POSTSECONDARY ELECTRONIC STANDARDS COUNCIL

# XML Technical Specification for Higher Education

Version 2.11

**Deleted:** 0

A publication of the

**Postsecondary Electronic Standards Council**

**Washington, DC**

**August 2002**

**Deleted:** June

**Working Draft**
**Last edited by Mike Rawlins**

(this page is intentionally left blank)

# XML Technical Specification for Higher Education

## Table of Contents

1 Introduction .......... 1

1.1 Purpose and Scope .......... 1

1.2 Intended Audience .......... 1

2 XML Forum Work Products .......... 2

2.1 General Guidelines .......... 2

2.1.1 General Naming Conventions .......... 2

2.1.2 Versioning Scheme .......... 2

2.1.3 URI, URL, File, and Directory Structure .......... 3

2.2 Core Components .......... 5

2.2.1 Metadata essential for XML syntax .......... 5

2.2.1.1 Data Types .......... 5

2.2.1.2 Aggregate Items .......... 6

2.2.1.3 Spreadsheet Organization and Columns .......... 7

2.2.1.4 Analysis Orientation .......... 8

2.2.2 Core Component Naming Conventions .......... 8

2.3 Best Practices .......... 9

2.3.1 General Design Considerations .......... 9

2.3.2 Schema vs. DTD .......... 9

2.3.3 Use of Elements vs. Attributes .......... 10

2.3.4 Element vs. Type .......... 13

2.3.5 Hide vs. Expose Namespaces .......... 18

2.3.6 Local vs. Global .......... 22

2.3.7 Namespaces - Zero, One or Many .......... 24

2.3.8 Variable Content Containers .......... 27

2.3.9 Nulls, Zeroes, Spaces, and Absence of Data .......... 27

2.3.10 Other Considerations .......... 29

3 XML Schema Development Roadmap .......... 29

4 Implementation Recommendations .......... 31

4.1 Why this section? .......... 31

4.2 General Requirements .......... 31

4.3 Message Handling .......... 33

4.3.1 Requirements .......... 33

4.3.2 Options and Near Term Recommendations .......... 33

4.3.3 Long term recommendations .......... 37

4.4 Security .......... 37

4.4.1 Requirements .......... 37

4.4.1.1 General Considerations .......... 37

4.4.1.2 Security Considerations .......... 38

4.4.1.3 Countermeasures and Remediation Strategies .......... 39

4.4.1.4 Encryption .......... 40

4.4.2 Near term Recommendations .......... 41

4.4.3 Long term recommendations .......... 41

4.5 Registries and Repositories .......... 42

4.5.1 Requirements .......... 42

4.5.2 Near term Recommendations .......... 43

4.5.3 Long term recommendations .......... 43

4.6 Electronic Trading Partner Agreements .......... 43

4.6.1 Requirements .......... 43

**Deleted:** 4
**Deleted:** 4
**Deleted:** 4
**Deleted:** 5
**Deleted:** 6
**Deleted:** 7
**Deleted:** 7
**Deleted:** 8
**Deleted:** 8
**Deleted:** 8
**Deleted:** 9
**Deleted:** 12
**Deleted:** 17
**Deleted:** 21
**Deleted:** 23
**Deleted:** 26
**Deleted:** 26
**Deleted:** 28
**Deleted:** 28
**Deleted:** 30
**Deleted:** 30
**Deleted:** 30
**Deleted:** 32
**Deleted:** 32
**Deleted:** 32
**Deleted:** 36
**Deleted:** 36
**Deleted:** 36
**Deleted:** 36
**Deleted:** 37
**Deleted:** 38
**Deleted:** 39
**Deleted:** 40
**Deleted:** 40
**Deleted:** 41
**Deleted:** 41
**Deleted:** 42
**Deleted:** 42
**Deleted:** 42
**Deleted:** 42

4.6.2 Near term Recommendations ........ 44

4.6.3 Long term recommendations ........ 44

**5 Reference Documents & Standards ........ 44**

5.1 Terms ........ 44

5.2 Acronyms ........ 45

**Deleted:** 43

**Deleted:** 43

**Deleted:** **43**

**Deleted:** 43

**Deleted:** 44

## Development of the XML Technical Specification

This specification is an ongoing output of the Technology Work Group of the XML Forum for Education. First organized in August 2000 on the recommendation of a PESC study group, the XML Forum has as its mission the establishment of Extensible Markup Language (XML) standards for the education community through collaboration. The Technology Work Group was charged with performing research on existing XML specifications and best practices and providing technical guidance to XML developers in the education space. This document is the result of its efforts over the past eighteen months. It will be updated periodically as national and international XML standards are established.

Michael Rawlins, Principal Consultant of Rawlins EC Consulting, collaborated with the Technology Work Group, adding to the process his experience in standards-setting bodies and knowledge of XML. Mike has over 15 years of experience as a technical consultant in information systems. He is vice chair of ANSI ASC X12 Subcommittee C on Communications and Controls and co-chairs X12C's Future Architecture Task Group, which is responsible for technical aspects of X12's work on XML. He participated in the ebXML effort, serving on the steering committee and leading the Requirements Project Team. Mike has a Masters of Science degree in Computer Science from the University of Texas at Dallas.

Although representatives of the IMS Global Learning Consortium, University of Wisconsin-Madison, Miami-Dade Community College, Brown University, and the US Department of Education were important members of the work group, several work group members deserve special recognition for their contributions to this document. Bruce Marton of the University of Texas-Austin and Chair of the XML Forum Architecture Committee provided guidance and perspective during development of this version. Karl Van Neste of the College Board has served as chair of the Technology Work Group and provided leadership and expertise to initial efforts. Steve Margenau of Great Lakes Educational Loan Services has chaired the Technology Workgroup and provided research and recommendations for key sections of the document. Richard Driscoll and others at Datatel provided review and editorial assistance in the publication of the document.

The Postsecondary Electronic Standards Council
One Dupont Circle, NW, Suite 520
Washington, DC 20036
(202) 293-7383
http://ww.StandardsCouncil.org

(this page is intentionally left blank)

## 1 Introduction

This specification was developed by members of the XML Forum for Education's Technology Work Group in consultation with its technical advisor. The purpose of this specification is to help guide the work of the XML Forum, providing recommendations to inform decisions that face the following groups:

- The Core Components Work Group in the development and maintenance of a data dictionary and data models in conjunction with the Technology Work Group
- The Technology Work Group in the development of schemas based on the data models
- The XML Forum, as an organization, as its structure changes to meet the needs of the higher education community
- The higher education community as it implements XML message data exchanges

This specification is a living document – it is expected to change and evolve with XML and its related standards.

The development of this specification served to clarify, for the XML Forum, the most efficient work processes and the ultimate deliverables of the standing and ad hoc work groups of the XML Forum.

Every effort was made to build on the experience and work done previously by other standards organizations within and outside of Higher Education: W3C, ebXML, IFX, X12, CommonLine, IMS, IEEE, and ISO, among others.

Keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in the Internet Engineering Task Force (IETF) Request for Comments (RFC) 2119.

### *1.1 Purpose and Scope*

The purpose of this document is to provide guidance in the development and maintenance of a data dictionary and XML sschemas. The scope of this specification includes the data which institutions and their partner's exchange in support of the existing business processes within Higher Education like administrative applications for student financial aid, admissions, and registrar functions.

### *1.2 Intended Audience*

The internal audience of this document is the members of the XML Forum for Education as well as the technical members of the education community at large wishing to use XML in their data exchanges.

# 2 XML Forum Work Products

## *2.1 General Guidelines*

### 2.1.1 General Naming Conventions

The following recommendations by the XML Forum's Technology Group for general conventional standards are used whenever possible.

- *Upper Camel Case (UCC)* SHALL be used. *UCC* style capitalizes the first character of each word and compounds the name following the conventions of the ebXML Technical Architecture v1.0.4, section 4.3:
- Acronyms SHOULD be avoided, but in cases where they are used, the capitalization SHALL remain.
  (*example*: XMLSignature).
- Underscore ( _ ), periods ( . ) and dashes ( - ) MUST NOT be used.

  | | | |
  |---|---|---|
  | (*examples*: | use HeaderManifest, | not Header.Manifest; |
  | | use StockQuote5, | not Stock_Quote_5; |
  | | use CommercialTransaction | not Commercial-Transaction.) |

- XML "type" names SHALL have "Type" appended to them.
  (*example*: type="NameType")
- Schema names adhere to the following conventions.

  1. Schema document names (the root element of a schema) SHALL be based on the business purpose of the document.

  2. Schema names that support the data dictionary SHALL be based on the category of definitions in that schema

  3. Schema physical file names SHALL be the same as the schema name, with a ".xsd" extension.

  4. Schema names SHALL remain constant across all versions.

**NOTE**: A list of acronyms used in this document can be found in section 5.2.

### 2.1.2 Versioning Methodology

The initial approved set of XML Forum schemas SHALL be designated 1.0. New versions, developed primarily for maintenance purposes or the inclusion of new documents, SHALL be deemed minor releases and incremented by .1. Major releases SHALL be incremented by 1.0. Major releases SHALL be designated under such circumstances as the following.

- Several new documents are developed
- Major additions are made to the data dictionary
- Changes to file, URL, or namespace schemes

- Changes in schema design approach

Versions SHALL be named by a four-character string formed by two digits indicating the major version followed by two digits for the minor version, using leading zeroes. Separate URLs, URIs, and directories SHALL be used for each version. Each schema SHALL have an attribute in the root element of PESCXMLVersion.

2.1.3 URI, URL, File, and Directory Structure

The base URI for namespaces in XML Forum schemas SHALL be http://schemas.PESCXML.org. This URI SHALL also be valid as the base URL for the network location of the XML Forum schemas and associated files. The version string MUST be appended to this base URI to form the URI relevant to the version.
(*example*: Version 1.0 has the URI http://schemas.PESCXML.org/0100)

The Forum plans to make several different types of files available on its web site. A brief description of the categories and their URL/URI specifications is listed below. Succeeding subsections specify further details. All path names below are located under the base version URL path.

- Schema files - This is the largest category of component and is further broken down in succeeding subsections. Schema files are located in the xsd path.
- XSLT Stylesheets - Although not "standard" work products of the forum, the Forum intends to provide here a facility for sharing and distributing commonly used transformation stylesheets. XSLT stylesheets are located in the xsl path.
- Sample instance documents - As will be described later, there are several sample instance documents per business document schema. These are located in the xmlExamples path.
- Documentation - Is located in the docs path.

For each area, the path has sub-paths for core and sectors (to be described shortly).

**Formatted:** Bullets and Numbering

### 2.1.3.1 Core Data Dictionary

**Deleted:** ¶

**Formatted:** Bullets and Numbering

Paths - core and baseTypes
Root files and URLs: xsd/core/coreMain.xsd and xsd/baseTypes/baseTypesMain.xsd

Core - The Core Components team, in a common "Core" data dictionary, SHALL define all aggregates and their maximum universe of member elements. This Core data dictionary SHALL be represented by a several schemas, divided into groups of related items as will be described in a later section. Names of these

**Deleted:** single

**Deleted:** .

**Deleted:** The root URI for Version 1.0 of the Core schema, for example, has the root namespace of "http://schemas/PESCXML.org/0100/core" and is associated with the namespace identifier "core".

schemas are derived from reasonable names assigned by the Core Components team to the groups.  Such files are xsd:included into coreMain.xsd.

BaseTypes - Base types represent re-usable "leaf level" data elements that are either simpleTypes or complexTypes (if they have aggregates).  They do not have child elements.  Base types are derived from standard W3C Schema data types by extension or restriction.   Examples of base types are numerics with range or sign restriction, strings with length restrictions, and enumerations. These may all be specified in the baseTypesMain.xsd schema file, or broken into groups of related items that are xsd:included into baseTypesMain.xsd.

Deleted: ¶

Formatted: Bullets and Numbering

### 2.1.3.2 Sector Dictionaries

Root files and URL path:  xsd/sectorName/sectorNameMain.xsd

In addition to the Core dictionary, the Core Components team MAY define one or more "Sector" data dictionaries.  A Sector dictionary applies to a defined functional sector of the postsecondary arena and supports requirements unique to that sector for aggregate membership, cardinality, pattern, and/or code values. A Sector dictionary SHALL be represented by a single schema. Sector content may be specified in the sectorNameMain.xsd schema file, or broken into groups of related items that are xsd:included in them.

Deleted: The root URI for version 1.0 of a Sector schema (named, for example, "SectorName") has the root namespace of "http://schemas.PESCXML.org/0100/ SectorName".

Deleted: ¶

Formatted: Bullets and Numbering

### 2.1.3.3 Instance Document Schemas

URL path:  xsd/sectorName/

Instance document definitions are declared under the appropriate Sector URI. Each instance document has a targetNamespace and schema file specific to that document.  The instance document schema imports the Core namespace and, where appropriate, the Sector namespaces.  For example. the root URI for version 1.0 of the Transcript schema within the Registrar and Administration Sector has the root namespace of "http://schemas.PESCXML.org/0100/RegAdmin/Transcript".

Deleted: , or under the Core URI

The instance document schema SHOULD have the basic structure of a root element with an anonomously defined complexType.  Each of it's first level children SHOULD be of a type specified in the Core or Sector library.  Where an instance document schema requires modifications to the types defined in either the sector or core, these modifications SHALL be expressed as named types following the declaration of the root element.  Such extended or restricted types shall reside in the namespace of the instance document schema.

NOTE:  The long term goal and strategy is to have as few such locally declared types as possible in instance document schemas and to migrate common content to sector and core libraries.

### *2.2 Core Components*

#### 2.2.1 Metadata essential for XML syntax

To facilitate creation of schemas, the following metadata items SHALL be recorded, but is not limited to, in the data dictionary for each element.

- Aggregate object name
- Element name
- Cardinality rules (
- Element description
- Element equivalence in other transaction(s)
- Data type (string, date, number, etc)
- Minimum length (NOTE:  May be specified in the Core and raised in the Sector.)
- Maximum length  (NOTE:  May be specified in the Core and lowered in the Sector.)
- Values of code elements

**Deleted:** these are OPTIONAL for aggregates in the core dictionary).

#### 2.2.1.1 Data Types

The following simplified list of datatypes SHALL be used for core component analysis, instead of the full set supported by XML schemas.  Each type has several OPTIONAL attributes that MAY be specified, as needed, for a particular data item.

- *Number* - precision (number of decimal places), minimum value, maximum value
- *String* (as defined by the W3C in *XML Schema Part 2: Datatypes*) - min length, max length, and pattern facets (such as NNN-NN-NNNN for Social Security Numbers).  Patterns, if used, MUST be specified using a regular expression language as defined by the W3C in *XML Schema Part 2: DataTypes Regular Expressions*.  Pattern facets may be specified in the Core, and restricted in a Sector.  If an element contains a member of a list, all potential list values MUST be specified (this resolves the issue with coded fields).

  **NOTE**:  If a string item is specified as mandatory in an aggregate item, it is RECOMMENDED to have a minimum length of 1.
- *Date*
- *Time*
- *DateTime*
- *Boolean*  - 0,1,true,false

When a data item is defined, it MUST be assigned a type from this set.  The attributes listed SHOULD be used to place restrictions on the allowed values.  If the attributes are not listed in the data item's definition, then there are no restrictions beyond the general restrictions implied by the datatype.

### 2.2.1.2 Aggregate Items

#### 2.2.1.2.1 Specification of Aggregates

Aggregate data items are composed of two or more data items. For aggregates the following apply.

- The included elements MUST be specified in sequence.The core dictionary SHOULD specify the maximum universe of included elements.
- Sector dictionaries MAY restrict included elements, and MAY add additional elements.
- Cardinality (how many times an included element may occur in the aggregate) SHALL be specified for aggregates in the core dictionary. The widest common range of cardinality shall be expressed in the Core. The cardinality of elements within aggregates in the Core is defined as that which is most applicable to the widest range of uses, with a goal of minimizing the need for modification in sector or document schemas. The defaults in most cases will be 0..1 or 1..1). Cardinality MAY be specified for aggregates in sector dictionaries and business document schemas. When creating a business document schema, cardinality SHOULD NOT be broadened from that specified in the sector dictionary.
- The cardinality SHALL be expressed as l..u where l is the lower number of occurrences and u is the upper number of occurrences. A wild card of "*" SHALL be used to indicate no upper limit. (For example, a cardinality of 1..1 means that the data item is mandatory in the aggregate and can occur only once. 0..1 means that the data item is OPTIONAL, and can occur no more than once. 0..* means that it is OPTIONAL and if it does occur there are no limits on how many times it can occur. )

**Deleted:** OULD NOT

**NOTE**: It is RECOMMENDED that judicious consideration be given before specifying an item in an aggregate as mandatory (minimum cardinality of 1).

#### 2.2.1.2.2 Issues Concerning Aggregates

The following recommendations are made for addressing issues regarding aggregates:

- *Over-riding the cardinality of an item in an aggregate on a per document basis*
  (*example*: a street address is mandatory in a reissue but is not mandatory in an adjustment.)
  It is RECOMMENDED that this type of definition not be supported (at least in Version 1) since it makes defining reusable aggregates more complex. One RECOMMENDED approach is to define street address with a cardinality 0..2 in an "address" aggregate, but define address 1..1 in the reissue and 0..1 in the adjustment.

- _Conditional use of items in an aggregate_ – As in the case of X12 EDI, these are the relational conditions often imposed on elements in segments.
  (*examples*: Use "a" or "b" but not both;
  if "a" then use "b", else use "c".)
  It is RECOMMENDED that conditionals not be supported (at least in Version 1) since it adds complexity to the analysis and construction of the schemas. Use of such conditional restrictions and edits, not being supported in the schemas, SHALL be the responsibility of the business applications that use the data.

### 2.2.1.3 Spreadsheet Organization and Columns

#### 2.2.1.3.1 Organization

Analysis spreadsheets SHOULD be organized as follows:

| | |
|---|---|
| • *Basic* | A simple data item. |
| • *Aggregates* | A group of basic items or other aggregates, specified in sequence. If a basic item is not re-used, the full specification MAY appear within the aggregate rather than being specified on a separate line. |

#### 2.2.1.3.2 Columns

Columns SHOULD be organized as follows:

- Aggregate Name
- Name of included item. If an aggregate is included within an aggregate, only the name of the aggregate SHOULD be listed - not the names of all of its children
- Description
- Datatype
- Cardinality - The number of times the included item can appear in the aggregate
- For each basic item:
  - Minimum length - OPTIONAL (String Only)
  - Maximum length - OPTIONAL (String Only)
  - Pattern - OPTIONAL (String Only)
  - List of values - OPTIONAL (String Only)
  - Minimum value - OPTIONAL (Number Only)
  - Maximum value - OPTIONAL (Number Only)
  - Comments – (*example*: Code sets or source)
  - Sector Library. For a sector library spreadsheet, designates the item as new in the sector library, present in the core library but modified in the sector library, or present in the core library and referenced in the sector library.

**NOTE**: Some reusable basic items MAY not have an aggregate name.

#### 2.2.1.4 Analysis Orientation

It is RECOMMENDED that the data dictionary use the core components as "abstract" items or types rather than the full set of all particular items.
*(example*: a general "party" is defined rather than specifying "student", "lender", or "guarantor" separately.)
This approach enhances reusability and simplifies maintenance.

#### 2.2.1.5 Code Lists

- Short, two or three character codes SHALL be used where deemed appropriate by the Core Components team instead of longer, more fully described words or phrases.

- For code lists that are created and maintained by the XML Forum, permitted values SHALL be listed in the data dictionary and as documentation in document schemas. Schemas SHALL NOT be written so as to provide run-time schema validation of codes against the permitted values in the list. The justifications for these rules are that we do not wish implementations to be delayed by administrative and procedural delays in adding codes to schemas, and that business applications are likely to perform their own code value checking, making schema validation redundant.

- For code lists that are created and maintained by organizations other than the Forum, the Core Components team SHALL determine whether or not schema validation is to be supported. The team SHALL make this decision based on factors such as the stability of the code list, size of the code list, and copyright status. Schemas SHALL NOT import or include schemas from other organizations for the purpose of code list validation.

### 2.2.2 Core Component Naming Conventions

ebXML core component naming conventions (based on ISO 11179) SHALL be used for a XML Forum logical component. Names for elements MAY be modeled after the IFX Forum's name fragment combinations for XML tags. The IFX Forum's name fragments SHOULD be used wherever an appropriate match exists with an XML Forum element name. Where a match does not exist, the necessary fragments SHALL be created by the XML Forum team responsible for the data dictionary.

## *2.3 Best Practices*

### 2.3.1 General Design Considerations

The XML Forum schemas are oriented primarily toward data interchange. This does not preclude designing schemas that have another primary orientation, such as for presentation, but rather the primary focus is for data exchange. The schemas are therefore data oriented, although in some cases they may mirror paper business documents. For these reasons, the content model is oriented toward semantics (or "content") rather than presentation or structure (content model contains some degree of presentation orientation mixed with semantics).

### 2.3.2 Schema vs. DTD

The World Wide Web Consortium XML Schema Language recommendation SHALL be used to describe data instead of DTDs or BizTalk Schema (by Microsoft).

XML Schemas SHALL be used for the following reasons.

1. XML Schemas are supported by the W3C, ebXML, and other organizations.

2. XML Schemas support greater content and data type validation than DTDs.

3. XML Schemas are stable and reached the W3C Recommendation status as of May 2, 2001.

4. XML Schemas support open-ended data models (allow vocabulary extensions and inheritance); DTDs do not.

5. XML Schemas provide a rich core of base data types; DTDs do not.

6. XML Schemas support data types and data type reuse via object-oriented-like mechanisms; DTDs provide only limited support.

7. XML Schemas are well-formed XML documents; DTDs require an understanding of the SGML syntax.

Well developed XML Schemas can perform content checking that is largely unavailable in DTDs. Since content or data checking is a large component of many software development efforts, these efforts can be reduced with XML Schemas.

Tools like **XML Spy** (from Altova, http://www.xmlspy.com/) support XML Schemas and DTDs. A user can generate a "first cut" at an XML Schema based on a DTD and continue to maintain the content model. Due to the advanced type definitions that are available in XML Schemas, a user cannot maintain the content model when converting an XML Schema to a DTD.

BizTalk Schema (framework) works only with the BizTalk Server product. It uses a proprietary schema syntax (XDR) that is incompatible with W3C XML Schemas. Microsoft has promised to eventually support W3C XML Schemas.

### 2.3.3 Use of Elements vs. Attributes

In the majority of circumstances, elements SHALL be used in the design of XML Schemas that support data exchange in the PESC realm.

XML Forum Schemas are oriented towards data exchange in the support of existing and future transaction families and their accompanying data structures. Elements provide a method for defining and expressing structure within a document via the containment of child elements. They also provide a means for validating the document's structure. Additionally, a structure composed of elements is more extensible in the face of future changes; i.e., elements are supportive of change.

Attributes MAY be used when defining information that is intrinsic to an element, but not a part of that element. Attributes are akin to metadata; they are useful for information that describes an element, such as ID numbers, URLs, types, and other references. Attributes cannot be hierarchical, they cannot contain child attributes or elements, their order cannot be controlled and therefore, cannot provide structure.

All element and attribute forms are unqualified. Element and attribute names imported from another namespace into a schema document must have a namespace prefix. Local names of the schema target namespace do not require a namespace prefix. Only the root element of an instance document requires a namespace prefix.

To illustrate the appropriate use of elements and attributes, consider an office building with multiple floors. Each floor has multiple tenants. (see Example-1.xml and Example-1.xsd) The following illustrates an XML document representing that structure, using elements to represent the building (Building), floors (Floor), and tenants (Tenant). An attribute (LevelNumber) is used to identify each floor.

Example-1.xml - (*Use of Elements vs. Attributes*)

```
<?xml version="1.0"?>
<PESCXML:Building
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.pescxml.org
http://schemas.pescxml.org/Example-1.xsd">
  <Floor LevelNumber="1">
    <Tenant>Smith</Tenant>
    <Tenant>Jones</Tenant>
    <Tenant>Zoltan</Tenant>
  </Floor>
  <Floor LevelNumber="2">
    <Tenant>North</Tenant>
    <Tenant>South</Tenant>
    <Tenant>East</Tenant>
    <Tenant>West</Tenant>
  </Floor>
  <Floor LevelNumber="3">
    <Tenant>Wealthy</Tenant>
  </Floor>
</PESCXML:Building>
```

Example-1.xsd - (*Use of Elements vs. Attributes*)

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Building">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Floor" type="FloorType"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="FloorType">
    <xsd:sequence>
      <xsd:element name="Tenant" type="xsd:string"
        maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="LevelNumber" use="required"/>
  </xsd:complexType>
</xsd:schema>
```

While it is possible to represent the same structure using only elements (Example-2.xml and Example-2.xsd), the document structure is more complex and a little more difficult to understand. It makes a clearer design to have the LevelNumber an attribute of Floor, rather than a child of Floor.

Example-2.xml - (*Use of Elements vs. Attributes*)

```
<?xml version="1.0"?>
<PESCXML:Building
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation=
    "http://schemas.pescxml.org http://schemas.pescxml.org/Example-2.xsd">
  <Floor>
    <LevelNumber>1</LevelNumber>
    <Tenant>Smith</Tenant>
    <Tenant>Jones</Tenant>
    <Tenant>Zoltan</Tenant>
  </Floor>
  <Floor>
    <LevelNumber>2</LevelNumber>
    <Tenant>North</Tenant>
    <Tenant>South</Tenant>
    <Tenant>East</Tenant>
    <Tenant>West</Tenant>
  </Floor>
  <Floor>
    <LevelNumber>3</LevelNumber>
    <Tenant>Wealthy</Tenant>
  </Floor>
</PESCXML:Building>
```

Example-2.xsd - (*Use of Elements vs. Attributes*)

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Building">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Floor" type="FloorType"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="FloorType">
    <xsd:sequence>
      **<xsd:element name="LevelNumber" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>**
      <xsd:element name="Tenant" type="xsd:string"
        maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

## 2.3.4 Element vs. Type

Core components SHALL be defined as types and elements SHALL be created from those types. Types allow for the re-use of a single definition of an element or group of elements. A type definition, including its contents, can be re-used by other element definitions, including an element definition with the same name (See Example-3.xml and Example-3.xsd). Reusing element definitions in different documents assists in eliminating confusion as to the format of a data item and its allowable contents. The question "Are these the same or not?" is eliminated.

Example-3.xml - *(Element vs. Type)*

```
<?xml version="1.0"?>
<PESCXML:Directions
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.pescxml.org
http://schemas.pescxml.org/Example-3.xsd">
  **<Location>**
    <HouseNumber>2334</HouseNumber>
    <FirstStreet>PO BOX 1400</FirstStreet>
    <SecondStreet>Dayton</SecondStreet>
    <CityName>Madison</CityName>
    <State>WI</State>
    <ZipCode>53704</ZipCode>
  **</Location>**
  **<Destination>**
    <HouseNumber>1610</HouseNumber>
    <FirstStreet>RT 2</FirstStreet>
    <SecondStreet>Chicken Farm Road</SecondStreet>
    <CityName>Maxwell</CityName>
    <State>MI</State>
    <ZipCode>53786</ZipCode>
  **</Destination>**
  **<Position>**
    <HouseNumber>1220</HouseNumber>
    <FirstStreet>PO Box 724</FirstStreet>
    <SecondStreet>15 St</SecondStreet>
    <CityName>Bowler</CityName>
    <State>IL</State>
    <ZipCode>53111</ZipCode>
  </Position>
</PESCXML:Directions>
```

Example-3.xsd - *(Element vs. Type)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Directions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Location" type="AddressType"/>
        <xsd:element name="Destination" type="AddressType"/>
        <xsd:element name="Position" type="AddressType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="AddressType">
    <xsd:sequence>
      <xsd:element name="HouseNumber" type="xsd:string"/>
      <xsd:element name="FirstStreet" type="xsd:string"/>
      <xsd:element name="SecondStreet" type="xsd:string"/>
      <xsd:element name="CityName" type="xsd:string"/>
      <xsd:element name="State" type="xsd:string"/>
      <xsd:element name="ZipCode" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

New types MAY be derived from existing types providing the capability to extend an element definition within the original type (See Example-4.xml and Example-4.xsd). Derived types can be useful for organizations whose requirements for a data item differ from requirements established within the PESC XML Forum realm.

Example-4.xml - *(Element vs. Type)*

```
<?xml version="1.0"?>
```

```
<PESCXML:Directions
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.pescxml.org
http://schemas.pescxml.org/Example-4.xsd">
  <Location>
    <HouseNumber>2334</HouseNumber>
    <FirstStreet>PO BOX 1400</FirstStreet>
    <SecondStreet>Dayton</SecondStreet>
    <CityName>Madison</CityName>
    <State>WI</State>
    <ZipCode>53704</ZipCode>
  </Location>
  <Destination>
    <HouseNumber>1610</HouseNumber>
    <FirstStreet>RT 2</FirstStreet>
    <SecondStreet>Chicken Farm Road</SecondStreet>
    <CityName>Maxwell</CityName>
    <State>MI</State>
    <ZipCode>53786</ZipCode>
    <CountryCode>CA</CountryCode>
    <PostalCode>POP 1K0</PostalCode>
  </Destination>
  <Position>
    <HouseNumber>1220</HouseNumber>
    <FirstStreet>PO Box 724</FirstStreet>
    <SecondStreet>Southwest Way</SecondStreet>
    <CityName>Bowler</CityName>
    <State>IL</State>
    <ZipCode>53111</ZipCode>
  </Position>
</PESCXML:Directions>
```

Example-4.xsd - *(Element vs. Type)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Directions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Location" type="AddressType"/>
        <xsd:element name="Destination"
```

```
          **type="InternationalAddressType"/>**
        <xsd:element name="Position" type="AddressType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  **<xsd:complexType name="AddressType">**
    <xsd:sequence>
      <xsd:element name="HouseNumber" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="FirstStreet" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="SecondStreet" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="CityName" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="State" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="ZipCode" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  **<xsd:complexType name="InternationalAddressType">**
    **<xsd:complexContent>**
      **<xsd:extension base="AddressType">**
        **<xsd:sequence>**
          **<xsd:element name="CountryCode" type="xsd:string"/>**
          **<xsd:element name="PostalCode" type="xsd:string"/>**
        **</xsd:sequence>**
      **</xsd:extension>**
    **</xsd:complexContent>**
  **</xsd:complexType>**
</xsd:schema>
```

In addition, when defined as a type, an item's requirements MAY vary between Nillable and non-Nillable.  Nil provides a way to specify that an element has no value in an individual document instance (see Example-5.xml and Example-5.xsd).

<u>Example-5.xml</u> - *(Element vs. Type)*

```
<?xml version="1.0"?>
<PESCXML:Directions
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.pescxml.org
http://schemas.pescxml.org/Example-5.xsd">
```

```
  <Location>
    <HouseNumber>1220</HouseNumber>
    <StreetAddress>
      <FirstStreet>Mississauga Avenue</FirstStreet>
      <SecondStreet xsi:nill="true"></SecondStreet>
    </StreetAddress>
    <CityName>Auckland</CityName>
    <State>NJ</State>
    <ZipCode>06743</ZipCode>
  </Location>
</PESCXML:Directions>
```

Example-5.xsd - *(Element vs. Type)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Directions">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element  name="Location" type="AddressType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="AddressType">
    <xsd:sequence>
      <xsd:element name="HouseNumber" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="StreetAddress" nillable="true"
        type="StreetAddressType"/>
      <xsd:element name="CityName" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="State" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
      <xsd:element name="ZipCode" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="StreetAddressType">
    <xsd:sequence>
      <xsd:element name="FirstStreet" type="xsd:string"/>
      <xsd:element name="SecondStreet" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

### 2.3.5 Hide vs. Expose Namespaces

Schemas SHALL be designed to hide Namespaces. Hiding Namespaces provides for XML instance documents that are relatively easy to read and understand, most notably when Schemas import definitions from another namespace. (See Example-6.xml, Example-6.xsd, Sector-6.xsd, and Core-6.xsd - An XML Document and Schema with Namespaces hidden, and Example-7.xml, Example-7.xsd, Sector-7.xsd, and Core-7.xsd - An XML Document and Schema with Namespaces exposed.) Hiding namespaces moves the complexity of a document's framework to the Schema level.

Example-6.xml - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<Example6:BorrowerInfo
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:Example6="http://schemas.pescxml.org/0100/Sector/Example_6"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.pescxml.org/0100/Sector/Example_6
http://schemas.pescxml.org/0100/Sector/Example_6/Example-6.xsd">
  <Person>
    <Name>John Mack</Name>
    <SSN>123-45-6789</SSN>
    <EnrollmentStatus>Full Time</EnrollmentStatus>
  </Person>
</Example6:BorrowerInfo>
```

Example-6.xsd - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  targetNamespace="http://schemas.pescxml.org/0100/Example_6"
  xmlns:Example6="http://schemas.pescxml.org/0100/Sector/Example_6"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  attributeFormDefault="unqualified"
  elementFormDefault="unqualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-6.xsd"/>
  <xsd:import namespace="http://schemas.pescxml.org/0100/Sector"
  schemaLocation="http://schemas.pescxml.org/0100/Sector/sector-
6.xsd"/>
```

```
  <xsd:element name="BorrowerInfo">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Person" type="sector:PersonType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

<u>Sector-6.xsd</u> - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  targetNamespace="http://schemas.pescxml.org/0100/Sector"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-6.xsd"/>
  <xsd:complexType name="PersonType">
    <xsd:complexContent>
      <xsd:extension base="core:PersonType">
        <xsd:sequence>
          <xsd:element name="EnrollmentStatus" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

</xsd:schema>
```

<u>Core-6.xsd</u> - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  targetNamespace="http://schemas.pescxml.org/0100/Core"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:complexType name="PersonType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="SSN" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Example-7.xml - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<Example_7:BorrowerInfo
  xmlns:Example_7="http://schemas.pescxml.org/0100/Sector/Example_7"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://schemas.pescxml.org/0100/Sector/Example_7
http://schemas.pescxml.org/0100/Sector/Example_7/Example-7.xsd">
  <Example_7:Person>
    <core:Name>John Mack</core:Name>
    <core:SSN>123-45-6789</core:SSN>
    <sector:EnrollmentStatus>Full Time</sector:EnrollmentStatus>
  </Example_7:Person>
</Example_7:BorrowerInfo>
```

Example-7.xsd - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org/0100/Example_7"
  xmlns="http://schemas.pescxml.org/0100/Example_7"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  attributeFormDefault="qualified"
  elementFormDefault="qualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-7.xsd"/>
  <xsd:import namespace="http://schemas.pescxml.org/0100/Sector"
  schemaLocation="http://schemas.pescxml.org/0100/Sector/sector-
7.xsd"/>
  <xsd:element name="BorrowerInfo">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Person" type="sector:PersonType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
```

```
</xsd:schema>
```

<u>Sector-7.xsd</u> - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org/0100/Sector"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  elementFormDefault="qualified"
  attributeFormDefault="qualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-7.xsd"/>
  <xsd:complexType name="PersonType">
    <xsd:complexContent>
      <xsd:extension base="core:PersonType">
        <xsd:sequence>
          <xsd:element name="EnrollmentStatus" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

</xsd:schema>
```

<u>Core-7.xsd</u> - *(Hide vs. Expose Namespaces)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org/0100/Core"
  elementFormDefault="qualified"
  attributeFormDefault="qualified">
  <xsd:complexType name="PersonType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="SSN" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Additionally, maintenance is easier as it is possible to change a Schema without impact to instance documents. Take, for example, the case of a Schema that imports component definitions from another namespace. If the imported definitions are moved to within the Schema that had been importing those definitions, or an additional Schema is added to those Schemas already supporting the instance document, every instance document requires updating with those changes.

### 2.3.6 Local vs. Global

It is RECOMMENDED by the Technology Work Group to use xFront's Venetian Blind Design paradigm. This design paradigm, which is well described on xFront's web site (http://www.xfront.com/GlobalVersusLocal.pdf accessed on June 5, 2001), supports reuse of type definitions and namespace hiding. xFront's Venetian Blind Design paradigm focuses on the development of types, which are then used as components for the main element.

By comparison, xFront describes two other design paradigms – the Russian Doll Design and the Salami Slice Design. The Russian Doll Design calls for an XML Schema that mirrors the instance document. The schema is bundled, like a set of Russian doll containers, one inside the other. This paradigm is compact but does not allow for type reuse and hence is largely impractical.

xFront's Salami Slice Design is entirely opposite of the Russian Doll Design. Each component is separately called and joined together in the end, like a salami sandwich. This approach provides for type reuse but does not allow developers to hide namespace complexities.

xFront's Venetian Blind Design paradigm focuses on the development of reusable types which are then used as components for the main element. The following is a schema example using the Venetian Blind Design paradigm (See Example-8.xml and Example-8.xsd). In this example, a library is made up of one to many books. Here, the main element is a "Library", which is made up of the base type "BookRecordType". Note that the data types "EmptyType", "US-StateType", and "StreetAddressExampleType" can be used in many different ways. They are the building blocks for the main record "BookRecordType".

Example-8.xml - (Local vs. Global)

```
<?xml version="1.0"?>
<PESCXML:Library
  xmlns:PESCXML="http://schemas.pescxml.org"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://schemas.pescxml.org
http://schemas.pescxml.org/Example-8.xsd">
  <Book>
    <Author>Ron Johnson</Author>
    <Title>Don't Step in That!</Title>
    <Cost>35</Cost>
    <Quantity>50</Quantity>
    <State>MD</State>
    <Street>1220 North 15 Street</Street>
    <USInd></USInd>
  </Book>
  <Book>
    <Author>Mildred Frank</Author>
    <Title>Golly, If I Only Had a Computer</Title>
    <Cost>52</Cost>
    <Quantity>175</Quantity>
    <State>VA</State>
    <Street>1610 North 11 Street</Street>
    <USInd></USInd>
  </Book>
</PESCXML:Library>
```

Example-8.xsd *(Local vs. Global)*

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://schemas.pescxml.org"
  xmlns="http://schemas.pescxml.org"
  elementFormDefault="unqualified">
  <xsd:element name="Library">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookRecordType"
          minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="BookRecordType">
    <xsd:annotation>
      <xsd:documentation>Many records containing book
        data</xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Cost" type="xsd:float" minOccurs="0"/>
      <xsd:element name="Quantity" type="xsd:integer"
        minOccurs="0"/>
      <xsd:element name="State" type="US-StateType"/>
      <xsd:element name="Street" type="StreetAddressExampleType"
        minOccurs="0"/>
      <xsd:element name="USInd" type="EmptyType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="EmptyType">
  </xsd:complexType>

  <xsd:simpleType name="US-StateType">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AK"/>
      <xsd:enumeration value="MD"/>
      <xsd:enumeration value="OH"/>
      <xsd:enumeration value="VA"/>
      <xsd:enumeration value="NC"/>
    </xsd:restriction>
  </xsd:simpleType>

  <xsd:simpleType name="StreetAddressExampleType">
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="2"/>
      <xsd:maxLength value="30"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

### 2.3.7 Namespaces - Zero, One or Many

The following are RECOMMENDED by the Technology Work Group.

- The Core schema SHALL specify its own namespace as its targetNamespace.
- All Sector schemas SHALL specify their own namespace as their targetNamespace.
- Each instance document schema SHALL specify its own namespace as its targetNamespace.
- References to components used within a schema (simpleTypes, complexTypes, etc.) from the W3C's XML Schema definition are qualified with xsd:.
- In situations where a PESC schema reuses components defined in a schema in another namespace, **Import** SHALL be used. **Import** allows a schema to reference a portion of another schema that resides in a namespace different from that of the referencing schema. This directly supports the PESC model of a schema hierarchy (Core, Sector, Instance Document).

See Example-9.xml, Example-9.xsd, Sector-9.xsd, and Core-9.xsd (below).

Example-9.xml - (Namespaces - Zero, One or Many)

```
<?xml version="1.0"?>
<Example9:BorrowerInfo
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:Example9="http://schemas.pescxml.org/0100/Sector/Example_9"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://schemas.pescxml.org/0100/Sector/Example_9
http://schemas.pescxml.org/0100/Sector/Example_9/Example-9.xsd">
  <Person>
    <Name>John Mack</Name>
    <SSN>123-45-6789</SSN>
    <EnrollmentStatus>Full Time</EnrollmentStatus>
  </Person>
</Example9:BorrowerInfo>
```

Example-9.xsd - *(Namespaces - Zero, One or Many)*

```
<?xml version="1.0"?>
<xsd:schema
  targetNamespace="http://schemas.pescxml.org/0100/Example_9"
  xmlns:Example9="http://schemas.pescxml.org/0100/Sector/Example_9"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  attributeFormDefault="unqualified"
  elementFormDefault="unqualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-9.xsd"/>
  <xsd:import namespace="http://schemas.pescxml.org/0100/Sector"
  schemaLocation="http://schemas.pescxml.org/0100/Sector/sector-
9.xsd"/>
  <xsd:element name="BorrowerInfo">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Person" type="sector:PersonType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Sector-9.xsd - *(Namespaces - Zero, One or Many)*

```
<?xml version="1.0"?>
<xsd:schema
```

```
  targetNamespace="http://schemas.pescxml.org/0100/Sector"
  xmlns:sector="http://schemas.pescxml.org/0100/Sector"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:import namespace="http://schemas.pescxml.org/0100/Core"
  schemaLocation="http://schemas.pescxml.org/0100/Core/core-9.xsd"/>
  <xsd:complexType name="PersonType">
    <xsd:complexContent>
      <xsd:extension base="core:PersonType">
        <xsd:sequence>
          <xsd:element name="EnrollmentStatus" type="xsd:string"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:schema>
```

<u>Core-9.xsd</u> - *(Namespaces - Zero, One or Many)*

```
<?xml version="1.0"?>
<xsd:schema
  targetNamespace="http://schemas.pescxml.org/0100/Core"
  xmlns:core="http://schemas.pescxml.org/0100/Core"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="unqualified"
  attributeFormDefault="unqualified">
  <xsd:complexType name="PersonType">
    <xsd:sequence>
      <xsd:element name="Name" type="xsd:string"/>
      <xsd:element name="SSN" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

The XML Forum SHOULD NOT use the **Redefines** option when defining its Schemas. A Schema **Redefine** operation performs an implicit **Include** operation. All of the components in the Schema that are the object of the **Redefine** are **Included** in the Schema performing the **Redefine**. However, **Redefine** takes things farther than **Include** by allowing the Schema performing the **Redefine** to extend or restrict components in the **Redefine**d Schema. Most likely this will not be necessary for the generic PESC definitions.

Use of **Redefine**s, however, MAY be advantageous for use by an organization that has additional requirements for a data item which fall outside the requirements defined in the PESC Schema. Like **Include**, **Redefine**s can be used for any Schema that does not have a targetNamespace. This allows an entity to **Redefine** (ie., **Include**) a PESC component schema, but modify that Schema with its own extensions/requirements.

### 2.3.8 Variable Content Containers

Variable content containers SHALL NOT be supported.

A container is an element that contains other elements. For example, a Chapter element that contains multiple Paragraph and Illustration elements describes the composition of a particular chapter in a book.

A variable content container accommodates those situations where the number of elements within the container is subject to change (i.e., items can be added to and/or removed from the container description as time goes on), and where the attributes associated with an element may vary from element to element.

The ability of a variable content container to support these requirements is powerful and handy in appropriate situations. Properly implemented, variable content containers lend themselves well to ad-hoc assembly of information units of similar nature. While useful for information publishing and presentation, variable content containers are beyond the needs of this implementation.

The methods by which these containers are implemented are relatively complex. Dealing with this complexity is made easier if this method of schema architecture is implemented by a single entity. In a situation where multiple entities are sharing schema structures, the complexities are too great to be comfortably managed in the same fashion by all participants.

This is in contrast with the intentions of the XML Forum - to keep the initial implementation at a level of simplicity facilitating broad and successful adoption. For the purposes of moving information formatted as XML from point to point in support of transactions, the complexity of variable content containers is not warranted.

### 2.3.9 Nulls, Zeroes, Spaces, and Absence of Data

The following rules SHALL apply in designing schemas and interpreting instance documents.

1. *Absence of data* - If an element is defined as OPTIONAL (minOccurs attribute value of zero) and the element does not occur in an instance document, semantics SHALL NOT be interpreted from the element other

than that the originator of the instance document did not include it. No default values are to be assumed. Likewise, if an attribute is declared as OPTIONAL ("use" attribute value of OPTIONAL) and the attribute does not occur in an instance document, semantics SHALL NOT be interpreted from the attribute other than that the originator did not include it. No default values are to be assumed.

**NOTE**: All string items defined with a minOccurs of one SHALL have a minimum length requirement of one character.

2. *Zeroes* - Zeroes, when appearing in a numeric element in an instance document, SHALL be interpreted as a zero value.

3. *Spaces* - Spaces sent as values for elements or attributes (of type string) in instance documents SHALL be interpreted as spaces. It is RECOMMENDED that leading and trailing spaces be removed, but when they appear they SHALL have semantic significance. The schema Whitespace facet directly affects spaces with its ability to collapse contiguous spaces, and replace tabs, carriage returns, and line feeds with a space. The default setting of Whitespace SHALL be used, as it preserves existing spaces, tabs, carriage returns, and line feeds as they are. Sending an element with just spaces is not the same as sending a nulled element (see #4 below).

4. *Nullability* - In certain cases, it MAY be desirable to convey that an element has no value (a null value) rather than indicating that it has a value of spaces or that it is not present in a document. In these cases, the originator of the instance document SHOULD convey explicitly that an element is null. An example is an address update for a previously transmitted address. The previous address had two address lines, whereas the current address has just one line. The originator of the document indicates that the second address line is removed by indicating that the element is nulled as follows:

   ```
   <AddressLineTwo xsi:nill="true"></AddressLineTwo>
   ```

   To support this the AddressLine element in the schema is defined as nullable via:

   ```
   <xsd:element name="AddressLine" type="xsd:string" nillable="true"/>
   ```

   When this type of nullable semantics are desired, the "nill" and "nillable" attributes SHALL be used (as opposed to spaces for strings or zeroes for numerics). The "nillable" attribute SHALL NOT be used in element declarations with a minOccurs of greater than zero. When there is a requirement that an element be OPTIONAL and not appear in an instance document, the minOccurs attribute with a value of zero SHALL be used in the element declaration. By default, any element defined in analysis as having a minimum occurrence of zero SHALL be represented in the schemas as nullable.

### 2.3.10 Other Considerations

1. _Permissive vs. Restrictive_ - Schemas SHALL be restrictive, i.e., the "ANY" content model is prohibited.
   *Rationale*: Schemas are intended for data interchange, with data content to be fully defined.

2. _Use of Notation_ - Notations to define data types or file types SHALL NOT be used.
   *Rationale*: Simplicity. Base XML schema features should provide all necessary functionality.

3. _Unparsed Entities_ – Unparsed entities SHALL NOT be supported.
   *Rationale*: Simplicity (requires use of Notation). No present requirement to handle or reference non-XML data sources.

4. _Mixed Content_ – Mixed content SHALL NOT be permitted in schema definitions, (i.e., an element with children SHALL NOT also have a text value).
   *Rationale*: XML Forum schemas are oriented toward data exchange and mixed content is contrary to most established data modeling philosophies.

## 3 XML Schema Development Roadmap

The roadmap for schema development relies on the interaction of the Core Components and Technology Work Groups within the XML Forum. This section describes the interaction of those groups for the development of new XML Schemas and the update and maintenance of existing XML Schemas

1. The Core Components Work Group forwards a spreadsheet for a dictionary (described earlier in this document) of related items to the Technology Work Group. These related items may comprise all data elements for the following.

   - the community (the "parent" dictionary),
   - a given business domain (a "sector library"),
   - a document or message, or
   - an aggregate.

   Sector libraries, documents, messages, and aggregates should be represented on spreadsheets separate from the parent dictionary.

2. The Technology Work Group creates XML Schemas appropriate to the scope of the items represented on the submitted spreadsheet. Base XML

Schema types common to the PESC community are defined in the parent dictionary. The parent dictionary schema may not correspond to an actual XML message set or document. When working with a dictionary for a business domain, a sector library is created if the requirements of that domain cannot be met by the definitions contained in the parent dictionary. Where applicable and appropriate, the Technology Work Group will use existing base Schema types. These existing types may be defined in the parent dictionary, sector library, or by other standards such as ISO. New base Schema types are created only as necessary - if the type does not exist in the parent or sector library, or if an existing type in the parent or sector library does not meet all requirements specified by the sector library dictionary.

3. There are cases where an existing dictionary is updated with new elements, the requirements for an existing element change, or one or more elements are removed. To accommodate these maintenance situations, a new spreadsheet with the changes highlighted is provided to the Technology Work Group, and steps 1 and 2 are repeated as necessary to update the applicable Schema(s). The Technology Work Group will monitor maintenance situations for the potential to eliminate a sector schema and/or sector-specific element definitions due to changes in requirements.

4. After creating an initial XML Schema for a document or message, the Technology Work Group creates at least two instance documents, a "typical" document and a "complete" document. The "typical" document shows how an instance document appears when all of the required fields, and some of the optional fields, exist. The "complete" document illustrates the instance document when all fields, required and optional, exist. These instance documents are used to both validate that the structure expressed by the corresponding Schema is correct, and that the instance document is indeed valid as per the Schema. Schemas and instance documents may be validated via the tool used to create them, and by a validating XML parser.

5. The Core Components Work Group reviews the "typical" and "complete" instance documents. Suggestions and corrections are returned to the Technology Work Group. The Technology Work Group reviews suggestions and corrections, and makes appropriate changes to the schema and sample instance documents. This step is repeated as often as necessary to provide example instance documents that effectively illustrate the data structures described by the Schema(s).

6. If there are multiple uses for the data group(s) described by a schema. The Core Components Work Group advises the Technology Work Group as to the contents of each set of example instance documents that illustrate a particular data exchange for a message set.

(*example*: For the various applications of the Common Record, or the varied CommonLine message sets, a set of instance documents illustrating each document/message set is created.)

7. The Technology Work Group creates instance documents with the contents defined by the Core Components Work Group in Step 6. These documents and their corresponding Schemas pass through the procedures in steps 4 and 5 as often as necessary. When approved by the Core Components Work Group, these example documents serve not only as illustrations, but may also be used by business partners to test the electronic portion of data exchange agreements.

8. Once accepted by the Core Components Work Group, the Schemas are considered ready for release to the community. The Architectural Chair solicits comments from the full Forum and then forwards a copy with comments to the Steering Committee for final approval. Upon approval, XML Schemas will be released and published to http://schemas.PESCXML.org.

# 4 Implementation Recommendations

## 4.1 Why this section?

The primary goal of the PESC XML Forum is to create XML schemas to enable common documents to be exchanged in the postsecondary education environment. However, in addition to this requirement, PESC members are also seeking guidance regarding how to implement XML and related technologies on the public Internet. Of primary interest to the Forum is helping its members follow a degree of uniformity so that their implementations are interoperable. This section presents a set of recommendations, accompanied by rationale, which should assist PESC members in making implementation decisions.

## 4.2 General Requirements

There are a number of general requirements that may influence many technology decisions. This section captures some of the more important requirements.

- *Cost* - Keeping implementation costs low is a very important consideration for most PESC members.
- *Time frame* - The Forum has identified two separate time frames around which to make recommendations. Recognizing that some PESC members will be implementing in the next few months, there is a short-to-near term time frame that anticipates minimal changes in existing systems. In addition, the Forum anticipates that systems may evolve as XML and related technologies continue to develop. Therefore, we have included a long-term time frame on most recommendations.

- *Compatibility with existing systems* - Compatibility with existing systems is of high importance in the short term. For the longer term this is a secondary consideration.
- *Open Source* - Public domain, open source software shall be considered as an option where it is available, but this is a secondary consideration.
- *Standard vs. Proprietary solutions* - The Forum will only recommend solutions that are based on standards from internationally or nationally accredited standards organizations, or recommendations from leading industry consortiums such as the World Wide Web Consortium (W3C). Solutions based on a single vendor's proprietary approach SHALL NOT be recommended.

The Technology Work Group has identified a number of use cases for which these recommendations are targeted.

- *School to school transcript exchange* - In response to requests received by a variety of means, a school batches a set of transcripts and sends them to another school. A third party, such as an EDI Server, may or may not be involved.
- *School to agency or organization IPEDS reporting* - Aggregate enrollment information, faculty, facility. These may be batch or real-time, TBD.
- *School to Department of Education direct loan application* - loan origination and disbursement. Batch and near real-time single applications.
- *Meteor* - Initiated by borrower or financial aid officer, a Meteor-enabled site goes to the National Student Clearinghouse, finds out who the loan holders are, sends inquiries to holders about the specific loans held by a borrower, receives responses, and displays them to the requester. Real-time application.
  **NOTE**: We are concerned with what happens between the Meteor enabled site and the clearinghouse & loan holders.
- *Organization to school test score reporting* - An organization such as the ETS sends test scores in batch to a school.
- *School reporting non-administrative student data to outside organizations* such as INS, state departments of health.

The following use cases are not within our scope:

- Interdepartmental exchange of student administrative data (campus to campus exchange within the same system may be within scope)
- School procurement with suppliers
- School Accounting and budgeting
- Development

## *4.3 Message Handling*

### 4.3.1 Requirements

- *Packaging* - A means must be provided to allow multiple XML documents to be packaged into one unit for transport.
- *Network protocol* - IP over the public Internet must be supported.
- *Reliable delivery* (once and only once) through software - This is not a near term requirement. It is expected that exceptions from overdue or duplicates will be handled by business applications or manual methods and not the communications/messaging software.
- *Return Receipts* - This is the ability of the messaging system to automatically generate return receipts upon receipt of a message. Generally, this is a "nice to have", but is not a high priority short-term requirement since current systems may not be able to support return receipts. It is expected that business applications will handle return receipts at a business document level where they are required (in other words, at the business application level rather than at the protocol level).
- *Routing* (identification internal sender & receiver) - There may be a requirement to provide the ability to route documents to an internal business application once they have been received by an organization. However, we are not certain that this functionality is needed and request input and comment.
  (*example*: all documents for a university might be sent to xml@university.edu, but transcripts and loan information might need to be sent do different business applications.)
- *Error handling and reporting* through software - This pertains to providing error handling and reporting being provided by the messaging system. Again, these are generally "nice to have", but not a high priority short-term requirement since current systems may not be able to support them. It is expected that business applications will provide this functionality where required.
- *Automated restart and recovery* - "Nice to have" but not a high priority.
- *Audit trails* - "Nice to have" but not a high priority.
- *Ability to specify or request quality of service* - This is not considered to be a near term requirement.
- *Platform independence* - This is a high priority since it is expected that many different platforms will be used.

For more information and discussion, refer to ebXML Transport Routing and Packaging Overview and Requirements at http://ebXML.org/specs/esreq.pdf.

### 4.3.2 Options and Near Term Recommendations

Packaging

- UNIX tar
  - *Pros* - Easily invoked by scripts.  Universal implementations on all UNIX platforms
  - *Cons* - Though utilities may be available on other platforms, these may not be commonly installed.
- Zip
  - *Pros* - Easily invoked by scripts.  Commonly available on Intel/Win32 platforms.
  - *Cons* - Though utilities may be available on other platforms, these may not be commonly installed.
- MIME
  - *Pros* - Default packaging if using multiple attachments with most SMTP mailers.
  - *Cons* - Not as well supported if not using an SMTP mailer.
- SOAP
  - *Pros* - Designed specifically for XML support.
  - *Cons* - Has not progressed to a full W3C recommendation.  Not currently widely supported in commercial, off-the-shelf products (mostly widely available as software development kits). SOAP with attachments (SOAP Messages with Attachments, W3C Note 11 December 2000, http://www.w3.org/TR/SOAP-attachments) may be advantageous, but this is not fully supported in the latest SOAP specification.
- ebXML
  - *Pros* - Meets all packaging and routing requirements as well as security requirements.
  - *Cons* - Little commercial support as yet; complex; dependent on SOAP with attachments.
- X12 996 File Transfer Transaction Set
  - *Pros* - Compatible with existing EDI Server.
  - *Cons* - Most likely will require X12 capable EDI system.  Attention must be paid to using X12 delimiters that aren't used in the XML syntax.
- SMTP - Multipart MIME attachments
- FTP and HTTP - Either Tar or Zip.
  **NOTE**:  FTP may not need packaging with mput and mget if files are sent to or retrieved from unique directories.
- EDI Server - X12 996 Transaction Set

Recommendation:  Dependent on chosen file transport protocol and system

Network Protocol

- IP over the public Internet is recommended.

Supported file transfer methods (based on using IP)

- FTP
  - *Pros* - Implementations on most platforms. Provides for immediate flagging of transmission failures. Easily invoked by scripts.
  - *Cons* - Must set up session with source or target (i.e., cannot operate in a store-and-forward or asynchronous fashion). Not well suited to situations that might require very fast response time. Username/password or anonymous logins are possible security vulnerabilities.
- SMTP
  - *Pros* - Implementations on most platforms. Store-and-forward model allows for asynchronous delivery.
  - *Cons* - Some mail systems may limit the size of attachments. Some systems may have problems with multiple attachments. Not as easily scripted as FTP. No immediate notification of delivery failure or delays.
- HTTP (Post and Fetch)
  - *Pros* - Browser implementations on nearly every platform; server implementations on most platforms. Allows for fast response times. Session security available with HTTPS
  - *Cons* - Not easily scripted or integrated with existing business applications; may require human action.
- Kermit, Xmodem or other TELNET or terminal session based protocol
  - *Pros* - Lowest common denominator, widely available.
  - *Cons* - Not as easily scripted or integrated with existing business applications, may require human action, username/password security vulnerability, performance not as good as other options.

Recommendation:

- FTP vs. SMTP - Discussion needed. Leaning toward FTP since it offers immediate notification of delivery errors. FTP or SMTP may be more appropriate for unattended operation.
- HTTP for business applications suited to a human running a browser, with relatively small file sizes.

Reliable Delivery

- There are no suitable options in the near term.

Return Receipts

- Some SMTP servers support return receipts, although not all do. FTP provides immediate notification of successful transmission. Beyond these, there are no suitable options in the near term.

Routing

- Separate destination address for each business application (*example*: transcripts@myuniversity.edu for transcripts, and financialaid@myuniversity.edu for financial aid)
  - *Pros* - Relatively easy to set up if there are a small number of destinations. Requires no special inner envelope.
  - *Cons* - Complex if there are a large number of internal destinations
- X12 envelope
  - *Pros* - Compatible with existing EDI Server.
  - *Cons* - Most likely will require X12 capable EDI system.
- ebXML
  - Pros - Meets all packaging requirements as well as security requirements.
  - Cons - Little commercial support as yet. Dependent on SOAP with attachments.
- Extensions to SOAP header (similar to ebXML)
  - *Pros* - Designed specifically for XML support.
  - *Cons* - Has not progressed to a full W3C recommendation. Not currently widely supported in commercial, off-the-shelf products.

Recommendation:

- For traffic not going through EDI Server, use separate destination addresses for each business application
- For traffic going through EDI Server, use X12 996.

Software-based error handling and reporting

- There are no suitable options in the near term.

Automated restart and recovery

- There are no suitable options in the near term.

Audit trails

- There are no suitable options in the near term.

Quality of Service (QOS)

- This is not considered to be a requirement in the near term.

Alignment with Frameworks (a.k.a. Web Services architectures)

- ebXML

    - _Pros_: Non-proprietary
    - _Cons_: May have more features than needed in some areas, and not enough in others. Few commercial implementations as yet.
- Microsoft .NET
    - _Pros_: Commercial implementation
    - _Cons_: Proprietary. Limited support outside of Intel/Win32 platform.

Recommendation:

- Do not align with or support any particular framework at this time, in the near term.

#### 4.3.3 Long term recommendations

It is premature at this time to make long term recommendations. The XML Forum will watch the trends and see how they develop. SOAP appears, at this time, to be the XML/IP message handling option most likely to gain widespread market acceptance due to its association with W3C. However, the ebXML message handling service is more capable, and several major vertical industry associations have adopted it.

### *4.4 Security*

#### 4.4.1 Requirements

The scope of these requirements is limited to exchanges of information between organizations. Specific security requirements may vary depending on the information being transmitted, business processes, business applications, and the institution. For example, FERPA does not require any specific technologies, but leaves it to institutions to select the technologies, according to their own requirements, that enable them to comply with the act. In addition, some institutions may require verification of receipt, while others may not.

In this section we identify some general considerations regarding security and attempt to identify some preliminary requirements.

##### 4.4.1.1 General Considerations

One useful way to assess security issues is to consider the following factors.

- _Risk areas_ - What are the general types of things (events, resources) that are at risk?
- _Risk exposure_ - What is the potential damage that could be incurred if a risk event happens?
- _Risk probability_ - What is the likelihood that a risk event will happen?

Once these are determined, appropriate countermeasures or remediation strategies may be determined.

In a classic model of computer security, the following areas generally address most risks:

- *Integrity* - Systems and data integrity are maintained. In the areas being addressed, this generally means that the receiver receives data as the sender, without alteration, has transmitted it. There are also concerns with integrity of the business transaction, the intentional breach of which might be considered as fraud.
- *Confidentiality* - Systems and data are protected from unauthorized access. The primary concern is maintaining the confidentiality of data while in transmission.
- *Availability* - Systems and data are available when needed. There are generally no concerns with this aspect of security, though they may be affected by general risks such as "denial of service" attacks.

#### 4.4.1.2 Security Considerations

Typical risk events and security considerations that may be of concern in document exchanges between institutions are listed below, with an initial assessment of risk probability and exposure. The requirements are generally the same for admissions and records related data (transcripts, applications, test scores, etc.) and financial aid data.

- *Transmission Confidentiality* - There are external requirements to keep certain information confidential from the general public. For example, the Family Educational Rights and Privacy Act (1974) requires that institutions keep student information reasonably confidential. Placing information like this "in the clear" on the public Internet may violate such requirements. It is the belief that most XML Forum messages may be subject to such requirements, therefore this is a high probability risk area. Breach of statutory or regulatory confidentiality requirements may involve criminal or civil liability, with associated fines or penalties. Transmission confidentiality is of high concern.
- *Persistent confidentiality* – Persistent confidentiality is concerned with maintaining the confidentiality of a transmitted document after it has been received (or before it is transmitted). Since the XML Forum is primarily concerned with formats of data for interchange and in facilitating interchange, this topic is beyond our immediate scope.
- *Fraud* - Fraud can be considered a breach of transaction integrity. Fraud can include events such as the following.
  - Third party assumes identity of sender and transmits an invalid or bogus message. In general, we regard this as unlikely but request verification. Risk exposure may vary widely depending on the business area. The probability for exposure are low for admissions and records information, but may be somewhat higher for student loan information.

    - Third party assumes identity of receiver and intercepts message. Again, not very likely but request verification. Risk exposure may also vary, however, when financial or personal information such is involved, the probability and exposure may be higher.
    - Sender denies sending a transmission - This area is of high concern.
    - Receiver denies receiving transmission - This area is of low concern.
    - Transmission altered - Unlikely, but request verification. Again, the exposure may vary depending on the business area.
- *Signature requirements* - There are legal means to make electronic signatures have the same legal force as written signatures. There may be circumstances in which an institution is required to pass along to another institution the electronic signature of an individual (or an indication that the individual has signed a document), but the specifics about requirements for institutions to sign documents exchanged with other institutions is unknown. For example, a lender may be required to pass along to a guarantor the electronic signature of a loan applicant, but the lender may not itself need to sign the loan application. In cases where an institution is acting as a third party, as in the case of on the behalf of a student or borrower with Sallie Mae, it may need to authenticate the end user. However, these types of "transitive trust" problems are beyond the immediate scope of this specification.

#### 4.4.1.3 Countermeasures and Remediation Strategies

Various types of countermeasures or remediation strategies may be appropriate depending on the risk probability and exposure. Not all of these necessarily need to be handled in messaging software, and might instead be handled by business applications or manual procedures. Countermeasures fall into the following broad categories:

- *Authentication* - In general terms, these are mechanisms designed to verify that a party is who they purport to be before granting them access to a resource. For transmitting documents across the public Internet, it generally refers to verifying the identities of the sender and receiver of the data.
- *Authorization* - In general terms, these are mechanisms that grant or deny access to a resource after the identity of the user has been authenticated. For document transmission, it generally deals with the ability to view or modify confidential data that has been encrypted.
- *Error detection* - This is a broad area that can include activities such as monitoring system access and error logs. These may or may not be part of the messaging software. It can also cover reviewing and validating data in business applications to detect aberrations or alterations. Error detection can be thought of as a way to detect actual or attempted security breaches rather than to prevent them.

#### 4.4.1.4 Encryption

One of the customary security countermeasures that addresses authentication and authorization is encryption. There are two broad categories of encryption. Each of which has various non-technology-related requirements for use.

- _Symmetric (private shared) keys_ - Data is encrypted by the sender and decrypted by the receiver using the same key. This is the simplest method, but the management and confidentiality of the key(s) becomes complex as more parties are involved. Symmetric keys are generally used only for confidentiality.
- _Asymmetric (two part public/private) keys_ - Asymmetric keys can be used for both encryption and authentication. For encryption, the sender encrypts the data using the receiver's public key, and the receiver decrypts it using the receiver's private key (known only to the receiver). For authentication, the sender uses the sender's private key, which is known only to the sender. This is commonly done by encrypting a so-called "message digest". The message digest can also serve as a means to detect whether or not the message was altered. The receiver then decrypts the digest using the sender's public key.

When asymmetric keys are used, private keys are always kept confidential to the owner but the public keys are shared. Various "trust hierarchies" or "trust models" may be devised to handle the public keys. Where parties are known to each other, the public keys may be exchanged on a bilateral basis by whatever means make sense. This may be appropriate for communities of limited size. However, when parties don't always know each other or when communities become large, it may be necessary to have a "trusted third party" that can "vouch for" the identity (and the public key) of each of the parties. Community members may retrieve the public keys of other parties from such a third party, as well as validate a key that they have been given directly.

We believe that the XML Forum's security requirements will determine that asymmetric keys will be most appropriate. This leads to the question of the best trust model. We anticipate that in the short term, informal bilateral exchange of public keys will be sufficient. However, we also anticipate that in the near to long term a trusted third party model will be required.

In addition, batch and real-time implementations may require somewhat different solutions. In the current environment, the EDI Server in Austin supports a relatively small community of participants who are known to each other. "Out of band" key exchange and management works well in this environment. This same model seems to hold true for CommonLine. The same model may hold true for real-time situations in which there is an application-to-application exchange of data. However, in real-time situations involving individuals at a browser, a third party trust model may be required. In addition, there may be situations in which an individual using a browser may act on behalf of an institution, for example, presenting the institution's security certificate instead of one assigned to that specific individual. These types of requirements need further research.

For more general information and discussion, refer to ebXML Transport Routing and Packaging Overview and Requirements.

### 4.4.2 Near term Recommendations

- *Key type* - Symmetric (private) or Asymmetric (public/private) - It is fairly clear that asymmetric keys are the best choice for supporting various authentication and non-repudiation requirements. Asymmetric keys used in conjunction with one-time session (or file) keys are recommended for encryption to ensure confidentiality.
- *Key management* - The EDI Server in Austin currently uses a PGP keyring to manage a small community of institutional users engaging in batch data exchanges. Public keys are generally exchanged with the Server via e-mail or some other "out-of-band" techniques, and these appear to be adequate for the near term. CommonLine is planning on near-real time exchanges using PGP. For a near term recommendation, we recommend the current practice of out-of-band key exchange. We also note that the Federal government has a GSA sponsored government wide initiative for a public key infrastructure - see http://hydra.gsa.gov/aces. The Forum should monitor the development of this infrastructure in developing a long term recommendation.
- *Digital signature approach* - The predominant current practice is to use PGP. As XML DSIG implementations become commonly available, now that it is a W3C recommendation, we recommend that the community consider it.
- *Encryption algorithm/software* - For a near term recommendation, we recommend current practice of using PGP. We also note that some government implementations, such as the Department of Education's Student Aid Internet Gateway, are using FTP over SSL 3.0 and the Diffie-Hellman Dynamic Key Exchange algorithm.

### 4.4.3 Long term recommendations

We recommend the development of a security policy framework. A policy framework gives a basis for trust in the community, and gives participants a basis for the extent to which they can trust a message. Such a framework might include a set of practices that organizations have to agree to in order to participate, defines how the trust framework works, provides a set of information that organizations have to supply in order to comply with the trust framework, describes how the trust framework works technically, describes the functionality provided, and specifies the roles and responsibilities of the entity managing the membership. This framework must take into account the fact that different government organizations (such as the INS or Department of Education) might have different explicit or implicit frameworks with which they expect campuses to comply.

In regard to the technical implementation of the framework, we recommend that the XML Forum adopt in the long term a set of technologies that are compatible with the recommended countermeasure technologies described in section 12.3 of the May, 2001 ebXML Message Service Specification. In particular, we direct attention to the table of section 12.3.10 that specifies recommended technologies for various usage profiles. These recommendations, in general, recommend use of XML/DSIG, SAML, and XML-Encryption as these technologies mature.

## *4.5 Registries and Repositories*

### 4.5.1 Requirements

Registries and repositories are an ongoing area of development in e-Business technology. Repositories are facilities for storing data. They can be thought of as the bookshelves in a library. Registries have information about the data in the repository, and assist in retrieving items from repositories

Registries and repositories generally deal with making information or software components (such as XML schemas or parts of schemas) available from a central location, and offer various ways to categorize and search. The types of information, items stored, and uses can vary widely. Some are little more than libraries of XML schemas or DTDs for common business documents. Others are designed to also store information on business processes or encoded models of them. Still others function much like telephone yellow pages, helping to identify companies, which offer certain goods and services.

The architecture of registries and repositories are generally determined by how they deal with certain requirements. Here are the requirements that are generally most important, and how we anticipate the XML Forum will view these requirements in the near term.

- *Types of objects to be stored* (items) - XML schemas are the primary item that we believe will need to be stored. These will cover both document schemas and "library" schemas. There may be a need to also store supporting materials such as spreadsheets.
- *Actors* (who uses them) - Most XML Forum members access the materials as users. A small number of persons on the Forum staff or a small number of technical volunteers may maintain the information and materials.
- *Activities supported* - The schemas MUST be available to users for run-time validation of instance documents and for developers to assist in designing applications (or transformations/conversions) that use the schemas.

- *Access controls* - Public read access is required. Create/update/delete access is required and restricted to a small technical staff.
- *Audit trails* - Audit trails generally track creation, update, or deletion of registry items, and may also track access. We don't anticipate a requirement for system based audit trails at this time on the basis that there will only be a limited number of persons with write access.

For more information and discussion, refer to ebXML Registry and Repository Part 1: Business Domain.

#### 4.5.2 Near term Recommendations

The Technical Work Group recommends that the Forum use the PESCXML.org web server, with a set of file system directories controlled by an administrator, as a registry and repository. Run-time schemas should be stored in these directories, and supporting documentation posted on the web site.

#### 4.5.3 Long term recommendations

The Forum should monitor developments regarding the maturity and acceptance of registries, particularly ebXML registries. When and if these become mature, accepted, and implemented by a number of standards bodies, the Forum may wish to consider using a facility that is hosted by another body such as DISA, OASIS, or UN/CEFACT.

### *4.6 Electronic Trading Partner Agreements*

#### 4.6.1 Requirements

This topic is covered primarily due to the ongoing ebXML work with Collaboration Protocol Agreement and Profile (CPA/CPP) and the Universal Description, Discovery, and Integration (UDDI) initiative. The ebXML CPA/CPP offers a mechanism for describing the IT aspects of an e-Business relationship in an XML document so that it can be used to automatically configure an e-Business system. UDDI offers similar features, but also offers mechanisms for discovering trading partners.

Regarding the requirements addressed by the ebXML CPA/CPP, we anticipate that the information required to configure systems so that different organizations may exchange data by manual means, and that the systems will be manually configured. We base this on the assumption that the amount of information needed can be kept to a level that makes manual means practical.

Regarding the requirements addressed by the UDDI initiative, we don't see a near term need for automated discovery of trading partners. We believe that organizations will know of or discover each other primarily through means involving human action.

### 4.6.2 Near term Recommendations

The Technical Work Group recommends against supporting electronic trading partner agreements in the near term, as there is no firm requirement for them.

### 4.6.3 Long term recommendations

As with the other implementation areas, the Forum will monitor the maturity and implementation of ebXML CPA/CPP and UDDI. When and if these mature and software becomes commonly available, then the Forum may wish to consider a different recommendation.

## 5 Reference Documents & Standards

| | |
|---|---|
| http://www.xfront.com/ | XML Schema Best Practices as maintained by Roger L. Costello |
| http://www.w3.org/XML | The current specification for XML Schemas |
| http://www.ibiblio.org/xml/ | XML resources at Ibiblio (Café Con Leche) |
| http://xml.coverpages.org/sgmlnew.html | Archives of Robin Cover's XML Cover Pages at OASIS (the Organization for the Advancement of Structured Information Standards |
| http://www.ietf.org/rfc/rfc2119.txt?number=2119 | Key words for use in RFCs to indicate requirement levels - Internet Engineering Task Force Request for Comments 2119 |
| http://ebxml.org | The current requirements specification for ebXML |

### *5.1 Terms*

| | |
|---|---|
| BizTalk | An XML schema language from Microsoft, released in 1999. |
| Cardinality | A quantity relationship between data elements. For example, one-to-one, zero-to-many and many-to-one express cardinality. These are referenced as 1..1, 0..u, u..1. |
| CommonLine | A standard for transmission of FFELP and Alternative Loan student loan data between schools and their varied service providers. |
| Namespace | A unique name that identifies an organization that has developed an XML schema. It serves as a prefix so that multiple |

| | |
|---|---|
| | schemas can be used to define tags in an XML document. XML namespaces provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified by URI references. |
| XML Type | A definition of an element or group of elements that can be re-used in the definition of other elements. |
| XML Schema | The definition of the content and structure used in an XML document, written in XML syntax. Schemas are more verbose than DTDs, but they can be created with many XML tools. |

### *5.2 Acronyms*

| | |
|---|---|
| CPA | Collaboration-Protocol Agreement (CPA). The message exchange agreement between two parties may be described by a Collaboration-Protocol Agreement (CPA). |
| CPP | Collaboration-Protocol Profile (CPP). The message exchange capabilities between two parties may be described by a Collaboration-Protocol Profile (CPP). |
| DSIG | Digital Signature Group (DSIG). DSIG proposed a standard format for making digitally-signed, machine-readable assertions about a particular information resource. |
| DTD | Document Type Definition (DTD). A language that describes the contents of an SGML or XML document, consisting of a set of mark-up tags and their interpretation. |
| ebXML | Electronic Business XML (www.ebxml.org) (ebXML) is a set of specifications that together enable a modular electronic business framework. ebXML enables a global electronic marketplace where enterprises of any size and in any geographical location can meet and conduct business with each other through the exchange of XML-based messages. ebXML is jointly sponsored by the United Nations (UN/CEFACT) and OASIS. |
| EDI | Electronic Data Exchange (EDI). The exchange of standardized document forms between computer systems for business use. |
| ETS | Educational Testing Service (ETS). A private educational testing and measurement organization. |

| | |
|---|---|
| FERPA | Family Educational Rights and Privacy Act (1974) (FERPA) allows students access to their educational records and limits the ability of others to access those records, except as authorized by law. |
| IEEE | The global association of engineers, scientists and allied professionals (www.ieee.org) |
| IFX | Interactive Financial Exchange (IFX) (www.ifxforum.org) |
| IMS | The IMS Global Learning Consortium (www.imsproject.org) is an organization developing and promoting open specifications for facilitating online distributed learning activities such as locating and using educational content, tracking learner progress, reporting learner performance, and exchanging student records between administrative systems. |
| ISO | The International Organization for Standardization (ISO). A network of national standards institutes from 140 countries working in partnership with international organizations, governments, industry, business and consumer representatives (www.iso.ch/iso/en/ISOOnline.openerpage) |
| PGP | Pretty Good Privacy (PGP). PGP is method of encrypting your data. It can also be used to apply a digital signature to a message without encrypting it. |
| SAML | Security Assertion Markup Language (SAML). SAML defines an XML framework for exchanging authentication and authorization information using industry-standard protocols and messaging frameworks. |
| SGML | Standard Generalized Markup Language (SGML). SGML is an international standard (ISO 8879) that prescribes a standard format for embedding descriptive markup within a document as well as a way of describing the structure of a document. |
| SOAP | Simple Object Access Protocol (SOAP). SOAP is an XML based lightweight protocol for the exchange of information in a decentralized, distributed environment. |
| UDDI | Universal Description, Discovery, and Integration (UDDI). UDDI is a platform independent, open framework for describing services using the Internet. It uses standards such as Extensible Markup Language. |

| | |
|---|---|
| URI | Universal Resource Identifier (URI). The generic set of all names and addresses that are short strings that refer to objects (typically on the Internet). |
| URL | Uniform Resource Locator (URL). A standard way of specifying the location of an object, typically a web page, on the Internet. |
| W3C | The World Wide Web Consortium (W3C) - the main standards body for the World-Wide Web. |
| X12 | Also known as "ANSI X12" and "ASC X12,". It is a standard from the American National Standards Institute (ANSI) for electronic data interchange (EDI). X12 is the primary North American standard for defining EDI transactions. |
| XDR | XML Data Reduced. An XML schema language from Microsoft, XDR was released in 1999 as a working schema as part of Microsoft's BizTalk initiative. |
| XML | Extensible Markup Language (XML). A subset of Standardized Generalized Markup Language aimed at document publishing, XML is an open standard for describing data and defining data elements in business-to-business documents. |
| XSD | XML Schema Data |
| XSL | XML Style Language. A style sheet format for XML documents. It is the XML counterpart to the Cascading Style Sheets (CSS) language in HTML. |