1997 20TH 2017

PESC

MULTIPLE STAKEHOLDERS. ONE VISION.

XML TECHNICAL SPECIFICATION

Version 2.2

April 11, 2016

A publication of the Technical Advisory Board

P20W Education Standards Council (PESC)

PESC APPROVED STANDARDS

FOR ADMISSIONS, REGISTRAR & FINANCIAL AID

Executive Summary

Business Problem
Historically, the companies, agencies, and institutions that comprise the education community at large have developed their systems, data models, and data standards independently. This has resulted in a lack of common data definitions and enterprise data standards across the education domain. While the need to conduct business electronically has increased dramatically in recent years, differing data definitions have made mapping and analyzing data between systems difficult, and, consequently, have hindered the building of standard interfaces that ensure data quality and consistency.

Solution
Recognizing the need to standardize its own business processes and data definitions, the U.S. Department of Education's Office of Federal Student Aid (FSA) in 2000 embarked upon a comprehensive plan to rationalize its existing systems and processes and base data exchange on XML. In 2002, this initiative was aligned with a broader community initiative when FSA began to work with the Postsecondary Electronic Standards Council (PESC) and the National Council of Higher Education Loan Programs (NCHELP) on the development and design of an XML standard for higher education. Based on the principles originally codified in FSA's XML Framework, this standard seeks to provide all stakeholders maximum benefit from the adoption of XML technology through better, faster, and cheaper information exchange.

Now under the direction of PESC, this community-wide standards initiative seeks to establish enterprise definitions for data that is commonly exchanged by members of the education community as well as the policies, procedures, and standards that will guide further developments.

In 2015, the U.S. Department of Education and PESC agreed upon a goal to join a Federal Government wide initiative to use a single model for XML information exchanges. The National Information Exchange Model (NIEM) [1] is the framework in which PESC standards will be developed and documented in the future. In addition, PESC intends to manage the Education Domain within NIEM.

To reach this goal, PESC has determined to evolve its standards so that they will become conformant with NIEM over time. The following steps will be required to meet this goal:

1. Convert core-main and sector library schemas so that they follow the NIEM based naming and design rules [2] specified in this document while maintaining the PESC namespace, library, and import structure. Because many simple element names do not meet the NIEM naming conventions, substitution groups will be added to include the new and old name in the schema so that instance documents will still be backward compatible.
2. When PESC creates a new standard or revises previously released standards, the workgroup will reference the new core and sector schema files. The PESC Technical Advisory Board (TAB) will provide tools to help with this migration.

3. When the domain stewardship of the NIEM Education domain is assigned to PESC, the core and sector schemas will be modified to include NIEM namespaces and schemas and placed in the Education domain as NIEM Reference and Extension schemas. TAB and PESC Change Control Board (CCB) will assure that these domain schemas are NIEM conformant. Substitution groups used to support backward compatibility of element names will be removed.
4. Any new or revised standards will then use the NIEM framework for development of PESC standards.

PESC XML Technical Specification

Information exchange specifications and the data models upon which they are based must expand and evolve as business needs change. To guide these processes, this document, the PESC XML Specification, and its companion volume, the Standards Forum Policies and Procedures Manual [3], have been developed. These documents are to be used in conjunction with the NIEM documentation to provide the methodology, standards, conceptual framework, and policies needed to assemble, maintain, and expand standard, consistent information exchange specifications.

Table of Contents

Executive Summary ..... *i*

Table of Contents ..... *iii*

Tables and Figure ..... *v*

1 Introduction *2*

1.1 Overview **2**

1.2 Purpose **2**

1.3 Scope **2**

1.4 Intended Audience **3**

1.5 Organization of the Document **3**

1.6 Assumptions **3**

1.7 Examples **3**

2 PESC XML Schema Structure *4*

2.1 Naming Conventions **6**

2.1.1 Element Name Examples 9

2.1.2 XML Type Naming 9

2.2 Namespace Conventions **10**

2.2.1 Description 10

2.2.2 Naming of Namespaces 10

2.2.3 Application of Namespaces 10

3 XML Schema Design Rules *13*

3.1 Global Definitions **13**

3.2 Data Definition **14**

3.3 XML Comments **16**

3.4 Element Construction **16**

3.5 Attribute Construction **16**

3.6 Complex Type construction **16**

3.7 SimpleTypes **16**

3.8 Group structure **17**

3.9 Model Groups **17**

3.10 Schema Assembly **17**

4 XML Schema Development Methodology *17*

4.1 Overview **17**

4.2 Gather Requirements and Data Definitions 17
4.3 Look up Data Definitions in Core and Sector Libraries 18
4.4 Metadata Essential for XML Syntax 18
4.5 Data Types 19
4.6 Aggregate Items 19
4.6.1 Specification of Aggregates 19
4.6.2 Issues Concerning Aggregates 20
4.6.3 Analysis Orientation 20
4.7 Assemble Schema 20
4.8 Test Schema 21
4.9 Deploy Schema 21
Appendix A: Revision History 22
Appendix B: References 22

Tables and Figure

Tables

Table 2.1 Representation Terms for names referencing simple content....9
Table 2.2 Element Name Examples....9
Table 3.1 Data definition sentence beginning....15

Figures

Figure 2.1 – Address Line Core Component Example....5

1 Introduction

1.1 Overview

This document, the Guidelines for XML Architecture and Data Modeling, provides information for developers and implementers of PESC-compliant XML. The reference information provided herein is based on standards and best practices that the PESC membership has developed on community-wide XML Schema development projects as constrained by the NIEM Naming and Design Rules [2].

The development of this specification served to clarify, for the Standards Forum, the most efficient work processes and the ultimate deliverables of the standing and ad hoc work groups that make up the Standards Forum for Education. As these work groups and their needs evolve and expand, so will this document, as it will in conjunction with changes to XML and its related standards.

1.2 Purpose

The purpose of this specification is to guide the work of the Standards Forum of Education, providing a framework for decisions that face the following groups:

- The Standards Forum for Education as an organization, as its structure changes to meet the needs of the higher education community
- The education community as it implements XML information exchanges

This document provides the standards and guidelines the education community can use to ensure consistent and efficient development of XML Schemas for information exchanges. It provides numerous XML examples, best practices, and patterns to which developers can refer.

1.3 Scope

The scope of the XML standards in this specification is the **data** that institutions and their partners exchange in support of the business processes within education, such as student financial aid, admissions, and registration. While schemas may be designed for other purposes, such as for data presentation, this is not the primary focus of the Standards Forum.

Since the business processes within Education require data interchange, PESC schemas are data oriented and may in some cases mirror paper-based documents. Their content models focus more on semantics (or "content") than on presentation or structure (where the content model contains some degree of presentation orientation mixed with semantics). Consequently, the guidelines and standards in this specification have a similar orientation.

The Guidelines for XML Architecture and Data Modeling provides introductory information on XML Data Modeling. Specifically, the document includes information on:

- XML Schema Structure
- XML Schema Design Rules

- XML Schema Development Methodology

1.4 Intended Audience

The intended audience of this document is the Standards Forum for Education as well as members of the education community at large wishing to use XML in their data exchanges. It is targeted to advanced XML Schema developers who need to build complex XML Schema message specifications, and assumes that the developer is familiar with XML Schema development.

1.5 Organization of the Document

The Guidelines for XML Architecture and Data Modeling consists of the following sections:

Section 1: Introduction provides a high level overview, scope, and assumptions of this document.

Section 2: PESC XML Schema Structure describes the layered schema structure and conventions that have been adopted by PESC.

Section 3: XML Schema Design Rules lists the key constraints on using XML schema language that will assure that PESC schemas can evolve to become NIEM conformant.

Section 4: XML Schema Development Methodology describes the design/build/test methodology that should be followed when developing PESC-compliant schemas.

Appendix A: Revision History provides a list of changes to this document since its initial publication.

1.6 Assumptions

The Guidelines for XML Architecture and Data Modeling is based on the following assumptions:

- This document is written assuming that the reader has an understanding of XML. It is not intended to be an XML tutorial. It only covers topics that can be clarified for the purposes of standardization. It is intended to be a guide for PESC members and the education community in general, to use when implementing XML throughout the community.
- The reader will have access to the PESC core and sector schemas that are used to construct information exchanges.

1.7 Examples

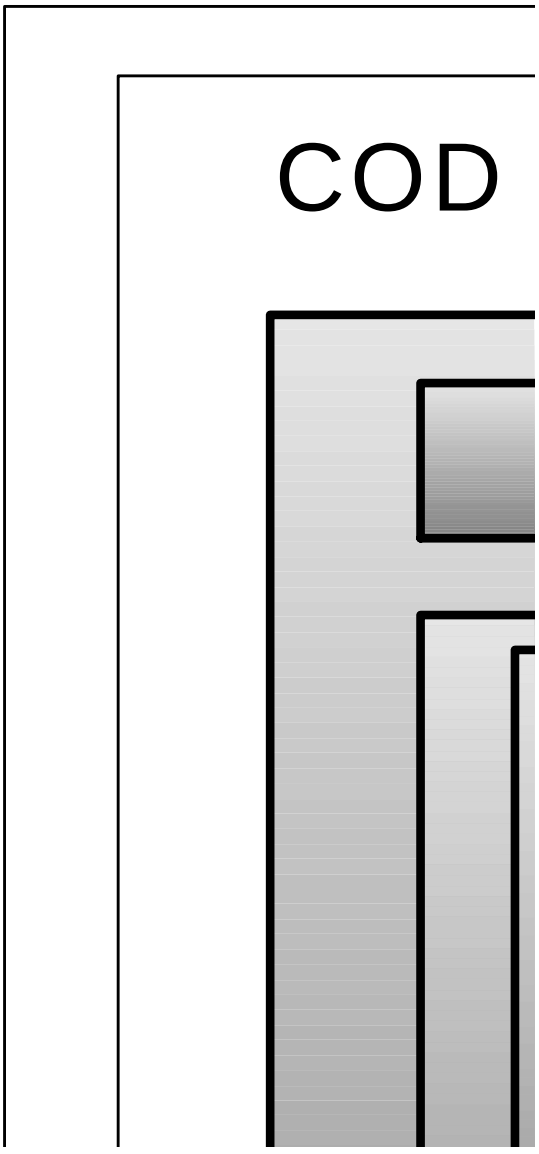
The XML schema examples in this document exclude certain required elements in PESC schemas such as annotation/documentation and the inheritance model for the sake of brevity and exposition. These additional schema content and constraints will be covered in section 3.

2 PESC XML Schema Structure

PESC XML Schema Development shall be done in a tiered manner to obtain maximum reusability. The tiers correspond to the scoping of the elements, from general use across all systems, to specific use for a particular message. The three tiers for development are:

- Core Component Library
- Sector Library
- Message Specification

These tiers, or levels, are illustrated in the following diagram:



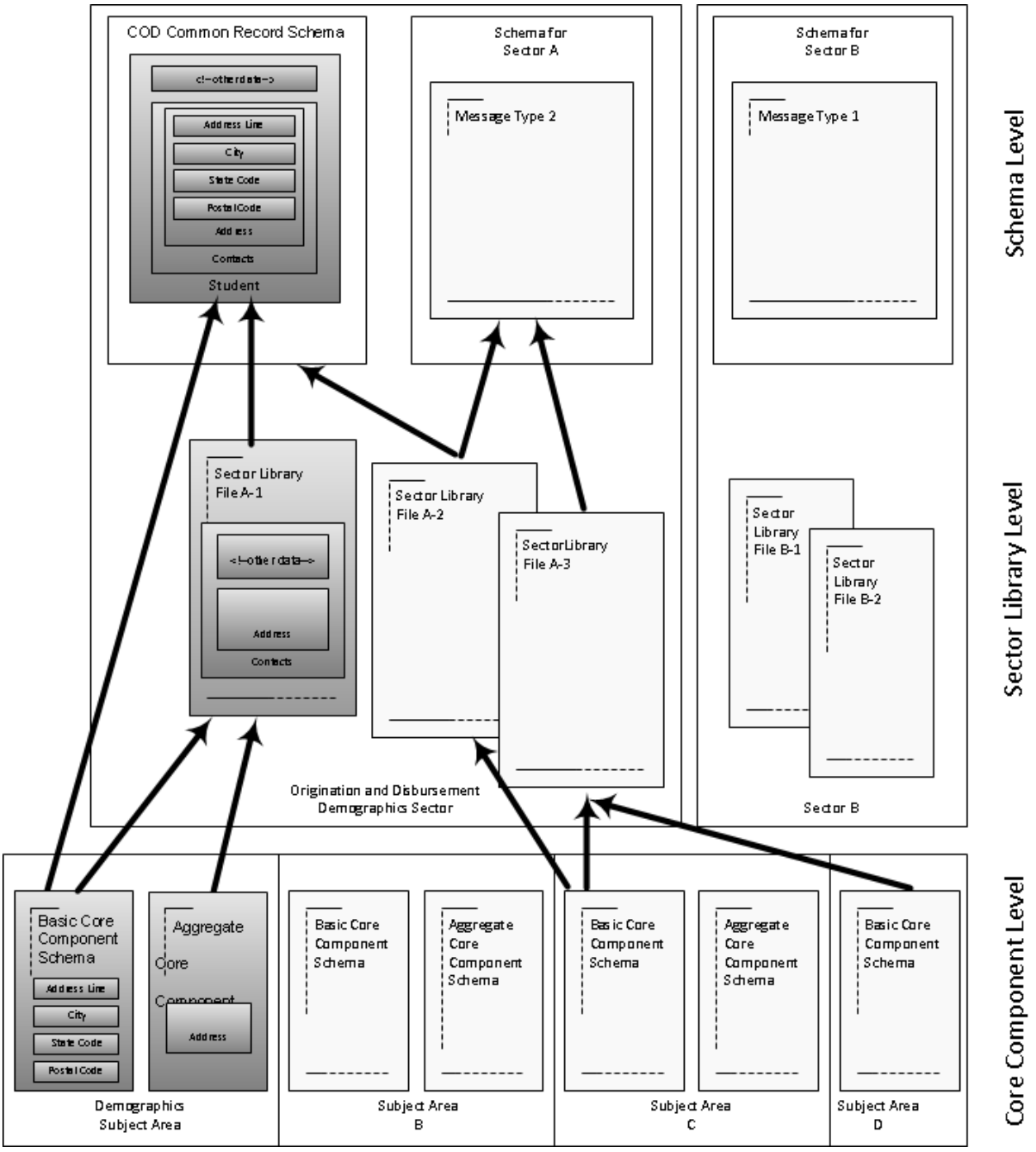


Figure 2.1 – Address Line Core Component Example

The diagram traces the use of a representative data entity, Address, through the levels of the XML modeling architecture, from its definition as a Core Component (bottom level), through its refinement (through the addition of context) into a specific component on the Sector Library Level, to its ultimate inclusion in a Message Specification Schema.

2.1 Naming Conventions

PESC's Core Component Naming convention is based on the Core Component naming convention described in the NIEM Naming and Design Rules [2]. The NDR rule number is associated with the sections below. While these rules apply to schema names, the name attribute defined in the "element" tags in the schema will also be used directly in exchange documents. The naming uses the notion of objects and their properties familiar to those familiar with UML and object oriented programming experience. A name is made up of the following components [Rule 10-55]:

- Object Qualifier Terms (0 or more)
- Object Class Terms (1)
- Property Qualifier Terms (0 or more)
- Property Terms (1)
- Representation Qualifier Terms (0 or more)
- Representation Terms (0 or 1)

PESC schema component names must use only the following characters [Rule 10-44]:

- Upper-case letters (`A–Z`)
- Lower-case letters (`a–z`)
- Digits (`0–9`)
- Hyphen (`-`)

The Hyphen (`-`) should only be used to separate parts of a single word phrase or value to make it comprehensible [Rule 10-45].

Other characters, such as the underscore (_) character and the period (`.`) character MUST NOT appear in component names in NIEM-conformant schemas [Rule 10-44].

PESC's Education Domain Element Naming Convention uses the following rules to produce a name:

- XML schema component names are English words [Rule 10-43]
- XML schema component names (`name = 'Name'`) are case sensitive. The first letter of each concatenated word should be in uppercase (unless an attribute), and the rest of the letters should be lowercase [Rule 10-48].
- Attribute names start with a lower case while all other schema components start with upper case [Rule 10-49].
- Nouns should be singular unless the concept is plural [Rule 10-52].
- Verbs should be in the present tense unless the concept itself is past tense [Rule 10-53].
- Articles, conjunctions, and prepositions should be avoided unless absolutely necessary for clarity [Rule 10-54].
- Length should be considered when creating schema component names (especially for XML element names), but not to the point of sacrificing context. Well known abbreviations and acronyms may be used, but only if commonly known and context is not lost [Rule 10-49].

- The name of a qualified *Object Class* shall be unique throughout the Schema and may consist of more than one word [Rule 10-56].
- The name of a qualified *Property Term* shall occur naturally in the definition and may consist of more than one word. A name of a *Property Term* shall be unique within the context of an *Object Class* but may be reused across different *Object Classes* [Rule 10-57].
- If the name of the *Property Term* uses the same word as the *Representation Term* (or an equivalent word), the *Representation Term* may be omitted [Rule 10-61].
- The name of the *Representation Term* must be one of the terms specified in the *List of Representation Terms* as included in this document and the NDR [Rule 10-62].
- A simple type with the "Code" representation term must use an enumeration list or be derived from a code type [11-10].
- Representation terms are used in all simple content components and only in complex content components that correspond to a content that is found in the Table 1 [Rules 10-63, 10-64].

Primary Representation Term	Secondary Representation Term	Definition
Amount	-	
BinaryObject	-	A set of finite-length sequences of binary octets.
	Graphic	A diagram, graph, mathematical curves, or similar representation.
	Picture	A visual representation of a person, object, or scene.
	Sound	A representation for audio.
	Video	A motion picture representation; may include audio encoded within.
Code		A character string (i.e.,letters, figures, and symbols) that for brevity, language independence, or precision represents a definitive value of an attribute.
DateTime		A particular point in the progression of time together with relevant supplementary information.
	Date	A particular day, month, and year in the Gregorian calendar.
	Time	A particular point in the progression of time within an unspecified 24-hour day.
	Duration	An amount of time; the length of a time span.
ID		A character string to identify and distinguish uniquely one instance of an object in an identification scheme from all other objects in the same scheme together with relevant supplementary information.
	URI	A string of characters used to identify (or name) a resource. The main purpose of this identifier is to enable interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols. A URI is either a Uniform Resource Locator (URL) or a Uniform Resource Name (URN). The specific syntax for each is defined by **[RFC 3986]**.
Indicator		A list of two mutually exclusive Boolean values that express the only possible states of a property.
Measure		A numeric value determined by measuring an object along with the specified unit of measure.
Numeric		Numeric information that is assigned or is determined by calculation, counting, or sequencing. It does not require a unit of quantity or unit of measure.
	Value	A result of a calculation.
	Rate	A representation of a ratio where the two units are not included.

Primary Representation Term	Secondary Representation Term	Definition
	Percent	A representation of a ratio in which the two units are the same.
Quantity		A counted number of nonmonetary units possibly including fractions.
Text	-	A character string (i.e., a finite sequence of characters) generally in the form of words of a language.
	Name	A word or phrase that constitutes the distinctive designation of a person, place, thing, or concept.
	List	A sequence of values. This representation term is used in tandem with another of the listed representation terms.

Table 2.1 Representation Terms for names referencing simple content

2.1.1 Element Name Examples

Below are examples of simple content element names constructed using this pattern:

Table 2.2 Element Name Examples

Object Qualifier	Object	Property	Representation Term	Element Name
High School	Student	Birth	Date	HighSchoolStudentBirthDate
-	Student	GradeLevel	Code	StudentGradeLevelCode
-	Course	Description	Text	CourseDescriptionText
-	Student	Applied	Indicator	StudentAppliedIndicator

2.1.2 XML Type Naming

In addition to the naming rules above, type names have a naming convention:

- Complex type names must end with "Type" [Rule 11-1]
- Simple type names must end with "SimpleType" [Rule 11-4]. (The current PESC schemas using "Type" instead of "SimpleType" will not be converted until creating NIEM reference schemas, but any new types should use this convention.)

For example, the elements illustrated above could have schema type names of:

- HighSchoolStudentBirthDateSimpleType
- StudentGradeLevelCodeSimpleType
- CourseDescriptionTextSimpleType
- StudentAppliedIndicatorSimpleType

A complex element for StudentAddress would be named StudentAddressType.

Note: When an object is a specialization of a more general object, it may be possible to reuse type definition using a more general definition. For example, we could use "PersonAddressType" to define addresses for all specializations of the Person Object including Student.

2.2 Namespace Conventions

2.2.1 Description

According to the World Wide Web Consortium (W3C), the purpose of XML namespaces is to "provide a simple method for qualifying element and attribute names used in Extensible Markup Language documents by associating them with namespaces identified in Uniform Resource Identifiers (URI) references."

In other words, XML Namespaces allow an organization to group their element names and attribute names in such a way to prevent conflicts in like named elements and attributes. This is done to manage element names and attribute names within a single organization and between multiple organizations.

PESC has developed an approach for the namespaces it will use, and it is defined in the following section.

2.2.2 Naming of Namespaces

The general approach for PESC namespaces is based on the guidelines in XML.gov [4], as well as the ASC X12 Reference Model for XML Design [5]. The root of all namespaces PESC will use in its XML Libraries will begin with the organizational identifier:

```
urn:org:pesc:
```

After the organizational identifier, additional terms will be used to indicate the schema (core, sector, or message), the schema name, and the schema version. Below are some examples of current name spaces:

Core schema: **urn:org:pesc:core:CoreMain:v1.13.0**
Sector schema: **urn:org:pesc:sector:AcademicRecord:v1.7.0**
Message schema: **urn:org:pesc:message:HighSchoolTranscript:v1.3.0**

2.2.3 Application of Namespaces

2.2.3.1 Description

Before discussing namespace usage specifically, consider a situation with four documents:

1. A Base Schema file, holding Core Component Definitions
2. A Sector Library Schema file, holding sector-level XML Schema objects
3. A Message Specification Schema file, holding the final XML message definition
4. An XML "Instance Document", holding a file that conforms to the Message Specification

Each of the first three documents contains XML Schema objects, which may be part of different namespaces. The elements need to be used in the XML Instance Document, with minimal complexity, but with having their unique locations (through namespace identification) preserved. This section describes exactly how each XML Schema document should be constructed to meet these goals.

There are several XML Schema mechanisms that function together to support namespace usage in XML documents. These mechanisms include:

Core Component XML Schema:

- Setting the `targetNamespace` attribute
- Setting the `elementFormDefault` and `attributeFormDefault` attributes to indicate how elements will be qualified in an instance document using this schema

Sector Library Schema

- Using `xs:import` as appropriate, to include the Core Component XML Schema
- Setting the `targetNamespace` attribute
- Setting the `elementFormDefault` and `attributeFormDefault` attributes to indicate how elements will be qualified in an instance document using this schema

Message Specification XML Schema

- Using `xs:import` as appropriate, to include the Sector Library XML Schema
- Setting the `targetNamespace` attribute

XML Instance Document

- Setting the `xmlns:[prefix]` attribute to reference the Message Specification
- Setting the `schemaLocation` attribute to reference the Message Specification
- Setting the `namespace` prefix on the elements of the Instance Document

2.2.3.2 Approach

There are several different ways to use these different settings within a set of included/including schemas, and the instance document that refers to them. Rather than list all of the combinations possible, we will focus on the recommended approach for PESC XML development. The approach is as follows:

- Core Component XML Schema:
  - Set the `targetNamespace` attribute to a value appropriate for the Core Component Library, such as:
    ```
    <xs:schema targetNamespace=
    "urn:org:pesc:message:HighSchoolTranscript:v1.3.0"
    ```
  - Set the attributes `elementFormDefault="unqualified"` and `attributeFormDefault="unqualified"`
- Sector Library Schema
  - Use the `xs:import` directive to import the Core Component XML Schema
  - Set the `xmlns:[prefix]` attribute to reference the Core Component Library
  - Set the `targetNamespace` attribute to a value appropriate for the Sector Library, such as:
    ```
    <xs:schema
    targetNamespace="urn:org:pesc:sector:thissector:v1.0.0"
    ```

    - Set the attributes elementFormDefault="unqualified" and attributeFormDefault="unqualified"
    - Create all element objects in this schema (do not use ones declared in the Core Component Library) - use only the `simpleType` and `complexType` objects from the Core Component Library, using the namespace prefix to qualify them.
- Message Specification XML Schema
    - Use the `xs:import` directive to import the Sector Library XML Schema
    - Set the `xmlns:[prefix]` attribute to reference the Sector Library
    - Set the `targetNamespace` attribute to a value appropriate for the Message Specification, such as:
      `<xs:schema targetNamespace="urn:org:pesc:message:thismessage:v1.0.0">`
    - Set the attributes `elementFormDefault="unqualified"` and `attributeFormDefault="unqualified"`
    - Create all top-level element objects in this schema (do not use ones declared in the Sector Library) - use only the `simpleType` and `complexType` objects from the Sector Library, using the namespace prefix to qualify them.
- XML Instance Document
    - Set the `xmlns:[prefix]` attribute to reference the Message Specification
    - Set the `schemaLocation` attribute to reference the Message Specification
    - Set the namespace prefix on the root element of the Instance Document to the one set for the `xmlns:[prefix]` attribute.

This approach will preserve unique traceability of types and elements through the different layers of the PESC XML Libraries, but require few steps on the part of the Instance Document authors in order to be compliant. A sample instance document would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<root:RootTagOfDocument
xmlns:root="urn:org:pesc:message:thismessage:v1.0.0"
xmlns:sector="urn:org:pesc:sector:thissector:v1.0.0"
xmlns:core="urn:org:pesc:core:CoreMain:V1.13"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:org:pesc:message:thismessage:v1.0.0
MessageSpec.xsd">
    <DocumentID>2003-08-17T09:30:47-05:00BatchID</DocumentID>
<!-- all other elements do not need to be qualified except for the root --
>
</root:RootTagOfDocument>
```

3 XML Schema Design Rules

3.1 Global Definitions

To avoid multiple definitions of the same name, NIEM requires all elements [Rule 9-35], attributes [Rule 9-47], and types [Rule 9-10, Rule 9-24] to be defined at the top level of the schema; that is, as direct children of the <xs:schema> element. Elements will then be referenced using the "ref" attribute in complex element and type definitions.

Example:

```
<xs:element  name="EmailAddress" type="core:EmailAddressType">
   .…
</xs:element>
<xs:attribute name="languageCode" type="core:languageCodeType"/>
<xs:element name="NoteMessage" type="core:NoteMessageType">
   .…
</xs:element>

<xs:attribute name="languageCode" type="core:LanguageCodeType>
   ….
</xs:attribute>
<!--The complex type below references the top level definitions above-->
<xs:complexType name="Email>
   <xs:ComplexContent>
      …
      <xs:sequence>
         <xs:element ref="core:EmailAddress"
minOccurs="0"maxOccurs="unbounded" />
         <xs:element ref="core:NoteMessage"/>
      </xs:sequence>
      <xs:attribute ref="core:languageCode"/>
   </xs:ComplexContent>
    . . .
</xs:complexType>
```

However, since current PESC instance documents are based upon schemas that use elementFormDefault = "unqualified" and PESC does not use global element definitions in its schemas, all elements except the root tag are not qualified with a prefix in current instance documents. Unfortunately, the XML Schema specification requires all elements defined as global elements in an XML schema to be qualified even if elementFormDefault="unqualified". As a result, if the PESC schemas are converted to have global elements, all PESC instant document tags would need to be qualified when a new version of the message specification was released. We believe that this would create programming work that is unnecessary and would discourage the community from using PESC standards.

To assure that PESC complies with the intent of NIEM global schema elements; that is, element names have one and only one definition, the CCB will enforce the rule that all schema elements have exactly the same type definition so, for example, if an element in one complex type is <xs:element name="A" type="core:Atype"/>, all definitions of an element named "A" must be

of type= "core:AType". Also, no element definition in a complex type will contain a <xs:simpleType> or <xs:complexType> local definition. With this in mind the example above would look like this:

```
<xs:complexType name="Email>
   <xs:ComplexContent>
      …
      <xs:sequence>
         <xs:element name="EmailAddress" type ="core:EMailAddressType
minOccurs="0"maxOccurs="unbounded" />
         <xs:element name="NoteMessage" type="core:NoteMessageType/>
      </xs:sequence>
      <xs:attribute name="languageCode" type="core:LanguageCodeType"/>
   </xs:ComplexContent>
    . . .
</xs:complexType>
```

3.2 Data Definition

To simplify automatic generation of documentation and to facilitate the human understanding, a textual definition needs to be provided for the declarations of elements [Rule 9-36], types [Rule 9-12, Rule 9-25], enumeration values [Rule 9-23], and attributes [Rule 9-48] using the `<xsd:annotation>` and `<xsd:documentation>` constructs.

The following table identifies the opening phrases used to introduce a data definition [Rule 11-31, 11-32, 11-33]:

Component Type	Starts with	Example
Simple Element	A <optional adjective><representation term or synonym>	BirthDate: A date on which the person was born StudentAttendanceCount: A count of the number of students attending class PersonFirstName: A first name of a person
Simple Element Indicator	True if <condition> false otherwise	StudentIndicator: True if a student; false otherwise
Simple Element Code	A code	TestNameCode: A code identifying a specific assessment
Complex Element	A or An	StudentRecord: A record of the accomplishments of a student
Complex Type	A data type	StudentRecordType: A data type that describes a record of the accomplishments of a student

Component Type	Starts with	Example
Simple Type	A data type	TestScoreValueSimpleType: a data type that describes the numeric score obtained on an assessment

Table 3.1 Data definition sentence beginning

Element: (future implementations)

```
<xs:element name="EmailAddress" type="core:EmailAddressType">
   <xs:annotation>
      <xs:documentation>The numbers, letters, and symbols used to identify an
electronic mail (Email) user within a network. </xs:documentation>
   </xs:annotation>
</xs:element>
```

Type:

```
<xs:complexType name="EmailType">
   <xs:annotation>
      <xs:documentation>A data type that encodes the numbers, letters, and
symbols used to identify an electronic mail (Email) user within a network
</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
      <xs:sequence>
         <xs:element name="EmailAddress" type="core:EmailAddress/>
         <xs:element name="NoteMessage" type="core:NoteMessage/>
      </xs:sequence>
   </xs:complexContent>
</xs:complexType>
```

Enumeration:

```
<xs:simpleType name="InstructionalActivityStatusCodeType">
   <xs:annotation>
      <xs:documentation>A simple type defining the academic status of a
student as determined by the method of instruction</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
      <xs:enumeration value="Regular" >
         <xs:annotation>
            <xs:documentation>Regular Student</xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      <xs:enumeration value="Correspondence" >
         <xs:annotation>
            <xs:documentation>Correspondence Student</xs:documentation>
         </xs:annotation>
      </xs:enumeration>

      <xs:enumeration value="Extension" >
         <xs:annotation>
            <xs:documentation>Extension Student </xs:documentation>
         </xs:annotation>
      </xs:enumeration>
      . . . .
```

```
  </xs:restriction>
</xs:simpleType>
```

3.3 XML Comments

Since top level comments cannot be associated with a particular element or type, comments may become separated from the constructs they are describing during editing or sorting. The annotation/documentation provides a superior mechanism for providing human readable explanations of the semantics of the construct. A schema with comments will not be flagged as non-conformant, but the practice is discouraged [Rule 9-77].

3.4 Element Construction

In NIEM, all elements are constructed from types that are extended with certain attributes. In NIEM, an element may not be defined using a `SimpleType` [Rule 11-6]. However, since PESC is not implementing the object model of NIEM at this time, and is not converting our `SimpleType` names to include "SimpleType", this rule will not be implemented until PESC is steward of the Education Domain.

- See the naming conventions for elements [Section 2.1]
- In addition to a name, an element definition must have a type attribute unless it is abstract [9-37].
- No element may have a default value (default attribute) [9-44]
- No element may have a fixed value (fixed attribute) [9-45]

3.5 Attribute Construction

- See the naming conventions for attributes [Section 2.1]
- An attribute definition has a type attribute [9-49]
- No attribute may have default values (default attribute) [9-56]
- No attribute may have fixed values (fixed attribute) [9-57]

3.6 Complex Type construction

Complex types with mixed content, a type defining both simple content and child elements, are not allowed [Rule 9-26, Rule 9-27].

It is not required in the W3C Schema language that you explicitly use `xs:complexContent` or `xs:simpleContent` in a complex type definitions. For PESC schemas to be NIEM conformant, all complex type definitions must include xs:simpleContent or xs:complexContent as a children of `xs:extension` or `xs:restriction` [Rule 9-28]. Note: `xs:simpleContent` is used in a complex type when the element has a text value but also has attributes.

3.7 SimpleTypes

Simple types have the following characteristics:

- Simple types must be declared at the top level of the schema.

- The name of the simple type must end in "SimpleType"[Rule 11-4].
- The name of the simple type must be in upper camel case [Rule 11-5].
- A simple type must have an annotation/documentation specifying its semantic definition

3.8 Group structure

The `xs:group` construct is used to provide a reusable set of elements to insert into types and elements. Because NIEM does not have a semantic model for groups, it is not allowed in NIEM conformant schemas [Rule 9-74].

3.9 Model Groups

W3C Schema language provides three model group constructs for organizing elements. `xs:all` allows all elements in any order in the instance document; `xs:choice` allows only a subset (usually one) of the elements ; and `xs:sequence` restricts the elements to the order they appear in the schema. NIEM disallows the `xs:all` construct [Rule 9-60] and requires `xs:choice` to be a child of `xs:sequence` [Rule 9-64] so that the complex type or element may be extended with more elements in a schema.

3.10 Schema Assembly

To construct a NIEM conformant schema, only the `xs:import` construct may be used to reference external schemas. The `xs:redefine` and `xs:include` are not allowed in NIEM conformant schemas [Rule 9-87, Rule 9-88].

In addition, each schema imported must have a namespace associated with it [Rule 9-89]:

```
<xs:import namespace="urn:org:pesc:message:HighSchoolTranscript:v1.3.0"
schemaLocation="HSTV1.3.0.xsd"/>
```

4 XML Schema Development Methodology

4.1 Overview

Designing an XML Schema is like designing an application, and it should follow a design, build, test life cycle. The basics steps are as follows:

- Gather Requirements and Data Definitions
- Look up Data Definitions in core and sector schemas
- Assemble Schema
- Test Schema
- Deploy Schema

4.2 Gather Requirements and Data Definitions

Just like building any new system, building an XML Schema starts with good requirements. This includes understanding the business process that will be modeled in XML, knowing the different business scenarios, and what data is modeled in each of the scenarios. It is important

for the Schema designer to understand what data will be needed and how the data is going to be used.

4.3 Look up Data Definitions in Core and Sector Libraries

During this phase of Schema development, a Schema designer will search the core-main and sector schemas for the data definitions gathered during the requirements gathering phase. For each data element that a Schema designer searches for, there are three possibilities.

- **Exact Match –** The Schema designer will use the Core Component, as is, to model this piece of data in XML.
- **Close Match –** The Schema designer will evaluate if a modification can be made to the Core Component that will allow this core component to be used in this Schema. If a modification can be made to update a Core Component, then the Schema designer must go through the change process outlined in the PESC Policies and Procedures Manual [4]. If this change cannot be made, then the Schema designer must follow the Enhancement Request Management process to create a new Core Component.
- **No Match –** The Schema designer must go through the Enhancement Request Management process to create a new Core Component, and get it stored into the XML Registry and Repository.

4.4 Metadata Essential for XML Syntax

To facilitate creation of schemas, the following metadata items should be recorded (these should not be considered a limit) in the data dictionary for each element.

- Simple Types
  - Element name
  - Data type (string, date, number, etc.)
  - Cardinality rules
  - Element description
  - Element equivalence in other transaction(s)
  - Element facets such as minimum length**,** maximum length, minimum values, maximum values, or patterns depending on the data type.
  - Values of code elements
- Aggregate Items (Complex Types)
  - Element name
  - Element sequence
  - Cardinality rules
  - Element description
  - Element equivalence in other transaction(s)

4.5 Data Types

The following simplified list of datatypes should 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*) - minimum length, maximum 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: Data Types Regular Expressions*. If a pattern facet is specified in the Core schema it may be modified by a Sector schema as long as that modification is a subset of the Core schema pattern. If an element contains a member of a list, all potential list values must be specified.
- **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.

4.6 Aggregate Items

4.6.1 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 common 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).
- 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.)

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

4.6.2 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 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 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 should be the responsibility of the business applications that use the data.

4.6.3 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 "entity" is defined rather than specifying "student", "lender", or "servicer" separately.)

This approach enhances reusability and simplifies maintenance.

4.7 Assemble Schema

Once the Schema designer has located the entire set of core components needed to model this interface, the Schema designer is now ready to combine these core components into a Schema. The Schema designer should attempt to leverage existing sector libraries, and existing message specifications when building a new schema. These existing resources will save the Schema designer time, and limit mistakes. The Schema designer should follow the design patterns and best practices outlined in this document, the PESC XML Technical Specification.

4.8 Test Schema

A Schema should go through two phases of testing. The first phase of testing is to try and model each of the required business scenarios. If any issues are found in this phase of testing, then the Schema designer must correct them by either going back to the Core Component Registry and Repository, or correcting the assembly of the Schema. The second phase of testing is Inter System Testing. The Schema should be used during a System's IST to test out the interface.

4.9 Deploy Schema

Once a Schema has completed the testing phase of Schema development, it is ready to be deployed. A Schema's deployment process should follow the same deployment process that application code goes through.

Appendix A: Revision History

DATE	SECTION/ PAGE	DESCRIPTION	Version	MADE BY
2/20/05	Whole Document	Initial Revision	1.0	M. Bolembach
2/15/15	Whole Document	Revised to use those NIEM rules that are useful in structuring PESC schemas but do not require NIEM schemas	2.0	PESC TAB
8/5/15	3.1 Global Definitions	Delay implementation of global element rule by assuring element definition equality in complex types	2.1	PESC TAB
4/11/16	Title/Footer/Header/ Body	Changed the title to PESC XML Technical Specification	2.2	PESC TAB

Appendix B: References

[1] NEIM, https://www.niem.gov/technical/Pages/version-3.aspx

[2] NIEM Technical Architecture Committee (NTAC), National Information Exchange Model Naming and Design Rules, V3.0, 2014

[3] PESC Standards Forum for Education, PESC Policies and Procedures

[4] XML.gov

[5] ASC X12, Reference Model for XML Design