2GEFS Roadmap

By Winchel "Todd" Vincent III
November 13, 2004, Version 0.2

Introduction

The California Administrative Office of the Courts ("CA AOC") initiated the Second Generation Electronic Filing Specifications ("2GEFS" or "Specifications") to facilitate the development and implementation of interoperable electronic filing solutions in the California courts. The Specifications include *Court Filing 2.0*, *Court Policy 2.0*, *Request-Response 2.0*, and *CMS-API 2.0*. The CA AOC commissioned and financed the development of the Specifications for application in California courts. Additionally, a number of private companies volunteered time and expertise to develop, implement, and test of the Specifications. This document introduces the *2GEFS Concepts Document*, *Court Filing 2.0*, *Court Policy 2.0*, *Request-Response 2.0*, and *CMS-API 2.0*. This document also includes an Appendix A that includes definitions of technologies relevant to the 2GEFS.

1. 2GEFS Concepts Document

The *2GEFS Concepts Document* is a high-level document that defines the 2GEFS terminology, architecture, concepts, and assumptions.[1] The *Concepts Document* is an excellent high-level overview of electronic court filing and the ideas upon which the 2GEFS are based. The *2GEFS Concepts Document* is recommended reading for administrators, managers, and system architects. Developers may also wish to read the *Concepts Document* to understand the 2GEFS at a high-level.

2. Court Filing 2.0

2.1. Filing

The *Filing Specification* is the most important of the 2GEFS, because an electronic court filing system could be built using the *Filing Specification* alone. *Filing XML* includes information necessary to:

- Initiate a case in a CMS, if a case does not exist;
- File one or more documents into a case;
- Add name, contact details, identifiers and descriptions and roles for people, organizations, and things associated with the case;
- Add charges or offenses to a case;

[1] See http://www.xmllegal.org/Documents/2GEFS/2GEFS_Concepts_PDF_Ver1_1_2003_07_31.pdf

- Add calendar information or send calendar information back to a filer;
- Add service of process information; and
- Add fee and payment information.

Filing XML is sent to a court's EFM wrapped in an XML envelope. Filing applications usually transmit enveloped *Filing XML* over HTTP on the Internet.

The *Filing Specification* includes a *Confirmation* schema, an *Envelope* schema, and a *Header* schema. Filers use the *Filing* schema to send electronic filings to courts. Courts use the *Confirmation* schema to send confirmations to filers.

2.2. Confirmation

Confirmation XML mirrors the elements in *Filing XML*, except that *Confirmation XML* also includes a confirmation number, a confirmation date, a confirmation time, and a confirmation filing status. Elements in *Confirmation XML* mirror elements in *Filing XML* based on the assumption that filing information may change as the *Filing XML* travels downstream from a filing application to a court. In case the information changes, or even if it does not, the confirmation serves as a receipt and notice of the information ultimately accepted by a court. As a result, there is a corresponding element in *Confirmation XML* for every element in *Filing XML*.

2.3. Filing and Confirmation Envelope

When transmitting *Filing XML* and *Confirmation XML*, there is usually a need to send information about the transmission itself. Transmission information is usually not important to the filing or confirmation transaction. As a result, it is common practice to envelope *Filing XML* and *Confirmation XML* in an XML envelope. Any XML envelope can be used to send and receive 2GEFS *Filing XML* and *Confirmation XML*.

The 2GEFS includes a simple 2GEFS *Envelope* that can be used to transmit *Filing XML* and *Confirmation XML*. Alternatively, a SOAP or ebXML envelope can be used. A 2GEFS *Envelope XML* and a SOAP XML envelope have the same structure. Both envelopes have a root element named **<Envelope>** and two children elements, **<Header>** and **<Body>**. The difference between the 2GEFS *Envelope XML* and a SOAP envelope is that a SOAP envelope includes additional attributes that have meaning to SOAP-aware processors and the namespaces are different. Otherwise, the envelopes are the same. Courts in California are using the 2GEFS *Envelope XML* for testing and are planning to implement the SOAP envelope for live filing.

2.4. Filing and Confirmation Header

Header XML can be used in the 2GEFS *Envelope XML* or in a SOAP envelope. In either case, *Header XML* is placed within the envelope's **<Header>** element. The SOAP specification explicitly states that application-specific header information is *not* defined by the SOAP specification and should be defined by the application. If SOAP is used for transmission, then 2GEFS *Header XML* should go inside the SOAP **<Header>** element as the first child element. SOAP attributes designed to tell SOAP applications whether to

understand or forward *Header XML* may be used based on the rules defined for each element in the *Header Specification*.

3. Court Policy 2.0

The *Policy Specification* is the second most important of the 2GEFS because it provides a standard format for communicating information about a court and the court's divisions. *Court Policy XML* is simply a configuration file that holds information unique to a court. *Court Policy XML* is *helpful* in a court filing system where there is only *one* court. It is *necessary*, as a practical matter, when expanding a system where there are *multiple* court locations or *multiple* court divisions within a single court.

Theoretically, *Court Policy XML* could contain a wide variety of information. Attempting to specify too much information in *Court Policy XML* is problematic because of the complexity and difficulty in writing code for all logical uses. As a result, the 2GEFS Court Policy only attempts to support a minimum of what could theoretically be in a court policy XML specification.

Court Policy XML includes the following information about a court:

- Unique names and identifiers for a court, its divisions, and its groups;
- Court clerk(s);
- Fee schedules;
- Court payment details (enough information to pay a court);
- Code tables (frequently used code tables from the courts case management system(s), such as case categories);
- Exchanges (names and unique identifiers for specific electronic information exchange points);
- Hours of operation (including cut-off times for electronic and paper filing);
- Accepted MIME types;
- Maximum filing size;
- Accepted credit cards; and
- Accepted reply to protocols and formats

Court Policy XML should be available to all organizations involved in electronic filing with the court, preferably over the Internet, for local or remote use. The *Court Policy Specification* includes a standard means of Internet publication although Internet publication is not absolutely necessary for the use of *Court Policy XML*.

4. Request-Response 2.0 and CMS-API 2.0

The *Request-Response* and *CMS-API Specifications* are closely related. The structure of the *CMS-API* maps class-for-class, method-for-method, and argument-for-argument with *Request-Response XML*. This makes mapping *Request-Response XML* to a CMS with an implemented *CMS-API* relatively easy.

Intended for internal court use, the *CMS-API Specification* defines an API designed to be built into or onto a court CMS to input and output information in a CMS. The *CMS-API* would most commonly be implemented to interface a clerk review module with the court CMS. For security reasons, it would be unlikely a court would want to expose the *CMS-API* to applications outside the court.

In contrast, *Request-Response XML* is used to send and receive XML requests and responses from systems *outside* the court. There is no restriction on using *Request-Response XML* inside the court, but the primary purpose is to provide a platform neutral XML format for service providers to request information from the court and for the court to respond back to service providers. This request-response transaction would take place through a court's EFM, providing security to court applications.

Neither *Request-Response XML* nor *CMS-API* is intended to provide a full and complete API to a CMS. The current 2GEFS *CMS-API* specifies only "core 1" classes, methods, and arguments. The purpose is to provide the minimum number of CMS calls necessary for electronic court filing applications. The current *CMS-API* core 1 supports *adding* and *updating* information included in *Filing XML* and a few additional "get" requests.

The *CMS-API Specification* defines a means to define extensions for additional calls that could satisfy additional requirements, such as the automated creation of *Court Policy XML*. Thus, the scope of *Request-Response XML* and *CMS-API* is, or could be, broader than *Filing XML*, although it is currently not as broad as a full and complete API for a CMS.

The following list shows the high-level organization of *CMS-API* and includes labels that designate basic core level requirements:

- **CMS** (core 1 and higher)
- **Court** (core 1 and higher)
- **Filing** (core 2 and higher)
- **Case** (core 1 and higher)
- **Document** (core 2 and higher)
- **Person** (core 1 and higher)
- **Organization** (core 1 and higher)
- **Codes** (core 2 and higher)
- **Calendar** (core 2 and higher)
- **Payment** (core 2 and higher)
- **CourtPolicy** (core 2 and higher)

Generally, methods in the API have the ability to (a) add, (b) get, (c) update, and (d) delete information. For example, the `CMS.AddCase` method adds a new case to a CMS. The `Case.AddDocument` method adds a new document to a case. `Case.GetPeople` gets all of the people associated with a case. Some operations are appropriate for the *CMS-API*, but not for *Request-Response XML*. For example, a CMS should have the

ability to programmatically update and delete information via an API. However, in most cases, *Request-Response XML* should not provide the ability for an application outside the court to update or delete information. Authorized "get" operations are more appropriate for *Request-Response XML*.

Like the *Filing Specification*, the *Request-Response Specification* includes an *Envelope* schema and a *Header* schema. The *Envelope* schema can be substituted with a SOAP envelope. The *Header* schema is used for both a 2GEFS *Envelope* and a SOAP envelope. The *Filing* and *Request-Response Envelope* and *Header* are exactly the same, except that they exist in different namespaces.

5. Schema Framework

The 2GEFS XML schemas are standard *W3C XML Schema* built according to the rules and best practices of the <xmlLegal> Schema Framework. To work with the 2GEFS, it is beneficial to have a good understanding of *W3C XML Schema*. It is not necessary to understand the rules and best practices of the <xmlLegal> Schema Framework. However, the developer who understands the Schema Framework may find it easier to write code around the 2GEFS. To extend the 2GEFS, it is necessary to understand the rules and best practices of the Schema Framework.

The <xmlLegal> "Schema Framework" is a set of best practices and rules for developing modular and interoperable XML Schemas. The Schema Framework supports a decentralized and distributed set of schema repositories and services. The Framework provides and supports version control, schema normalization, schema management and maintenance, and consistent publishing rules for schema discovery and documentation. The Framework also facilitates the creation of data dictionaries.

1.5.1. Philosophy and Assumptions

The <xmlLegal> Schema Framework builds on the idea that there are "vertical" and "horizontal" domains in which XML standards can be created. See *Scope* section for more information.

The Schema Framework recognizes that different applications have different data requirements. These differing 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. As a result, the Framework supports many different schemas that can be used in an interoperable way.

1.5.2. Scope

From a vertical perspective, the 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 schema results in a common set of building block and primitive schemas. Primitive schemas, for example, include *Person*, *Organization*, and *Address*.

1.5.3. Modularization

The Schema Framework supports modular XML Schemas that are used as building blocks to build other, more complex schemas, messages, protocols, forms and documents. For example, *Person*, *Organization*, and *Address* schema can be used as building blocks for a more complex *Court Filing* schema, *Juvenile Complaint* schema, or *Rap Sheet* schema.

Modularization facilitates schema reuse and customization as well as code reuse all while maintaining interoperability. Modularization also reduces the total size of individual XML formats.

1.5.4. 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. Schemas that follow the Framework's rules and best practices are "normalized." Normalization greatly enhances schema use and reuse, schema management, and interoperability.

1.5.5. Version Control

The 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.

1.5.6. Schema Repositories

The Schema Framework supports distributed and decentralized Internet-based schema repositories as well as local schema repositories. Schema repositories contain schema documentation and data dictionaries, as well as the schema themselves.

1.5.7. Intellectual Property

<xmlLegal> Schemas are licensed under a modified General Public License.[2] The GPL allows the royalty free use and distribution of schema provided that the rules of the Schema Framework are followed. GPL modifications, for example, provide a legal framework that ensures <xmlLegal> normalization and version control practices are followed. These practices, in turn, help to ensure interoperability.

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

Appendix A: Definitions

2GEFS Versions

The <xmlLegal> Schema Framework supports a precise version control system. The format for version numbers is a 0-padded two- or three-digit number combined with an optional string "Test." The string "Test" means that the schema is still undergoing drafting or testing.

During Phase 1 and 2, the 2GEFS went through several versions, as indicated in the table below:

Specification	Date	Version
Filing	July 2003	Test03
Filing	May 2003	Test04
Filing	July 2004	Test05
Filing	August 2004	01
Policy	July 2003	Test02
Policy	May 2004	Test03
Policy	July 2004	Test04
Policy	August 2004	01
Request-Response	July 2003	Test01
Request-Response	May 2003	Test02
CMS-API	July 2003	0.2.4
CMS-API	May 2003	0.2.5

Application Programming Interface ("API")

An "Application Programming Interface," or "API," is a defined gateway into a computer information system. The gateway may be used to enter information into the system, extract information from the system, or otherwise manipulate information in the system.

A non-technical person might analogize a software API to an electrical socket in a home and a corresponding plug that fits into the socket. An API for a United States socket defines three holes of a particular size, shape, and arrangement into which a plug with corresponding prongs fits. An API for a German socket may require a plug with three prongs, but the size, shape, and arrangement is different.

An API for software works the same way. Software developers use an API so they know how to write code to fit data into the system. If a computer system does not have a defined or easy-to-use API then it may be difficult or impossible to programmatically enter, extract, or manipulate information in the system.

A user interface or terminal used by a person for manual data entry or extraction is usually not called an API.

Attribute

See definition of XML.

Base64 Encoding

"Base64 Encoding" is a means of transforming a binary electronic file into an encoded text representation. Base64 encoding happens to result in text that does not include any of the XML-reserved characters. XML reserved characters must be escaped. In short, because base64-encoded text does not include any XML reserved characters, it is "XML-safe" and is easy to insert into an XML document. Base64 encoding is an accepted means of inserting a binary electronic document, such as a Microsoft Word document or a PDF document, into XML. The disadvantage of base64-encoding is that the encoded text is approximately 33% larger than the original binary file. This bloats the resulting XML. If an XML file is too large, then it can result in performance problems for XML software, such as parsers.

Case Management System ("CMS")

A "CMS" is an application for supporting court operations. A CMS records and manages information about court cases, records, calendars, finances, and other court information. A CMS will sometimes have an existing interface that is different and distinct from the "Clerk Review Software." CMSs vary among courts and vendors. Some CMSs have well-defined APIs, some CMSs have poor APIs, and some CMSs have no API at all.

Document management ("DMS") capabilities are included in the CMS definition unless specifically noted otherwise.

Clerk Review Software or Module ("Clerk Review")

The "Clerk Review Software" is a web- or application-based interface available to court administrators and other court personnel. The Clerk Review Software is used for reviewing incoming e-filings and for performing other administrative tasks. The Clerk Review Software can be separate from or part of a CMS.

Court Filing 1.0

See definition of Legal XML Court Filing 1.0.

Document Management System ("DMS")

A "DMS" receives, manages, stores, and retrieves, electronic court documents. A DMS may be part of a CMS or may be different software altogether. In this document, DMS capabilities are included in the term CMS unless specifically noted otherwise.

Document Object Model ("DOM")

The "Document Object Model" or "DOM" is a W3C standard API for XML documents.[3]

[3] See http://www.w3.org/DOM/.

Document Type Definition ("DTD")

A "document type definition" ("DTD") is a set of rules that define the type, number, and order of elements that may appear in an XML document.[4] The rules of a DTD are set out in "declarations."[5] The following is a set of declarations that define an "Address" document in XML:

```
<?xml version="1.0" ?>
<!DOCTYPE Address [
<!ELEMENT Address (Street+,City,State,PostalCode,Country) >
<!ELEMENT Street (#PCDATA) >
<!ELEMENT City (#PCDATA) >
<!ELEMENT State (#PCDATA) >
<!ELEMENT PostalCode (#PCDATA) >
<!ELEMENT Country (#PCDATA) >
]>
```

The first declaration in the DTD is for the "Address" element. According to the declaration, the "Address" element may contain five other elements, namely, "Street," "City," "State," "PostalCode," and "Country." Each of these elements may contain "#PCDATA," which is text.[6] The "+" on the end of "Street" means there may be *one or more* "Street" elements within "Address."[7]

There are other rules used to define the number of elements that may appear in the document. Additionally, attributes may be specified in declarations corresponding to individual elements.[8] There are numerous other XML rules that are beyond the scope of this document.[9]

Electronic Document

The term "electronic document" refers to any type of electronic document format that includes "formatting," "logical structure," and "data." Examples of an electronic document formats are Microsoft Word, Adobe Portable Document Format ("PDF"), Corel Word Perfect, TIFF Images, and Hypertext Markup Language ("HTML").

An XML document or an XML instance document is not an "electronic document" under this definition. Under this definition an XML document + a stylesheet is equivalent to an electronic document.

Electronic Filing or E-filing

"Electronic Filing" or "E-Filing" is an electronic document delivered to a court by electronic means. "Electronic documents", "electronic court documents" and similar terms are synonymous unless the context provides otherwise.

---

[4] W3C XML 1.0 Recommendation, http://www.w3.org/TR/1998/REC-xml-19980210#sec-prolog-dtd.
[5] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#NT-markupdecl.
[6] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#syntax.
[7] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#sec-element-content.
[8] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#attdecls.
[9] Id. See generally XML 1.0 Recommendation.

Electronic Filing Manager ("EFM")

An "EFM" is an application (or applications) used to process electronic filings. An EFM implements Court Filing and Request-Response transmissions, and accesses and interprets *Court Policy XML*. If the EFM communicates with a CMS it does so using the CMS API (via Court Adaptor Application middleware when needed). Parts of the EFM are usually exposed outside the court's firewall. The EFM then talks to the CMS via the CMS-API. In this way, the CMS-API is not exposed in a non-secure environment.

Element

See definition of XML.

Filing Application

An application used by Filers to prepare electronic filings. A Filing Application may be a desktop application or a web-based application provided by a Service Provider, or a subsystem of a larger system such as a law firm's practice management system.

Georgia Interoperability Testing

Georgia Courts Automation Commission's ("GCAC") mission is to encourage and facilitate automation in courts in the State of Georgia. In furtherance of this mission, GCAC sponsored a Court Filing Interoperability Pilot Project ("Interoperability Pilot"). The purpose of the Interoperability Pilot was to provide Georgia courts and attorneys with proof-of-concept of electronic court filing and to better understand the need for, and barriers to, developing court filing systems and standards in Georgia.

The Pilot had two phases: Phase I completed in July and August 2001, and Phase II completed in October and November 2001. The Interoperability Pilot closed in November 2001. Two live electronic filing systems resulted from the Interoperability Pilot: a juvenile electronic filing system and a child support electronic filing system.

Hypertext Transport Protocol ("HTTP")

"Hypertext Transport Protocol" or "HTTP" is a text-based messaging format used for Internet communications. It is an Internet Engineering Task Fore ("IETF") standard, RFC 2616, http://www.ietf.org/rfc/rfc2616.txt. HTTP is the protocol used to send web pages to and from web browsers and web servers. It is a very stable and well-understood Internet standard.

Instance Document

An "instance document" is a well-formed and valid XML document based on a schema.

Legal XML Court Filing 1.0

"Legal XML Court Filing 1.0" is the July 24, 2000 Legal XML Court Filing recommended standard. Legal XML published the *Court Filing 1.0* DTD before Legal XML became OASIS/Legal XML. OASIS/Legal XML does not have the *Court Filing 1.0* DTD available on its website. The *Court Filing 1.0* DTD was used in the Georgia

Interoperability Pilot Project and is available on the Georgia State University Electronic Court Filing Website.[10]

Parser, Non-Validating Parser, and Validating Parser

A "Parser" is software used to process and XML instance document. A "Non-Validating Parser" is software used only to determine whether an XML document is well-formed. A "Validating Parser" is a parser that validates well-formed XML against a schema. Although all parsers and validating parsers are supposed to implement the W3C XML 1.0 and XML Schema standards the same way, there are slight nuances among them that may interfere with interoperability.

"schema"

In this document, the term "schema" with a lower-case "s" means either a DTD or an XML Schema. The term "Schema" with a capital "S" means an XML Schema.

Secure Sockets Layer ("SSL")

"Secure Sockets Layer" or "SSL" is a protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a private key to encrypt data that is transferred over the SSL connection. Both Netscape Navigator and Internet Explorer support SSL, and many Web sites use the protocol to obtain confidential user information, such as credit card numbers. By convention, URLs that require an SSL connection start with https: instead of http:. See http://wp.netscape.com/eng/ssl3/.

Service Provider

A person or organization that provides electronic filing services or software to filers, courts, or justice users.

Service Provider Application

Software applications developed or operated by a Service Provider.

Simple Mail Transfer Protocol ("SMTP")

"Simple Mail Transfer Protocol" or "SMTP" is an Internet Engineering Task Force ("IETF") standard for transporting electronic mail, RFC 821, http://www.ietf.org/rfc/rfc0821.txt. SMTP is the standard protocol used to send Internet email. It is a very stable and well-understood Internet standard.

SOAP

"SOAP" is a W3C recommendation for an XML transmission protocol. SOAP is often called an "envelope" because it serves as an XML wrapper around other XML. For example, a SOAP envelope could wrap *Filing XML* prior to transmission over the Internet.

[10] See http://e-ct-file.gsu.edu/CourtFilings/Interoperability/.

A non-technical person could analogize SOAP to a paper envelope using to mail a letter via post. SOAP is the paper envelope. The user of SOAP is responsible for addressing the SOAP enveloped, writing the letter, packaging the letter in the envelope, and sending the envelope to its destination.

See http://www.w3.org/TR/soap12-part0/, http://www.w3.org/TR/2003/REC-soap12-part1-20030624/, http://www.w3.org/TR/2003/REC-soap12-part2-20030624/.

Stylesheet

Stylesheets apply formatting to an XML instance document. There are two types of stylesheets, CSS Stylesheets and XSL Stylesheets.

Uniform Resource Indicator ("URI")

A "Uniform Resource Identifier" or "URI" is a compact string of characters for identifying an abstract or physical resource. The syntax for URIs is defined by an IETF in RFC 2396 at http://www.ietf.org/rfc/rfc2396.txt. URIs can come in various formats. The most recognizable URI is a web address, such as:

- http://www.ietf.org/rfc/rfc2396.txt

A "URN" is a type of "URI."

Valid XML or Validation

It is possible, although not required, to "validate" well-formed XML using a DTD or an XML Schema.[11] Validating well-formed XML with a DTD or an XML Schema means that the structure of the well-formed XML is checked by software, called a parser, to see if it matches the rules specified in the declarations of the schema. If the well-formed XML does not conform to the schema, then the parser will report a validation error.[12]

Web Service

A "Web Service" can be generically defined as an Internet-based address where XML can be sent and received or requested and received. An XML over HTTP connection could be considered a web service.

Web services technologies include, among other things, XML, SOAP, SOAP Attachments (formerly called Direct Internet Message Encapsulation or "DIME"), and Web Services Description Language ("WSDL"). The W3C has a web services activity. More information can be found at http://www.w3.org/2002/ws/.

[11] Id. http://www.w3.org/TR/1998/REC-xml-19980210#dt-valid.
[12] Id.

Well-Formed XML

"Well-formed XML" is an XML document that follows the simple rules that (1) every document must have a single root element and (2) for every element there must be a "begin tag" and a corresponding "end tag" that does not overlap with other begin and end tags.[13] Note, elements (begin and end tag combinations) may be nested within other elements, but tags may not overlap.[14]

For example, the following is well-formed XML:

```
<Address>
  <Street>2356 Peachtree Street</Street>
  <Street>Suite 2000</Street>
  <City>Atlanta</City>
  <State>Georgia</State>
  <PostalCode>30302</PostalCode>
  <Country>U.S.A.</Country>
</Address>
```

The following, however, is illegal according to the XML specification because the tags overlap and are *not* well-formed:

```
<Bold>This is some<Italics>bad</Bold> XML</Italics>.
```

World Wide Web Consortium ("W3C")

The Word Wide Web Consortium ("W3C") is a well-known industry standards organization. The W3C is responsible for developing and maintaining XML and related XML technologies. See http://www.w3.org/.

XML

"XML" is an acronym for e**X**tensible **M**arkup **L**anguage ("XML"). XML is a technical standard developed by the W3C [15] and defined at http://www.w3.org/TR/REC-xml/. Since early 1998, when the W3C recommended XML 1.0 as a standard, XML and related W3C standards have gained widespread acceptance in the technical community as "smart," web-based information management technologies.

XML and related standards are used to create "document formats" by combining customized "elements" and, optionally, "stylesheets." XML elements look similar to Hypertext Markup Language ("HTML") elements. For example, the following is an HTML element:

```
<FONT Size='12'>I agree to give you a peppercorn in exchange for your services.</FONT>
```

[13] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#sec-well-formed.
[14] Id.
[15] World Wide Web Consortium Website, http://www.w3.org.

An ***element*** is a combination of a "begin tag" and "end tag" and everything in between the two tags.[16] Some elements may only contain text. Some elements may contain other elements (i.e., tags and text).[17] Elements may also be empty (i.e., contain no text).[18] Elements may have ***attributes***, which are assigned "values."[19]

In the example above, the element name is "FONT." The "FONT" element has an attribute named "Size" with an attribute value of "12." Elements are nested within other elements to create a hierarchy of "marked-up" text. A complete hierarchy of marked-up text is an "XML document"[20] also called an "instance document." The following is an example of a simple XML document with legal elements.

```
<Legal>
   <Contract>
      <Clause>
         <Paragraph>
         I agree to give you a peppercorn in exchange for your services.
         </Paragraph>
      </Clause>
   </Contract>
</Legal>
```

HTML is a standardized set of about 90 pre-defined elements that web designers use to create HTML documents (web pages).[21] The problem with HTML is that it is a *dumb* "document format."[22] Indeed, a significant disadvantage of HTML is that most of its predefined set of tags do not have a meaningful relationship to the text within them. For example, the following HTML element, with the addition of the "color" attribute, would look colorful in a web browser:

```
<FONT Color='Red'>I agree to give you a peppercorn in exchange for your services.</FONT>
```

However, the HTML **<FONT>** element does not provide meaningful information to a reader, a search engine, or any other information system about the meaning of the text within the element. A web browser knows it should display the text in red, but it knows nothing else about the text. More meaningful mark-up would look like this:

---

[16] XML 1.0 Recommendation, http://www.w3.org/TR/1998/REC-xml-19980210#sec-starttags.
[17] Id. http://www.w3.org/TR/1998/REC-xml-19980210#NT-content.
[18] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#NT-EmptyElemTag.
[19] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#NT-Attribute.
[20] Id. at http://www.w3.org/TR/1998/REC-xml-19980210#sec-documents. Note, an "XML document" should not be confused with an "electronic document." Generally, an XML document, from a human and legal perspective is not a complete document, unless it includes a stylesheet. That is, the combination of an XML document and a stylesheet corresponds to an electronic document and most closely to the traditional notion of a paper document.
[21] HTML 4.01 Specification, http://www.w3.org/TR/html4/index/elements.html.
[22] Winchel "Todd" Vincent, III, "What is the Best Format for E-CT-Filing," http://gsulaw.gsu.edu/gsuecp/CourtFilings/DocumentFormat/.

<Contract Color='Red'>I agree to give you a peppercorn in exchange for your services.</Contract>

Unfortunately, custom elements such as **<Contract>** are not allowed in HTML because they are not defined by the HTML standard. XML, unlike HTML, is not a set of defined elements. Rather, XML is a "grammar" (or "syntax") that can be *used to define any number of custom elements*. Using XML, the developers of document or data formats can create industry-specific (e.g., legal-specific) elements, such as **<Contract>**, **<CourtFiling>**, or **<Transcript>**, that can hold information important in their industry or to their specific application.

XML Document

An "XML Document"[23] is well-formed XML as defined by the W3C. An XML Document is the same an instance document. An "XML document" is a term-of-art used in the W3C XML 1.0 specification. The term applies only to the mark-up (i.e., the tags) and the text in the document. An "XML document" according to the W3C XML 1.0 specification, does *not* include a stylesheet.

XML Namespaces

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.[24] For example, if an element name is "Table," XML namespaces allow a developer to qualify the context of the element with a namespace and an associated namespace prefix. Below is an example using "Table" as an element name using namespace prefixes (that would have associated namespaces) distinguishing their context:

<Furniture:Table>

<Math:Table>

XML Schema

"XML Schema" is a W3C recommended standard that improves on DTD technology. XML Schema perform the same role and function as a DTD. That is, an XML Schema is a set of rules that define the type, number, and order of elements that may appear in a well-formed XML document. The two most important differences between a DTD and an XML Schema are that:

- XML Schema can be used to validate data types. For instance, an XML Schema can be used to ensure that text within an element is an integer, a string of text, a date, or a time. DTDs, on the other hand, cannot validate data types.
- XML Schema are well-formed XML documents, while DTDs are not.

[23] See http://www.w3.org/TR/REC-xml/#sec-documents.
[24] See http://www.w3.org/TR/REC-xml-names/.

Because a DTD and an XML Schema are the same type of technologies, in this document, the term "schema" with a lower-case "s" means both a DTD and a XML Schema. "Schema" with an upper-case "S" means an XML Schema as defined in this section.

XML Schema are more useful and powerful than DTDs. However, XML Schema are also more complex. Additionally, a disadvantage of XML Schema in comparison to a DTD is that, while tools for DTDs have existed for some time and are mature, tools for XML Schema are relatively new and not as mature.

The following is part of an XML Schema that defines an Address format. This example is not complete, but it is enough to compare with the DTD example included in these definitions.

```
<xsd:complexType name="Address" mixed="true">
    <xsd:sequence>
        <xsd:element ref="Line" minOccurs="0" maxOccurs="5"/>
        <xsd:element ref="Suburb" minOccurs="0"/>
        <xsd:element ref="City" minOccurs="0"/>
        <xsd:element ref="State" minOccurs="0"/>
        <xsd:element ref="County" minOccurs="0"/>
        <xsd:element ref="PostalCode" minOccurs="0"/>
        <xsd:element ref="Country" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>
```

XSL and XSLT

"XSL" stands for e**X**tensible **S**tylesheet **L**anguage. "XSL" is another W3C recommended standard that compliments XML. XSL is a language for expressing stylesheets. Stylesheets are used to make XML documents look like real documents or to manipulate or change an XML document.

For example, in the <Contract> examples above, a reader would not want to see the begin and end <Contract> tag surrounding the text within the tags. An XSL stylesheet can be used to apply formatting (such as **bold** or red) to the text within the tags and, at the same time, to make the tags disappear.

XSL has two parts. "XSL Transformations," or "XSLT," is one part. XSLT is a powerful language that allows a programmer to change or transform an XML document into another XML document or into another type of electronic document, such as an Adobe PDF document, a Microsoft Word document, or a text document.