<xmlLegal> Schema Framework Reference Documents

By Winchel "Todd" Vincent III

Winchel "Todd" Vincent III is an attorney and technical consultant. Mr. Vincent has been Project Director of Georgia State University Electronic Court Filing Project (http://e-ct-file.gsu.edu/) since 1997. He is currently the lead consultant on the California Administrative Office of the Court Second Generation Electronic Filing Project ("2GEFS"). In 1998, Mr. Vincent founded the international standards effort Legal XML (http://www.legalxml.org). He has successfully implemented live E-Filing and other XML based legal systems in Georgia and California. Mr. Vincent is a frequent and prolific speaker, writer, and educator on law and technology. He is currently a private consultant and can be reached at winchel@mindspring.com.

This is the fourth article in a series discussing the use of XML in court and justice applications. The first article, published in the November 2003 issue of the e-Filing Report, *summarized the Georgia Courts Automation Commission ("GCAC") Juvenile and Child Support Electronic Court Filing Projects. The first article described the Georgia Juvenile Project's use of XML documents. The second article stated that the XML documents were not based on JXDD and went on to detail a number of issues with JXDD. The third article compared and contrasted JXDD with the <xmlLegal> Schema Framework in an effort to show how the two works are complementary, not competitive.*

This article is a more in-depth look at the <xmlLegal> Schema Framework that describes four example XML Schema and XML instance documents, sometimes called "Reference Documents."

Introduction

This article is a detailed look at the <xmlLegal> Schema Framework and four example "Reference Documents" based on the Schema Framework. Part 1 of this article describes the Schema Framework. Part 2 of this article revisits terms and definitions from the third article, and defines four new terms: (1) "reference document," (2) "building block schema," (3) "source document," and (4) "output document." Part 3 of this article describes four example "reference documents." The examples include a State Police Complaint, a Juvenile Complaint, a Case Count Report, and a German Address.[1] The source documents, output documents, XML, XML Schema, schema documentation, and data dictionaries are available on the <xmlLegal> website at: http://www.xmllegal.org/Papers/GlasserLegalworks.htm.

1. <xmlLegal> Schema Framework

1.1. Schema Framework Introduction

The <xmlLegal> "Schema Framework" is a set of best practices and rules for developing modular, object-oriented XML Schemas. The Schema Framework provides and supports version control, schema normalization, schema management and maintenance, and consistent publishing rules for schema discovery and documentation. The Schema Framework supports a decentralized and distributed set of schema repositories. The Framework is vocabulary neutral and can use terminology from any dictionary or source. The Framework also facilitates the creation and analysis of dictionaries from XML Schemas.

1.2. Philosophy and Assumptions

The purpose of the <xmlLegal> Schema Framework is specify rules and best practices for XML Schema development that lower the time and cost to develop systems and to share information among different systems. Underlying the Framework's rules and best practices is the recognition that different applications have different data requirements. These differing data requirements are often legitimate. For example, a system used to track terrorists is different than a system to record driver's licenses, both of which are different than a court filing system. Despite, or, perhaps, in spite, of such differences, these applications are increasingly required to share information. As a result, the Schema Framework supports different schemas, created by different authors, that can be used to meet different application requirements, but in an interoperable way.

The <xmlLegal> Schema Framework is based on the assumption that multiple autonomous public and private organizations will maintain schema repositories. As a result, the Schema Framework supports a distributed and decentralized schema repository system. No single organization, person, or process must be in control of all schemas. However, it is expected that any single public or private organization or a consortium of organizations will control schemas in their domain. A related assumption relevant to court and justice XML is that federal, state, and local governments will create and maintain schemas and will maintain and mange schemas in repositories. The Schema

[1] The examples use the Commonwealth of Kentucky for illustration purposes only. The source documents come from other jurisdictions.

Framework, with its distributed repository system, is equally able to handle federal, state, and local government schemas and schema repositories.

The <xmlLegal> Schema Framework assumes that international communication is an important and fundamental requirement of information sharing, especially justice information sharing. Accordingly, the Schema Framework supports English and non-English language schema.

1.3 Scope

The <xmlLegal> Schema Framework builds on the idea that there are "vertical" and "horizontal" domains in which XML specifications (i.e., "reference documents") can be created.[2] From a vertical perspective, the <xmlLegal> Schema Framework supports XML formats for court, justice, legislative, transcript, and contract information. From a horizontal perspective, the Schema Framework defines a set of common rules and best practices for creating messages, forms, and documents within each vertical domain in a consistent way. The intersection of the vertical and horizontal domains results in a common set of building block. Building block schemas, for example, include *Person*, *Organization*, and *Address*.

1.4 Modularization

The <xmlLegal> Schema Framework requires the creation of modular schemas and provides rules and best practices for combining and managing modular schemas. This is *object oriented* schema development. Under the rules of the <xmlLegal> Schema Framework, a schema for a message, a form, or a document, such as a criminal complaint, would not be one schema, but rather a set of schemas with one schema acting as the primary schema. The Schema Framework's requirements for modular schema combined with the Framework's naming conventions forces compatibility with International Standards Organization ("ISO") 11179. Indeed, <xmlLegal> Schemas are fully compatible with ISO 11179 naming conventions. Further, <xmlLegal> Schema fit nicely into UML models.

1.5. Normalization

<xmlLegal> Schemas follow defined rules of construction, some of which are required by the W3C XML Schema specification, some of which are industry best practices, and some of which are <xmlLegal> conventions and best practices. Normalization enhances schema use and reuse, schema management, and interoperability.

1.6 Version Control

The <xmlLegal> Schema Framework has a strict version control system. Strict version control enhances interoperability by ensuring that there is a mechanical (programmatic) means of discovering the appropriate schema and validating instance documents based on the schema. Strict version control also makes iterative schema and software development easier.

1.7 Intellectual Property

<xmlLegal> Schema are licensed under a modified General Public License ("GPL").[3] The GPL allows the royalty free use and distribution of schema provided that the rules of the Schema Framework are followed. Unlike a GPL tailored for source code, the <xmlLegal> GPL is tailored for XML Schema. The <xmlLegal> GPL, for example,

[2] As founder of and contributor to Legal XML, this writer often discussed standardization in terms of

provides a legal foundation that ensures <xmlLegal> normalization and version control practices are followed. This is essential if applications that share information are to take advantage of schema validation. Combined with the Schema Framework's technical rules and best practices, this legal foundation helps to ensure interoperability.

2. Definitions

2.1 Schema

A "schema" is a DTD, W3C XML Schema, OASIS Relax NG Schema, or the like. These are each different, and in some respects incompatible, approaches to tagging data. The <xmlLegal> Schema Framework currently supports the W3C's XML Schema, but it could just as easily support OASIS' Relax NG Schema.

2.2. Data Dictionary

A "data dictionary" is a set of defined terms. A data dictionary does not mandate a data structure. A good dictionary will include many, well-defined terms. If the dictionary contains synonyms, then a good dictionary has a mechanical means to determine that two terms are synonyms. A data dictionary can be expected to evolve over time to include additional terms or to depreciate older, unused terms as language and terminology change. An XML data dictionary should be in a simple XML format, such as the W3C Resource Description Framework ("RDF")[4] or perhaps one of the emerging ontology formats.[5]

2.3. Data Model

"vertical" and "horizontal" standards.

[3] See http://www.xmllegal.org/Legal/GeneralPublicLicense.htm.

[4] See http://www.w3.org/RDF/.

[5] See, for example, the W3C's OWL at http://www.w3.org/2001/sw/WebOnt/.

A “data model” is a highly defined “data dictionary.” Terms in a data dictionary, after all, do not exist in isolation. Most or all terms have a relationship to other terms. In XML terminology, this means that terms have content models (e.g., children elements), enumerations, and data types. There is no data model that fits all applications. It is beneficial, however, to have agreed or example definitions for terms and reference data models. It is also beneficial to have references for enumerated values for common terms (e.g., Gender might have the following values “Male,” “Female”, “Unknown” and “Other”). Example data models are beneficial as a reference.

2.4. Schema Framework

A “schema framework” is a set of best practices and conventions for creating, storing, and managing *arbitrary* schema. A “schema framework” does not define terms or data models, but uses terms and data models from other sources. For example, the <xmlLegal> Schema Framework can use terminology from GJXDM. Practice has shown that the use of a “schema framework” greatly reduces the time it takes to create, manage, develop, store, and write code and applications around schema.

2.5. Building Block Schema

A “building block schema” is a common schema used (and reused) as the foundation of larger, more complex schemas. For example, in the <xmlLegal> Schema Framework, there are a number of common building block schemas, such as *Address*, *Phone*, *Email*, *Person*, and *Organization*. Building block schema are used to build a larger schema, such as a *State Police Complaint*.

2.6. Reference Document

In <xmlLegal> Schema Framework terms, a "reference document" is a set of schemas, with one primary schema and, very likely, a number of building block schemas that defines a format for a message, form, or a document. For illustration purposes, this article describes four examples reference documents (a) a *Juvenile Complaint*, (b) a *State Police Complaint*, (c) a *Case Count Report*, and (d) a *German Address*.

2.7. Source Document

A "source document" is the original paper or non-XML electronic form on which the "reference document" is based. For example, the State Police Complaint's source document is a paper document that has been used by police officers in a real jurisdiction.

2.8. Output Document

An "output document" is the combination of XML that validates against one or more schema and an XSLT or XSL-FO stylesheet. The result is an electronic document, such as an HTML, PDF, SVG, or RTF document.[6]

3. Example Reference Documents

This section describes four reference documents and the process for creating the reference documents. The reference documents are a State Police Complaint, a Juvenile Complaint, a Case Count Report, and a German Address.

3.1. Data Analysis

Before creating any schema, it is important to do thorough data analysis (and define detailed requirements). The data analysis methodology for developing an <xmlLegal> schema (or reference document) is a "bottom-up" approach that begins with a source document and people who use the source document. For example, the primary

[6] Other means of creating output documents are possible. Other formats for output documents are possible.

sources for terminology for each of the three Justice XML reference documents described in this article were (1) the paper forms that are actually in-use and (2) the people[7] who use and collect the data on the paper form. It makes sense to start with the source document and source people because the document and people who use the document necessarily have or know all required and optional data.[8] As a secondary source, a number of standard national and state data dictionaries were consulted and used as references. It makes sense to consult standard national and state data dictionaries to ensure that common terminology is used consistent with generally understood meanings.

3.2. Juvenile Complaint

To show the process of developing an <xmlLegal> Schema Framework set of schemas, let us first consider an example *Juvenile Complaint*. The example *Juvenile Complaint* originated as a Microsoft Word document in a Sheriff's Department. The Sheriffs print the document, make copies, and distribute copies to officers in the field. When an officer detains a juvenile, the officer fills-in blanks on paper and submits the completed paperwork to a Juvenile Court.

To create schemas that represent the *Juvenile Complaint*, a developer would study the source document, do data analysis, and identify building block schemas that fit data in the source document. Most of the building block schema would likely exist in the <xmlLegal> Schema Repository. If the building block schemas do not exist, then new building blocks can be created. The building block schema in the *Juvenile Complaint*, for

[7] "People" include local court and justice agency staff such as sheriffs, administrative assistants, court clerks, judges, and lawyers as well as consultants working with local staff.

[8] The assumption that a "source document" has all required data also assumes that the source document is up-to-date and there are no new requirements identified. In practice, when forms are reviewed closely, some new data is added, old, unnecessary data is deleted, and data may be reorganization. Knowing all required data often requires interviews with the people who use the data and often changes over time.

example, include the following schema, all of which are common <xmlLegal> building block schema:

- Address
- Date
- Email
- Offense
- Organization
- Person
- Phone
- Signature Block
- Time

Once a developer identifies building block schemas, the schemas would be moved into a *Juvenile Complaint* project and a primary schema called *Complaint* would be created. The primary *Complaint* schema would include elements that mirror the paper form used by the Sheriffs. The example *Juvenile Complaint* includes the following elements (with an associated building block represented in parentheses):

- File Number
- Child (Person)
- Mother (Person)
- Father (Person)
- Legal Custodian (Person or Organization)
- Offenses (Offense)
- Custody

    - Detention
- Release
- Coperpetrators (Person)
- Victims (Person or Organization)
- Witnesses (Person)
- Investigating Officer (Person)
- Investigating Officer Agency (Organization)
- Complainant (Person)
- Complainant Signature (Signature Block)

As you can see, a building block schema represents most of the elements in the primary *Complaint* schema. With the building block schema identified, the primary schema can be created based on Schema Framework rules.

If you are familiar with a *Juvenile Complaint* from your jurisdiction, you may be thinking, “That looks good, but here, we call an ‘Offense’ a ‘Complaint’ and the order of the information is different.” From an <xmlLegal> Schema Framework perspective, this is expected and acceptable. If you follow the rules of the Schema Framework, then you can create a *Juvenile Complaint* for your jurisdiction that meets your jurisdiction’s specific requirements, yet still comply with the rules, and profit from the benefits, of the Framework. Of course, if this *Juvenile Complaint* meets your needs, then you could use it as is. In either case, the advantage of using the Schema Framework is that your schemas fit into a system that make it easy to use, reuse, manage, document, publish, and access a large number of schemas.

3.3 State Police Complaint

Now that you have been introduced to the process of creating an <xmlLegal> set of schemas, let us consider a second example -- a *State Police Complaint.* Like the *Juvenile Complaint*, the *State Police Complaint* is created based on a source document. The *State Police Complaint* includes the following building block schema:

- Address
- Date
- Email
- Organization
- Person
- Phone
- Signature Block
- Time

You will notice that the building block schemas for the *State Police Complaint* are exactly the same as the as the building block schemas for the *Juvenile Complaint,* except that the *State Police Complaint* does not include the *Offense* building block.[9] Closer inspection of the schema and schema documentation will show, however, that while the common building block schemas are from the same "genre" (e.g., *Address*, *Person*, *Organization*), the schema are not the same exact "object."

<xmlLegal> schemas, because they are modular and are associated with different projects, exist as separate files and live in different contexts with different versions numbers. The benefit of this approach is that a large number of reference documents are *not* "tightly bound" to a set of building blocks (although they could be). Anyone

familiar with the “dll hell” problem on Microsoft Windows platforms (before Microsoft .NET) will understand this issue. If 100 reference documents (i.e., schema) were tightly bound to one *Address*, then when and if the *Address* changes, the 100 reference documents would no longer be compatible. From a validation and interoperability perspective, tightly bound schemas creates a situation like the San Andreas Fault – one slip of the fault line will cause a tremor and possibly an earthquake. Depending on the slippage and resulting magnitude, the ripples created throughout the tightly bound system could be catastrophic and would be costly. Understand, when a schema that has been tightly bound to one or more reference documents changes, it is unlikely that older reference documents will continue to validate against the schema (i.e., schema hell). From a schema and XML perspective, this complicates or eliminates interoperability. From a more practical development and coding perspective, it is simply costly.

As a related matter, while the *Address* building block exists in both the *Juvenile Complaint* and the *State Police Complaint*, closer inspection will reveal that the *State Police Address* schema is a modified version of the *Juvenile Address* schema. Under the rules of the <xmlLegal> Schema Framework, schemas of the same genre can be either a copy of another schema (“loosely bound”), a subset schema[10] of another schema (“loosely bound”) or a completely different schema (“no binding”).

In these examples, the *State Police Address* is a subset of the *Juvenile Address*. The *Juvenile Address* breaks an address line into subparts, such as street number, street

[9] Slightly different requirements resulted in the absence of an *Offense* building block in the *State Police Complaint*.

name, apartment number, etc. The client overseeing the *State Police Address* did not want such a breakdown. The result is that XML created based on the *State Police Address* schema would validate against the *Juvenile Address* schema, but the opposite would not necessarily be true. The loose binding would, however, make it easy to map and transform XML from one schema to the other. Hence, interoperability is easy to achieve, even thought there is no single *Address* schema for all applications. The result of loosely binding schema within a standard schema framework is an adaptable and powerful system. The result of a tightly binding schema (with or without a framework) is an inflexible and brittle system.[11]

3.4. Case Count Report Schema

The *Juvenile Complaint* and the *State Police Complaint* are good examples of similar reference documents that can use the same genre of building block schema. The *Case Count Report* reference document is an example of a set of schemas that do not use common building block schemas, but nevertheless fit into the Schema Framework. The *Case Count Report* schema is used to count case information passed from a Court Clerk's Office to an Administrative Office of the Courts. The schemas record statistics for cases on the docket, disposed cases, the age of disposed cases, and pending cases, each grouped by (a) D.W.I . or D.U.I.D., (b) theft or worthless checks, (c) drug offenses, (d) assault, (e) traffic, (f) other criminal cases, and (e) total cases.

The building block schema for the primary *Case Count* schema include:

---

[10] The term "subset schema" has been used in <xmlLegal> documentation for a number of years. It is conceptually similar to the more recent GJXDM idea of "subset schema" although the mechanics are quite different.

[11] At some level, schemas in a modular schema framework must be tightly bound to other schemas. Tight binding must happen for a single reference document. As a rule of thumb, tight binding may be appropriate for related reference documents, but tight binding on a large scale should be carefully considered.

    - Case Age
- Cases
- Count
- Dispositions
- Pending Cases

None of these building blocks fit the usual schema genres, such as *Address, Person,* or *Organization*. Nevertheless, the schemas fit into the Schema Framework and benefit from the association. Among other things, because the *Case Count* schemas are a part of the Framework, it is easy to generate data dictionaries in a common format that can be fed and reused in larger data dictionaries.

3.5 German Address

Although a German address is not directly relevant to United States Court and Justice XML, international exchange of justice information is \ relevant to fighting international crime and terrorism. The example *German Address* schema is a subset[12] of an *American-English Address* schema, but for the use of German words for element and attribute names. The result is a one-to-one mapping of terminology that facilitate and simplifies automated translation from one language into another.[13] This feature of the <xmlLegal> Schema Framework is important not simply for international communication, but also for communication in cities and States with diverse populations, such as California, Florida, or New York.

3.6. Examples Available on <xmlLegal> Website

[12] See footnote 10.
[13] A good translation assumes an accurate mapping of words from one language to words in another language, but this is an issue with all translations, regardless of the mechanics.

The four example reference documents are available on the <xmlLegal> website at http://www.xmllegal.org/Papers/GlasserLegalworks.htm. The website includes source documents, output documents, XML, XML Schema, schema documentation, data dictionaries, and an example schema repository. Some resources are available to free members. Other resources are available only to paid members. Also available to paid members is a discussion mailing list in case you would like to ask questions or receive more information about the <xmlLegal> Schema Framework.