DISCUSSION DRAFT
3-24-04

OPEN XML COURT INTERFACE

Electronic Filing Manager Test Plan

March 24, 2004

counterclaim

MTG Management Consultants

TABLE OF CONTENTS

I. INTRODUCTION

I. INTRODUCTION

This document is intended to describe the software quality assurance testing to be used during the development phase of the Open XML Court Interface (OXCI) project. This document describes what will be tested and the testing process, and presents a schedule of tests and an explanation of the test reports produced.

A. RELATED DOCUMENT

References to the following document may be found within this design document:

Title	Date	Source
OXCI EFM Design Document	January 2004	counterclaim, Inc., MTG Management Consultants, L.L.C.

B. APPROACH

The tests performed will be functional unit tests of key software components. Each module of the Electronic Filing Manager (EFM) identified in the design document will have an associated set of unit tests. Currently, OpenEFM maintains an Apache Ant build environment. The suite of tests provided will be accessible as an Ant build task. The unit tests will be implemented using JUnit, an open source, functional unit testing library. Both Ant and JUnit can execute under a J2EE 1.3 environment.

C. TEST PASS/FAIL CRITERIA

JUnit provides test results of three possible types, success, failure, or error, as follows:

- Success is a result signifying the entire test executes and a successful outcome occurs.
- Failure results when the entire test executes and the test criteria are not correctly met.
- Error results occur when the entire test could not be executed due to a problem, such as poor server configuration.

Each functional unit test will result in one and only one of the outcomes listed above.

D. TESTING REQUIREMENTS

The EFM will run in a J2EE 1.4-compliant environment. Any set of hardware and operating system capable of providing this environment should be acceptable for testing.

E. TEST DELIVERABLES

JUnit can produce test results in the form of HTML documents. These HTML pages display information about the execution of tests. The success, failure, or error outcome of any test can be viewed in detail or summarized by groups.

These statistics will be used throughout the test phases to provide an accurate assessment of the compliance of the software. At the end of each test phase, a report will be provided. This report will detail the level of compliance met, and explain any failing tests.

The list of tests and packages provided in the Test Items section is likely to change during the development cycles. The first iteration test report will consist of a list of all tests and a description of their outcomes. The second iteration test report will include a list of new tests and packages that may have been added. The final iteration test report will be a definitive list of tests provided for the EFM.

F. SCHEDULE

The functional unit tests will be developed concurrently with the application logic of the software. The end of each development iteration will be followed by a brief testing period. During this period the entire test suite will be run on the EFM. At the end of this period, a report detailing the outcome of the tests will be provided. The expected duration of the testing period is 2 to 3 days.

II. TEST ITEMS

II. TEST ITEMS

Functional unit tests will be provided for every module of the EFM. This will include basic testing, such as ensuring that each data object correctly sets and returns all internal data elements. This will also include advanced tests, such as verifying that each action executes correctly. Actions include the code that is executed during every instance that a person interacts with the user interface. Data objects that facilitate the filing of a document, or a query and response traveling through the system, will also be tested.

The EFM is implemented in a Java class package hierarchy. Each package represents code that serves a similar function and is used in a similar way. This modular and object-oriented approach allows for certain aspects of the application logic to be re-implemented in a convenient manner. The packages included in the EFM are listed and described below. Each package will have a set of JUnit tasks associated with it to ensure its proper function.

A. ADMIN PACKAGE

This package implements the user interface of the system. It is divided into three categories, virtual objects, form objects, and actions as follows:

- The virtual objects store data representing objects such as filings, documents, and cases. The virtual objects are lightweight Java classes, and they provide only access functionality to the data they store. No application logic is contained in the virtual objects. The virtual objects are used to store session-specific information.
- The form objects represent HTML forms in the graphical user interface (GUI). Methods that validate data gathered by an HTML form may be stored in the form objects. The form and virtual objects are both used by actions.
- Actions represent the application logic of the user interface. Actions are generally used once the user clicks something in order to determine the next step the application should take.

B. BILLING PACKAGE

The billing package handles interfacing with a payment transaction service. A billing module implementation that uses the VeriSign Payflow Pro service is provided.

C. CLIENT PACKAGE

The client package provides Java classes that can be used to submit court filings to the EFM. Implementations providing client access via HTTP, SOAP, and ebXML are provided.

D. CONFIG PACKAGE

The config package contains classes that handle the configuration of a running instance of the EFM. These classes handle the parsing of the XML configuration files and the creation of the described modules.

E. CONNECTOR PACKAGE

The connector package handles implementations of the specific CmsConnectors. Generic classes are provided and a functional Java Remote Method Invocation (RMI) connector has been implemented.

F. CORE PACKAGE

The core package provides implementations of the essential objects used by the EFM. Most notable are the server, controller, and component object. The server oversees the general running of the application, handling tasks such as starting and stopping the application. The controller contains all of the individual modules being used by the system. It also implements a few critical system tasks. The controller acts as the kernel of the application. The component object is used by modules which are loaded into the controller. This object handles basic controller component interaction.

G. COURT POLICY PACKAGE

This package contains classes that represent a court policy XML file. The controller class uses the court policy object when it must access court information to make an application control flow decision. Classes are also provided which host the XML in the Web environment. This package is configurable from the XML configuration file.

H. LOGGER PACKAGE

The logger package handles classes that manage logging EFM events to disk. A simple file system logger is provided.

I. VALIDATOR PACKAGE

This package contains classes that handle the validation of XML files. Implementations are provided for both Document Type Definition (DTD) and Schema-based validation.

J. MODEL PACKAGE

The model package is the largest package in the EFM. It contains classes that represent all the types of data used by the system. This includes things such as courts, Electronic Filing Service Providers (EFSPs), filings, documents, and IDs Application logic is not generally stored in these objects.

K. NOTIFICATION PACKAGE

This package provides a simple class for sending e-mail notifications. The EFM controller sends e-mail notifications to parties involved with filings when specific system events occur, such as the acceptance or rejection of a filing.

L. SECURITY PACKAGE

The security package provides classes used to ensure the authentication of an EFSP wishing to file documents with the EFM. Classes exist that define a manager and that represent a violation. A simple username and password manager implementation is also provided.

M. TRANSCEIVER PACKAGE

The transceiver package contains classes that are used to accept XML court filings and queries over various protocols. Implementations are provided which work with HTTP, SOAP, and ebXML.

N. UTIL PACKAGE

The util package contains classes used by various modules of the system. These classes handle tasks such as reading a file on the system, generating XPath queries based on arbitrary XML, and manipulating date and time objects. Generally speaking, these classes act as libraries that contain helper functions that other developers may find useful at a later point in time.

APPENDIX A
REVISION HISTORY

REVISION HISTORY

Version	Date	Revised By	Description
0.1	2-18-04	Mr. Jim Beard counterclaim, Inc.	Initial creation of the document as an outline.
0.2	3-24-04	Mr. Beard counterclaim, Inc.	The first complete version of the document.