaction component shall be expressed in syntax that conforms to XML (current version is 1.0).

c. The payload of the message business information shall be neutral but may need to use entity references to comply with the XML syntax requirements pursuant to the XML 1.0 specification.

d. Each message shall contain three distinctive components: the technical data, the business data and the metadata components.

3.5 Technical Data

a. The first part of the message shall be the processing instructions expressed in the document type declaration: A mechanism shall be employed to allow the payload section of the message to be handed off to the appropriate application without having to incur the overhead of parsing the complete message.

b. A multi-byte character set declaration shall be used to help facilitate diverse language characters (multi-lingual).

c. The declaration shall be in the form required by the current version of XML to specify Unicode UTF-8 or UTF-16 encoding.

Eg. <?xml version=”1.0” encoding=”UTF-8”?>

d. The next component is the routing information (a technical component). Each message shall have enough information in this section to uniquely identify each party and be able to address each party on the Internet.

3.6 Metadata

a. The message shall use a root element tag of to identify it as an ebXML compliant transaction message.
    [EDITORS NOTE - THE TRP GROUP SHALL EITHER ADOPT THIS OR SPECIFY AN ALTERNATIVE IDENTIFICATION ELEMENT OR METHOD]

b. The next components of the metadata identify the business process, the version of that process, the owner of the process and where another application can retrieve a mapping template to apply to the business data component. Any further information that may be required to facilitate the business requirements of that message.

c. The header component shall allow for extensibility to meet future requirements.

d. A method for guaranteeing non-repudiation of a message shall be built into the Transport, Routing and Packaging Specification.

e. A method shall be determined to ensure that each message is delivered or, in the alternative, an appropriate error message is generated. The method shall also allow that each message is delivered once and once only (i.e., no duplicate deliveries of the same message) between applications (communication end points).

f. Each message and each message set shall have the ability to be uniquely identified to meet the business needs of providing archiving for an audit trail.

4. Participation Requirements

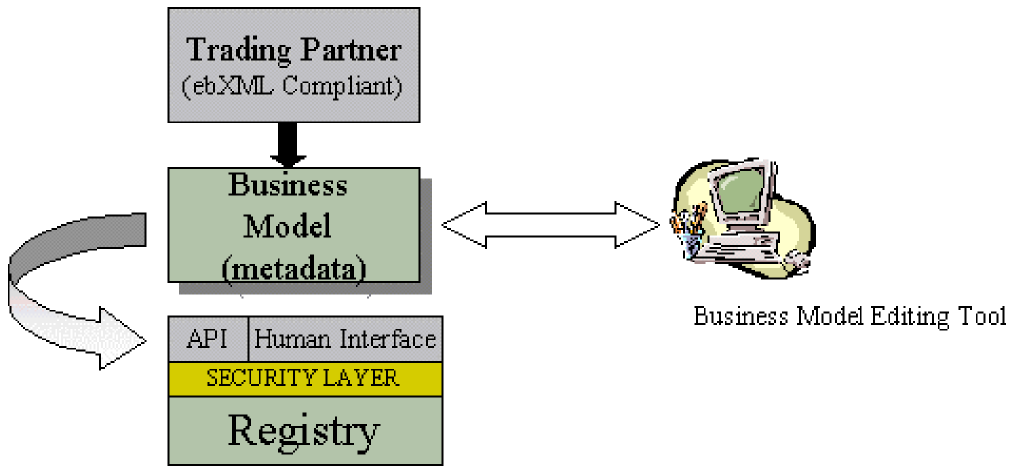


Figure 4.1 How Trading Partners Create metadata

NOTE - Change diagram : Human Interface (from above) Create HI box above API Please add a box to show repository, arrow from reg to rep.

a. A trading partner creates a model of its business processes.

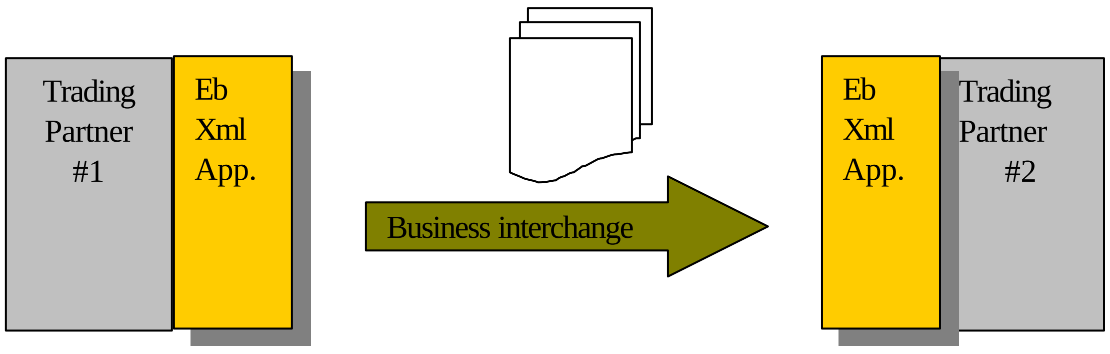
b. The model is registered with a registry or another mechanism which allows other businesses to query it for the information. The owner is authenticated using a set protocol.

c. The registry then submits the data to a repository. The repository may be public or privately controlled. Trading partners can even utilize their own repositories.

d. The repository then makes the information available via the registry. The registry and repository shall be synchronized.

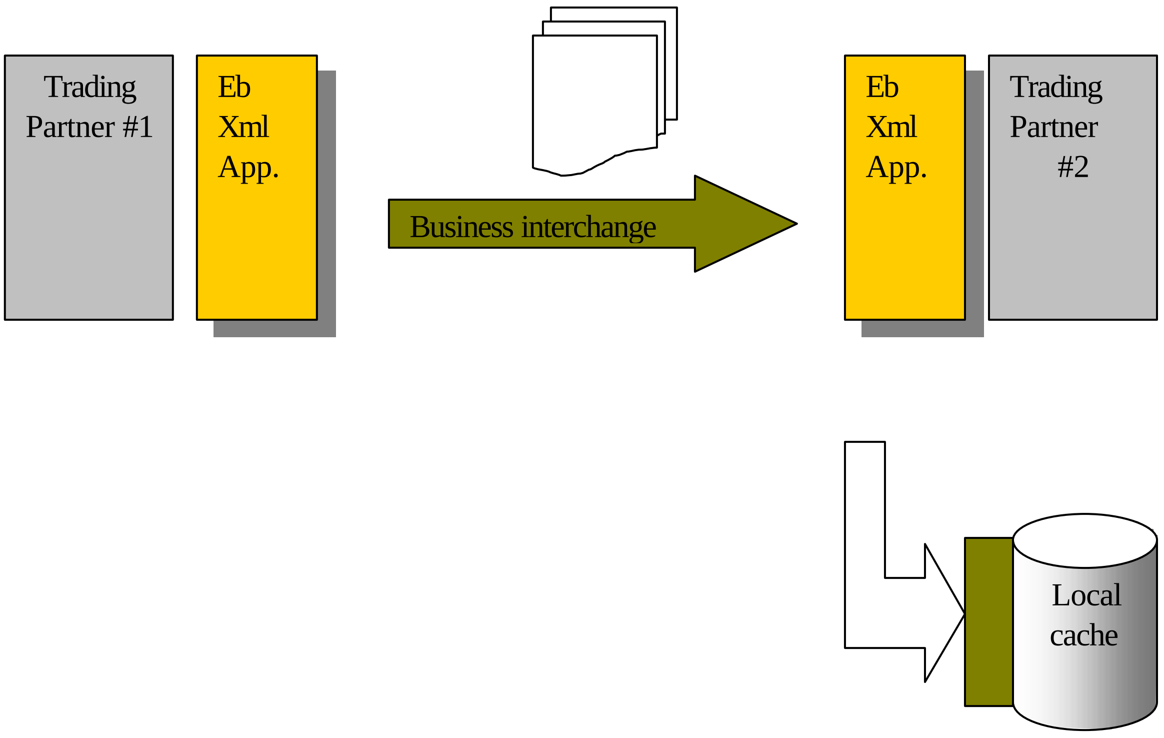
e. The trading partner can now send a message to another ebXML capable trading partner. The message sent to the second partner can be classified in two distinct categories: meta information and business information.

Figure 4.1.4 - A Business Interchange



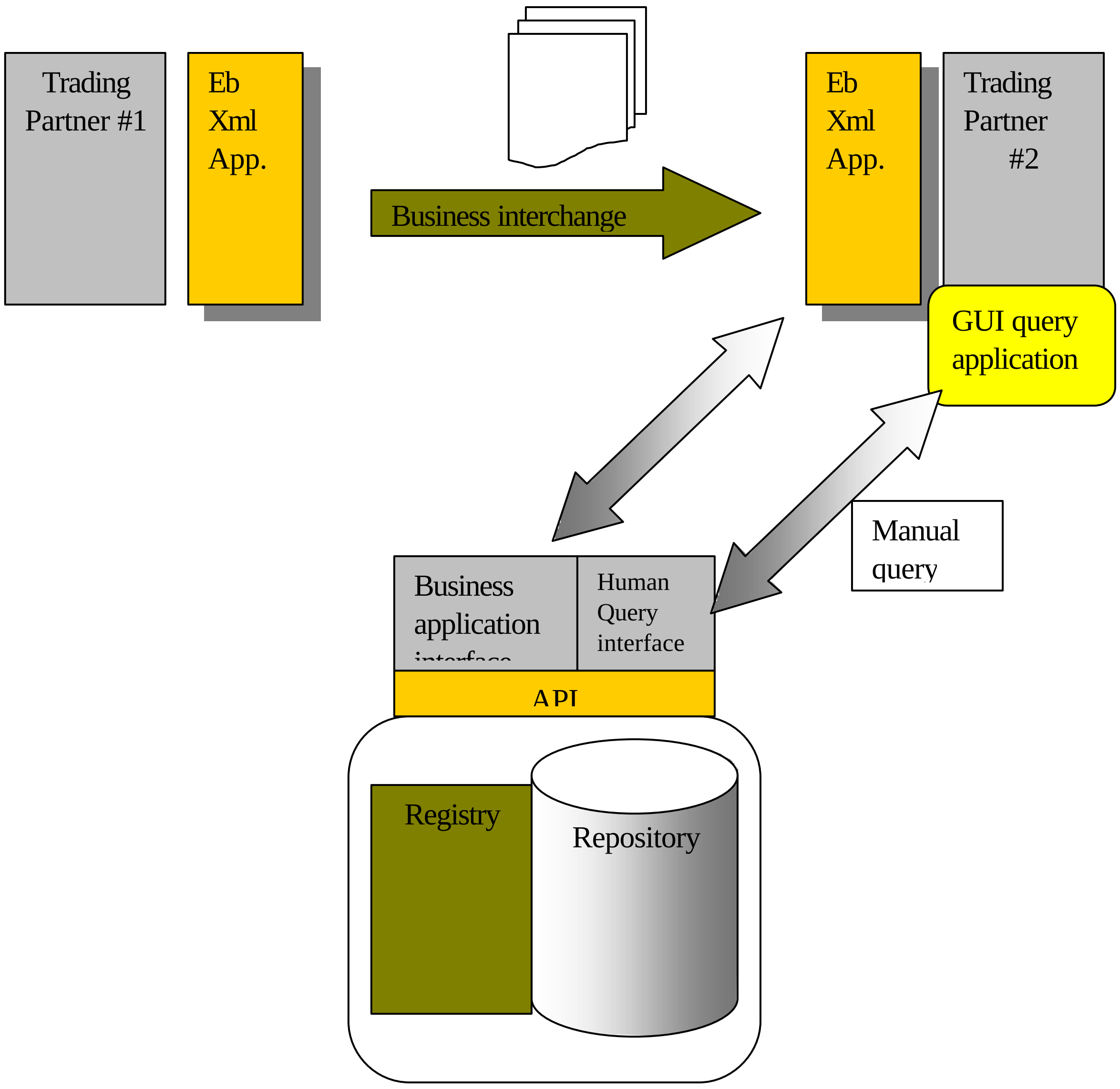
f. When trading partner #2 receives the incoming message, it gets handed to the ebXML application by the ebXML compliant server that receives it. The ebXML application then checks its own cache to see if the process object is available (versioning rules will apply).

Figure 4.1.5 - A Trading Partner receives an incoming business message.



g. If the object is available, the transaction can be acted upon.

h. If it is not found, the ebXML application shall then checks the registry and repository for the item.



i. If the object is found, the transaction can now be acted upon.

j. A third stage manual search can be employed to locate the object.

5. Conformance Issues and Testing

5.1 Conformance Definitions

a. Conformance is being able to measure that an implementation adhears to the requirements that are inherent in the specification

b. Conformance can only be tested for items which are defined in the specification.

   Conformance has three pieces: (1) the conformance statement or claim, (2) metrics to determine if it conforms and (3) a testing program with procedures.

Conformance components

ebXML shall provide a conformance
Conformance for Certification of Repository Authorities.

[KNOWN ISSUE - The conformance section (to be added to this document) shall address the need for versioning of specific components]

[KNOWN ISSUE - There is a larger issue of versioning which needs to be addressed]