Public Works and Government Services Canada | Travaux publics et Services gouvernementaux Canada

Recommendations for placement of CPSIN IJI Data Dictionary Data Elements into an ebXML / ISO-IEC 11179 Metadata Facility.

Oct 29, 2003

Prepared for Public Works and Government Services Canada

DISCLAIMER:

The contents of this document are considered confidential and classified as such. No member of the public should have access to this document. Readers are hereby cautioned not to leave copies of this document in un-secure environments.

Authors: Duane Nickull, ***duane@yellowdragonsoft.com***
C. Mathew Mackenzie, ***matt@yellowdragonsoft.com***

Table of Contents

1.0 Statement of Work

1.1 Background

In support of Government Online (GOL) strategy, the second phase of the Business Transformation Enablement Program (BTEP) is focused on achieving business and information inter-operability between government information systems. The BTEP's IM e-Enabler objective is to provide automated information management services to support delivery of Government of Canada services. The IM e-Enabler first timeline initiatives are focused on developing the means to test, demonstrate and establish procedures that would enable government projects to create and share "defined" processes for creating "shareable" information. The ebXML framework being the preferred source of logical and physical component designs and specifications, an ebXML registry/repository is required to provide the mechanism for registering, discovering and sharing design specifications for information and business processes.

Version 1.0 of ebXML specified a methodology for creating and managing a set of reusable data elements called Core Components. The Core Components work continued under the auspices of the United Nations CEFACT committee and is guided by the United Nations Unified Modelling Methodology (UMM).

The initial stage of this project will work with placing the Canadian Public Safety Information Network (CPSIN) Integrated Justice Initiative (IJI) data model into an instance of the ebXML version 2.1 Registry/Repository. The work involves placing all 353 identified data elements of that dictionary into the registry in alignment with the principle methodology in the ebXML Technical Architecture as a set of Reusable Data Elements for a specific domain. In Core Components methodology terms, these would be called "Business Information Entities" or BIE's, being data elements to be used within a specific context (see later sections for more explanation of "Context").

The goal/objective of establishing a registry repository is to facilitate the further development and use of a core justice dictionary for both national and international audiences, if deemed appropriate following the proof-of-concept project. The Registry will eventually facilitate integration of instance data with many other key systems and enable system/applications inter-operability.

This is the third and final of a series of reports generated as part of this project. At the end of this report is a set of recommendations for continuing the work.

1.2 Customer Requirement

PWGSC as a participant in the IM e-Enabler project requires that individual data elements of the IJI data dictionary be represented in a Registry/Repository to facilitate several use cases. The data elements in the registry/repository shall facilitate several use cases including, but not limited to the following:

1. Enable data modelers to use the data elements to build transaction sets in multiple syntaxes and representations.

2. Enable business or domain analysts to maintain a complete data dictionary and share it with multiple stakeholders.

3. Facilitate harmonization of data models across multiple domains within the government and international justice spaces.

4. Serve as a pathfinder, pilot project to proof the concept of a registry centric infrastructure.

5. Enable key stakeholders to analyze the benefits of a registry centric concept of operations.

6. Enable reuse of data elements across multiple domains.

7. Expressing the current IJI data dictionary in one or more ontology's within the registry.

8. Enable programmers ad systems analysts to build applications against the functionality prescribed by the registry/repository system.

9. Validate the Core Components technical specification methodology and provide feedback into that teams work.

1.3 Work Components

This report outlines various work items done in order to facilitate storing the CPSIN IJI data dictionary in an ebXML metadata Registry/Repository.

Before finding a serialization (a format for expression) for the individual Data Elements, a study of the current work within various standards bodies was necessary to determine which standards solutions are available and relevant to this project. The findings are that several major standards were not specialized enough to be a complete solution, yet a solution is directly attributable to utilizing key components of many of these standards. While some overlap exists between some of the standards, there are also some gaps. The feeling is that the work described hereinafter is highly relevant to completion of several of these standards and may provide valuable input.

Reliance on work done in these standards was indispensable as an asset while building a solution. After examining the OASIS Content Assembly Mechanism (CAM) and Business Collaboration Mechanism (BCM) initiatives, it was deemed that they are relevant, yet not currently able to be implemented as a complete solution. Many concepts of CAM are used within this document.

Accordingly, a recommendation to the PWGSC contract authority is that the project should define its own storage format for placing the data dictionary within a registry. That will also take into account several other initiatives in this project and determine a format that may subsequently be transformed into a standard for Data Element Metadata (DEM) representation internationally and within the Government of Canada context. The DEM format will be built by harmonizing data models from the UN/CEFACT CCTS and UMM, ISO/IEC 11179, ISO TC 154, ebXML Registry Information Model (RIM) and xml.

The CPSIN IJI data dictionary v 1.0, establishes custom data element standardization rules. To quote:

> *"The common unit for transferring information (data plus understanding) between organizations is the data element (ISO/IEC FDIS 11179-1:1999). Data elements exist as fields and columns in the databases files of CPSIN partners. They are documented as elements and attributes in the CPSIN partner organizations dictionaries, data models, object models and XML schemas."*

The Data Element component of the model is the ideal starting point for CPSIN-related data design exercises. Once the data elements are stored in a registry/repository, the CPSIN logical model will serve as a classification ontology.

2.0 Assumptions

Given the IJI data model as a starting point, it is recommended that the IM e-Enabler Proof of Concept project should build an XML representation of a meta-metadata object for each "Data Element" object in the model. There are currently 353 data elements in the CPSIN IJI data dictionary as of version 1.0. This recommendation is based on several key assumptions:

1. That instance data will likely be expressed in an XML format, yet must be able to render in UML, HTML, PDF, XML Schema, XML DTD and other formats.

2. That the set of requirements expressed in section 3.2 (below) is accurate. Assuming they are all requirements, XML is currently the only syntax choice for representing the Data Element Metadata within the Registry/Repository.

3. That the ebXML Registry Services Specification and associated Registry Information Model v. 20 and higher, are the best standards to support CPSIN data dictionary modeling and interchange (both are based on ISO 11179)

3.0 Approach and Solution

The proposed methodology to solve the problem is based on a four point plan of action.

1. Document the requirements from all stakeholders of the data elements. Ensure all stakeholders were represented and their requirements well documented. (UMM & Business Collaboration Framework {BCF} methodologies)

2. Review and reconcile all the relevant data models and derive a syntax neutral data element metadata model. Take careful steps to ensure the model will meet all the current and future functional requirements of the stakeholders.

3. Develop a serialization (expression) of the Data Element Metadata model in XML. Account for future forwards and backwards compatibility and ease of implementation from a programmers' perspective.

4. Develop a strategy and methodology for using the Data Element Metadata to develop new transactions and reconcile older data models as well as any new use cases introduced by key stakeholders.

NOTE: Setting out to reconcile the five different data model approaches is a monumental challenge.

3.1 Requirements for Data Element Metadata serialization

In order for data elements to be placed and managed within an ebXML registry, they must be serialized into a format that allows them to be bound to the Registry. A serialization is a format, which includes both the syntax and the taxonomy for expressing a Data Element. There are no formal standards for defining a format for such a binding or serialization. The UN/CEFACT Core Components Working Group defines a data model that was used as the basis for this work.

Before a format was defined, it was important to capture the requirements for what the Data Elements must be capable of supporting. In addition to the metadata requirements outlined in the IJI Data Dictionary, each Data Element Metadata (DEM) object should be capable of conveying the following information:

1. An XML schema and/or DTD may be derived or expressed from the DEM object, yet the DEM object must not preclude other formats of instance data from being used within an operational system in the future. Target output types include XML schema, XML DTD, HTML and PDF. It may also provide eForms capabilities.

2. The DEM objects shall be readable by both humans and application actors within an infrastructure and that the applications shall be able to consistently derive structure from the DEM objects. This requires a language with terse and exact parsing rules that leave no room for variance between commercial implementations of parsers or proprietary byte handling routines.

3. The DEM objects can explicitly point at or otherwise reference a UML or other modeling artifact via a variety of protocols (examples – HTTP/S, LDAP, FTP). This places a pre-requisite for a mechanism like xlink or hypertext linking.

4. The Data Element metadata shall have a binding to a set of RIM metadata and/or shall minimize replication of Registry meta-metadata instances except where required for data portability.

5. The DEM shall not constrain the final representation in any way, yet must be capable of facilitating multiple implementation serializations (syntax bindings) as represented via the UN/CEFACT core components technical specification diagram.

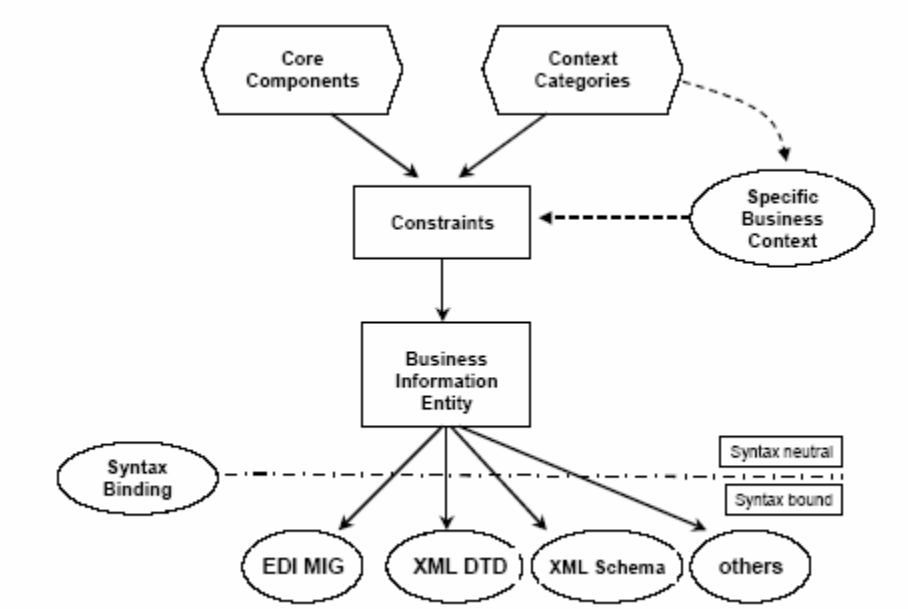


Figure 1 – Core Component concepts from UN/CEFACT CCTS Specification v 2.0

NOTE: Al of these concepts are explained in much greater detail within this report.

6. The DEM shall be capable of conveying semantics of the core Data Dictionary Data elements in more than one language and syntax. Initially, English and French will be supported.

7. The DEM must be in a format capable of expressing multi-byte character encoding such as UTF-16 in order to facilitate internationalization.

8. The DEM must be capable of being transformed easily into other DEM formats (such as the UN/CEFACT ATG2 Core Components/Business Information Entities Meta-metadata format and work by the OASIS CAM and BCM groups when those groups have completed their work.)

9. The DEM must be capable of declaring semantic equivalencies to other existing metadata objects. This is a requirements based on an understanding that integration with existing systems will be essential.

10. The DEM must be capable of containing an intrinsic relationship to context declarations in order to facilitate the above requirements, possibly in addition to the registry relationships expressed within the CPSIN data dictionary, ebXML RIM and ISO/EIC 11179 parts 1-5.

11. The DEM must facilitate both basic (atomic) Data Elements as well as more complex aggregates. The aggregates to be designated as UN/CEFACT aggregate core components (ACCs) and represented as aggregate business core components using XML schema.

12. The DEM should be written in a way so programmers can write implementations, yet if the DEM model changes, the implementations will not be broken. This is referred to as forwards compatibility.

Given the data model, a top down approach starting with a Data Element object is appropriate for a starting point. The object model shall follow ISO/IEC 11179 MetaData Registries representation formats, yet account for specializations provided by the IJI Data Dictionary work, Government of Canada (GoC) stakeholders, UN/CEFACT CCTS, ISO TC 154 and ebXML RIM. This model should be a superset so that metadata instances of all 353 Data Elements in the current CPSIN/IJI gold copy of the data dictionary can be placed into a Registry using this format.

4.0 Analyzing the Data Element Models

The logical model for expression of XML Data Element Metadata (DEM) has been derived by combining 5 works. Those works include the CPSIN IJI Data Dictionary version 1.0, the UN/CEFACT Core Components Technical Specification version 2.0, ISO/IEC 11179 (various works) and the ebXML Registry Information Model v 2.5.

Each of these models is examined in greater detail below.

4.1 UN/CEFACT Core Component and Business Information Entity model

Below is the model for UN/CEFACT Core Components from the version 2.0 technical specification.

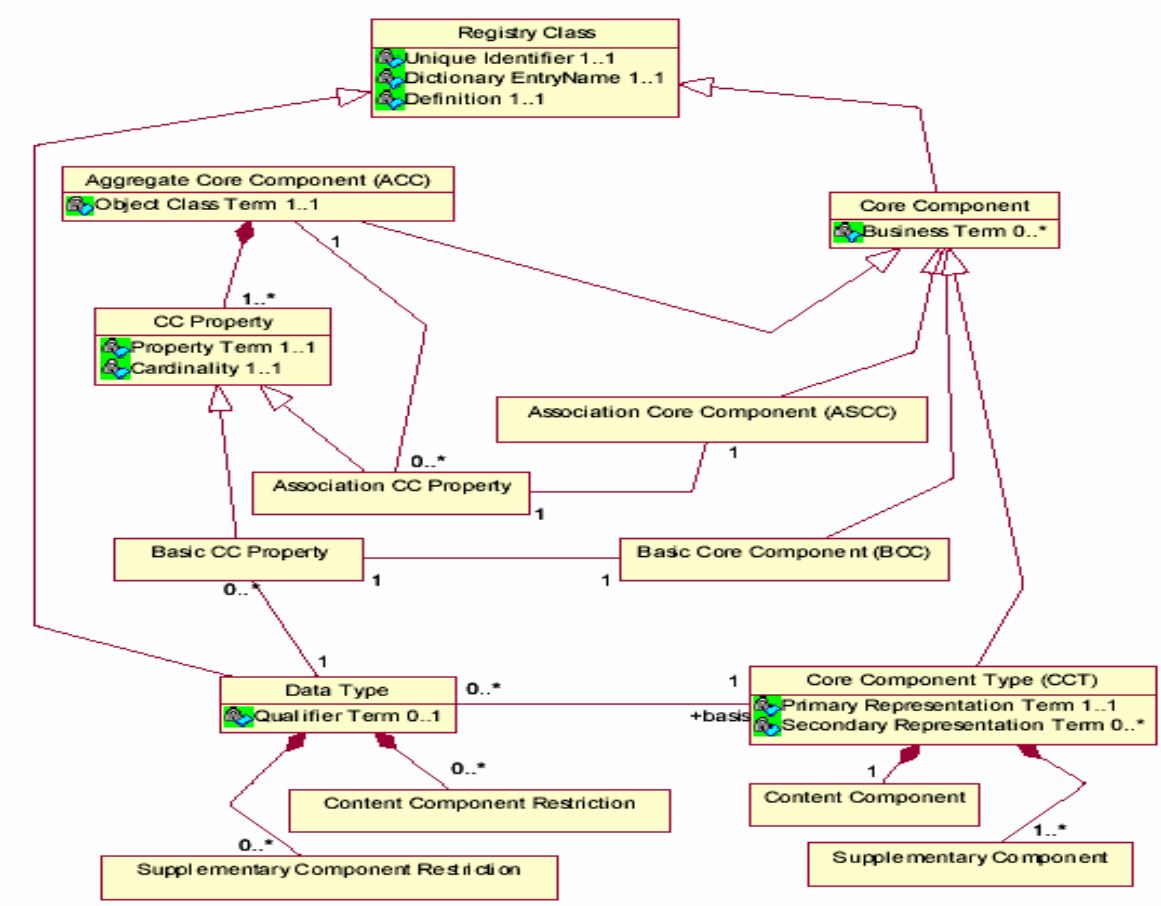


Figure 2 – Core Component model from UN/CEFACT CCTS Specification v 2.0

The UN/CEFACT Core Components Technical Specification version 2.0 contains a logical data model for a core component. The terminology of the Core Components working group's BIE is synonymous with the CPSIN IJI definition of a Data Element. The difference between the UN/CEFACT CCWG Core Component is that the UN/CEFACT Core Components are envisioned for a global set of business collaborations vs. the CPSIN IJI Data Dictionary has been scoped solely for the domain of justice, albeit international.

NOTE: Please note that the term "Data Types" is not always synonymous with the semantics used within different agencies.

4.1 CPSIN IJI Data Element model

The following data model for Data Elements is from the CPSIN IJI Data Dictionary. Some of the syntax neutral concepts expressed within it are likely better suited for XML specific metadata expressions using the W3C XML Schema format. This format is ideal for expressing cardinality, data types etc.

Name	Definition
The full name of the data element.	A statement that explains the meaning of the data element.

Property	Values/description
Version	A number, 1.0 for example, will describe the version of a data element for the purposes of change control.
Domain	The Domain represents the spectrum of values for a data element. Domains are documented above.
Minimum size	The minimum number of units in a layout including literals.
Maximum size	The maximum number of units in a layout including literals.
Layout of representation	The layout of characters in data element values expressed by a character string representation for display purposes.
Context	Used to further define the setting in which a data element will be used.
Permissible values	The set of representations of permissible instances of the data element, according to the representation form, layout, data type and maximum and minimum sizes specified in the corresponding attributes. The set can be specified by name, by reference to a source, or by enumeration of the representation of the instances. Only sample values have been listed to date and they are specified by English Name only. For complete lists of values that are currently being considered by the Working Group, please refer to the Coded Elements document found elsewhere on the CPSIN Data Dictionary CDROM or contact the Data Standards Secretariat.
Element Identifier	A language independent unique identifier for a data element within a registration authority.

Property	Values/description
Registration Authority	Any organization authorized to register data elements.
Registration status	A designation of the position in the registration life cycle specified by a registration authority. For more information, see Section 10.4 Registration Status.
Entity name	The CPSIN Logical Data Model entity or entities in which the data element appears. Elements that are part of key structures may appear in more than one entity.
Topic	The CPSIN Logical Data Model Topic to which the data element fundamentally belongs.
Comment	General comments about an attribute that may help to clarify a data concept.

Synonymous name	Context
An expanded form of the data element name, required to avoid ambiguity in data structures supported by the CPSIN Logical Data Model.	An explanation of the role played by the data element when this name is used. e.g. Parent Project Identifier for Project Identifier. See Section 8.3.1 Use of Rolenames in the CPSIN Data Dictionary.

Familiar name	Context
An alternative word or phrase that is commonly used to refer to the data element.	An explanation of the environment or circumstances in which the name is used, e.g. City Code for Telephone Address Area Code.

Figure 3 – IJI Data Element components from CPSIN IJI Specification v 1.0

Note: The term “Representations” as used within the CPSIN IJI Data Dictionary is not always synonymous with the use of it within the UN/CEFACT Core Components group.

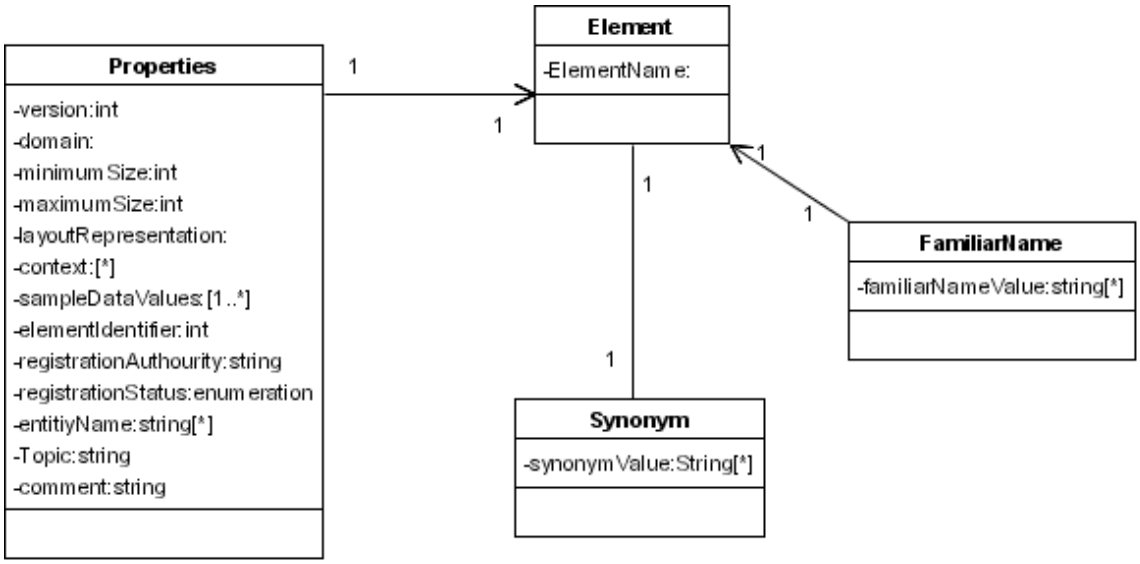


Figure 4 – UML Expression of CPSIN IJI Data Element model from Specification v 1.0

While this model is very close the CCTS model, it will have to be reconciled with the other models in order to place instances into the registry. The representations will have to be expanded into a separate set of elements in order to facilitate multiple serializations of the metadata for applications outside the registry.

4.3 ISO/IEC 11179 - 2002

Metadata Attribute Name		Application System	
	PV End Date (each PV)	(Not applicable)	
	Value Domain (VD) Context	Registry	
	VD Entry Name	ISO English-Language Country Short Name	
	VD Definition	All short, ISO-recognized English-language names of all countries.	
	VD Description	(Not applicable)	
	VD Entry Identifier	{RAI} 5678:1	
	Datatype	CHARACTER VARYING	
	Datatype Schema/Source	ANSI ISO SQL	
	Maximum Characters	44	
	Format	(Not applicable)	
	Unit of Measure	(Not applicable)	
	Precision	(Not applicable)	
	VD Origin	ISO 3166-1:1997	
	VD Explanatory Comment	The value domain includes only the subset of names that designate countries; it does not include names of territories.	
3	Representation Class Attributes		
	Representation Class	Name	
	Representation Class Qualifier	Short	
4	Data Element Name and Identifier		
	DE Name Context	Registry	Facility Data System
	DE Name	Country Mailing Address Name	Country.Mailing_Address.Name
	DE Entry Identifier	{RAI} 5394:1	
5	Other Data Element Attributes		
	DE Example	Denmark	
	DE Origin	Application system	
	DE Comment	This data element is required for delivery of mail outside the country of origin.	
	Submitting organization	Office of Enforcement and Compliance Assurance	
	Stewardship Contact	Facility Data Systems Administrator	
6	Data Element Concept and Conceptual Domain		
	Data Element Concept (DEC) Context	Registry	
	DEC Name	Country Identifier	
	DEC Definition	An identifier for a primary geopolitical entity of the world.	
	Object Class	Country	
	Object Class Qualifier	Mailing Address	
	Property	Identifier	
	Property Qualifier	(None)	
	DEC Entry Identifier	{RAI}12468:1	
	Conceptual Domain (CD) Context	Registry	
	CD Name	Country	
	CD Definition	The primary geopolitical entities of the world.	
	CD Entry Identifier	{RAI} 2468:1	

	Metadata Attribute Name	Application System
	CD Origin	ISO 3166:1
	Value Meaning (for each VM)	The primary geopolitical entity known as <China>
	VM Begin Date (for each VM)	19970110
	VM End Date (for each VM)	(Not applicable)
	VM Identifier (for each VM)	<Assigned by system as 1001…1230: one to each VM>
7	Classification Type Examples	Classification Values for Classification Type
	Keyword	Country, Address, Mailing
	Group	Mailing Address
	Object	Address, Country
	Layer of Abstraction Type	Specialization
8	Registration and Administrative Status	
	DE Registration Status	Recorded
	DE Administrative Status	In Quality Review
	VD Registration Status	Standard
	VD Administrative Status	Final
	DEC Registration Status	Recorded
	DEC Administrative Status	In Quality Review
	CD Registration Status	Standard
	CD Administrative Status	Final

Figure 4 – ISO/IEC Data Element Components

4.4 ebXML Registry Information Model

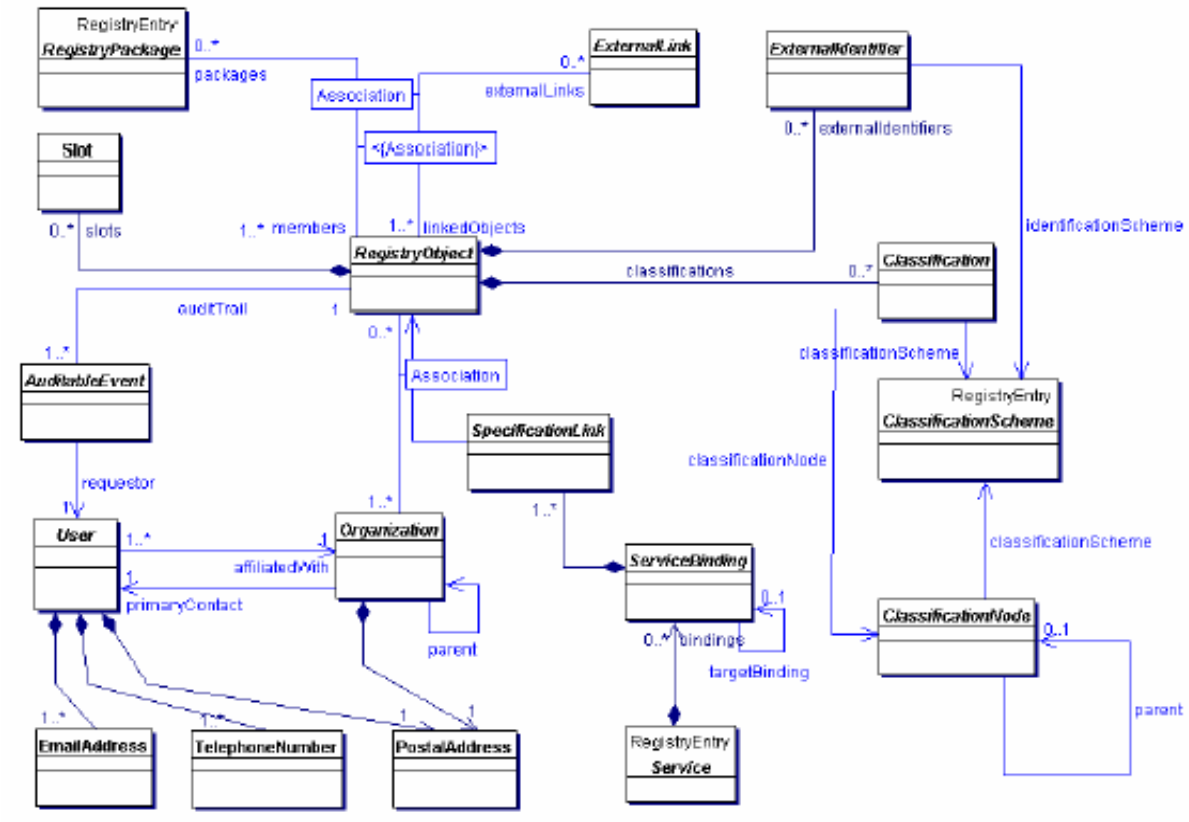


Figure 5 – ebXML Registry Information Model v 2.5

There is also a large degree of overlap with the IJI and CCTS data models. All of these models have an identifier, versions, common name and associations with the authority that is responsible.

5.0 Model Reconciliation Issues and Proposed Solutions

For the Data Elements of the CPSIN IJI data dictionary to be stored/managed in an ebXML Registry/Repository system, there is need for alignment between the CPSIN Data Element Status code values and the ebXML Registry code values. Both seem to be derived from ISO/IEC 11179-3, yet the ISO/IEC standard adds another layer of complexity to creating a prototype implementation.

The first recommendation is to align the terminology used to describe certain terms.

IJI Term	**CCTS**	**ISO/IEC 11179**	**ebXML RIM**
Element Name	Dictionary Entry Name	Data Element Entry Name	Registry Object Name
Definition	**Definition**	**Definition**	
Version	Version		MajorVersion; MinorVersion
Not defined			Expiration date
Domain			Classification

			Node(s)
Minimum size	Primitive Type Minimum length (from data type)		
Maximum size	Primitive Type Maximum length (from data type)		
Layout representation	Primary Representation; Secondary Representation; Expression Type (from Data Types)…		
Context	[not contextually specific until BIE]		Classification Node(s) (RIM 2.5 section 9.4
Sample Data values	Representation Term		n/a
Element Identifier	Unique Identifier		Uuid
Registration Authority	Registrar, Registration Authority, **Submitting Organization**	Responsible Organization; **Submiting Organization**	Responsible Organization; **Submitting Organization**
Registration Status	[handled by ebXML RIM]	tba	Status
Entity Name			
Topic			
Comment			
Synonym	Business Term		
Familiar name	Dictionary name?		

Table 1 – Comparison of various components

Each of the IJI data Element attributes will be broken out individually and discussed in the next section of this document. This will guide the XML serialization solution.

As per ISO/IEC 11179, the entire set of data element attributes will be grouped together into 4 sub areas in a new model. The new model will classify all existing attributes into one of the subclasses of Data Element

The ebXML Registry Information Model makes the attribute “Status” mandatory. Since this may be primarily used to machine access, there may be a secondary of separate “Status” asserted by one or more organizations who use the DEM. It is not mandatory that these two Status attributes be synchronized since on can be retrieved programmatically from the Registry and the other one can be read from the instance; but it is a recommendation that the RIM status can also be accessed via the DEM instance serialization. Further reasons are set forth in section 5.12.

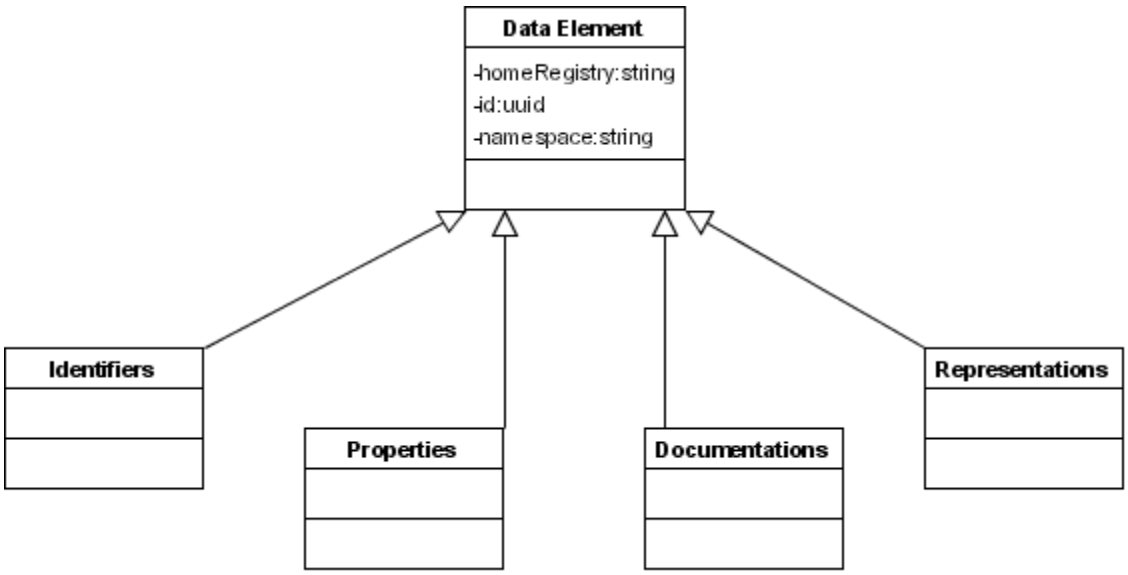


Figure 7 – UML expression of IJI Data ELement

5.1 Element Name

The IJI data dictionary declares that "element name" means "The full name of the Data Element". In general, the IJI data dictionary also follows the ISO 11179 naming conventions of an Object Class Term, Property term and a representation term. Optionally, a further qualifier term may be present. The names of Data Elements are represented alphabetically as a list.

The Core Components Technical Specification applies a similar convention. CCTS says a *Naming Convention* is necessary to gain consistency in the naming and defining of all *Core Components*, *Data Types* and *Business Information Entities*. The IJIJ data dictionary has done a brilliant job of applying such a naming convention.

One of the primary reasons for such is that consistency must facilitate comparison and meaning during the discovery and analysis process and also will guide locating meaningful data elements while modeling new processes or making new schemas.

The UN/CEFACT CCTS uses a format of dot notation to concatenate terms representing class term, property term and representation terms for the Dictionary Entry Name (the term that is synonymous with the IJI's "element name"). IJI uses a space between words.

Example: IJI data element "Being Type Identifier" would become "Being.Type.Identifier" if expressed using the CCTS format.

CCTS uses only words from the Oxford English dictionary for data element names. This does not preclude them from being expressed in other languages.

Recommendation – use dot notation for Element Name.

While further research may be needed to determine the technical advantages of each, the favor was to use the dot notation format from UN/CEFACT in order to better align with UMM and the ebXML RIM work. Other groups within government may be likely to use the CCTS format and harmonization of terminology in formats that may be familiar to others could b useful.

Provision must also be made for more that one language expression of a data element name. Both French and English should be initially supported. Allowing other agencies to assert their names for data elements may aid integration.

5.2 Definition

IJI defines the meaning of *Definition* as “A statement that explains the meaning of the data element”.

The requirements are many. Multi-lingual support should be a must. The onus to be able to explain semantics is also linked to the concept of context. A meaning may change as the context in which a *data element* is used changes.

CCTS declares that the *definition* shall be consistent with the requirements of ISO 11179-4 Section 4 and will provide an understandable meaning, which should also be translatable to other languages. This expands on IJI and meets the requirements of a Canada wide IJI data dictionary that may someday become multi-language.

Recommendation – Definition as a property assertion.

Definition is really a property of a data element asserted by an organization. It will be useful to be able to cross reference multiple definitions in order to semantically map different data elements from one data format to another to achieve integration.

A primary assertion of definition should be made by the Responsible Organization (RO), a data steward.

5.3 Version

The IJI data dictionary has a notion of versioning Data Elements. This is equally important and represented within the ISO/IEC 11179 Metadata Registry (MDR), the ebXML RIM and CCTS. The *mask* or *representation layout* of the IJI version attribute is a simple majorVersion.minorVersion.

There are some potential dependencies that are not talked about within the IJI data dictionary.

Recommendation – version broken into major, minor and incremental

Recommend that version become a property of a data element and expressed under the <Property> element in an xml serialization. The xml format is designed not to place any dependency on the version attributes for programmers/applications. It is further recommended to break the version into three sub-types of version: majorVersion, minorVersion and incrementalVersion. Each of these attributes will have a data type of integer.

The following format is recommended.

```
<Properties assertedBy="Canadian Public Safety Information Network">
     <Property name="version.major" value="1" />
     <Property name="version.minor" value="0" />
     <Property name="version.incremental" value="0" />
...
```

5.4 Domain

A Domain, by definition of the IJI Data Dictionary v 1.0, is a group of predefined characteristics for a data element that is determined by its Representation Term. It is a set of values or a range of possible values.

A data type defines in more physical terms what the data element can contain.
Examples include character strings, numbers, dates and times. This is required for the specification of fields in record layouts, database table columns and XML tags to define length and acceptable representation.

Recommendation:

The Domain attribute should be asserted as a property of the Data Element to preserve the specific set of values defined within the IJI data dictionary. While it may be redundant with the <Representation> elements, it is important to preserve the syntax neutral concepts developed by the IJI data stewards (RO).

The representation in the XML Serialization will be as follows:

```
<Properties assertedBy="RO_Identifier_goes_here">
      ...
      <Property name="domain" value="dDate" />
      ...
```

5.5 Minimum Size

Minimum size is a representation attribute to specify the minimum number of units in a representation.

Recommended: attribute of Property and specific to context.

Move this to an attribute of the Properties element. This may also be further constrained within a specific context by imposing syntax specific constraints upon this attribute. For instance, the Properties attribute may be expressed by stating the type attribute is “minimumSize” and the value is “2”. In the <Representations> branch that is contextually specific to W3C XML Schema format, the values are further constrained to a choice of one – two byte value from an enumerated list of twelve choices.

```
<DataElement>
  ...
   <Properties assertedBy=”responsibleOrganizationIdentifier”>
      <Property type=”minimumSize”
                value=”2”
   </Properties>
   <Representations>
         <Representation type="http://www.w3.org/2001/XMLSchema" context="all">
         <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                     targetNamespace="urn:component-foo">
         <!--schema here-->
         <xsd:element name="BeingBirthMonth">
           <xsd:complexType>
```

```
            <xsd:simpleContent>
              <xsd:extension base="xs:string">
                  <xsd:attribute name="value" use="required">
                    <xsd:simpleType>
                      <xsd:restriction base="xs:NMTOKEN">
                        <xsd:enumeration value="01"/>
                        <xsd:enumeration value="02"/>
                        <xsd:enumeration value="03"/>
                        <xsd:enumeration value="04"/>
                        <xsd:enumeration value="05"/>
                        <xsd:enumeration value="06"/>
                        <xsd:enumeration value="07"/>
                        <xsd:enumeration value="08"/>
                        <xsd:enumeration value="09"/>
                        <xsd:enumeration value="10"/>
                        <xsd:enumeration value="11"/>
                        <xsd:enumeration value="12"/>
                      </xsd:restriction>
                    </xsd:simpleType>
                  </xsd:attribute>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
...
```

5.6 Maximum Size

Like minimum size, Maximum size is a representation attribute to specify the maximum number of units in a representation. See the 5.5 Minimum Size for recommended solution.

5.7 Layout of Representation

The best solution for a layout of representation was to place this information in the native metadata constraint language for the instance output type. This information is kept under the path:

```
        //DataElement/Representations/Representation
```

An example is to place an XML Schema fragment to make assertions of constraints on final instance data in XML format. XML DTD may also work for this purpose, but to a lesser effect.

Instance data is also acceptable under this branch. If the output is in PDF format, it is possible to place a PDF binary data blob into the XML tree within a CDATA section. An HTML or XHTML representation for eForms may also be acceptable.

5.8 Context

Section 9.4 of the ebXML Registry Information Model discusses using the classification schemas and nodes as a mechanism to express contextual classifications of registry objects. This is the methodology we also recommend based on the ease of which this mechanism can express

multiple contextual classifications and may be extended to meet future or other requirements in this area.

The UN/CEFACT CCTS v2.0 identifies 8 context driver categories.

- Geopolitical
- Business Process
- Supporting Business Process
- Role
- Official Constraint
- Systems Capabilities
- Product Classification
- Industry Classification

A problem does exist however with respect to registry classification scheme bloating. If all the CCTS context classifications are used, the registry classification scheme would be unmanageably large.

Each individual context category has tree parts to it. A context category identifier, a qualifier or identifier for a list of set of code values that are acceptable to express the values for that context, and a value or set of values specific to an individual context. It may also be a good idea to include the identifier for an agency or place where a stakeholder could retrieve additional information about a specific context coded value list.

Furthermore, the list of eight context categories may not be complete and care should be taken to build a Data Element in a format that will not break should another context be added.

A Data Element of the type envisioned in the CCTS is sufficiently high enough that it is reusable over several contexts. When a Core Component Data Element is constrained within a specific set of context category values, it becomes a Business Information Entity or BIE.

Recommendation: Inline and using Registry Classifications

The IJI Data Dictionary includes a Data Element property of Context. Because the IJI Data Dictionary was developed within a specific domain (context), the Data Elements should be considered BIE's and are specific to certain sets of context values. We recommend keeping the Property attribute of context in the Properties section of the Data Element.

The IJI data dictionary can be further contextually classified by one or more schemes by utilizing the ebXML Registry Classification schemes in a hierarchical manner to assert context. By testing this, we identified potential problems that may happen regarding two issues:

- The order in which data modelers apply context categories to their modeling efforts; and
- A single registry may bloat if multiple classification schemes are used to represent all possible choices of context values.

The latter is worth a closer look. For example, if you chose to express just 4 context categories and had 50 values for each possible context, you would have to create 50^4 classification schemes to express one specific order of classification. In reality, the numbers are much larger.

Context	Classification scheme	Number of values (approx.)
Geopolitical	ISO 3166-2 ISO 639	1,650 (165 countries * 10 regions) * the number of languages

Industry Classification	NAICS	3,950
Business Process	UN/CEFACT Catalog of common business processes	50 (unknown at this time)
Supporting Process	UN/CEFACT Catalog of common business processes	50 (unknown at this time)
Official constraints	Unknown (United Nations + each nations legislation (and United Nations)	5000
Role	Depends on processes.	25 (guess)
Systems Capabilities	Unknown	25 (guess)
Product Classification	UN/SPSC	3,250 +

If you account for every possible combination, this may not work very well or take a long time to implement.

Recommend: Limited use and an XML serialization of Context Assertion/Declaration sets.

Highly recommend that such projects proceed on a limited basis until more is learned about how to declare, store and locate context groupings.

A solution to have a 2 or three deep nested hierarchy with limited context values expressed as a classification scheme, then serialize a set of context definitions into an XML Registry Object is probably the best way to make these declarations. The user would then be able to chose between multiple context declarations under a certain node to retrieve a complete context declaration.

The context format was written as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContextAssertion homeRegistry="http://ebxml.pwgsc.gc.ca:8080"
                  value="urn:uuid:4a593056-3509-0766-2e7b-4e154030423f"
>
  <!--Category = ( Geopolitical |
                  OfficialConstraint |
                  Process |
                  ProcessRole |
                  SupportingRole |
                  ProductClassification |
                  IndustryClasification |
                  SystemCapability
                  )-->
  <Declaration category="Geopolitical"
               qualifier="ISO-3166-2"
               agencyURL="http://www.iso.org"
               value="CA-ON" />
  <Declaration category="IndustryClassification"
```

```
               qualifier="NAICS-2002"
               agencyURL="http://www.naics.org"
               value="9221"
               description="Justice, Public Order, and Safety
Activities" />
  <Declaration category="SystemCapability"
               qualifier="ServiceOutputSyntax"
               value="xml schema xsd" />
</ContextAssertion>
```

The model for the context declaration is simple. A root element of ContextAssertion has only one child element ("Declaration"). Declaration is an empty element having only attribute values for category (an enumerated list of the eight context categories), qualified (a token to identify the qualifier for the value), agencyURI (a resource identifier to aide in assessing context values bu providing a link to more information about the qualifier), a value (the actual context value) and an optional description.

5.9 Permissible values

Permissible values is part of the representation domain. Recommend this is left to the final <Representation> element. See sections 5.5 and 5.6. Permissible values are best expressed using XML schema enumerated list declarations in the <Representation> part of the DEM.

Permissible values can be expressed inline if needed as per the original IJI Data Dictionary metadata. They are a <Property>.

5.10 Element Identifier

Notwithstanding another requirement for using UUID as a location-neutral binding, the IJI data Dictionary has a notion of identifiers assigned by the agency assuming the role of ISO/IEC 11179 *Responsible Organization* or RO. A well documented problem with RO assigned identifiers is that they are not themselves unique and cannot facilitate the requirements of a complex design time or runtime architecture.

The ebXML RIM 2.5 requires that each and every Registry Object has a UUID in the form of a DCE 128 bit algorithm. The DCE algorithm is thought to be adequate for purposes of establishing an Universally unique identifier or UUID.

The problem with the DCE UUID is that it in itself is insufficient to meet the needs of the Binding between the instance and DEM objects. Decoupling the URI for the location of a registry may be a great way to facilitate both federated registry deployment and tracking of a single data element within various registries.

RECOMMENDATIONS:

1. To use both the Registry Authority assigned UUID's and the RO assigned UID's inline in each DEM object.

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataElement
  home="http://ebxml.pwgsc.gc.ca:8080"
  id="urn:uuid:6e60580b-4538-2615-0c2c-3e034c430445"
  xmlns="http://ns.cpsin.org/data-element/1.0"
  >
...
```

2. That the need for binding the two together can be addressed by programmatically accessing the registry and specifying the UUID at the time the DEM is placed into the Registry. A program to generate the DCE 128 bit UUID's should be used to make these prior to them being submitted to the registry.

The ebXML Registry specification states that the Registry must be capable of generating a UUID in the DCE 128 bit format if one is not supplied at the time a Registry Object is loaded into the Registry. For now – a methodology of allowing the registry to make the UUID, then modifying the DEM to reflect this can be done, albeit a clumsy and un-elegant solution.

The mechanical identification and retrieval of this data element may be subsequently discovered by using a combination of the home URI (includes the protocol for retrieval – example ldap://, http:// or ftp://) and contains assertions by various parties of how to identify the Data Element. The CCTS specification has a concept of a dictionary name in plain English, using dot notation to concatenate concepts. The IJI data Dictionary has a similar notion of a name, using spaces to join.

OtherIdentifiers

Each element may have more than one identifier. Those using Document Assembly specification work for example may wish to use other identifiers rather than the Registry assigned UUID. Additionally, an identifier to represent a common dictionary term (such as the UN/CEFACT CCTS) may be useful.

Recommendation – multiple identifiers

The XML expression for the additional identifiers may be expressed as follows:

```
<Identifiers>
  <Identifier type="responsibleOrgURL"
              value="http://www.sgc.gc.ca/iji-iij/CPSIN ITS e.asp" />
  <Identifier type="ElementIdentifier"
              value="014" />
  <Identifier type="DataDictionaryName"
              value="Gender.Identifier"
              xml:lang="en-CA" />
  <Identifier type="EntityName"
              value="Being, Gender" />
  </Identifiers>
```

The "Identifiers" element contains only one child element ("Identifier"). Each Identifier element has no children, two mandatory attributes ("type" – to represent the qualifier for the entity who asserts the identifier and "value+ - the actual value of the identifier) and one optional attribute (xml:lang to facilitate multi lingual support).

This model is good to ensure forwards and backwards compatibility with implementations. Additionally, existing code bases will be unaffected by any new identifiers added to the DEM object over its' lifecycle.

The IJI "Entity name" attribute of a data element has been classified as an identifier. The synonym may be placed here as well.

IJI Element Identifier

Element identifier is already assigned for many of the data elements within the IJI data dictionary. They start a 0001 and go up to 0353. These are preserved.

UN/CEFACT Data Dictionary Name

The CCTS dictionary name is also classified as an identifier. It is thought to be semantically meaningful to humans who are familiar with the semantics of words described in the oxford English dictionary. The data format above facilitates that other language data dictionary names may also be used within this structure to facilitate global re-use of data elements.

IJI Entity Name

This is the identifier for the CPSIN IJI logical model entity or entities in which the data element appears. This may be very hard to track and synchronize in an inline serialization since it will have to be updated each time a new entity uses a specific Data Element.

It is recommended that the relationships between the IJI Data Elements and the Entities that use them be reflected in the Registry via the "Associations" mechanism. This alleviates the problems of synchronization. The ebXML Registry is sufficient to handle this requirement.

Responsible Organization URL

The IJI data dictionary describes a 'Registration Authority" (RA). The RA is called a Responsible Organization within ISO/IEC 11179 or RO. The RO identifier is needed to identify the RO/RA actor. A URL is probably the best way to identify each RA/RO since they are unique and also can lead to more details of how to contact the RA/RO. See below 5.11 also.

5.11 Registration Authority

See directly above also.

Within the Data model for data elements are three identifiers that are assigned by the Responsible Organization (a term used to identify the Responsible Organization or "RO"). The XML Fragment proposed for the XML serialization is expressed as follows:

```
  <Identifiers>
<Identifier type="responsibleOrgURL"
            value="http://www.sgc.gc.ca/iji-iij/" />
...
```

This presents a possible solution to use a URL to identify the RO. The proposed solution for the CPSIN Registry will be to use the value http://www.sgc.gc.ca/iji-iij/.

5.12 Registration Status

The CPSIN version 1.0 gold data dictionary has the following permissible values for a Data Element:

- Proposed
- Under Construction
- Withdrawn
- Accepted -Code Values Pending
- Approved
- Retired

With the following extensions based on ISO/IEC 11179

- Certified
- Standardized

The ebXML RIM v 2.5 specifies the following as permissible values for a RegistryObject (NOTE: RegistryObject may be deemed a *Superclass* of Data Element)

The following values are from ebXML RIM 2.5 and derive an alignment of terms/semantics

- Submitted
- Approved
- Deprecated
- Undeprecated
- Withdrawn

Solution:

The following table maps the CPSIN data dictionary status attribute to the ebXML RIM attribute for status.

CPSIN Term	**EbXML Rim Term**	**Notes:**
Proposed	Submitted	There are several terms which do not exist in the RIM attribute enumerated list of values. "Submitted" most closely matches the semantics for "Proposed"
Under Construction	Submitted	There is no equivalent to "Under Construction within the ebXML RIM. Propose to use Submitted as equivalent. "Under Construction" seems to be a specialization of submitted (ie – submitted plus an explanation)
Withdrawn	Withdrawn	Perfect match
Accepted – Code Values Pending	Submitted	Accepted – code Values pending is another specialization of Submitted. (see "Under Construction" above).
Approved	Approved	Matches
Retired	Deprecated	Deprecated means it is no longer in date. Seemed to match better with "retired" than "withdrawn", which indicates an item has been withdrawn completely. If it is retired, it may still be desired that it is reference-able from the Registry.

Recommendation: IJI Status is a Property of a Data Element

Proposed to make the following format a way to express the status in both ebXML RIM and CPSIN IJI terms. The Registry itself may keep track of the official status in the RIM metadata for

each Data Element Object within the registry and the xml serialization of each Data Element object will retain it's own status property assertion as defined by IJI.

The Registration status assertion is a property of a registry object and should be expressed in a way that allows an organization of type Responsible Organization (RO is defined as the organization responsible for this object) to be associated as making the assertion.

```
<DataElement>
  ...
   <Properties assertedBy=”responsibleOrganizationIdentifier”>
      <Property type=”registrationStatus”
                value=”( Proposed |
                         UnderConstruction |
                         Withdrawn |
                         Accepted-CodeValuesPending |
                         Approved |
                         Retired )”

  ...
```

CPSINStatus attribute shall have a choice of one of the values from the enumerated list. This is deemed necessary to preserve any specializations of the registry status assertion needed by IJI data dictionary stakeholders.

It is not clear if there is a requirement to synchronize these two status's (or even if it is possible given the discrepancies between the ebXML RIM and the IJI Data Dictionary.

5.13 Entity Name

The entity name attribute of each Data Element is a property asserted by the IJI Data Steward. It can be expressed within the properties branch of the Data element as follows:

```
  <Properties assertedBy="RO_Identifier_goes_here">
      <Property name="EntityName" value="being, person" />
```

5.14 Topic

The Topic attribute may be preserved as a property asserted by the RO. Expression of such may be as follows:

```
  <Properties assertedBy="RO_Identifier_goes_here">
      <Property name="topic" value="" />
```

5.15 Comment

A comment (as defined by the IJIJ data Element dictionary) is a type of annotation that is used for the RA/RO to provide additional details that they deem relevant to pass over to users of the data element.

Recommendation – expansion of comment into Documentation.

```
  <Documentations>
    <Documentation type="comment|note|instruction|other"
                   locale="en_CA"
```

```
                  mimeType="text/html">
                  <![CDATA[<html><body>Element Approved but further
                  research needed for values</body></html>]]>
  </Documentation>
  <Documentation type="comment|note|instruction|other"
                  locale="fr CA"
                  mimeType="text/html">
    <![CDATA[<html><body>viva la difference!</body></html>]]>
  </Documentation>
<Documentations>
```

The XML format for representing the data element will contain a “Documentation” element with three attributes. This has also been designed to allow for extensions without having any impact on existing implementations.

Each Documentation element has three attributes – “type” (must be a choice from an enumerated list of either comment | note | instruction | other); “locale” (must be a choice from an enumerated list of the ISO 3166-2 country and region code list) and mimeType (to specify the MIME type of the Documentation). This format supports full multi lingual capabilities to express annotations for extending the base set of details.

Adding content to this branch of the XML representation must be done or approved by the RA/RO only. Others who wish to add content should submit their requests to the RA/RO.

5.16 Synonymous name

Synonymous name is specialization of Identifier. It may be represented in the <Identifiers> element. In order to facilitate multi lingual support and re-use, a language attribute that may use ISO 639 language code identifiers to identify the natural language of each synonymous name may be a good idea as follows:

```
<Identifiers>
  <Identifier type="SynonymousName" value="text" xml:lang="en-CA" />
</Identifiers>
```

5.17 Familiar Name

Familiar name is a type of identifier. It will be represented in the <Identifiers> element branch of the model. It should also be capable of being expressed in natural languages other than English.

```
<Identifiers>

    <Identifier type="familiarName" value="text" xml:lang="en-CA" />

</Identifiers>
```

6.0 Context Declaration Mechanism

Note: This is a vast area and will need to be discussed later in greater detail. Further work must be done to ensure all stakeholders are represented.

A model for the declaration of context may be derived from the UN/CEFACT Core Components Technical Specification v 2.0 (CCTS). Within that document, exists a model for declaring sets of contexts. The model is shown below.

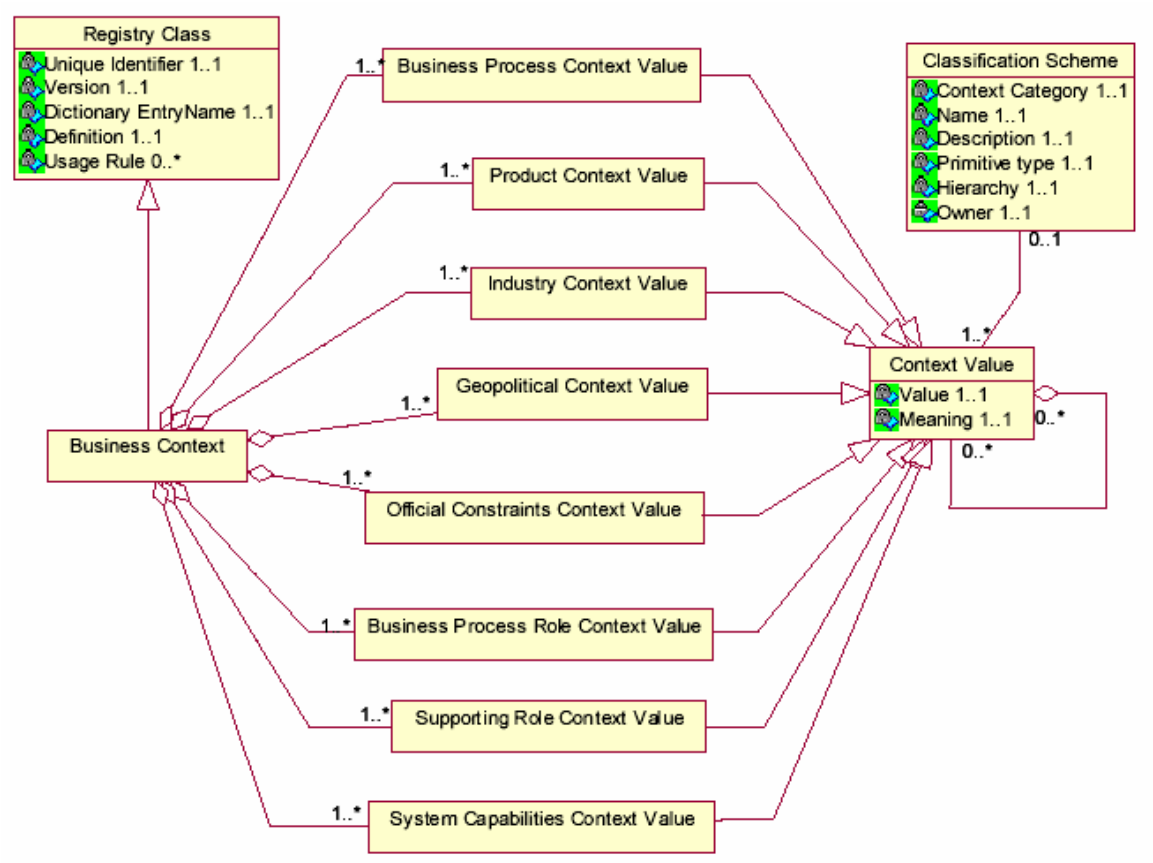


Figure 8 – Core Component model for Context from UN/CEFACT CCTS Specification v 2.0

The context format may be expressed using XML as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ContextAssertion homeRegistry="http://ebxml.pwgsc.gc.ca:8080"
                  value="urn:uuid:4a593056-3509-0766-2e7b-4e154030423f"
>
  <!--Category = ( Geopolitical |
                  OfficialConstraint |
                  Process |
                  ProcessRole |
                  SupportingRole |
                  ProductClassification |
                  IndustryClasification |
                  SystemCapability
                  )-->
  <Declaration category="Geopolitical"
               qualifier="ISO-3166-2"
               agencyURL="http://www.iso.org"
               value="CA-ON" />
  <Declaration category="IndustryClassification"
               qualifier="NAICS-2002"
```

```
               agencyURL="http://www.naics.org"
               value="9221"
               description="Justice, Public Order, and Safety
Activities" />
  <Declaration category="SystemCapability"
               qualifier="ServiceOutputSyntax"
               value="xml schema xsd" />
</ContextAssertion>
```

This format has several design considerations that should be incorporated into any final design.

a. It's hierarchy is simple and adding additional context categories at a later date will not break any existing implementations by fault of not being able to process the existing categories.
b. It allows for multiple qualifiers for context values. It is not fixed to any one set of values.
c. It is flexible and allows context sets to be declared that are incomplete (example – have accounted for only 3 out of 8 context categories).

NOTE: This is work under development. Next steps – develop an XML Schema for constraining this data model and publishing it.

7.0 Assembly of Metadata at Design Time (Assembly Document)

[To be completed later]

The assembly guide is meta-metadata. The purpose of the assembly guide is to provide a map to build an XML schema as an output. The assembly guide document is used at design time only. It must perform a variety of features.

The sample below was used for the demonstration in Ottawa, Canada in September 2003. Further work is needed to derive a functional application and rule set.

The assembly guide is an important part of the methodology for allowing users to build new documents based on the data elements in the registry.

```
<? xml version="1.0" ?>
<AssemblyGuide outputSyntax="W3CSchema"
               defaultContentRegistry="http://ebxml.pwgsc.gc.ca:8080">
  <DataElement root="true"
               name="Being" >
    <DataElement name="BeingTypeIdentifier"
                 useDataElement="uuid" />
    <DataElement name="0014-Gender.Identifier.xml"
                 useDataElement="uuid" />
    <DataElement name="BeingBirthYear"
                 useDataElement="uuid" />
    <DataElement name="BeingBirthMonth"
                 useDataElement="uuid" />
    <DataElement name="BeingBirthDay"
                 useDataElement="uuid" />
  </DataElement>
</AssemblyGuide>
```

8.0 Methodology for Extraction of Data Element Metadata

[To be completed later]

The exact methodology for extracting a Business Information Entity (BIE) from a Data Element is work to be done in the future.

For the purposes of this paper, a methodology was developed by placing several BIE's inline and extracting the correct schema fragment based on a set of contexts (see section 6.0 – Context above).

Java code was written to consume one data element and one context declaration and a parameter to express the output type. For purposes of the demonstration, the output type was the W3C XML schema.

The code below requires three separate jar files to run. Xerces.jar, jdom.jar and xml-apis.jar. All are available freely on the internet.

This code is constructed to allow it to be called as a helper class from a new main. The new main could easily consume the Assembly Guide, then extract the exact metadata from each Data element and dynamically build a new Schema.

A custom extension could also be written to use a registry client application to request each data element directly from the registry.

This sample proves that dynamic content assembly is possible and easy to accomplish.

NOTE: Formatting of the code was not preserved.

```
/**
 * This class will return a representation of a DataElement
 * given type and context.
 *
 * @author Matthew MacKenzie (matt@yellowdragonsoft.com)
 * @version $Id:$
 */
import org.jdom.Element;
import org.jdom.Document;
import org.jdom.Namespace;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.util.*;
import java.io.*;

public class GetRepresentation
{
          public static final String DATA ELEMENT NAMESPACE
                              = "http://ns.cpsin.org/data-element/1.0";
          private static final String XSD NAMESPACE
                              = "http://www.w3.org/2001/XMLSchema";
          private InputStream xmlStream;
          private Element rootNode;
          private List representations;
          private static final Namespace DATA ELEMENT NS
                              =
Namespace.getNamespace(GetRepresentation.DATA ELEMENT NAMESPACE);
          private static final Namespace XSD NS
                              = Namespace.getNamespace(GetRepresentation.XSD NAMESPACE);

          private static final String REPRESENTATIONS TAG = "Representations";
          private static final String REPRESENTATION TAG = "Representation";
```

```
	/**
	 * Constructor.  Argument is an input stream of the DataElement XML to be
parsed.
	 */
	public GetRepresentation(InputStream xmlStream)
	{
		this.xmlStream = xmlStream;
	}

	/**
	 * Retrieves a representation give type and context.  If the
	 * type matches XSD NAMESPACE, the bare XSD is returned, otherwise
	 * the whole Representation element is returned.  If nothing exists for
	 * the given parameters, null is returned.
	 */
	public Element retrieveRepresentation(String type, String context)
		throws JDOMException, IOException
	{
		if (this.rootNode == null)
		{
			if (this.xmlStream == null)
				throw new IOException("XML Stream is null!");

			this.rootNode = new
SAXBuilder().build(this.xmlStream).getRootElement();

			if
(!this.rootNode.getNamespace().equals(GetRepresentation.DATA ELEMENT NS))
				throw new JDOMException("Root node is not in the
right namespace (" +

	GetRepresentation.DATA ELEMENT NAMESPACE + ")");

		}

		if (this.representations == null)
		{
			Element repsXml =
this.rootNode.getChild(GetRepresentation.REPRESENTATIONS TAG,


			this.representations =
repsXml.getChildren(GetRepresentation.REPRESENTATION TAG,


		System.out.println("Found " + this.representations.size()
				+ " representation(s) of this data element."
				+ "  The one below is for context: "
				+ context);

			if (this.representations == null)
				return null;
		}

		Iterator repIter = this.representations.iterator();

		while (repIter.hasNext())
		{
			Element rep = (Element)repIter.next();
			if (rep.getAttributeValue("type").equals(type) &&
				rep.getAttributeValue("context").equals(context))
			{
				if (type.equals(GetRepresentation.XSD NAMESPACE))
				{
					return rep.getChild("schema",
GetRepresentation.XSD NS).detach();
				}
				return rep.detach();
			}
		}
		return null;

	}

	public static void main(String[] args)
		throws Exception
	{
		if (args.length < 3)
		{
```

```
                                System.out.println("USAGE: GetRepresentation <file> <rep
type> <context>");
                                System.exit(-1);
                        }
                        GetRepresentation gr = new GetRepresentation(new
FileInputStream(args[0]));
                        Element rep = gr.retrieveRepresentation(args[1], args[2]);
                        new XMLOutputter().output(rep, System.out);
                }
}
```

8.1 Binding between instance data and metadata

If the CPSIN IJI data dictionary is to be placed into an ebXML Registry, there is a binding that must be present between instance data and the Data Dictionary to facilitate discovery of metadata by agencies that consume the data as part of services they may invoke. The binding must be both at design time (via an *Assembly Document*) and at run time (via a reference between individual data elements and the BIE's)

The concept of Assembly Document will be discussed at a later date in section 7.0

For the runtime instance data binding, XML 1.0 provides a good mechanism to use for a runtime instance data binding requirement - called a FIXED attribute.

RECOMMENDATION: That a UUID be used as a fixed attribute to each data element in instance data. The UUID must be in a form that can transmit the following information:

1. Present the user a location where they can retrieve the metadata object associated with the instance data elements
2. Present the user with the protocol to be used for retrieving the metadata and
3. Present the user with a Universally Unique Identifier in order to:
    a. recognize other instances of that metadata concept and
    b. retrieve a copy of one and one only metadata object from the registry.

By affixing the UUID as a fixed value attribute, XML instance data that is subject to a parse to check for well formed-ness will have the UUID's present and not incur any subsequent lag in runtime in-efficiency. A validating parse against either an XML schema or an XML DTD will result in the FIXED attribute being attached to the in-memory serialization of the XML instance and will be available to application and human actors.

This should be formalized and will not be difficult to do.

RECOMMENDATION: se of URN's in the format of a backwards URL with a UUID affixed in order to convey all the requirements.

9.0 Sample XML Expression of IJI Data Element as CC's/BIE's

Based on the data model and an interpretation of the ISO/IEC 11179 and ebXML Registry models, the following sample illustrates how an XML serialization of the Data element model may be specified. A schema has been developed for this model and is attached hereto as Appendix "A".

```
<?xml version="1.0" encoding="UTF-8" ?>
<DataElement
```

```
      home="http://ebxml.pwgsc.gc.ca:8080"
      id="urn:uuid:6e60580b-4538-2615-0c2c-3e034c430445"
      xmlns="http://ns.cpsin.org/data-element/1.0"
      >
<Identifiers>
      <Identifier type="responsibleOrgURL"
             value="http://www.sgc.gc.ca/iji-iij/CPSIN_ITS_e.asp" />
      <Identifier type="ElementIdentifier" value="014" />
      <Identifier type="DataDictionaryName"
                  value="Gender.Identifier"
                  xml:lang="en-CA" />
      <Identifier type="EntityName"
                  value="Being, Gender" />
</Identifiers>

<Properties assertedBy="Canadian Public Safety Information Network">
      <Property name="version.major" value="1" />
      <Property name="version.minor" value="0" />
      <Property name="version.incremental" value="0" />
      <Property name="registration.status" value="APPROVED" />
      <Property name="domain" value="dDateValue" />
       <property name=”context” value=”IJI Context” />
      <Property name="topic" value="Person" />
      <Property name="familiar.name"
                value="Person.Gender.Identifier"
                context="A numeric value corresponding to the
                         gender which a person belongs" />
      <Property name="synonyms"
                value="Animal.Gender.Identifier"
                context="A numeric value corresponding to the
                gender which an Animal belongs" />
 </Properties>

<Documentations>
       <Documentation type="comment|note|instruction|other"
                      locale="en_CA"
                      mimeType="text/html">
      <![CDATA[<html><body>Element Approved but further
               research needed for values</body></html>]]>
       </Documentation>
       <Documentation type="comment|note|instruction|other"
                      locale="fr_CA"
                      mimeType="text/html">
      <![CDATA[<html><body>viva la difference!</body></html>]]>
      </Documentation>
<Documentations>

<Representations>
      <Representation type="http://www.w3.org/2001/XMLSchema"
        context="urn:uuid:4a593056-3509-0766-2e7b-4e154030423f">
      <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="urn:component-foo">
      <!--schema here-->

      <xsd:element name="Sex">
        <xsd:complexType>
          <xsd:simpleContent>
```

```
              <xsd:extension base="xsd:string">
                <xsd:attribute name="value" use="required">
                  <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                      <xsd:enumeration value="01 - Male"/>
                      <xsd:enumeration value="02 - Female"/>
                      <xsd:enumeration value="03 - Asexual"/>
                      <xsd:enumeration value="04 - Transgendered - in
transition"/>
                      <xsd:enumeration value="05 - Transgendered -
complete to female"/>
                      <xsd:enumeration value="06 - Transgendered -
complete to male"/>
                      <xsd:enumeration value="07 - Hemaphrodyte"/>
                      <xsd:enumeration value="08 - Unisexual Species"/>
                      <xsd:enumeration value="09 - Not applicable"/>
                      <xsd:enumeration value="10 - Other"/>
                    </xsd:restriction>
                  </xsd:simpleType>
                </xsd:attribute>
                <xsd:attribute name="uuid" use="fixed" >
                  <xsd:simpleType>
                    <xsd:restriction base="xsd:string">
                      <xsd:enumeration value="urn:uuid:6e60580b-4538-
2615-0c2c-3e034c430445" />
                    </xsd:restriction>
                  </xsd:simpleType>
                </xsd:attribute>
              </xsd:extension>
            </xsd:simpleContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="GenderIdentifier">
          <xsd:complexType/>
        </xsd:element >
        </xsd:schema>
        </Representation>

        <!--Start of another context here-->
        <Representation type="http://www.w3.org/2001/XMLSchema"
context="urn:uuid:6563671c-5008-464c-5b38-1377054b5a7a">
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="urn:component-foo">
        <!--schema here-->

        <xsd:element name="SexIdentifier">
          <xsd:complexType>
            <xsd:simpleContent>
              <xsd:extension base="xsd:string">
                <xsd:attribute name="value" use="required">
                  <xsd:simpleType>
                    <xsd:restriction base="xsd:NMTOKEN">
                      <xsd:enumeration value="01 - Homme"/>
                      <xsd:enumeration value="02 - Femme"/>
                      <xsd:enumeration value="03 - Asexual"/>
                      <xsd:enumeration value="04 - Transgendered - dans
la transition"/>
```

```
                    <xsd:enumeration value="05 - Transgendered -
accomplissez a la femme"/>
                    <xsd:enumeration value="06 - Transgendered -
accomplissez a la homme"/>
                    <xsd:enumeration value="07 - Hemaphrodyte"/>
                    <xsd:enumeration value="08 - Esp&#233;ce
D'Unisexual"/>
                    <xsd:enumeration value="09 - Non applicable"/>
                    <xsd:enumeration value="10 - Autre"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
              <xsd:attribute name="uuid" use="fixed" >
                <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="urn:uuid:6e60580b-4538-
2615-0c2c-3e034c430445" />
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:attribute>
            </xsd:extension>
          </xsd:simpleContent>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="GenderIdentifier">
        <xsd:complexType/>
      </xsd:element >
      </xsd:schema>
      </Representation>


  </Representations>

</DataElement>
```

10.0 – Recommended Future Work

There are several next steps that should be taken to keep this work going in the correct direction. The below is a partial list of recommendations:

10.1 Circulate the XML Schema for the Data Element

This will involve ensuring the rules for representation from the IJI data dictionary are captured and expressed properly. Additional input from appropriate standards groups would also be a good idea.

10.2 Complete an XML schema for a Context Declaration

Seek input and finalize a v1.0 schema for expressing Context. Seek input from appropriate UN/CEFACT groups.

10.3 Complete an XML schema for the Assembly Guide

10.4 IJI Data Dictionary placed into Registry

Complete loading of the IJI data dictionary into a permanent registry/repository in the proper XML format

10.5 Build a prototype application

It would be a good exercise to build a sample application that will allow schema designers to design and generate XML schemas automatically based on content in the registry and the context in which they will use it.

10.6 Reconcile this work with the CCRIM work

[TBD]

10.7 Reconcile this work with the UN/CEFACT ATG Core Components work

[TBD]

10.8 Circulate this work to standards groups to seek additional input.

[TBD]

10.9 Harmonize the CPSIN IJI data dictionary with at least one other Data dictionary

[TBD] Using the registry for this.

Appendix “A” – XSD Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Author: Yellow Dragon Software Corporation (Matthew MacKenzie,
Duane Nickull) -->


<xs:schema targetNamespace="http://ns.cpsin.org/data-element/1.0"
xmlns:de="http://ns.cpsin.org/data-element/1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="DataElement">
    <xs:annotation>
      xs:documentation>Specification of DataElement.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Identifiers">
          <xs:annotation>

        <xs:documentation>Collection element to hold 1-unbounded
Indentifier instances.</xs:documentation>
                                      </xs:annotation>
                                      <xs:complexType>
                                             <xs:sequence>
                                                    <xs:element
name="Identifier" maxOccurs="unbounded">

        <xs:annotation>

        <xs:documentation>Simple type/value element representing a piece
of information which canonically identifies data
element.</xs:documentation>

        </xs:annotation>

        <xs:complexType>

        <xs:attribute name="type" type="de:IdentifierTypes"
use="required"/>

        <xs:attribute name="value" type="xs:string" use="optional"/>

        </xs:complexType>
                                                           </xs:element>
                                             </xs:sequence>
                                      </xs:complexType>
                               </xs:element>
                               <xs:element name="Properties"
minOccurs="0">
                                      <xs:annotation>

        <xs:documentation>Collection element to hold 1-unbounded
Property instances.</xs:documentation>
                                      </xs:annotation>
```

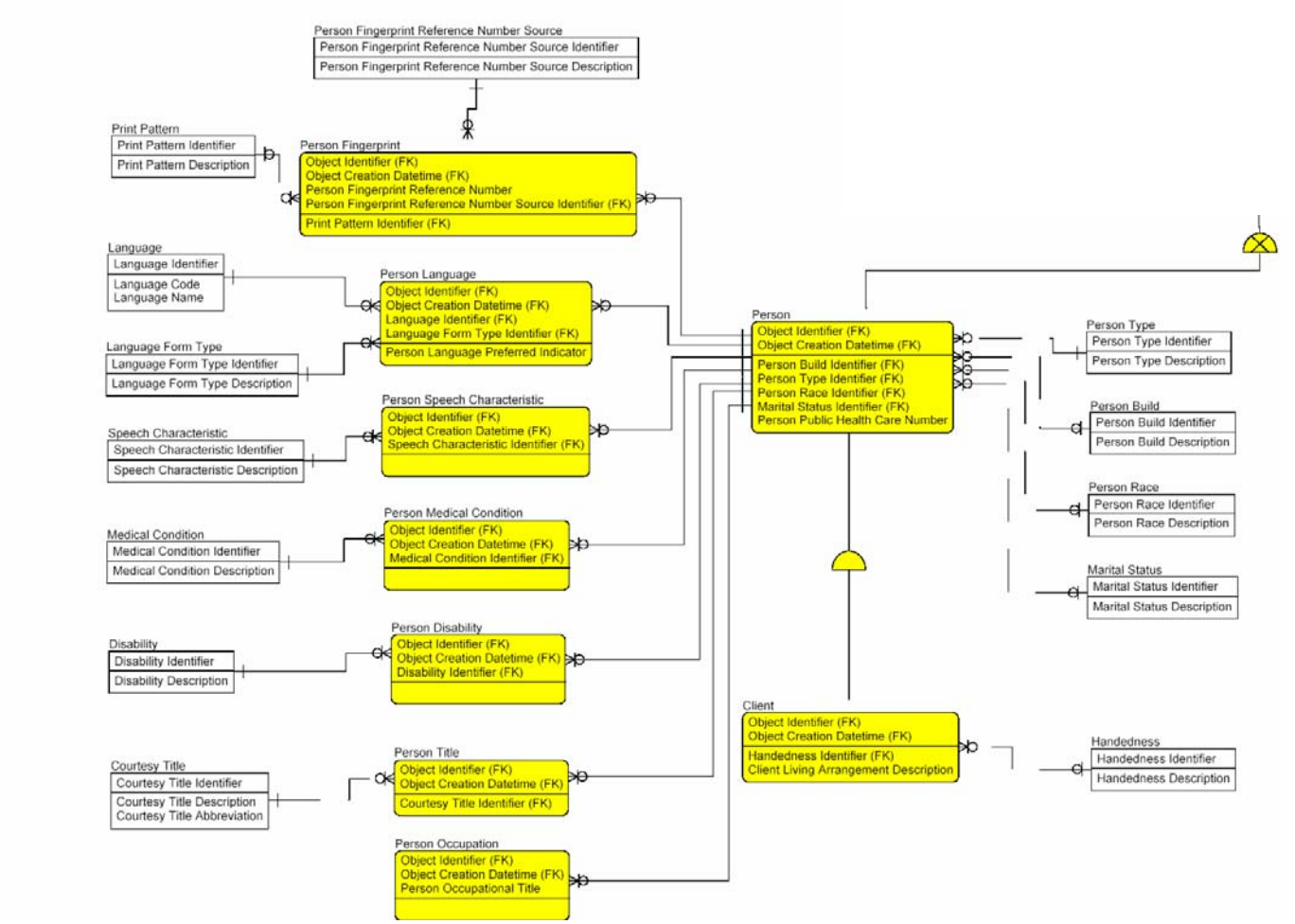
```
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element
name="Property" maxOccurs="unbounded">

        <xs:annotation>

        <xs:documentation>Element providing metadata storage for a
DataElement.  Property types are added to the PropertyTypes simpleType
in the schema, allowing extensibility of metadata without structural
changes to the overall schema.</xs:documentation>

        </xs:annotation>

        <xs:complexType>

        <xs:attribute name="name" type="de:PropertyNames"
use="required"/>

        <xs:attribute name="value" type="xs:string" use="optional"/>

        <xs:attribute name="context" type="xs:string" use="optional"/>

        </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                                <xs:attribute
name="assertedBy" type="xs:string" use="optional"/>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="Documentation"
minOccurs="0">
                                        <xs:annotation>

        <xs:documentation>Collection element to hold documentation
entries.  Entries can be differentiated by locale, type and
mimeTYpe.</xs:documentation>
                                        </xs:annotation>
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element
name="Entry" maxOccurs="unbounded">

        <xs:annotation>

        <xs:documentation>An entry in this DataElement's
documentation.</xs:documentation>

        </xs:annotation>

        <xs:complexType>

        <xs:simpleContent>

        <xs:extension base="xs:string">

        <xs:attribute name="locale" type="de:Locales"/>
```

```
        <xs:attribute name="type" type="de:DocumentationTypes"/>

        <xs:attribute name="mimeType" type="de:DocumentationMimeTypes"
use="optional" default="text/plain"/>

        </xs:extension>

        </xs:simpleContent>

        </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                                <xs:element name="Representations">
                                        <xs:annotation>
                                                <xs:documentation>List of
representations of this DataElement.  A representation is essentially a
schema in any format imaginable, although it is suggested that an XML
schema format is used.</xs:documentation>
                                        </xs:annotation>
                                        <xs:complexType>
                                                <xs:sequence>
                                                        <xs:element
name="Representation" maxOccurs="unbounded">

        <xs:annotation>

        <xs:documentation>A structure Representation of this
DataElement.  It is recomended that the type attribute of this element
be the namespace value for the schema language being used.  For xsd,
that would be http://www.w3.org/2001/XMLSchema.</xs:documentation>

        </xs:annotation>

        <xs:complexType>

        <xs:simpleContent>

        <xs:extension base="xs:string">

        <xs:attribute name="type" type="xs:string"/>

        <xs:attribute name="context" type="xs:string"/>

        </xs:extension>

        </xs:simpleContent>

        </xs:complexType>
                                                        </xs:element>
                                                </xs:sequence>
                                        </xs:complexType>
                                </xs:element>
                        </xs:sequence>
```

```
                        <xs:attribute name="home" type="xs:anyURI"
use="required"/>
                        <xs:attribute name="id" type="xs:string"
use="required"/>
                </xs:complexType>
        </xs:element>
        <xs:simpleType name="IdentifierTypes">
                <xs:annotation>
                        <xs:documentation>List of acceptable Identifier
types.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="responsibleOrgURL"/>
                        <xs:enumeration value="submittingOrgURL"/>
                        <xs:enumeration value="ElementIdentifier"/>
                        <xs:enumeration value="DataDictionaryName"/>
                        <xs:enumeration value="EntityName"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="PropertyNames">
                <xs:annotation>
                        <xs:documentation>List of property names in
use.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="version.major"/>
                        <xs:enumeration value="version.minor"/>
                        <xs:enumeration value="version.incremental"/>
                        <xs:enumeration value="registration.status"/>
                        <xs:enumeration value="domain"/>
                        <xs:enumeration value="topic"/>
                        <xs:enumeration value="familiar.name"/>
                        <xs:enumeration value="synonyms"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="Locales">
                <xs:annotation>
                        <xs:documentation>List of acceptable locales.
Used primarily for choosing an appropriate culture for Documentation
Entries.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="en_CA"/>
                        <xs:enumeration value="en"/>
                        <xs:enumeration value="en_GB"/>
                        <xs:enumeration value="en_US"/>
                        <xs:enumeration value="fr_FR"/>
                        <xs:enumeration value="fr"/>
                        <xs:enumeration value="fr_CA"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DocumentationMimeTypes">
                <xs:annotation>
                        <xs:documentation>List of recognized
documentation formats.  Remember to use CDATA when doing anything other
than text/plain.</xs:documentation>
                </xs:annotation>
```

```
                <xs:restriction base="xs:string">
                        <xs:enumeration value="text/plain"/>
                        <xs:enumeration value="text/html"/>
                        <xs:enumeration value="text/xml"/>
                        <xs:enumeration value="application/pdf"/>
                        <xs:enumeration value="application/ms-word"/>
                        <xs:enumeration value="text/rtf"/>
                        <xs:enumeration value="application/octet-
stream"/>
                </xs:restriction>
        </xs:simpleType>
        <xs:simpleType name="DocumentationTypes">
                <xs:annotation>
                        <xs:documentation>List of documentation
types.</xs:documentation>
                </xs:annotation>
                <xs:restriction base="xs:string">
                        <xs:enumeration value="comment"/>
                        <xs:enumeration value="note"/>
                        <xs:enumeration value="instruction"/>
                        <xs:enumeration value="other"/>
                        <xs:enumeration value="warning"/>
                        <xs:enumeration value="copyright"/>
                        <xs:enumeration value="restrictions"/>
                        <xs:enumeration value="description"/>
                        <xs:enumeration value="abstract"/>
                </xs:restriction>
        </xs:simpleType>
</xs:schema>
```

Appendix "B" – IJI Object Model for "Person"



Appendix "C" – Java Code for Extracting BIE's from Data Elements.

```
/**
 * This class will return a representation of a DataElement
 * given type and context.
 *
 * @author Matthew MacKenzie (matt@yellowdragonsoft.com)
 * @version $Id:$
 */
import org.jdom.Element;
import org.jdom.Document;
import org.jdom.Namespace;
import org.jdom.JDOMException;
import org.jdom.input.SAXBuilder;
import org.jdom.output.XMLOutputter;

import java.util.*;
import java.io.*;

public class GetRepresentation
{
	public static final String DATA_ELEMENT_NAMESPACE
			= "http://ns.cpsin.org/data-element/1.0";
	private static final String XSD_NAMESPACE
			= "http://www.w3.org/2001/XMLSchema";
	private InputStream xmlStream;
	private Element rootNode;
	private List representations;
	private static final Namespace DATA_ELEMENT_NS
			=
Namespace.getNamespace(GetRepresentation.DATA_ELEMENT_NAMESPACE);
	private static final Namespace XSD_NS
			= Namespace.getNamespace(GetRepresentation.XSD_NAMESPACE);

	private static final String REPRESENTATIONS_TAG = "Representations";
	private static final String REPRESENTATION_TAG = "Representation";


	/**
	 * Constructor.  Argument is an input stream of the DataElement XML to be
parsed.
	 */
	public GetRepresentation(InputStream xmlStream)
	{
		this.xmlStream = xmlStream;
	}

	/**
	 * Retrieves a representation give type and context.  If the
	 * type matches XSD_NAMESPACE, the bare XSD is returned, otherwise
	 * the whole Representation element is returned.  If nothing exists for
	 * the given parameters, null is returned.
	 */
	public Element retrieveRepresentation(String type, String context)
		throws JDOMException, IOException
	{
		if (this.rootNode == null)
		{
```

```
                    if (this.xmlStream == null)
                          throw new IOException("XML Stream is null!");

                    this.rootNode = new
SAXBuilder().build(this.xmlStream).getRootElement();

                    if
(!this.rootNode.getNamespace().equals(GetRepresentation.DATA_ELEMENT_NS))
                          throw new JDOMException("Root node is not in the
right namespace (" +

      GetRepresentation.DATA_ELEMENT_NAMESPACE + ")");

            }

            if (this.representations == null)
            {
                    Element repsXml =
this.rootNode.getChild(GetRepresentation.REPRESENTATIONS_TAG,

              GetRepresentation.DATA_ELEMENT_NS);

                    this.representations =
repsXml.getChildren(GetRepresentation.REPRESENTATION_TAG,

         GetRepresentation.DATA_ELEMENT_NS);

                    System.out.println("Found " + this.representations.size()
                                                      + " representation(s) of
this data element.");

                    if (this.representations == null)
                          return null;
            }

            Iterator repIter = this.representations.iterator();

            while (repIter.hasNext())
            {
                    Element rep = (Element)repIter.next();
                    if (rep.getAttributeValue("type").equals(type) &&
                          rep.getAttributeValue("context").equals(context))
                    {
                          if (type.equals(GetRepresentation.XSD_NAMESPACE))
                          {
                                return rep.getChild("schema",
GetRepresentation.XSD_NS).detach();
                          }
                          return rep.detach();
                    }
            }
            return null;

      }

      public static void main(String[] args)
            throws Exception
      {
            if (args.length < 3)
            {
                    System.out.println("USAGE: GetRepresentation <file> <rep
type> <context>");
                    System.exit(-1);
```

```
            }
            GetRepresentation gr = new GetRepresentation(new
FileInputStream(args[0]));
            Element rep = gr.retrieveRepresentation(args[1], args[2]);
            new XMLOutputter().output(rep, System.out);
      }
}
```