COMMONWEALTH OF PENNSYLVANIA
DEPARTMENT OF PUBLIC WELFARE

INFORMATION TECHNOLOGY STANDARD

Name Of Standard: **Enterprise XML Standards**	Number: **STD-EASS002**
Domain: **Componentware**	Category: **Data / Enterprise XML Standards**
Date Issued: **New**	Issued By Direction Of:
Date Revised: **11/19/2007**	James Weaver, Dir of Div of Tech Engineering

Abstract:

The purpose of this document is to establish standards for Extensible Markup Language (XML) schemas for Application Program Interfaces (APIs) and their associated data type libraries within the Department of Public Welfare.

General:

For an overview of XML, basic terminology, components, and general industry guidelines please refer to tutorials and documentation based on the World Wide Web Consortium's (W3C) standards which can be located at www.w3.org/xml/schema.html.

These XML standards addresses are to be used when creating APIs, when data is being shared between the Department and its business partners, or between applications within the Department. A business partner can be an external business entity (e.g. Social Security Administration, Internal Revenue Service, etc.) or another agency within the Commonwealth (e.g. Revenue, Labor and Industry, etc.).

Standard:

All XML schemas and documents created within DPW will comply with the general W3C standards and the specific DPW standards described in this section. Any deviations from DPW standards will require Data Administration (DA) and Department approval.

The DPW standard indicates that a schema is to be used rather than a Data Type Definition (DTD). This is due to the inherent lack of flexibility in DTDs in defining data within an XML document. The use of a DTD instead of a schema will require the approval of the Data Administrator (DA) and the Department. Those applications already using DTD will be permitted to continue with their application. However, during any reconstruction, restructuring or changes to the application, the developer will be expected to switch from DTDs to schemas.

An XML schema will be placed in a schema library where it can be accessed by other applications requiring access to the XML documents described by that schema. XML will be used as a mechanism for transferring data between applications via COM+, DCOM, SOAP, and other interfaces. All such interfaces will be represented by schemas in the schema library as long as they are able to parse the XML schemas. The schema will also be presented in the Department's data dictionary as a data element construct. The data dictionary tool will provide a hyperlink to the common type library where the XML schema resides

Use of the schema standard implies the use of Namespaces, the extensive use of publicly available XML schemas, and a strong recommendation to publish and document all XML schemas used within the Pennsylvania Department of Public Welfare. By creating standard XML element schemas, the overall number of schemas and the maintenance overhead is reduced. Thus XML schemas must be designed to support multiple functions whenever possible.

Where XML schemas are used to define interfaces to application functions, the schemas exist in the DPW enterprise, as well as, in the enterprise of the business partner. In general, a set of schemas are defined for each method in the API.

The schemas should be named for the method that uses them, with a suffix indicating the direction of the data flow represented by the schema.

This example is from the PROMISe application:

Method name:	GetTPLResourceInfo
Input schema name:	GetTPLResourceInfoReq
Output schema name:	GetTPLResourceInfoResp

This example is from the MCI application:

Method name:	CaseInitiation
Input schema name:	CaseInitiationIn
Output schema name:	CaseInitiationOut

This standard does not address the physical location of the schemas. The present practice is for schemas to be stored in Visual Source Safe and to be deployed to the file system of the DPW server where they are used. For schemas used by business partners, a copy of the schema is given to the business partner for use by that enterprise.

Future applications will mandate that some schemas be accessible by multiple business partners or the general public. A commonly accessible repository must be identified to support this requirement.

Documenting Schemas

The convention in documenting schemas is to use both the annotation and comment element. In the H-Net data type libraries, the following convention is currently being used:

```
<!--Minor Civil Division-->
    <xsd:simpleType name="MinorCivilDivision">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
                Minor Civil Division of the entity's residence.
            </xsd:documentation>
        </xsd:annotation>
```

```
    <xsd:restriction base="xsd:string">
      <xsd:maxLength value="20"/>
    </xsd:restriction>
  </xsd:simpleType>
```

In the above example, the <!-- Minor Civil Division -- > represents a name, while the <xsd:annotation> <xsd:documentation xml:lang=”en”> represents the annotation.

The rationale behind this approach was to make the data type libraries easier to read for people as they contain a number of individual simple and complex element constructs.

The Department’s accepted standard for XML documents and schemas is to use the <xsd:annotation> element. This element contains two child elements. The first is <xsd:appinfo> which can be used for XML parsers to process comments. The other <xsd:documentation> allows add-in comments for human consumption.

It should be noted that no data will be housed in comments. The comments are available for someone viewing or reading the XML document or schema but are not for drawing data out of applications.

Schema Structure

In the past, the structure of XML schemas was not to exceed five (5) layers or levels. This has proved to be an unsupportable standard as more applications require a greater indentation level. If the application required more than five layers the schema, it was to be justified and approved through the DA. The current standard will be to use the least amount of indentation that is reasonable based on the application. The goal is to make the XML schema usable in more than one function. XML does not have a physical limit for the number of layers that a structure can extend but limiting the number of layers will allow for more efficient code. The structure will extend to the lowest level that contains meaningful and pertinent data elements.

Tag Naming Standards

Individual data elements defined within an XML schema are called tags. In naming a tag, the following standards apply:

- The tag name will be the Official Name (as documented in the data dictionary) less white space found in the Department’s data dictionary for that element. This name is not the database name but the definitive name given that particular database name as defined in the data dictionary and must be a full English word that clearly conveys the data element meaning.
- When defining a type in an XML schema, if composing a complexType string the standard is to include the word “Type” at the end of the descriptive name. This practice will identify this mini-schema as a complex type and allow for easy access and reuse.

XML Data Types

The Department’s data dictionary will contain the data types for SQL Server, Oracle, the mainframe, and XML. The XML data type for a data element must be compatible with these other non-indexed data types for this element as defined within the data dictionary.

Aggregate construction will depend on business rules and how the data is normally used. The convention is for the tag to be broken down to the simplest form that is typically used or known enterprise-wide. The following examples explain this further:

- ZipCodeMain and ZipCodeExtension will be two separate XML tags because there are times when only ZipCodeMain is required for applications. If both forms of the zip code were included as a single XML tag, applications would require users to enter both parts of that zip code.
- The standard convention for date is to use the full date string. Applications can break the string down further on their own if necessary but the enterprise standard is as follows: date information will be all in one string and use the **date** type as defined in the W3C schema standard. The **date** type represents dates in the following format: yyyy-mm-dd. This format requires the use of leading zeros when single digit dates are used (Example: February 3, 2002 would be written 2002-02-03)

The data type used will follow the W3C conventions for data types. The application must use the data type that will correctly represent the typical usage of that data.

Data Type Libraries

Often the developers of XML schemas will want to create simple and complex types that can be shared and used as building blocks within other XML schemas. Some of the standard types are already provided through the XML Introductory type library. However, developers may want to create their own types to be held in a centralized repository called a data type library.

All of the MCI/MPI data elements defined by the business team that are defined as a type are located in a data type library. It is elements such as these or other data elements that are referred to in the actual schemas. Instead of having one single data type library, the libraries will be broken into several different files. One of these files will be the data type library for those types that are truly common and are used most often by applications. The second and subsequent files will contain application specific data type libraries. When an application includes a common type library in an XML schema the schema will require a good bit of parsing, but if the files are smaller the parsing time will be greatly reduced. Breaking up the common type library as stated above will remedy these difficulties.

Within the data type library, only those elements that exist as defined by a type will be permitted. If an element is not defined as a type (either simple or complex) it will not be included in the data type library but within the schema that defines it. Take the following example:

PriorName and BirthName will not be part of the data type library because both are simple elements that follow the complex type “NameType” data element definition. When defining Prior Name or Birth Name in an XML Schema, the appropriate method follows:

```
<xsd:element name="PriorName" type="NameType"/>
<xsd:element name="BirthName" type="NameType"/>
```

Notice that the element name is listed and then refers to the type as defined in the data type library.

The data type library can be found in Visual Source Safe in the following location:

DPW Enterprise IntranetSchemas\<environment>\SchemaApplication\Common

where <environment> is A (production), C (TFP) or D (Development)

String and Numeric Data Types

It is generally accepted that if a field is of data on which mathematical operations could reasonably be performed, or is used as a surrogate key determined by a mathematical operation (such as a simple ascending number often used as a primary key) then the field should be defined as a numeric data type. Otherwise it should be defined as a text data type of appropriate size.

On this basis, code fields (such as county code or state code) should not be defined as numeric. The same arguments can apply to other ”numbers’’ that generally use numeric characters but are not used for arithmetic or keys (telephone numbers, social security numbers, zip codes, etc). This standard eliminates the concerns about the suppression of leading zeros common with most numeric data types.

Optional / Nillable Elements

Optional (specified by minOccurs=0) is used on an element to indicate that the element may be omitted from the XML. The nillable attribute is used on an element to indicate that the element may have a null value. These properties may be used separately or in combination for different purposes. The following describes how these properties might be used on schemas for API’s.

If neither property is assigned to an element then the element must be present and must have a non-null value. Such an element may be described as a required field. This is appropriate for the schema for an API where the method will do an insert to row in a database and where the element is a required field in the database.

Nillable without Optional indicates that the element must be present and may have a null value. This is appropriate for the schema of an API where the method will do an insert to a row in database and where the element is permitted to have no value (null).

Optional without Nillable indicates that the element may be absent, but if it is present it must have a non-null value. This is appropriate for the schema for an API where the method will do an update to a row in a database and 1) if the element is absent it will not be updated and 2) if the element is present it is updated to a non-null value.

Optional and Nillable indicates that the element may be absent, and if it is present it may be null. This is appropriate for the schema of an API where the method will do an update to a row in a database and 1) if the element is absent it will not be updated and 2) if the element is present it may be updated to a null or non-null value.

Exemptions from this Standard:

Any deviations from this published standard must be reviewed and approved by the Department’s Data Administration (DA) Section within the Division of Technology Engineering (DTE).

Refresh Schedule:

All standards and referenced documentation identified in this standard will be subject to review and possible revision annually or upon request by the DPW Information Technology Standards Team.

Standard Revision Log:

Change Date	Version	Change Description	Author and Organization
05/09/01	1.0	Initial creation – XML Schemas for APIs	Unknown
8/19/02	1.1	Edited XML Schemas for APIs document for style	Beverly Shultz
10/25/02	1.0	Initial creation – XML Data Domain Standards	Stephen Stoner/Srini Subramanian
12/04/02	1.1	Edited for style - XML Data Domain Standards	Beverly Shultz Diverse Technologies Corporation / Deloitte Consulting
6/25/03	1.2	Replacing an existing standard - XML Data Domain Standards	Heather Brandt
5/26/04	1.2	Reviewed XML Data Domain Standards content – no change necessary Edited XML Schemas for APIs document for content – no change necessary	Dale Woolridge
6/06/06	1.1	Reviewed XML Schemas for API's document for content – no change necessary	Dale Woolridge
6/13/06	1.2	Reviewed XML Data Domain Standards content – no change necessary	Dale Woolridge
11/19/07	2.0	Updated content of XML Data Domain Standards and XML Schemas for API's and combined the standards into one document	DTE / DADD