DISCUSSION DRAFT
10-22-04

OPEN XML COURT INTERFACE

Electronic Filing Manager Architecture

October 22, 2004

MTG Management Consultants, L.L.C.
1111 Third Avenue, Suite 2700
Seattle, Washington 98101-3201
206.442.5010 206.442.5011 *fax*
www.mtgmc.com

TABLE OF CONTENTS

Page

TABLE OF CONTENTS

(continued)

I. INTRODUCTION

I. INTRODUCTION

The Open eXtensible Markup Language (XML) Court Interface (OXCI) consortium of state courts intends to produce a middleware implementation for electronic filing for use within all levels of state courts for the receipt, transmission, and validation of electronic filings, court orders, and associated data. The middleware will provide a uniform open source implementation of an Electronic Filing Manager (EFM), compliant with the specifications developed by the LegalXML Electronic Court Filing Technical Committee (TC) of the Organization for the Advancement of Structured Information Standards (OASIS). However, these interface specifications are still in development and are not a sufficient basis for a complete implementation. This document is intended to define the technical design requirements for developing a complete architecture for electronic filing.

A. OBJECTIVES

OXCI has identified conflicts between the existing Court Filing standards and the other technical requirements of the OXCI court filing architecture. For instance, the current Court Filing standard, Version 1.1, is defined as a Document Type Definition (DTD), which is inconsistent with the OXCI requirement that schemas be used rather than DTDs. While the next Court Filing specification, "Blue" is to be based on schemas, it is still a work in progress.

In addition to identifying conflicts, OXCI has also identified gaps in the architecture that must be addressed prior to implementation. For instance, although OXCI will require court documents to comply with the Court Filing XML (CF XML) standard and that the documents be transmitted using the Simple Object Access Protocol (SOAP) as the messaging protocol, the interface between the Court Filing and SOAP standards has not yet been fully defined.

The purpose of this document is to develop the requirements and architecture down to a level of detail that addresses the conflicts and gaps in the existing family of standards and to provide strong guidance, where possible, about preferred solutions.

B. DOCUMENT ORGANIZATION

As stated above, this document provides the requirements and high-level architectural design decisions for the court filing architecture. The Court Filing, Query/Response and Court Policy schemas are described in a separate deliverable, the XML Schemas document. The detailed implementation decisions for the EFM software component are described in the EFM Software Requirements and EFM Software Design documents.

The rest of this document is organized as follows:

- Section II includes the architectural requirements of the OXCI EFM.
- Section III lists a number of design questions that must be resolved including candidate solutions and the OXCI decision for each.
- Section IV proposes a system and software architecture including the components, interfaces, classes, and use cases.

There are also two appendices:

- APPENDIX A provides a glossary of the acronyms used in this document.
- APPENDIX B includes a bibliography of reference material used in the creation of this document.
- APPENDIX C includes a revision history for this document.

II. REQUIREMENTS

II. REQUIREMENTS

This section lists the architectural requirements for the OXCI Electronic Filing Manager. The requirements are organized according to the following categories:

- *Functional Requirements* describe the users, functions, and components of the EFM.
- *Information Requirements* provide a description of the court information standards and documents supported by the EFM.
- *Integration Requirements* describe the required integration approach using the Web services standards.
- *Policy Requirements* provide a description of other requirements regarding the appropriate use of technologies.
- *Business Model Requirements* describe several business models that must be supported by the EFM.

A. FUNCTIONAL REQUIREMENTS

The functional requirements for the EFM architecture include the users and functions of electronic filing systems and the components that compose a complete EFM.

1. Users and Functions

The EFM architecture must support the users and functions defined in the OASIS LegalXML Court Filing 1.1 standard (hereafter referred to as the Court Filing standard). The Court Filing standard lists the users and user-specific functions of an electronic filing system as follows:

- Attorneys and Pro Se defendants and plaintiffs.
  - » File pleadings.
  - » Send and receive notifications.
  - » Review pleadings, orders, and notices of individual cases.
  - » Open criminal cases.
  - » Open civil cases.

- Judicial officers and judicial support staff.
  - » File orders.
  - » Send and receive notifications.
  - » Review pleadings, orders, and notices of individual cases.
- Court clerks.
  - » File orders and notices within court.
  - » Send and receive notifications.
  - » Review pleadings, orders, and notices of individual cases.
  - » Keep the court files, including sealed, confidential records.
  - » Provide access to court files.
- Clerk staff.
  - » Receive, index, and file pleadings, orders, and notices for litigants, attorneys, judges, and clerk of court.
  - » Review queued entries prior to docketing.
  - » Review pleadings, orders, and notices of individual cases.
- System administrators (super users).
  - » Maintain (add, delete, modify) user lists.
  - » Maintain databases.
  - » Maintain court policy.

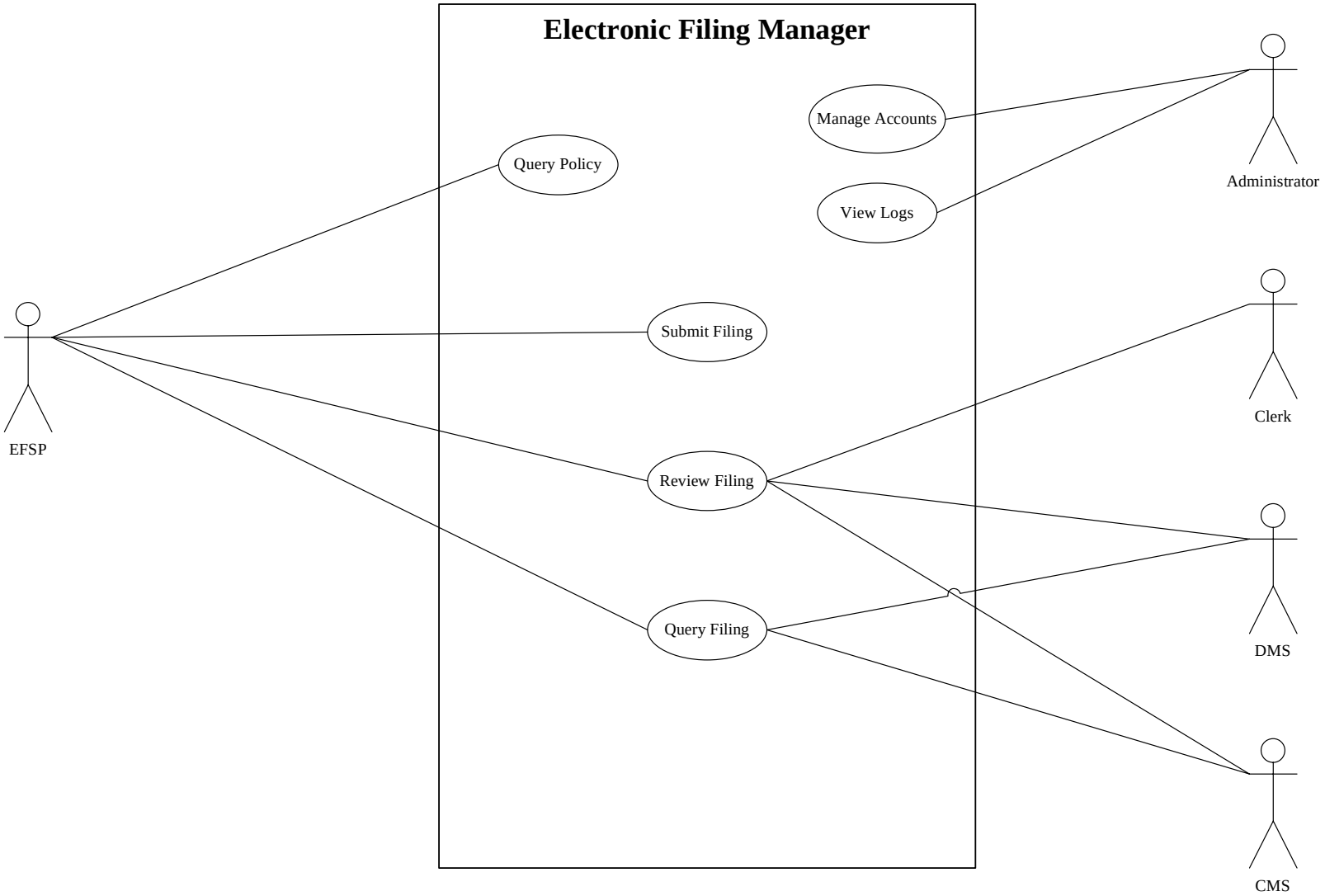
In this document, we will use the term "filer" to refer to any individual or entity that makes an electronic filing. Common examples of filers are attorneys; judicial officers; court clerks; public sector entities such as city, county, state, and federal governments; and private sector entities including corporations.

EXHIBIT I presents a Unified Modeling Language (UML) system context diagram that distills the users and functions described in the Court Filing standard into six use cases. The use cases include the following:

- *Submit Filing* in which a filer submits a filing with the court.
- *Review Filing* in which a clerk reviews and accepts or rejects filings submitted to the court.

OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

USE CASE DIAGRAM



- *Query Filing* in which a user requests information on a specific case from the court Case Management System (CMS).
- *Query Policy* in which a user obtains the policies specific to a court.
- *Manage Accounts* in which an administrator creates, modifies, and removes accounts and privileges on the EFM.
- *View Logs* in which an administrator reviews the system and application logs for the EFM.

The architecture must define a specific sequence of events and exchanges for each of the above use cases. These use cases will be developed in detail in Section III.

2. Court Filing Components

The EFM architecture must also support the components of an electronic filing system defined in the Court Filing standard. The Court Filing standard and the "Standards for Electronic Filing Processes" document defines five basic components: the EFM, the Electronic Filing Provider (EFP), the Electronic Filing Service Provider (EFSP), the Case Management System (CMS), and the Document Management System (DMS). The Court Filing standard defines each of these components as follows:

EFM

An EFM (or management system) is middleware that receives, presents, and manages electronic filings; the EFM is also considered to be the server in the electronic filing process.

EFP

An EFP is a front-end application that prepares and submits filings. The EFP is the application on the filer's side of the electronic filing architecture, and it is also called the client.

EFSP

The EFSP is the architectural component that supports a user's creation of a filing for submission to a court. The component may be provided by a court or a separate entity, such as a commercial vendor.

CMS

A court CMS manages the receipt, processing, storage, and retrieval of data associated with a case and performs actions on the data.

DMS

A DMS manages the receipt, indexing, storage, and retrieval of electronic and nonelectronic documents associated with a case.

B. INFORMATION REQUIREMENTS

The EFM Architecture must adhere to open standards for describing both general information and court-filing-specific information. This includes adherence to the W3C XML Schema and OASIS LegalXML Court Filing standards.

1. W3C XML Schema

The World Wide Web Consortium (W3C) is an organization of companies and individuals that develop and promote the open standards that define the Web, including HyperText Transfer Protocol (HTTP), HyperText Markup Language (HTML), and XML. XML 1.0 is the protocol recommended by the W3C for describing structured information on the Web. Although XML 1.0 includes a standard for defining XML structures, the Document Type Definition (DTD), W3C now recommends the use of a later standard, XML Schemas, for defining XML data exchange structures.

In support of the W3C standards, the EFM architecture must adhere to the XML 1.0 standard for all information exchanged between agencies and use XML Schemas for defining all XML structures.

2. OASIS LegalXML Court Filing

OASIS is a nonprofit consortium of organizations and individuals dedicated to the development and promotion of XML standards for electronic business in a wide range of industries. The LegalXML member section of OASIS supports the legal and justice community and includes TCs developing XML standards for court filing, notary services, Legislatures, and others. The Court Filing TC focuses on standards for electronic filing of court information. In order to support the widest range of courts and court information, the Court Filing TC has a policy of making many portions of its specifications "over-inclusive but optional."

OXCI requires compliance with the Court Filing standards; therefore, the EFM architecture must support the following Court Filing specifications:

Court Filing

The Court Filing 1.1 specification, developed in collaboration with Consortium of State Court Administrators (COSCA) and the National Association of Court Managers (NACM), is an OASIS Proposed Standard. The specification includes a DTD defining a **legalEnvelope** structure for encapsulating legal filings. The DTD also defines elements for submitting and confirming filings with a court and placeholders for query and response elements to be defined in the Query/Response specification. The data elements in the Court Filing 1.1 specification have been reconciled with elements in the Rap Sheet, Regional Information Sharing Systems (RISS), and American Association of Motor Vehicle Administrators (AAMVA) XML through the Justice XML initiative.

Because XML Schemas were designed to replace DTDs, it should be possible to convert the Court Filing DTD to a schema with little or no loss of interoperability. However, the DTD also includes elements that overlap with elements in the Web service protocols that are described in subsection II.C. Therefore, at a minimum, the EFM architecture should modify the Court Filing schema to minimize or eliminate the overlap of elements. Unfortunately, this results in incompatibility with strict implementations of the current Court Filing specification. Consequently, the OXCI EFM will not attempt to maintain compatibility with the Court Filing 1.1 specification.

The Court Filing TC is currently developing a follow-on specification code-named Court Filing "Blue." According to the working definition developed at the December 2003 meeting of the TC, "OASIS LegalXML Court Filing Blue is a set of specifications that provides the ability to electronically exchange information between and among the courts, their partners, and customers." At the start of this project, the requirements for Court Filing standard Blue were still in development and included the following principles:

- Leverage existing data and messaging standards.
  - » W3C XML Schemas.
  - » Global Justice XML Data Model (GJXDM) 3.0.
- Support court-specific extensions.
  - » W3C XML Namespaces.
  - » LegalXML Court Policy.
- Support multiple levels of interoperability.
  - » Level 1: LegalXML Court Filing Blue envelope.
  - » Level 2: Level 1 with a supported messaging standard, such as Electronic Business XML (ebXML) Messaging Service 2.0, and server authentication.

- » Level 3: Level 2 with user authentication and access controls.

- Establish recognized methods for messaging and communication.
  - » HTTP.
  - » Simple Mail Transfer Protocol (SMTP).
  - » File Transfer Protocol (FTP).
  - » Secure Sockets Layer (SSL).
  - » Web services (SOAP, Web Services Description Language [WSDL], Universal Description, Discovery and Integration [UDDI]).
  - » ebXML.
  - » Secure/Multipurpose Internet Mail Extensions (S/MIME).
  - » Synchronous or asynchronous operation.
- Support security.
  - » XML Signatures to support authentication, non-repudiation and document integrity.
  - » XML Encryption to support sealed documents.
  - » Security Assertion Markup Language (SAML).
  - » Public-key certificates.
  - » Privacy to support protection of payment information.
- Comply with governmental standards.
  - » Federal Information Processing Standards (FIPS).
  - » National Crime Information Center (NCIC) standards.
  - » Health Insurance Portability and Accountability Act of 1996 (HIPAA).
- Support court functional standards.
  - » COSCA/NACM.
- Support all court types and court filing types.
- Support all payload types.
  - » Portable Document Format (PDF).
  - » Tagged Image File Format (TIFF).
  - » XML documents (court document).

  - » Others.
- Support compatibility independent of vendor or product.
- Support version control.

The OXCI EFM must support the Court Filing Blue specification when it reaches the Proposed Recommendation stage. In the interim, the OXCI EFM has developed a Court Filing schema which is documented in the XML Schema deliverable. OXCI will submit the Court Filing schema to LegalXML for review, comment, and consideration in future specifications. Ultimately, OXCI will target Level 2 interoperability with other Court Filing Blue implementations. The specific design decisions will be discussed in Section III.

Query/Response

In December 2002, the Court Filing TC published a draft Electronic Court Filing Query and Response specification. The Query/Response specification defines an XML DTD for sending queries through the EFM and receiving responses from the court CMS and/or DMS. The query and response elements are designed to be encapsulated within a Court Filing 1.1 **legalEnvelope** element. The specification also defines a set of queries to be supported by a compliant CMS or DMS.

The OXCI EFM must conform to the Query/Response specification when it reaches the Proposed Recommendation stage. The Court Filing TC has halted further development of the Query/Response specification until the Court Filing Blue specification is more clearly defined. In the interim, OXCI has developed a Query/Response schema which is documented in the XML Schema deliverable. OXCI will submit the schema to LegalXML for review, comment, and consideration in future specifications.

Court Policy

The Court Filing TC created a subcommittee to develop a Court Policy specification. In November 2002, the subcommittee published a draft of a Court Policy Interface Requirements document that was reviewed and rejected by the TC. The draft explained that the "over-inclusive but optional" principle of the current Court Filing necessitates a means for an interface that describes a court's rules and administrative procedures. The Court Policy specification was designed to meet the "need for all involved with electronic filing (courts, parties, attorneys, prosecutors, and so forth) to know the expectations and/or constraints placed on the data elements and other aspects of a given electronic filing system. The information that describes a Court Policy for a given court will likely be fairly static and could therefore be described in a schema published in a well-known location by each court.

The OXCI EFM must support the Court Policy specification when it eventually reaches the Proposed Recommendation stage. The Court Filing TC has halted further development of the Court Policy specification until the Court Filing Blue specification is more clearly defined. In the interim, OXCI has developed a Court Policy schema which is documented in the XML Schema deliverable. OXCI will submit the schema to LegalXML for review, comment, and consideration in future specifications.

CMS Application Program Interface

The Court Filing TC created a CMS Application Program Interface (API) subcommittee to develop standards for interfacing between the EFM and a CMS or DMS. In 2001, the subcommittee produced a requirements document titled "EFM-CMS Interface Requirements," which describes the requirements for the CMS API. The document proposes that the CMS API should conform to the Court Filing, Court Policy, and Query/Response specifications and provides additional requirements regarding transactions, error handling, and security. The TC suspended further development of the CMS API specification until other architectural issues were resolved. The EFM must conform to the CMS API specification once work resumes and it eventually reaches the Proposed Recommendation stage. In the interim, the EFM must adhere to the requirements defined in the EFM-CMS Interface Requirement document. However, it is not reasonable to assume that every CMS and DMS will adhere to the CMS API standard. Therefore, OXCI recommends that each EFM implementation include court-specific CMS and DMS Adapters that provide CMS API-compliant interfaces to the CMS and DMS.

C. INTEGRATION REQUIREMENTS

With the Internet as the prime example, the advantages of building architectures around open standards are well established. The leading software vendors have now embraced the Web services standards as the best way to design applications for global interoperability. In order to maximum independence from the underlying architectures and to maximize compatibility with current and future applications, the EFM architecture must support the baseline Web service protocols defined by the W3C and OASIS, including:

- Simple Object Access Protocol (SOAP).
- WSDL.
- UDDI.

The function and advantages of each of the baseline Web service protocols are described in the following subsections.

1. SOAP

SOAP is the fundamental enabling technology for Web services. Developed originally by a group of software vendors, including Microsoft and IBM, the SOAP specification was submitted to the W3C in 1999 for adoption as a standard. SOAP provides basic messaging layer functions using XML to support language- and platform-independent remote procedure calls (RPCs) across the Internet infrastructure. To date, two versions of the SOAP specification have been published. The SOAP 1.1 specification is technically a W3C "Note," indicating its status as an interesting proposal but granting it no specific endorsement by the W3C. This is most likely due to long-standing Intellectual Property Rights (IPR) with Microsoft, IBM, and other vendors that were finally resolved in 2002. However, SOAP 1.1 has been widely implemented and is in fact a well-tested "de facto" standard. The SOAP 1.2 specification, a minor update to SOAP 1.1, is in active development and has reached the W3C Candidate Recommendation stage. There are also a number of important extensions to SOAP, such as the SOAP Messages with Attachments, XML Signature, SAML and WS-Security specification, that can provide additional features to the messaging layer as they are needed.

2. WSDL

The WSDL specification defines an XML-based protocol for describing a Web service. The WSDL specification was codeveloped by Microsoft and IBM and submitted to the W3C in 2001. The current version, the WSDL 1.2 specification, is a W3C Working Draft. WSDL includes all the information that a system needs to locate a SOAP-based Web service on a remote system, connect and bind to the Web service, issue RPCs, and receive the results.

One of the required properties of any Web service is that it be self-describing. Therefore, the EFM must provide WSDL definitions for each Web service-based interface.

3. UDDI

The UDDI specification defines an XML-based protocol for both publishing WSDL definitions in a registry and for querying registries for Web services and schemas. UDDI enables application developers to identify published Web services that meet certain criteria, facilitating the outsourcing of those components of the application. Microsoft and IBM codeveloped the UDDI specification and submitted it to OASIS for approval. Currently, UDDI versions 2 and 3 have reached the Committee Specification stage in the standards process and the UDDI Specification TC intends to submit the UDDI Version 2 specification for approval as an Approved OASIS Standard.

In Web service deployments involving small numbers of organizations and interfaces, WSDL information can be easily exchange directly, obviating the need for a central registry. However, as the use of Web services scales up to include more organizations and more interfaces, a registry standard such as UDDI becomes essential. Therefore, the EFM must eventually support publication of WSDL definitions for each of the supported Web services to UDDI-based registries.

D. POLICY REQUIREMENTS

In addition to the functional, information, and integration requirements, OXCI has identified some policy requirements for the EFM architecture. The EFM should:

1. Use freely licensed and open source technologies.

The "open" in the OXCI name indicates a preference for using freely licensed and open technologies for achieving electronic court integration. An extension of this principle is a preference for using open source software. Using open source software allows anyone to view or modify the code at any time and contribute those modifications back to the public codebase. As a policy, the EFM architecture should minimize its dependence on technologies that are either proprietary or known to have IPR restrictions that would limit implementation of the architecture. In addition, all source code necessary to implement or extend the EFM should be made freely available.

2. Simplify the architecture to minimize the cost and complexity of implementation.

The resources available for implementing electronic court filing can vary considerably according to the size and function of the court. Typically, small courts do not have the financial resources or technical capabilities for expensive or complex information technology projects. If the EFM architecture is to be applicable to both small and large courts, the essential portions of the architecture must be simplified to minimize the cost and complexity of a baseline implementation.

3. Scale to support both large and small courts.

The EFM must be simple and inexpensive enough to be practical for implementation by small courts. However, the architecture must also be sufficiently extensible so that courts with greater resources can add the additional functionality that they require.

4. Support all the electronic filings for a single court.

The EFM must be able to support all the filings required by a court. That is, no single court should need to implement multiple EFMs. There is a wide variety of legal cases including criminal, civil, juvenile, family, traffic, tax, bankruptcy, military, and tribal laws, and numerous combinations of these cases may occur in a single court. The EFM must therefore be flexible and extensible enough to support any and all of these case types.

5. Support filing fees and payments managed through an external payment system.

The EFM must be able to pass through payment requests and receipt information of filing fees and other payments to the clerk review interface and the CMS. However, the EFM will not be responsible for collecting or distributing the fees.

6. Support antivirus checking.

The EFM must support the checking of any documents for viruses before the documents are transmitted to another system. This includes documents being filed into the DMS or sent in response to a query. The antivirus interface will be defined in the Software Design deliverable.

E. BUSINESS MODEL REQUIREMENTS

At the December 2002 face-to-face meeting of the LegalXML Court Filing TC, Mr. Dallas Powell, representing the Tybera Development Group, Inc., presented an analysis of court filing business models titled "Architectural Models, Business Decisions, and Interoperability Issues." The paper described and identifies issues with each of the following models:

- *Court Control Model* in which the court manages both the EFSP and the EFM.
- *Vendor Control Model* in which vendors provide both the EFSPs and the EFM.
- *Split Control Models* in which vendors provide the EFSPs that connect to the EFM managed by the court.
- *Single Source Control Model* in which courts and automated legal firms all provide their own EFSP and EFM.

As Court Filing standards continue to evolve, it is unclear which, if any, of these models will predominate. Most likely, all three models will be implemented to some degree. In order to be

applicable to a wide range of courts, the EFM must be flexible enough to support each of these models.

* * * * * *

This section presented the requirements of the EFM as defined in the RFP and in the LegalXML Court Filing standards. The next section will identify the key design issues that need to be addressed prior to implementation of the EFM and will use these requirements as a context for arriving at a solution for each issue.

III. DESIGN DECISIONS

III. DESIGN DECISIONS

The Court Filing standards provide good definitions for the application-layer components of electronic court filing applications. However, many of the standards are incomplete and some conflicts exist between the standards. In addition, the standards do not address the other components that make up a complete system specification. OXCI has identified the key design decisions that will provide the framework for the EFM implementation. The following table summarizes the design issues, candidate solutions, and OXCI's decisions described in this section.

Design Issue	Candidate Solutions	OXCI Decision
Network Protocols	TCP/IP, UDP/IP	TCP/IP
Communication Protocols	HTTP 1.1, FTP, SMTP	HTTP 1.1
Messaging Protocols	SOAP 1.2 With Attachments and WS-Security, ebXML Messaging 2.0	ebXML Messaging Service 2.0
Authentication Protocols	SSL, XML Signature	EFSP: SSL as needed; Filer: None
Encryption Protocols	SSL, XML Encryption	SSL; Future ebXML Messaging support for XML Encryption as needed
Application Data Structure Definitions	XML DTD, XML Schema, RDF	XML Schema
Application Envelope Schema	Court Filing 1.1 **legalEnvelope** element, New GJXDM 3.0-based **FilingSubmissions** element, None	Court Filing Blue when available; New GJXDM 3.0-based **FilingSubmissions** element
Application Object Schema	Court Filing 1.1, New GJXDM 3.0-based Court Filing, Query/Response and Court Policy Objects	Court Filing Blue When Available; GJXDM 3.0-based Court Filing, Query/Response and Court Policy Objects in Interim
Service Description Schema	WSDL, ebXML CPP/A 2.0, None	WSDL
Collaboration Agreement Schema	WSEL, ebXML CPP/A 2.0, None	None
Registry and Repository Schema	UDDI, ebXML Registry, None	UDDI (future)
Database Interface	Relational, Object-Oriented	Relational
Location of the Clerk Review Interface	In the CMS, in the EFM	In the EFM

In the following subsections, we describe each design issue to be considered, compare the candidate solutions, decide on a selected solution, and provide the reasoning behind the decision.

A. NETWORK PROTOCOLS

Network protocols refer to the protocols in the Open System Interconnection (OSI) network model that provide functions such as global network addressing and routing. Although there are a wide range of network protocols in use, such as IPX and Appletalk, since the global adoption of the Internet, the Internet Protocol (IP) family of protocols, have emerged to become the de facto standards for network connectivity. Therefore, there are basically two choices for network protocols: Transmission Control Protocol/Internet Protocol (TCP/IP) and User Diagram Protocol/Internet Protocol (UDP/IP).

1. TCP/IP

TCP/IP is a reliable protocol for encapsulating and transmitting data between hosts on IP networks. The TCP packet header includes sequence numbers and acknowledgements that automatically segment and reassemble information in the correct order and retransmit information that is lost in transmission. TCP is the protocol underlying the Internet's predominant application protocols, including HTTP and FTP used on the Web and SMTP used for e-mail.

2. UDP/IP

UDP/IP is a high-performance protocol for encapsulating and transmitting data between hosts on IP networks. UDP packets include a minimal header that does not provide reliable delivery. Applications that typically use UDP include multimedia applications that require high performance but do not require reliable delivery.

Decision: TCP/IP

The EFM uses TCP/IP as the network protocol for the following reasons:

- UDP/IP does not support reliable delivery.
- TCP/IP is required by all of the major Internet application protocols.

B. COMMUNICATION PROTOCOLS

Communication protocols (also called application protocols in the OSI network model) provide the functionality that drives the key applications on the network. The key Internet communication protocols are HTTP, FTP, and SMTP.

1. HTTP 1.1

HTTP is an Internet protocol that supports the transfer of information from a Web server to a Web client, typically a Web browser. The information may be either static files or information that is dynamically generated by the Web server. Because HTTP is so widespread and well supported, many applications have also adopted HTTP as the standard communication protocol for transferring all types of data that are not necessarily related to the Web.

2. FTP

FTP is an Internet protocol that supports the transfer of files from a file server to a client. Files available through FTP are almost universally static files.

3. SMTP

SMTP is the Internet protocol for the exchange of e-mail. Although SMTP can be used for messaging between applications, it is typically only used for transmitting mail between users.

Decision: HTTP 1.1.

The EFM uses HTTP as the communication protocol for the following reasons:

- HTTP is widely used for exchange between applications.
- HTTP supports firewall transparency; that is, it is allowed through most firewalls.
- Although the required messaging protocol, SOAP, theoretically supports HTTP, FTP, and SMTP, the SOAP binding is only fully defined over HTTP.

C. MESSAGING PROTOCOLS

Messaging protocols provide functionality needed for integration across enterprise applications such as application-layer transport, routing and packaging (TRP), encapsulation, security, and RPCs. There are several XML-based messaging protocols, including XML-RPC, but the two most common

messaging protocols based on Web services are SOAP and ebXML Messaging Service (ebMS) 2.0 protocol. The table below lists the critical and secondary features associated with each of these protocols. The advantages of each protocol are shown in bold.

Component	SOAP (GXA)	ebMS 2.0
SOAP Version	1.2 w/Attachments	1.1 w/Attachments
SOAP Status	Recommendation	Note (well tested)
Security Protocol	WS-Security	Included
Security Protocol Status	Candidate Recommendation	**Standard**
Reliable Messaging Protocol	WS-Reliable Messaging	Included
Reliable Messaging Protocol Status	Working Draft	**Standard**
Critical Feature		
XML Signature	Yes	Yes
XML Encryption	**Yes**	Future
Error Reporting	**Better**	Good
Secondary Feature		
HTTP Binding	**Better**	Good
RPC	**Better**	Good
Time Stamps	**Create, Expire, Receive**	Create
Manifest	No	**Yes**
Routing	Requires WS-Routing	**Yes**
Support	Microsoft, IBM, BEA	freebXML, Sun, HP, IBM, Sybase, XML Global

1. SOAP 1.2 Messaging With Attachments and WS-Security

Based on the requirements, the EFM must conform to either the SOAP 1.1 or SOAP 1.2 specifications. However, both SOAP specifications require extensions that address additional functions such as encapsulation and security. Microsoft is leading the development of many of these SOAP extensions and refers to the combination of SOAP and these extensions as the Global XML Web Services Architecture (GXA). Two of the GXA extensions are the SOAP Messaging with Attachments and WS-Security specifications. The SOAP Messaging with Attachments specifications extend SOAP 1.1 and 1.2 to support encapsulation using either the Multipurpose Internet Mail Extensions (MIME) or Direct Internet Message Encapsulation (DIME) protocols. The WS-Security specification adds security extensions for supporting the XML Digital Signature and XML

Encryption specifications within SOAP 1.2 (but not SOAP 1.1). The WS-Reliable Messaging specification which will support acknowledgements and retransmissions is currently an OASIS Working Draft.

2. ebMS 2.0

ebXML is a joint effort between OASIS and the United Nations Centre for Trade Facilitation and Electronic Business (UN/CEFACT) which are responsible for the Electronic Data Interchange (EDI) standards. ebXML is dedicated to the development of standards for electronic business, focused on the needs of small- and medium-sized businesses. ebXML is responsible for several OASIS and UN/CEFACT specifications, including:

- ebXML Messaging.
- ebXML Registry.
- ebXML Collaborative Partner.
- ebXML Implementation.
- ebXML Business Process.
- ebXML Core Component.

ebMS 2.0 is an Approved OASIS Standard based on SOAP 1.1 with Attachments. It includes extensions for XML Digital Signatures. The ebXML Messaging TC decided to wait to support XML Encryption until it became a W3C Recommendation and XML Encryption was approved as an OASIS Recommendation in December 2002. The next version of the ebXML Messaging specification, version 3.0, will support XML Encryption.

ebMS 2.0 also supports reliable messaging. SOAP is designed for sending one-way messages. SOAP extensions for reliable messaging associate RPC requests with their responses and handle application-layer acknowledgements and retransmissions. Without support for reliable messaging in the messaging protocol, applications need to implement this functionality, often in a nonstandard way.

Decision: ebMS 2.0.

The EFM uses ebMS 2.0 as the messaging protocol for the following reasons:

- ebMS 2.0 is a more mature specification. It is an approved standard while WS-Security is a Committee Recommendation and WS-Reliable Messaging is a Working Draft.

- The ebMS is well supported by both free and commercial messaging solutions including freebXML Hermes, XML Global GoXML Messaging, and Sun ONE Integration Server.
- The ebXML framework is freely licensed, while GXA still has some unresolved IPR issues.

D. AUTHENTICATION PROTOCOLS

Security protocols control access to information. Among other security functions, the EFM must support authentication and nonrepudiation. Authentication refers to the verification that a user is whom he/she claims to be. According to the *Information Security Handbook*, nonrepudiation refers to an "authentication that with high assurance can be asserted to be genuine, and that cannot subsequently be refuted." The court needs the ability to authenticate the EFSP in order to limit which vendors are authorized to submit electronic filings with the court. The court also needs the ability to authenticate the filer in order to link the filer to the filing and to limit which filers are authorized to submit electronic filings with the court. The two leading protocols for authentication of Web services are SSL and XML Signatures.

1. SSL

SSL is a communications-layer protocol for authenticating and encrypting a wide range of network communications. SSL is most often used to secure HTTP communications between a Web server and a browser. However, SSL is also frequently used to secure HTTP communications between Internet applications, including Web service applications. SSL supports authentication by password or by public-key certificate.

2. XML Signature

The W3C has published a Recommendation titled "XML Signature Syntax and Processing" for describing digital signatures using XML. Digital signatures are values computed using various cryptographic techniques that can be attached to messages to validate the identity of the sender and/or the integrity of the message. Digital signatures are usually created using public-key certificates. Both ebMS 2.0 and SOAP support the XML Signature specification.

Decision: SSL for EFSP; EFSP will authenticate the filer.

The EFM authenticates the EFSP and passes through any authentication of the filer. The EFM uses SSL for authenticating the EFSP for the following reasons:

- The EFSP communicates directly with the EFM. Therefore, the connection can be secured at the communication layer between the HTTP server and the HTTP client.
- Nonrepudiation of the EFSP is not a critical requirement for the EFM.

The EFSP authenticates the filer, and the particular mechanism for filer authentication is not within the scope of this architecture. However, some courts may require the filer's authentication information for the following reasons:

- The filer does not always communicate directly with the EFM. Therefore, the filer's authentication needs to be embedded in the application information rather than at the communication layer.
- The ability to irrefutably link the filer to the filing through nonrepudiation is critical to the integrity of electronic court filing systems. Digital signatures are much better than passwords in meeting this requirement.

Therefore, if the filing includes the XML Signature of the filer, the EFM will pass this information through to the CMS and DMS.

Beyond the authentication protocol, the EFM should support additional access controls to protect against denial-of-service attacks or any other malicious attacks against the EFM and its back-end systems including the CMS and the DMS. At a minimum, the EFM should be protected by a firewall that supports access controls lists (ACLs) that filter by TCP/IP ports and/or addresses. In this configuration, the court should consider limiting external access to the EFM to the IP addresses of the EFSPs. Combined with SSL authentication, this will provide two-factor authentication of all incoming traffic.

An additional layer of protection could be achieved by installing a second ebXML Message Service Handler (MSH) external to the firewall and routing all messaging through both the external MSH and the MSH in the EFM inside the firewall. However, this configuration will not be tested as part of this project.

E. ENCRYPTION PROTOCOLS

The EFM also needs to support confidentiality. Although most court filings are public record, some filings will be sealed and access to them would be tightly restricted. Confidentiality is achieved by encrypting the information in such a way that only the authorized recipients have the ability to decrypt the data. The two leading protocols that support encryption of Web services are SSL and XML Encryption.

1. SSL

In addition to authentication, SSL supports encryption at the communication layer. However, this only protects the information during the transmission from the sender to the receiver. In the case of an electronic filing, this means that the communication between the EFSP and the EFM could be encrypted using SSL, but SSL would not encrypt the filing within the EFM.

2. XML Encryption

The W3C has published a Recommendation titled "XML Encryption Syntax and Processing" for representing encrypted information in XML. XML Encryption is related to XML Signature although the combination of digital signatures and encryption is currently not defined in either standard. Unlike SSL, XML Encryption supports encryption at the application layer, which means that the data is encrypted both during transmission and on the EFM.

Decision: Use SSL as Needed; Future ebMS support for XML Encryption as needed.

The EFM uses SSL as needed now and will support ebMS 3.0 compatibility with XML Encryption for the following reasons:

- XML Encryption supports encryption of the data from end-to-end.
- Although encryption is an important feature of the EFM, it is required only in special filings. In the interim, these filings should be encrypted during submission using SSL or may be managed outside the electronic filing process.

F. APPLICATION DATA STRUCTURE DEFINITIONS

The EFM must adopt a standard for the definition of XML structures. The W3C has published three specifications for defining XML Structures: the XML DTD, XML Schema, and Resource Description Framework (RDF) specifications.

1. XML DTD

Using XML DTDs is the method for defining XML structures defined within the XML 1.0 specification. The current LegalXML Court Filing specifications are defined using DTDs. However, DTDs are limited in their support for defining data types and complex structures.

2. XML Schema

The XML Schema specifications are W3C Recommendations that suggest XML Schemas as replacement for DTDs. XML Schemas support more complex data types and can be very specific about the type, length, and even the allowed values for a particular data element.

3. RDF

The "RDF Model and Syntax Specification" is a W3C Recommendation for describing structures comparable to XML Schemas with metadata that enable improved search capabilities and, eventually, intelligent software agents. The RDF is the basis for the Semantic Web. Mr. Tim Berners-Lee, the founder of the W3C, describes the Semantic Web as "an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

Decision: XML Schema.

The EFM uses XML Schema for the following reasons:

- The RFP identifies the use of Schemas as a requirement for the EFM.
- The Schema standard supports more data types and more specific data definitions than DTDs.
- It is not clear that the applications that interface to the EFM, such as the CMS or DMS, will have the need or capability for using the additional metadata functionality provided by the RDF specification.

G. APPLICATION ENVELOPE SCHEMA

An application-specific envelope schema is needed to support the encapsulation of multiple documents and common header information in a single filing. The Court Filing 1.1 specification defines a **legalEnvelope** element that also supports the routing and bundling of court filing documents. The draft GJXDM 3.0 specification also includes many of the same elements.

1. Court Filing 1.1 **legalEnvelope** Element

The Court Filing 1.1 specification defines a root-level **legalEnvelope** element that contains the following elements:

- **messageIdentification** is a string used by the sending application to identify the message.
- **to** identifies the recipient of the transmission.
- **from** identifies the sender, and may provide the information needed to send a confirmation or response.
- **replyto** supplies the information of where to send the confirmation or response.
- **cc** identifies others receiving the transmission.
- **bcc** identifies others receiving the transmission.
- **memo** provides human-readable text, ignored by applications.
- **creation** identifies the date and time that the envelope was created.
- **dataIntegrity** is a place holder for the method used to validate the integrity of a message's content.
- **paymentInformation** identifies how the filer intends to pay, is paying, or has paid any court fees.
- **authentication** shall be used for authenticating the sender or for some other element containing a digital signature.
- **legal** is a generic tag preceding all legal-related XML.

There is some overlap between the elements in the **legalEnvelope** and the ebMS headers, specifically regarding the **messageIdentification**, **creation**, and **dataIntegrity** elements.

2. New GJXDM 3.0-Based **FilingSubmissions** Element

The GJXDM 3.0 specification includes replacements for many but not all of the elements defined in the Court Filing 1.1 specification. A new, hybrid **FilingSubmissions** element may be defined using XML Schema that incorporates elements from both the GJXDM 3.0 and the Court Filing 1.1 **legalEnvelope** element as needed.

3. No Application Envelope

Since the messaging layer already includes encapsulation support through MIME or DIME, many applications will not need an application envelope. However, the messaging layer is only used for transmission so any common information placed in the messaging header will be discarded once the application has been received.

*Decision: Court Filing Blue when available; New GJXDM 3.0-based **FilingSubmissions** element in the interim.*

The EFM uses a new **FilingSubmissions** element based on the GJXDM 3.0 for the following reasons:

- The GJXDM 3.0 supports compatibility with other justice implementations.
- The **FilingSubmissions** object supports common filing information across several filing documents.

The **FilingSubmissions** element is defined using XML Schema and will incorporate elements from the GJXDM 3.0 and the Court Filing 1.1 **legalEnvelope** element as needed. The **FilingSubmissions** element is defined in the XML Interface Specifications deliverable.

H. APPLICATION OBJECT SCHEMA

The EFM must adopt a standard schema for the application objects that describe the content of the filing, query, and policy exchanges. The two alternative models for court filing, query/response and court policy application objects are the Court Filing 1.1 specifications and new objects based on the GJXDM 3.0.

1. Court Filing 1.1 Elements

The Court Filing 1.1 standard defines elements and structures for the filing and confirmation objects. Although it is designed to support extension data elements, such as query/response, many of these extensions are still in development. In addition, implementations of the Court Filing standard, including the Georgia Courts Automation Commission Court Filing Interoperability Pilot, have identified deficiencies in the Court Filing 1.0 and 1.1 specifications. These known deficiencies include the following:

- The DTD does not provide sufficient validation capabilities. Simple validation against the DTD is not sufficient for interoperability.
- The “over-inclusive and optional” philosophy sometimes results in multiple ways to achieve the same result (e.g., encapsulation).
- The definitions of the elements in the Court Filing specification do not provide sufficient detail.

- The specification does not include a process for validation of compliance with the specification.
- The specification does not define security practices or protocols.

Despite these deficiencies, the Court Filing 1.1 specification represents the most complete open specification for electronic court filing developed to date.

2. New GJXDM-Based Court Filing, Query/Response, and Court Policy Objects

The successor version to Court Filing 1.1, currently referred to as Court Filing Blue, will include a completely new object-based content model for court filing using the elements and structures defined in the Justice XML 3.0 specification. The Georgia Technology Research Institute (GTRI) published a draft specification for Justice XML 3.0 in December 2002, which defines a set of core horizontal objects common to information exchanges across the justice process. The draft also defines an extension mechanism for defining "Activity Objects" that support integration within specific vertical industries. Justice XML 3.0 will include a set of "Court Filing Activity Objects" that specifically support court processes including electronic filing. As the Justice XML 3.0 core objects are finalized, GTRI will work with representatives from LegalXML and OXCI to develop the Court Filing Activity Objects. Although the process for developing Court Filing Blue is still being defined, the LegalXML Court Filing TC has stated that the Justice XML 3.0 and the Court Filing Activity Objects will provide the framework for Court Filing Blue.

Decision: Court Filing Blue objects when available; New GJXDM 3.0-based Court Filing, Query/Response and Court Policy Objects in the interim.

The EFM uses new court filing, query/response and court policy objects for the following reasons:

- The new objects will adopt the use of schemas to address the deficiencies in the Court Filing 1.1 DTD.
- The use of the GJXDM 3.0 specification will also improve compatibility with other justice standards by increasing the number of common elements over ten-fold.

I. SERVICE DESCRIPTION SCHEMA

An essential property of any Web service is that it be "self-describing." Web services typically provide this feature through service description and transport binding schema which are used at design-time for implementing client interfaces to the Web service. The two most common

specifications for describing Web services are the WSDL specification and the ebXML Collaboration Protocol Profile (CPP) and Agreement specification.

1. WSDL

As discussed in Section II, WSDL represents the baseline specification for describing the network location and transport-layer bindings of a Web service as well as the methods and interfaces available through the Web service. At a minimum, the EFM must support WSDL to meet the standard definition of a Web service.

2. ebXML CPP/A 2.0

The ebXML CPP/A 2.0 specification is an extension to WSDL. In addition to the features supported through WSDL, CPP/A also enables the parties in a Web-service exchange to publish additional details regarding each party's message exchange capabilities and business processes as their CPP. Although CPP/A is a member of the ebXML suite of specifications and is compatible with the ebXML Messaging and Registry and Repository specifications, there is no interdependence between the specifications.

3. No Requirement for Service Descriptions

A valid alternative to the use of WSDL and ebXML CPP/A would be to minimize complexity and forgo the requirement that the EFM interfaces be self-describing. Although technically not Web services, the EFM interfaces could still be based on SOAP and take advantage of other Web service specifications. The disadvantage of this approach would be that developers implementing interfaces to the EFM would still need the details regarding the address information, bindings, and protocols necessary to interface to the Web service.

Decision: WSDL.

The EFM will support WSDL for the following reasons:

- Supporting a service protocol simplifies the implementation of client interface to the EFM.
- WSDL is simpler to implement and supported by a wider range of development tools than the ebXML CPP/A specification.

Service description is required for the development of the EFSP, CMS, and DMS interfaces and must be supported by the OXCI EFM.

J. COLLABORATION AGREEMENT SCHEMA

Another feature of certain Web services is the ability to negotiate collaboration agreements between parties involved in the exchange that describe service-level parameters such as Quality of Service (QoS) and cost and security characteristics. These specifications are used at design-time to develop contract agreements regarding the messaging, bindings, and security protocols in common between the parties to be used in a specific exchange. Two specifications for negotiating collaboration agreements are the Web Service End-point Language (WSEL) and ebXML CPP/A.

1. WSEL

The WSEL specification developed by IBM defines a schema for negotiating collaboration agreements. WSEL includes descriptions for the QoS, cost, and security characteristics of a Web service interface and the typical sequence of operations supported by the interface. However, the WSEL has not yet been submitted by IBM to a standards organization for approval. WSEL is very early in the standards process, and it is unclear whether WSEL will receive the critical mass of support needed to become an accepted standard.

2. ebXML CPP/A 2.0

In addition to service description, the ebXML CPP/A specification also supports the negotiation of collaboration agreements between parties. Specifically, it facilitates the creation of agreements by determining the intersection of the CPPs from both of the parties participating in the exchange. In December 2002, the ebXML CPP/A 2.0 specification was approved as an Approved OASIS Standard.

3. No Requirement for Collaboration Agreements

A valid alternative to the use of WSEL and ebXML CPP/A is to minimize complexity and to not require a specific collaboration agreement schema. This capability is not a required feature of Web services and could be omitted without impacting the complexity of interfacing to the EFM.

Decision: No support for collaboration agreement.

The EFM does not require collaboration agreements for the following reasons:

- WSEL and ebXML CPP/A are not yet widely supported by Web service development tools.

- It is unclear whether there is a definite need for the automation of contracts between the parties involved in electronic court filing.

K. REGISTRY AND REPOSITORY SCHEMA

Registries are directories that enable organizations to publish the WSDL for their Web services so that developers may discover them at design-time. Repositories store the additional objects and schemas that developers need to implement interfaces to the Web services. The two most common specifications for describing Web service registries are the UDDI and ebXML Registry specifications.

1. UDDI

As discussed in Section II, UDDI represents the baseline standard for Web service registries. The OASIS UDDI Specification TC has published UDDI Version 2 and Version 3 as Committee Specifications.

To be scalable to a large number of clients and interoperable with other Web services, the EFM should support the UDDI specification.

2. ebXML Registry

The ebXML Registry and Registry Information Model 2.0 specifications define registries that provide similar registry capabilities to a UDDI Web services registry while also providing object repositories that support the submission and retrieval of objects that define Web services. eXML Registry version 2.0 is an Approved OASIS Standard. ebXML Registry version 2.1 is currently an OASIS Committee Specification.

3. None

A viable alternative to implementing UDDI or ebXML Registry would be to minimize complexity and not use a registry or repository standard. The disadvantage of this approach would be that, without a central registry, the location of Web services will be much more difficult.

Decision: UDDI.

The EFM will support the UDDI specification for registries for the following reasons:

- The need for central registries for Web services increases as the number of organizations and interfaces using Web services increases. Assuming that most EFM implementation involve multiple EFSPs filing with multiple courts, support for a Web services registry will simplify implementation of the EFSP interfaces to the EFMs.
- UDDI is simpler to implement and better supported than the ebXML Registry specification.

The registry and repository is not a required feature and will not be supported by the initial implementation of the OXCI EFM. However, as the number of implementations grows, registries and repositories will become essential for scalability, and therefore, will be supported.

L. DATABASE STRUCTURE

A critical component of the EFM will be some sort of database storage system for temporarily queuing filings and storing and retrieving filing information. The database will support system reliability by queuing filings and notices until they can be reviewed by the clerk or delivered to the CMS, DMS, or EFSP. The EFM architecture should adopt as a standard one of the two basic types of database interfaces, which are relational and object-oriented.

1. Relational Database Interface

Relational databases are the most common type of database designs. Relational databases store information in tables with each row in a table representing a set of related data. Relational database management systems (RDBMSs) such as Oracle, Sybase, Structured Query Language (SQL) Server, or MySQL support the vast majority of existing databases.

Standard relational database interfaces include the Open Database Connectivity (ODBC), and the Java Database Connectivity (JDBC), which use SQL. These interfaces are widely supported by the majority of databases, third-party applications, and database support tools. Relational database interfaces are also well supported by a large pool of individuals with relational database administration experience and skills.

2. Object-Oriented Database Interface

Although relational database interfaces store and retrieve tabular data efficiently, they do not support the storage and retrieval of object-based data, such as in the Justice XML 3.0 model. In these cases, it is often better to use an object-oriented database interface which provides native support for the storage and retrieval of objects. This simplifies implementation by reducing the database interface complexity that the application needs to implement.

Object database management systems (ODBMSs) natively support storage and retrieval of object-oriented data. However, many RDBMSs also provide object-relational adapters that support transparent storage and retrieval of objects. Standard object-oriented and object-relational database interfaces include ActiveX Data Objects (ADO) and Java Data Objects (JDO).

Decision: Relational Database Interface.

The EFM uses a standard relational database interface for internal storage of filings and filing information for the following reasons:

- Most courts and legal organizations already have a relational database infrastructure.
- The ebXML MSH requires a relational database interface.

M. LOCATION OF THE CLERK REVIEW INTERFACE

This EFM design issue involves a decision on the location of the clerk review interface. After filings are submitted to the court, the court clerk reviews the filing and accepts or rejects the filing. If the filing is accepted, the filing documents are stored to the DMS and a case is opened in the CMS. If the filing is rejected, the filing should be returned to the filer with a message explaining why the filing was rejected. When filings are rejected, the filing documents should not be stored in the DMS and no case should be opened in the CMS. The two most logical locations for the clerk review interface are in the CMS and in the EFM.

1. In the CMS

The CMS and DMS are the key applications in courts that manage their cases electronically. The court clerk regularly uses the CMS user interface, and adding a review interface for electronic filings should be trivial. Some CMSs may even already provide an interface for reviewing electronic filings.

2. In the EFM

The EFM could also support the clerk review interface, preferably as a Web interface. Although the EFM would be a new application for the court clerk, the interface could be as simple as a Web browser form. Because the EFM will already require an HTTP server to support the ebXML interface, the addition of a basic clerk review interface would not significantly increase the complexity of the EFM application.

Decision: In the EFM.

Clerk review should occur in the EFM for the following reasons:

- The EFM is the gateway into the court and provides electronic validation of the court filings. Clerk review is an extension to include the clerk in the validation process. As the Court Filing and Court Policy specifications evolve to become better defined, the ability of the EFM to electronically validate the filing should eventually improve to the point where no manual review by the clerk is necessary.
- Supporting clerk review at the EFM reduces the complexity of the interfaces with the CMS and DMS. If the clerk rejects the filing, the filing and documents never reach the CMS or DMS. If clerk review occurs at the CMS, rejections by the clerk would require the rollback (nullification) of several transactions to delete the documents from the DMS and to update the status of the filing in the EFM.

Although the EFM should include a basic clerk review interface, it should also support court-specific configurations in which the basic interface is replaced with more sophisticated clerk review interfaces that pull in data from a CMS and/or DMS. Therefore, the clerk review interface should be implemented as a replaceable module with well-defined APIs.

* * * * * *

Now that the key design issues have been identified and resolved, we can use the requirements and design decisions as the basis for an implementation. In the next section, we propose an architecture to support electronic court filing based on the requirements and design decisions.

IV. DESIGN ARCHITECTURE

IV. DESIGN ARCHITECTURE

In this section, we propose an architecture for the EFM implementation. The architecture is based on the requirements defined in Section II and the design decision discussed in Section III. The EFM architecture discussion is organized under the following sections:

- *System Architecture* – Introduces the system architecture including the system components.
- *System Interfaces* – Describes each system interface in detail, including the structure and content of each interface.
- *Software Components* – Describes each software component in detail, including the standards for implementing each component and interfacing between components.
- *EFM Framework* – Introduces the Counterclaim OpenEFM framework for implementing the EFM application and graphical user interface (GUI) components.
- *EFM Interfaces* – Proposes UML-based prototypes for the internal interfaces necessary to implement the EFM application component.
- *EFM Classes* – Proposes UML-based prototypes for the Java classes necessary to implement the EFM application component.
- *Use Cases* – Illustrates the UML-based interactions between classes for each of the six use cases defined in the requirements.

A. SYSTEM ARCHITECTURE

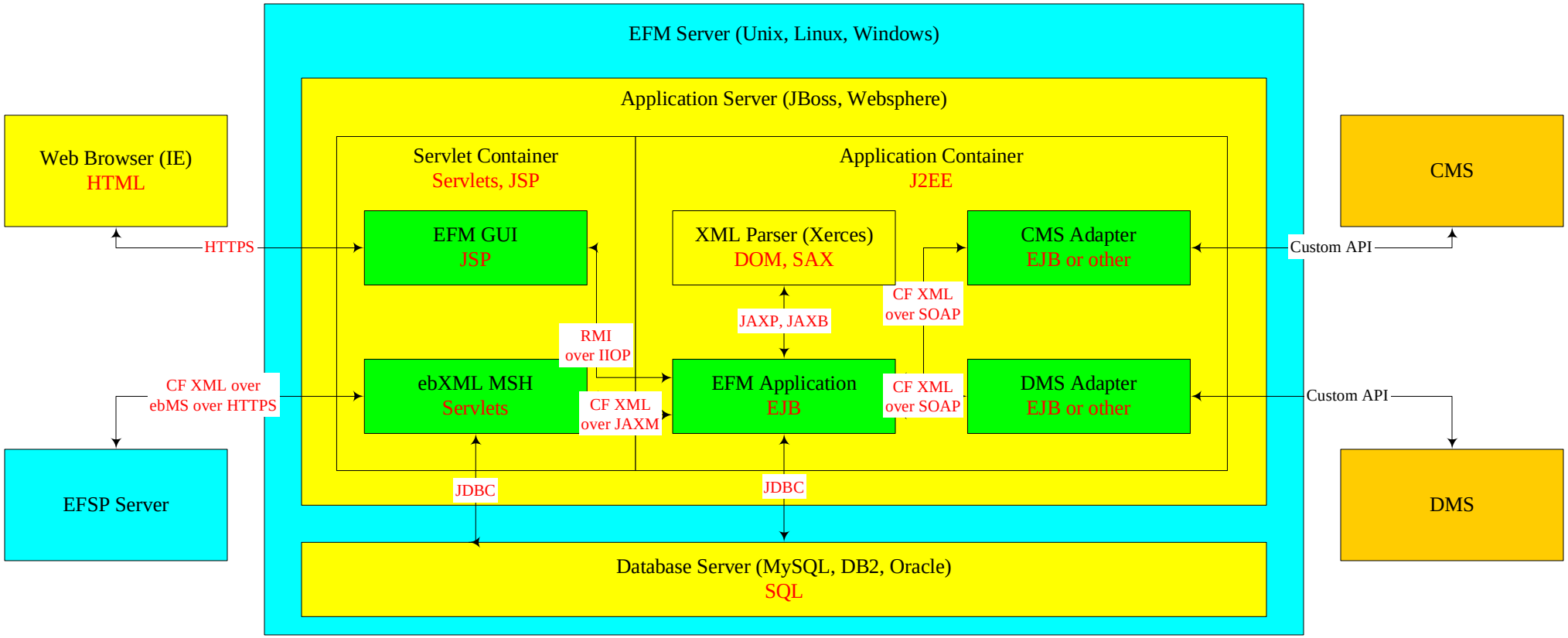
A proposed architecture for the EFM is shown in EXHIBIT II. Each of the system components in the exhibit is described below. The software components within each system component are described in subsection IV.C.

1. EFM Server

The EFM server represents the middleware that connects the EFSP, CMS, DMS, and Web browser components. It includes the Web server, application server, and database server components. The OXCI EFM server may be hosted on a Linux, Unix, or Windows platform.

OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

DEPLOYMENT VIEW



2. EFSP Server

The EFSP server is the filing service provider's system that interfaces with the EFM server. It may, in fact, be another implementation of the OXCI EFM server using the architecture described above.

3. CMS

The CMS is the case management system which stores case information, including information about documents filed with the court. The CMS is specific to each court or organization.

4. DMS

The DMS is the document management system that stores electronic or imaged versions of court documents. The DMS is specific to each court or organization.

5. Web Browser

The Web browser represents the client system and application used to support interactive filings, clerk review, and EFM administration. The OXCI EFM will be tested using the Internet Explorer Web browser.

B. SYSTEM INTERFACES

The four types of system interfaces supported by the architecture are:

- Court Filing.
- Query/Response.
- Court Policy.
- Payments.

These interfaces are implemented in XML and are collectively referred to as the CF XML interfaces. The following subsections describe the content of each interface and the system components that implement them.

1. Court Filing

The court filing interfaces include the submission of filing objects from the EFSP to the EFM, from the EFM to the CMS and DMS, and the return of one confirmation object for each filing. The filing objects are structured as follows:

SOAP Header
    ebMS Header
SOAP Body
    ebMS Manifest
    FilingSubmissions Element
        FilingSubmission Elements
MIME attached documents

The confirmation objects are structured as follows:

SOAP Header
    ebMS Header
SOAP Body
    ebMS Manifest
    FilingConfirmations Element
        FilingConfirmation Elements
MIME attached documents

2. Query/Response

The query/response interfaces include the query requests from the EFSP to the EFM, from the EFM to the CMS and DMS, and the response from each query request. Each query request must apply to a specific case – these interfaces are not designed to support queries across cases. The query objects are structured as follows:

SOAP Header
    ebMS Header
SOAP Body
    ebMS Manifest
FilingQueries Element
        Query Elements
MIME attached documents

The response objects are structured as follows:

SOAP Header
    ebMS Header
SOAP Body
    ebMS Manifest
FilingResponses Element
        Response Elements
MIME attached documents

3. Court Policy

The court policy interfaces include policy requests from the EFSP to the EFM and the return of the corresponding policy object from the EFM to the EFSP. The court policy interface is implemented in the EFM as a query and response as described above.

4. Payments

The payment interface supports payment requests and payment confirmations according to the UBL specification. The payment requests and confirmations are included as MIME attached documents as follows:

SOAP Header
SOAP Body
MIME attached documents
    PaymentRequest or Payment

C. SOFTWARE COMPONENTS

The system components in EXHIBIT II include a number of software components. Each software component is described below, including the implementation and interface standards for each component, and the products supported by the OXCI EFM for implementing that component.

1. Web Server

The Web server provides the container for all the HTTP interfaces. The Web server is compliant with the Java Servlet and Java Server Pages (JSP) specifications and hosts two subcomponents: the EFM GUI and an ebXML MSH servlet. The OXCI EFM will be tested with the Tomcat Web server which is bundled with most J2EE-compliant application servers.

EFM GUI

The EFM GUI represents the server-side code used to support clerk review, and EFM administration. The EFM GUI will not support filing into the local EFM or other EFMs or review of previous filings. The EFM administration functions supported by the GUI will not include database administration. The GUI is generated using JSP and interfaces with Web browsers using HTTPS and with the EFM application using Java Remote Method Invocation (RMI) over the Internet Inter-ORB Protocol (IIOP).

MSH

The MSH represents the message service handler that connects the EFSP with the EFM application. The MSH is implemented as a servlet. Between the MSH and the EFM, messages conform with the CF XML specifications over a Java API for XML Messaging (JAXM) interface. Between the MSH and the EFSP, messages conform with the CF XML specifications over an ebMS 2.0 interface. The MSH interface must be defined in WSDL.

The MSH must support both synchronous and asynchronous messages. For instance, in the SubmitFiling use case (see EXHIBIT VI-1 in subsection IV.G), the response from the FilingManager to the EFSP is a synchronous response in the same HTTPS session that indicates a filing disposition of "received." In the ReviewFiling use case (see EXHIBIT VI-2), the two updateFilingStatus messages from the FilingManager to the EFSP are asynchronous messages indicating filing dispositions of "accepted"/"rejected" and "filed."

2. Application Server

The application server provides the container for the business-logic components and interfaces to the back-end systems including the CMS and DMS. The application server is compliant with the J2EE 1.4 specifications and hosts the EFM application, XML parser, CMS adapter and DMS adapter subcomponents. The OXCI EFM will be tested with the JBoss and IBM Websphere application servers. It should be compatible with any J2EE-compliant application server.

EFM Application

The EFM application subcomponent provides the core business-logic for the application. The EFM application is implemented in Enterprise JavaBeans (EJB) 2.0 beans including entity beans and Container Managed Persistence (CMP). The internal framework, interfaces, and classes within the EFM application are detailed in the next three subsections.

XML Parser

The XML parser provides a service for traversing and extracting data stored in XML. The XML parser must support the Document Object Model (DOM) and Simple API for XML (SAX) specifications. The EFM application calls the XML parser using the Java API for XML Parsing (JAXP) interface. The OXCI EFM will be tested with the Xerces XML parser.

CMS and DMS Adapters

The CMS and DMS adapters are modular interfaces that connect the EFM application with the CMS and DMS. The CMS and DMS adapters may or may not be implemented using Java. The EFM will include a Java CMS Connector, defined using EJB 2.0 session beans, that provides a Web service interface between the EFM and any non-Java CMS adapters. This interface will support the CF XML over SOAP over HTTPS and be defined in WSDL. In the case of a Java CMS or DMS, the adapter will implement the CMS Connector interface directly and connect to the CMS or DMS using Java RMI. The adapter interfaces with the CMS or DMS using the custom API specific to that application.

The CMS Connector and CMS and DMS adapters must support both synchronous and asynchronous messages. For instance, in the ReviewFiling use case (see EXHIBIT VI-2), the initial response from the CMS and DMS adapter to the CMS Connector is a synchronous response in the same HTTPS session that indicates a filing disposition of "received." The updateFilingStatus messages from the CMS and DMS adapter to the CMS Connector are asynchronous messages indicating filing dispositions of "filed."

3. Database Server

The database server provides persistence for the MSH and the EFM application. The database server must support SQL and interfaces with the MSH and EFM application using the JDBC interface. The OXCI EFM will be tested with the MySQL, Oracle, and DB2 databases. It should be compatible with any relational database with a JDBC interface.

D. EFM FRAMEWORK

OXCI has selected the OpenEFM by Counterclaim as the development framework for the OXCI EFM application and GUI components. Leveraging an existing implementation will reduce the time and cost of implementation and will hopefully position the EFM for better interoperability with other court filing products in the future.

OpenEFM was developed by Counterclaim and released to the public as open source software licensed under the Mozilla licensing model. OpenEFM was developed in Java for Linux or Unix platforms and supports the Court Filing functional standards. OpenEFM includes the following components:

- A Web interface for submission of court filings using HTTP over SSL (HTTPS).
- A SOAP over HTTPS interface for submission of court filings.
- A security manager that handles authentication using SSL and HTTP cookies.
- A skeleton for a Query/Response interface.
- A skeleton for CMS interfaces that includes specifications for SOAP or Java RMI interfaces.
- A LegalXML validator that validates filing against any DTD derived from the Court Filing DTD. Administrators can customize validation using simple Xpath queries.
- An audit logger that records events to a log file.
- An ID Dispenser that generates Universally Unique Identifiers (UUIDs) and time stamps for each filing.
- A rudimentary Web interface for clerk review. If the clerk accepts the filing, it is sent to the CMS interface.
- A sample client that demonstrates how an EFSP can submit a filing using SOAP.

OpenEFM was developed using principles that are shared with the OXCI EFM architecture, such as:

- Support for XML standards, especially LegalXML Court Filing.
- Preference for open technologies and open source software.
- A free licensing model.

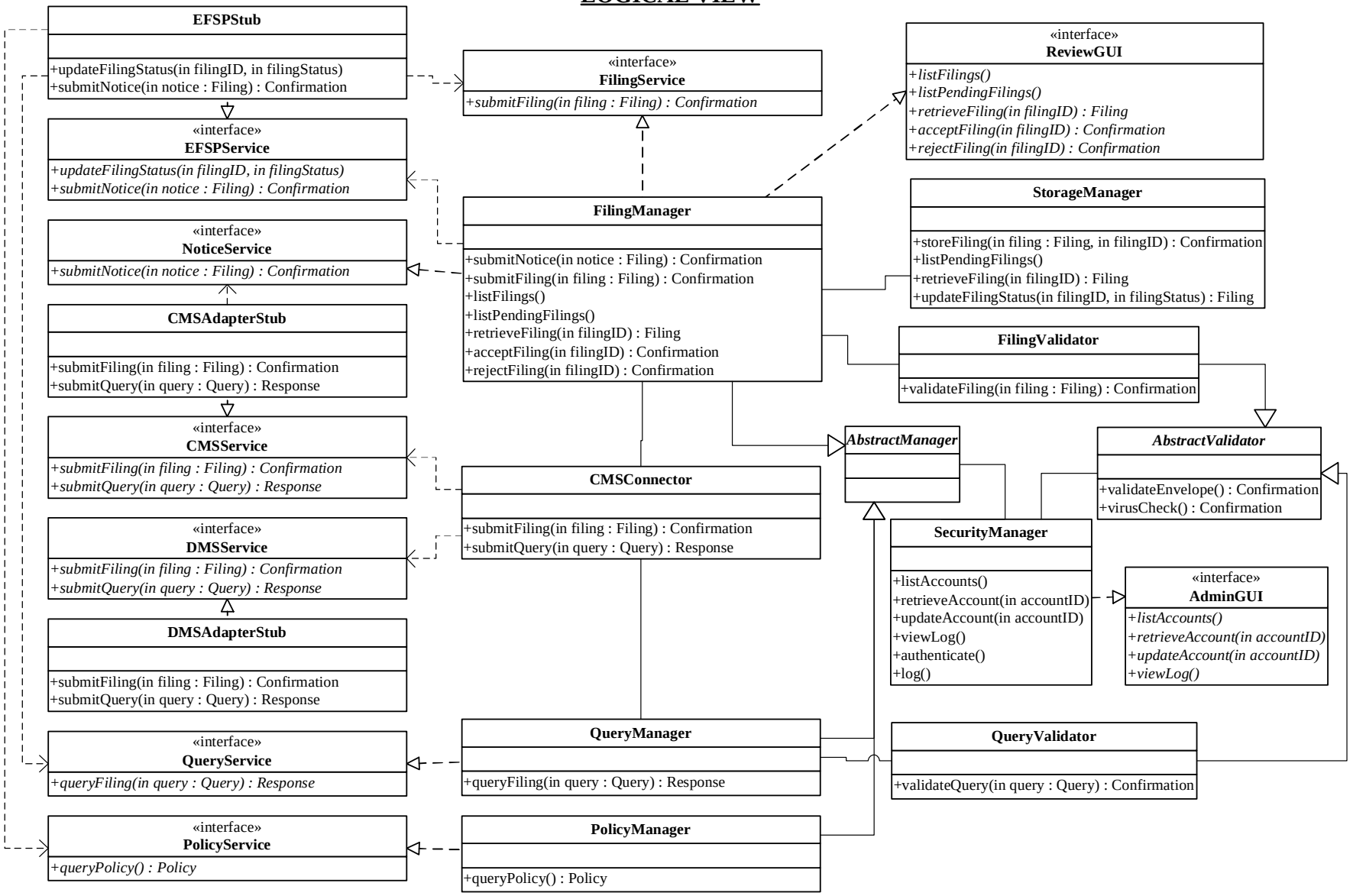
Although the OpenEFM does not yet support all the features of the OXCI EFM architecture, it provides a framework that should be easily adapted to support the requirements of the OXCI EFM.

E. EFM INTERFACES

EXHIBIT III presents an abstract software architecture for the EFM application component. The EFM application component includes the following interfaces and methods:

OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

LOGICAL VIEW



- **FilingService**
  - » submitFiling(filing: Filing): Confirmation
- **QueryService**
  - » queryFiling(query: Query): Response
- **PolicyService**
  - » queryPolicy(): Policy
- **ReviewGUI**
  - » listFilings()
  - » listPendingFilings()
  - » retrieveFiling(filingID): Filing
  - » acceptFiling(filingID): Confirmation
  - » rejectFiling(filingID): Confirmation
- **AdminGUI**
  - » listAccounts()
  - » retrieveAccount(accountID)
  - » updateAccount(accountID)
  - » viewLog()

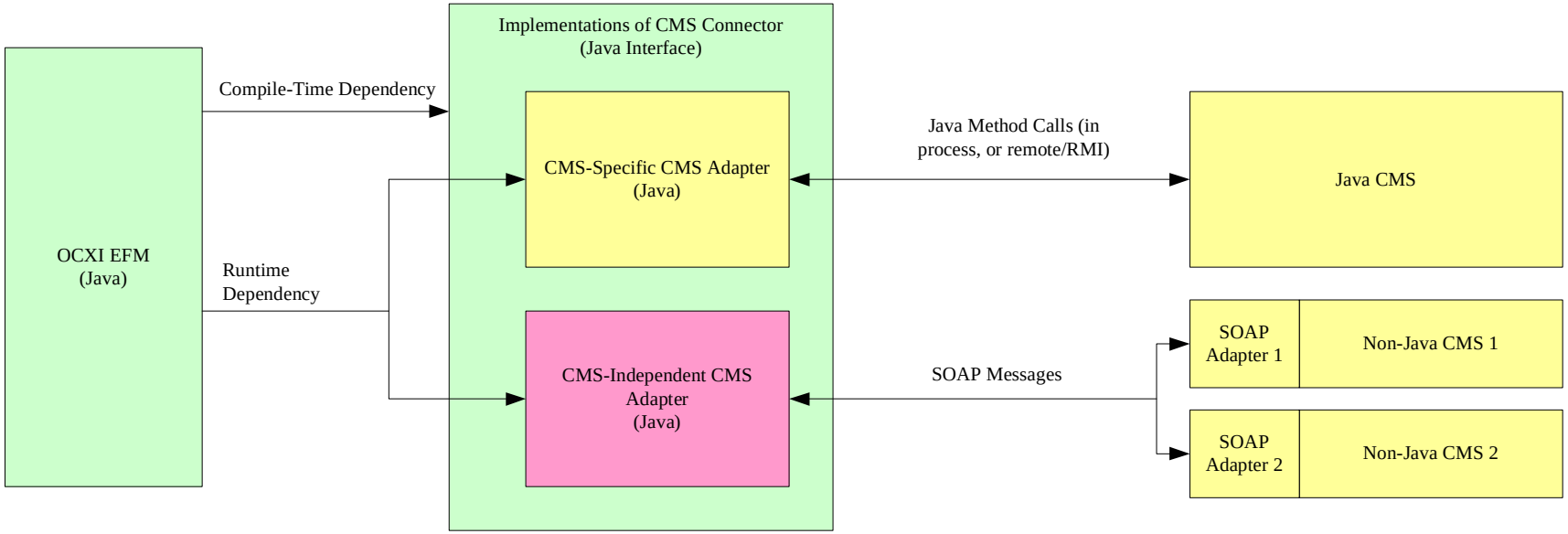
EXHIBIT IV presents a deployment view of the EFM to CMS/DMS interfaces. EXHIBIT III presents a class view and also includes the following interfaces and methods on the EFSP, CMS, and DMS:

- **EFSPService**
  - » updateFilingStatus(filingID, filingStatus)
  - » submitNotice(notice: Filing): Confirmation
- **CMSService**
  - » submitFiling(filing: Filing): Confirmation
  - » submitQuery(query: Query): Response

EXHIBIT IV

OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

EFM TO CMS/DMS INTERFACE DEPLOYMENT VIEW



Exists now in OpenEFM

Will exist in OXCI

Not in OXCI (built by each CMS vendor or court)

Opportunities for Standardization:

The efiling standard will presumably only include the SOAP messages between the CMS-independent adapter and all non-Java CMSs.

However, the CMS Connector Java interface should be viewed as a standardized component (at least de facto), as anyone writing a Java CMS that will plug in directly to OXCI will be dependent on that interface.

- **DMSService**
  - » submitFiling(filing : Filing): Confirmation
  - » submitQuery(query: Query): Response

F. EFM CLASSES

The EFM architecture includes the following data classes:

- **Filing**
- **Confirmation**
- **Query**
- **Response**
- **Policy**

The filing object migrates through a number of states according to the events in the filing process. EXHIBIT V presents a state diagram for the filing object that includes the following states in order of succession:

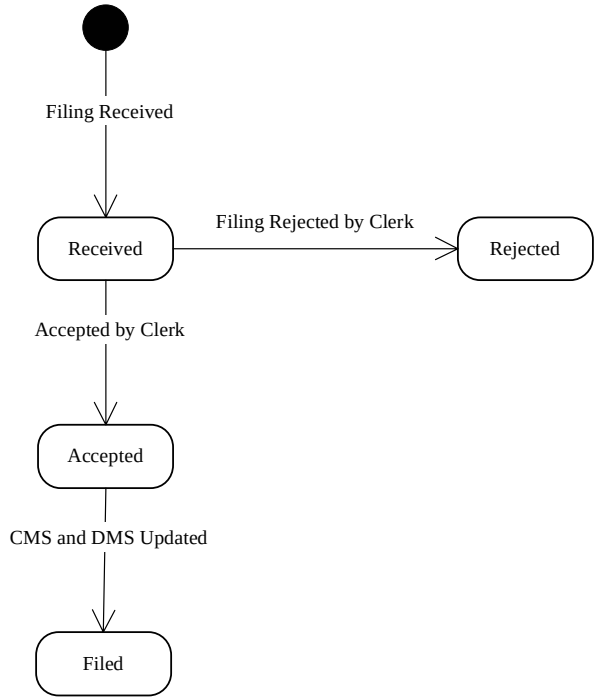
- *Received* denotes a filing object that has been submitted and is awaiting review by the court clerk.
- *Accepted* denotes a filing object that has been accepted by the clerk but has not yet been accepted by the CMS and DMS.
- *Filed* denotes a filing object that has been accepted by the CMS and DMS. This is a final state.
- *Rejected* denotes a filing object that has been rejected by the clerk. This is a final state.

The EFM architecture also includes the following system classes:

- **AbstractManager**, which represents an abstract transaction manager. This class encapsulates the functionality common to the **FilingManager**, **QueryManager**, and **PolicyManager** classes.

OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

FILING STATE DIAGRAM



- **FilingManager**, which manages the filing and notice processes and provides the functionality for clerk review. This class implements the **FilingService**, **NoticeService**, and **ReviewGUI** interfaces.
- **QueryManager**, which manages the query and response processes. This class extends the **AbstractManager** class and implements the **QueryService** interface.
- **PolicyManager**, which manages court policy requests. This class extends the **AbstractManager** class and implements the **PolicyService** interface.
- **SecurityManager**, which manages authentication and logging. This class implements the **AdminGUI** interface.
- **StorageManager**, which manages data storage.
- **AbstractValidator**, which represents an abstract schema validator. This class encapsulates the functionality common to the **FilingValidator** and **QueryValidator** classes. It includes methods for validating the envelope and validating the attached documents using external virus checking software.
- **FilingValidator**, which validates court filings and notices. This class extends the **AbstractValidator** class.
- **QueryValidator**, which validates court queries and response. This class extends the **AbstractValidator** class.
- **CMSConnector**, which connects the **FilingManager** and **QueryManager** objects the **CMSService** and **DMSService** interfaces.

EXHIBIT III also includes the following classes that represent the realizations of interfaces on the EFSP, CMS, and DMS:

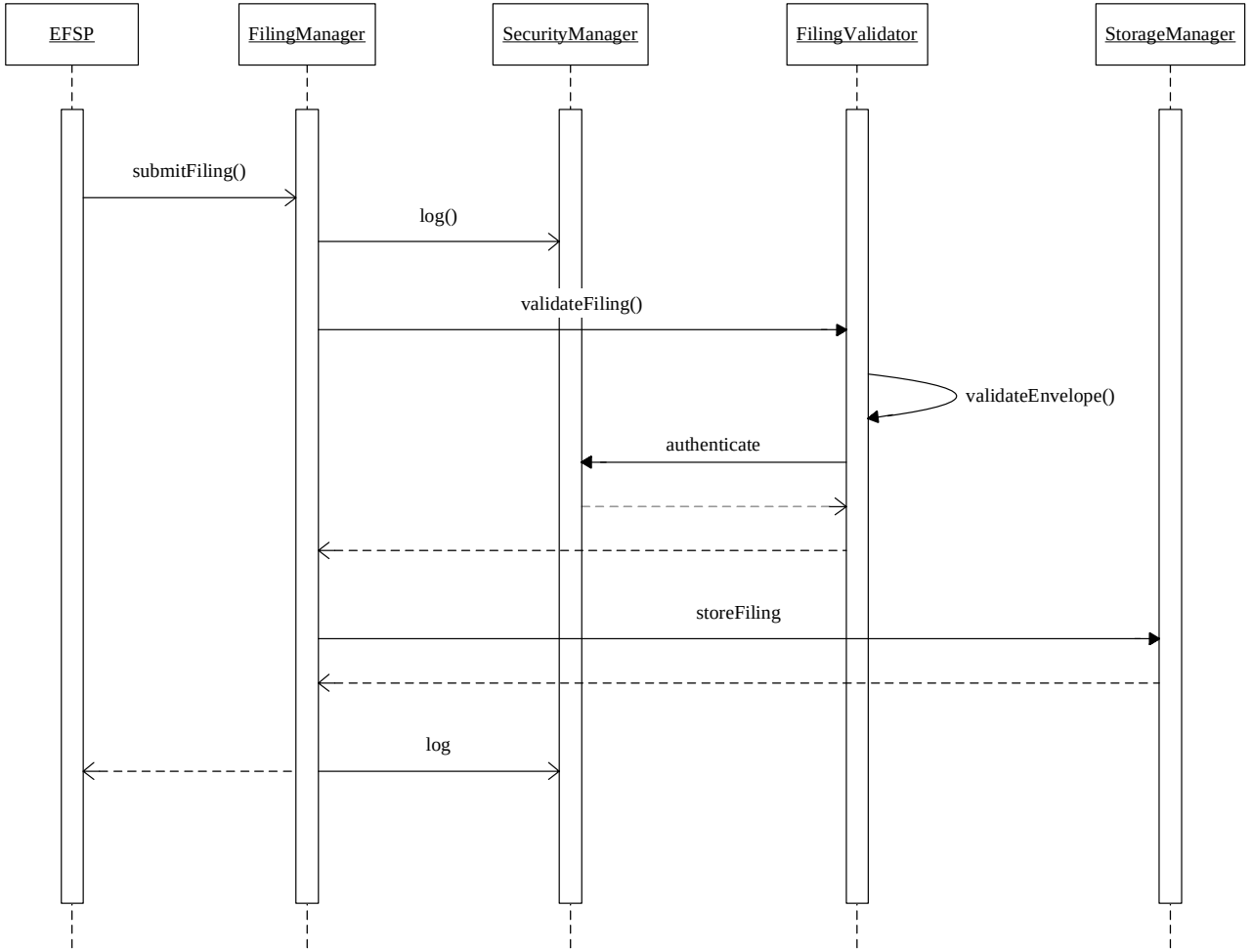
- **EFSPStub**, which implements the EFSPService interface.
- **CMSAdapterStub**, which implements the CMSService interface.
- **DMSAdapterStub**, which implements the DMSService interface.

G. USE CASES

EXHIBITS VI-1 through VI-6 present sequence diagrams for each of the six use cases defined in Section II. The use cases include:

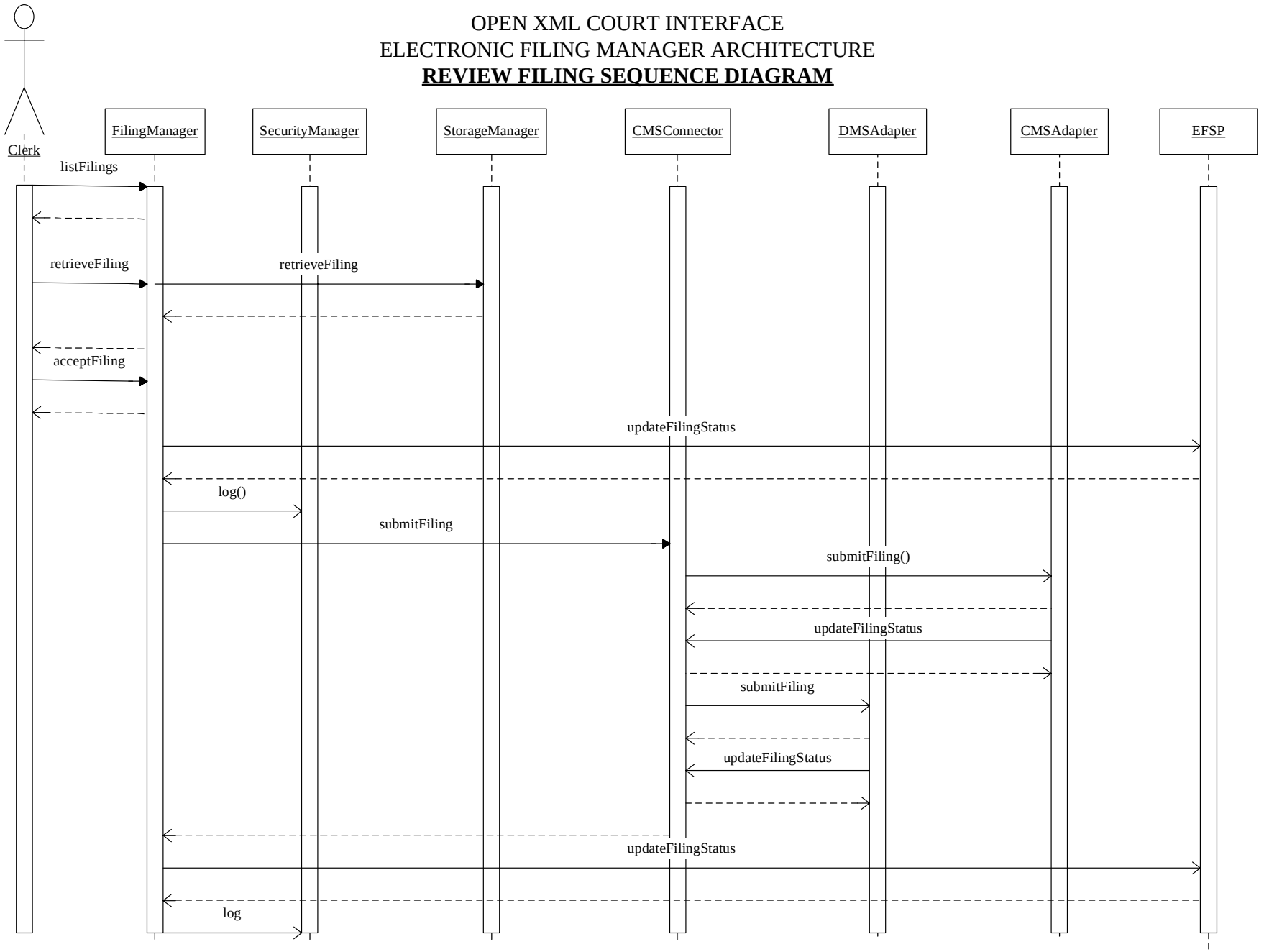
OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

SUBMIT FILING SEQUENCE DIAGRAM



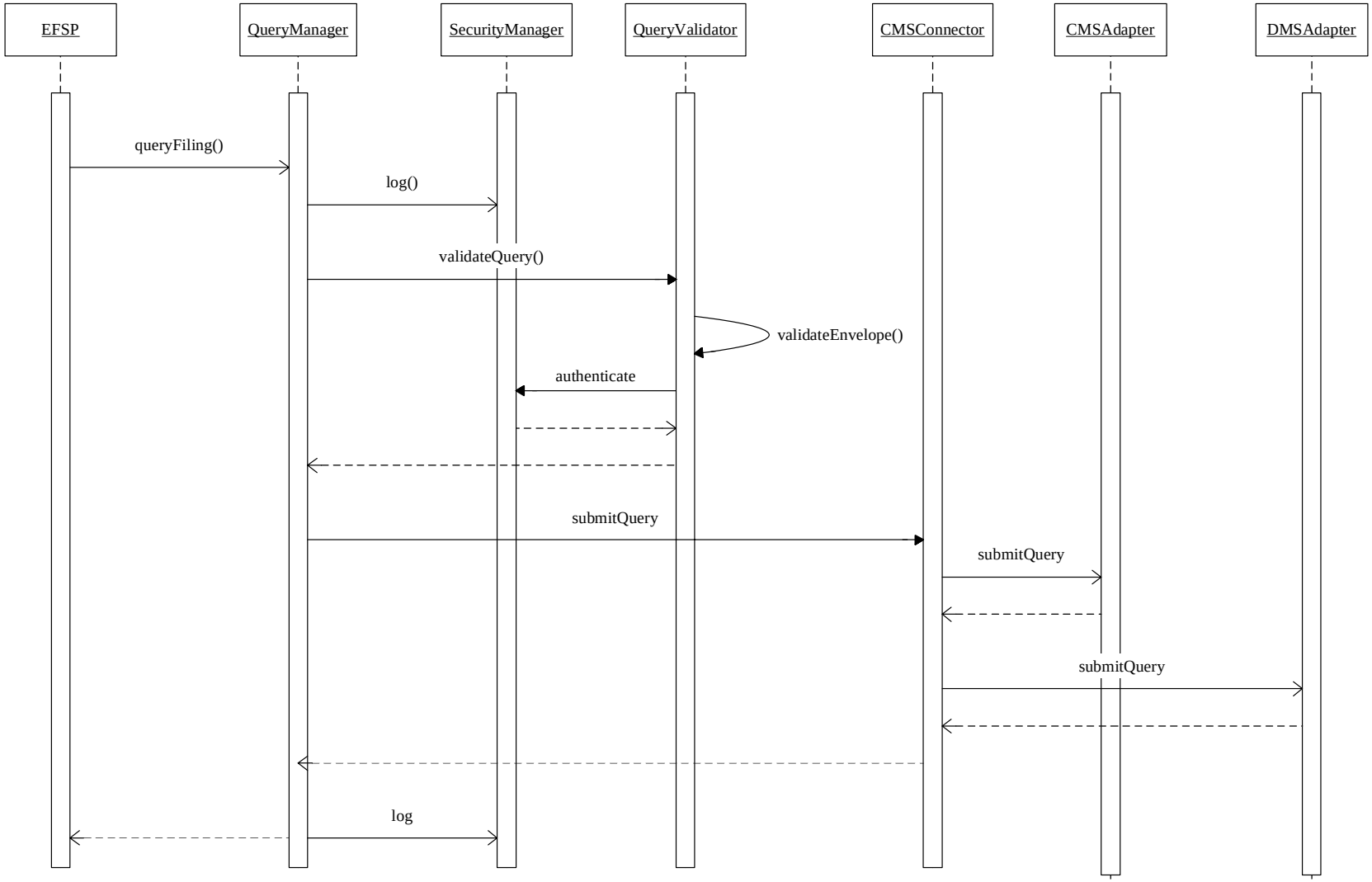
OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

REVIEW FILING SEQUENCE DIAGRAM



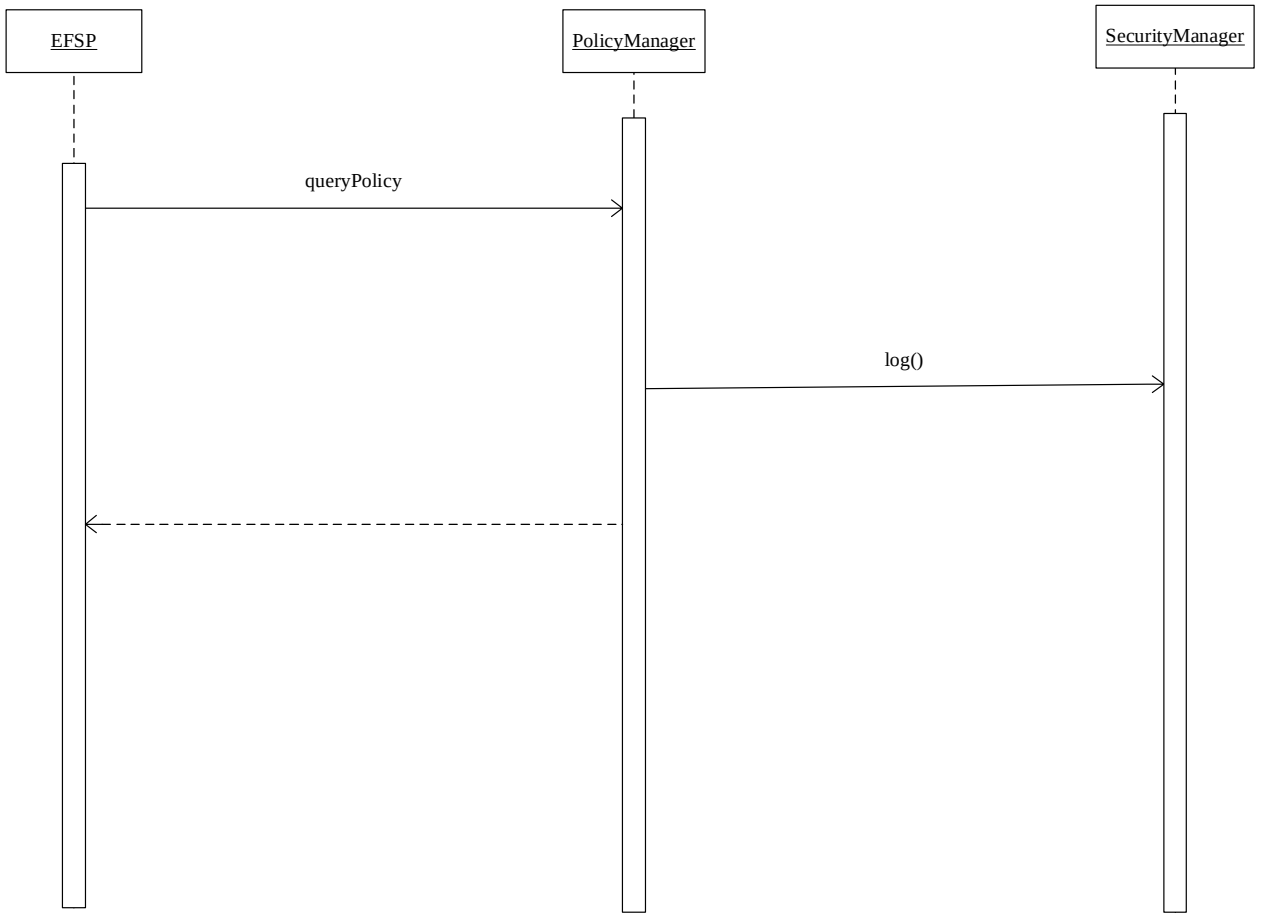
OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

QUERY FILING SEQUENCE DIAGRAM



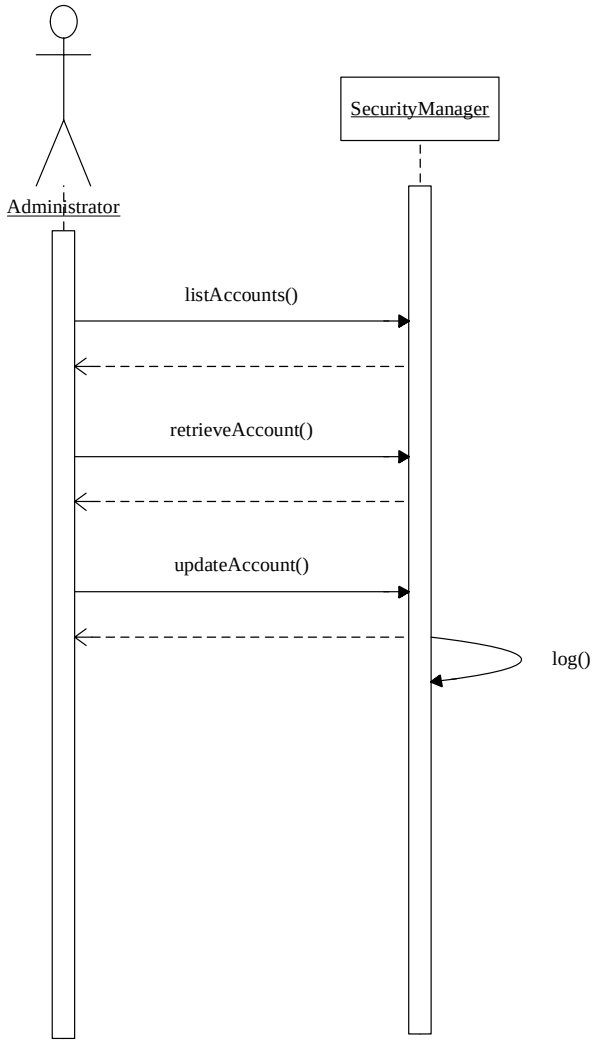
OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

QUERY POLICY SEQUENCE DIAGRAM



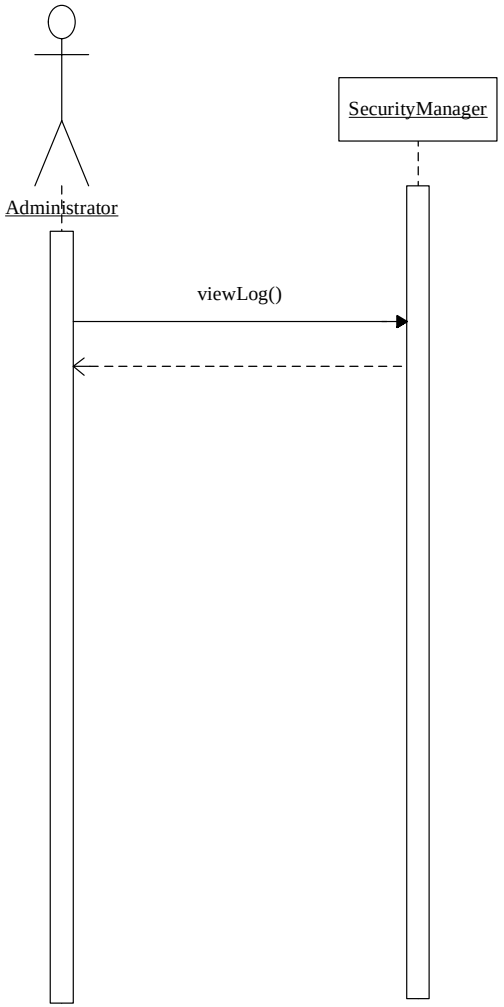
OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

MANAGE ACCOUNTS SEQUENCE DIAGRAM



OPEN XML COURT INTERFACE
ELECTRONIC FILING MANAGER ARCHITECTURE

VIEW LOGS SEQUENCE DIAGRAM



- *Submit Filing* in which a filer submits a filing with the court through the EFM.
- *Review Filing* in which a clerk interfaces with the EFM to review and accept or reject filings submitted to the court.
- *Query Filing* in which a user queries the court CMS through the EFM.
- *Query Policy* in which a user obtains the policies specific to a court from the EFM.
- *Manage Accounts* in which an administrator interfaces with the EFM to create, modify, or remove accounts and privileges on the EFM.
- *View Logs* in which an administrator interfaces with the EFM to review the system and EFM application logs.

Each diagram illustrates the sequence of exchanges between classes in the use case. Each exchange is illustrated as a method between two classes.

* * * * * *

The architecture defined in this document provides a good framework for the implementation of an EFM application based on open standards that is scalable and flexible enough to support a wide range of court models and systems.

APPENDIX A
GLOSSARY

GLOSSARY

AAMVA	American Association of Motor Vehicle Administrators
ACL	access controls list
ADO	ActiveX Database Objects
API	Application Program Interface
CF XML	Court Filing XML
CMP	Container Managed Persistence
CMS	Case Management System
CMS/API	Case Management System/Application Program Interface
CPA	Collaboration Protocol Agreement
CPP	Collaboration Protocol Profile
CPP/A	Collaboration Protocol Profile/Agreement
COSCA	Consortium of State Court Administrators
DIME	Direct Internet Message Encapsulation
DMS	Document Management System
DOM	Document Object Model
DTD	Document Type Definition
ebXML	Electronic Business eXtensible Markup Language
ebMS	ebXML Messaging Service
EDI	Electronic Data Interchange
EFM	Electronic Filing Manager
EFP	Electronic Filing Provider
EFSP	Electronic Filing Service Provider
EJB	Enterprise JavaBeans
FIPS	Federal Information Processing Standards
FTP	File Transfer Protocol
GTRI	Georgia Technology Research Institute
GUI	graphical user interface
GXA	Global XML Web Services Architecture
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ID	identification
IE	Microsoft Internet Explorer
IETF	Internet Engineering Task Force
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
IPR	Intellectual Property Rights
IPX	Internetwork Packet Exchange
J2EE	Java 2 Enterprise Edition
JAXM	Java API for XML Messaging
JAXP	Java API for XML Parsing
JDBC	Java Database Connectivity
JDO	Java Database Objects
JSP	Java Server Pages
LAMP	Linux, Apache, MySQL, and PERL
MIME	Multipurpose Internet Mail Extensions
MSH	Message Service Handler
NACM	National Association of Court Managers
NCIC	National Crime Information Center
NCSC	National Center for State Courts
OASIS	Organization for the Advancement of Structured Information Standards
ODBC	Open Database Connectivity
ODBMS	Object database management system
OSI	Open System Interconnection
OXCI	Open XML Court Interface
PDF	Portable Document Format
QoS	Quality of Service
RDBMS	Relational database management system
RDF	Resource Description Framework
RFC	Request for Comments

RFP	Request for Proposal
RISS	Regional Information Sharing System
RMI	Remote Method Invocation
RPC	remote procedure call
SAML	Security Assertion Markup Language
SAX	Simple API for XML
S/MIME	Secure/Multipurpose Internet Mail Extensions
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TC	Technical Committee
TCP/IP	Transmission Control Protocol/Internet Protocol
TIFF	Tagged Image File Format
TRP	transport, routing and packaging
UDDI	Universal Description, Discovery and Integration
UDP/IP	User Datagram Protocol/Internet Protocol
UML	Unified Modeling Language
UN/CEFACT	United Nations Centre for Trade Facilitation and Electronic Business
UUID	Universally Unique Identifiers
W3C	World Wide Web Consortium
WS	Web Services
WSDL	Web Service Description Language
WSEL	Web Service End-point Language
XML	eXtensible Markup Language

APPENDIX B
REFERENCES

REFERENCES

In the creation of this document, we consulted a large number of standards specifications and other documents related to electronic court filing. These documents included the following:

A. OASIS SPECIFICATIONS

1. M. Halverson, "Electronic Court Filing 1.1 Proposed Standard," OASIS LegalXML Member Section Electronic Court Filing Technical Committee, *http://www.oasis-open.org/committees/legalxml-courtfiling/documents/filing1.1/22072002cf1-1.pdf*, July 22, 2002.

2. S. Durham, "Electronic Court Filing Query and Response Standard (draft)," OASIS LegalXML Member Section Electronic Court Filing Technical Committee, October 22, 2002.

3. D. Bergeron, "Court Policy Interface Requirements," OASIS LegalXML Member Section Electronic Court Filing Technical Committee, *http://www.oasis-open.org/committees/ legalxml-courtfiling/documents/court_policy/court_policy_20021018.pdf*, October 14, 2002.

4. OASIS, "Definition of Court Filing Blue," OASIS LegalXML Member Section Electronic Court Filing Technical Committee, December 12, 2004.

5. M. Yuan, S. Spohn, "EFM-CMS Interface Requirements version 6," LegalXML, Inc., June 4, 2001.

6. OASIS, "Message Service Specification version 2.0 (Approved Standard)," OASIS ebXML Messaging Services Technical Committee, *http://www.oasis-open.org/committees/ebxml-msg/ documents/ebMS_v2_0.pdf*, April 1, 2002.

7. OASIS, "Collaboration Protocol Profile and Agreement Specification version 2.0 (Approved OASIS Standard)," OASIS ebXML Collocation Protocol Profile and Agreement Technical Committee, *http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf*, September 23, 2002.

8. OASIS, "OASIS/ebXML Registry Information Model v2.0 (Approved OASIS Standard)," OASIS/ebXML Registry Technical Committee, *http://www.oasis-open.org/committees/ regrep documents/2.0/specs/ebrim.pdf*, April 2002.

9. OASIS, "OASIS/ebXML Registry Services Specification v2.0 (Approved OASIS Standard)," OASIS/ebXML Registry Technical Committee, *http://www.oasis-open.org/committees/regrep/ documents/2.0/specs/ebrs.pdf*, April 2002.

10. D. Ehnebuske, B. McKee, D. Rogers (eds.), "UDDI Version 2.04 API Specification (OASIS Committee Specification)," OASIS UDDI Specifications Technical Committee, *http:// uddi.org pubs/ProgrammersAPI-V2.04-Published-20020719.pdf*, July 19, 2002.

11. P. H.-Baker, C. Kaler, R. Monzillo, A. Nadalin, "Web Services Security Core Specification (Working Draft 09)," *http://www.oasis-open.org/committees/wss/documents/WSS-Core-09-0126-merged.pdf*, January 26, 2003.

12. P. Hallem-Baker, E. Maller, "Assertions and Protocol for the OASIS SAML (Approved Standard)," OASIS Security Services Technical Committee, *http://www.oasis-open.org/ committees/security/docs/cs-sstc-core-01.pdf*, May 31, 2002.

13. G. Beaver, "Verification of ebXML Messaging for use with government," OASIS e-Government Technical Committee, November 26, 2003.

B. W3C SPECIFICATIONS

1. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler. W3C, "Extensible Markup Language (XML) 1.0 (Second Edition)," W3C Recommendation, *http://www.w3.org/TR/2000/REC-xml-20001006*, October 2000.

2. D. Fallside (ed), "XML Schema Part 0: Primer," W3C Recommendation, *http://www.w3.org/ TR/2001/REC-xmlschema-0-20010502/primer.html*, 2 May 2001.

3. D. Eastlake, J. Reagle, D. Solo, "XML-Signature Syntax and Processing," W3C Recommendation, *http://www.w3.org/TR/2002/REC-xmldsig-core-20020212*/, February 2002.

4. D. Eastlake, J. Reagle, "XML Encryption Syntax and Processing," W3C Candidate Recommendation, *http://www.w3.org/TR/2002/CR-xmlenc-core-20020802/*, December 2002.

5. D. Box et al, "Simple Object Access Protocol (SOAP) 1.1," W3C Note, *http://www.w3.org/ TR/2000/NOTE-SOAP-20000508*, May 8, 2000.

6. M. Gudgin, M. Hadley, J. J. Moreau, H. Frystyk Nielsen, "SOAP 1.2 Part 1: Messaging Framework," W3C Candidate Recommendation, *http://www.w3.org/TR/2002/CR-soap12-part1-20021219*, December 20, 2002.

7. O. Lassila, R. Swick (eds.), "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, *http://www.w3.org/TR/1999/ REC-rdf-syntax-19990222*, February 22, 1999.

8. J. Barton, S. Thatte, H. F. Nielsen, "SOAP Messages with Attachments," W3C Note, *http:// www.w3.org/TR/2000/NOTE-SOAP-attachments-20001211*, December 11, 2000.

9. F. Nielsen, H. Ruellan, "SOAP 1.2 Attachment Feature," W3C Working Draft, *http://www.w3.org/TR/2002/WD-soap12-af-20020924/*, September 24, 2002.

10. E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Description Language (WSDL) 1.1," W3C Note, *http://www.w3.org/TR/2001/NOTE-wsdl-20010315*, 15 March 2002.

11. R. Chinnici, M. Gudgin, J. J. Moreau, S. Weerawarana, "Web Services Description Language (WSDL) 1.2," W3C Working Draft, *http://www.w3.org/TR/2003/WD-wsdl12-20030124*, January 24, 2003.

C. JAVA SPECIFICATIONS

1. B. Shannon, "Java 2 Platform Enterprise Edition Specification v1.4," Sun Microsystems, *http://java.sun/com/j2ee*, November 24, 2003.

2. D. Coward, "Java Servlet specification version 2.3," Sun Microsystems, *http://java.sun.com/products/servlet*, August 13, 2001.

3. E. Pelegrí-Llopart, JavaServer Pages Specification version 1.2," Sun Microsystems, *http://java.sun.com/products/jsp*, August 21, 2001.

4. J. Ellis, L. Ho, M. Fisher, "JDBC 3.0 Specification," Sun Microsystems, *http://java.sun.com/products/jdbc*, October, 2001.

5. N. Kassem, A. Vijendran, R. Mordani, "Java API for XML Messaging (JAXM) Specification v1.1," Sun Microsystems, *http://java.sun.com/xml/jaxm*, June 2002.

6. R. Mordani, S. Boag, "Java API for XML Processing (JAXP) version 1.2," Sun Micrososystems, *http://java.sun.com/xml/jaxp*, September 6, 2002.

D. IETF RFCS

1. R. Fielding, J. Gettys, J. C. Mogul, H. Frystyk, T. Berners-Lee, "RFC 2616: Hypertext Transfer Protocol – HTTP/1.1," IETF, *http://www.ietf.org/rfc/rfc2616.txt*, January 1997.

2. Palme, A. Hopman, N. Shelness, "RFC2557: MIME Encapsulation of Aggregate Documents, such as HTML (MHTML)," *http://www.ietf.org/rfc/rfc2557.txt*, March 1999.

3. H. F. Nielsen, H, Sanders, R. Butek, S. Nash, "Internet Draft: Direct Internet Message Encapsulation (DIME)," IETF, *http://www.ietf.org/internet-drafts/draft-nielsen-dime-02.txt*, June 17, 2002.

E. OTHER DOCUMENTS

1. T. Bousquin, "Initial Draft of RFP Elements," OXCI Advisory Committee, September 16, 2002.

1. M. Kindl, J. Wandelt, W. Roberts, C. Medlin, "Justice XML Data Dictionary (JXDD) v3.0 – Status, Design and Development," Information Technology and Telecommunications Laboratory, Georgia Technology Research Institute, December 16, 2002.

2. "Global Justice XML Data Model v3.0.0.3," Department of Justice, Office of Justice Programs, *http://justicexml.gtri.gatech.edu*, November 28, 2003.

3. W. Caelli, D. Longley, M. Shain, *Information Security Handbook*, Macmillan, London, 1991.

4. Counterclaim, "OpenEFM Whitepaper," *http://www.counterclaim.com/openefm.htm*, June 13, 2002.

5. McMillan, T Carlson, "inCounter: An Open Source Electronic Filing Demonstration Project," NCSC, *http://www.court-tech.org/inCounter/inCounter-whitepaper-v5.pdf*, September 2002.

6. W. T. Vincent, "Georgia Courts Automation Commission Court Filing Interoperability Pilot Lessons Learned Document," *http://e-ct-file.gsu.edu/*, Georgia State University, December 4, 2001.

7. W. T. Vincent, "Georgia Courts Automation Commission Court Filing Interoperability Pilot Lessons Learned Document II," *http://e-ct-file.gsu.edu/*, Georgia State University, May 20, 2002.

8. D. Powell, "Architectural Models, Business Decisions, and Interoperability Issues," Tybera Development Group, *http://www.tybera.com*, November 7, 2002.

9. F. D. Kasparek, J. Greacen, T. Bousquin, "Standards for Electronic Filing Processes (Technical and Business Approaches)," National Center for State Courts, *http://www.ncsconline.org/D_Tech/Standards/Documents/pdfdocs/Recommended_%20Process_%20standards_%2011_27_02.pdf*, November 7, 2002.

10. "Tomcat 4 Servlet/JSP Container," Apache Software Foundation, *http://jakarta.apache.org/tomcat/tomcat-4.1-doc/*, 2002.

11. "Jboss Application Server version 3.2.3," Jboss Group, *http://jboss.org/*, 2003.

12. "Websphere Application Server," IBM, *http://www.ibm.com/software/info1/Websphere/*, 2003.

13. "freebXML Hermes Message Service Handler," Center for E-Commerce Infrastructure Development, Department of Computer Science and Information Systems, University of Hong Kong, *http://www.freebxml.org/msh.htm*, 2003.

14. "GoXML Messaging," XML Global, *http://www.xmlglobal.com/prod/messageservice/*, 2003.

15. "Xerces2 Java Parser," Apache Software Foundation, *http ://xml.apache.org/xerces2-j/*, 2003.

16. "MySQL Database Server," MySQL AB, *http://www.mysql.com/products/mysql/*, 2003.

17. "DB2 Universal Database," IBM, *http://www-306.ibm.com/software/data/db2*, 2003.

18. Sierra Systems, "Prototype for XML-based Efiling of Criminal Complaints," October 2003.

19. Georgia Technology Research Institute, "Rules for Subset Schemas," *http://justicexml.gtri.gatech.edu/rules_for_schema_subsets.html*, February 2004.

APPENDIX C
REVISION HISTORY

REVISION HISTORY

Version	Date	Revised By	Description
0.9	1/31/03	Mr. James Cabral	This was the initial version of this architecture document.
1.0	2/12/03	Mr. Cabral	This version included several revisions by the OXCI Steering Committee.
2.0	1/6/04	Mr. Cabral	This version updated the architecture to support the LegalXML Court Filing Blue requirements, including support for the GJXDM 3.0 release.
2.1	4/26/04	Mr. Cabral	This version included a new requirement for antivirus checking and added detail to the CMS/DMS adapters, payment interface, and security. This version also added EXHIBIT IV.
3.0	10/22/04	Mr. Cabral	This version reflects the architecture "as built" in the EFM. It includes asynchronous messaging requirements for the EFSP, CMS, and DMS interfaces; the removal of the "Submit Notice" use case; and the removal of the "two-way" functional requirement.