# POI 2.0 Vision Document

**by Andrew C. Oliver, Marcus W. Johnson, Glen Stampoultzis, Nicola Ken Barozzi**

### 1. Preface

This is the POI 2.0 cycle vision document. Although the vision has not changed and this document is certainly not out of date and the vision has not changed, the structure of the project has changed a bit. We're not going to change the vision document to reflect this (however proper that may be) because it would only involve deletion. There is no purpose in providing less information provided we give clarification.

This document was created before the POI components for Apache Cocoon were accepted into the Cocoon project itself. It was also written before POI was accepted into Jakarta. So while the vision hasn't changed some of the components are actually now part of other projects. We'll still be working on them on the same timeline roughly (minus the overhead of coordination with other groups), but they are no longer technically part of the POI project itself.

### 2. 1. Introduction

## 2.1. 1.1 Purpose of this document

The purpose of this document is to collect, analyze and define high-level requirements, user needs, and features of the second release of the POI project software. The POI project currently consists of the following components: the HSSF Serializer, the HSSF library and the POIFS library.

- The HSSF Serializer is a set of Java classes whose main class supports the Serializer interface from the Cocoon 2 project and outputs the serialized data in a format compatible with the spreadsheet program Microsoft Excel '97.
- The HSSF library is a set of classes for reading and writing Microsoft Excel 97 file format using pure Java.
- The POIFS library is a set of classes for reading and writing Microsoft's OLE 2 Compound Document format using pure Java.

By the completion of this release cycle the POI project will also include the HSSF Generator

and the HWPF library.

- The HSSF Generator will be responsible for using HSSF to read in the XLS (Excel 97) file format and create SAX events. The HSSF Generator will support the applicable interfaces specified by the Apache Cocoon 2 project.
- The HWPF library will provide a set of high level interfaces for reading and writing Microsoft Word 97 file format using pure Java.

## 2.2. 1.2 Project Overview

The first release of the POI project was an astounding success. This release seeks to build on that success by:

- Refactoring POIFS into imput and output classes as well as an event-driven API for reading.
- Refactor HSSF for greater performance as well as an event-driven API for reading
- Extend HSSF by adding the ability to read and write formulas.
- Extend HSSF by adding the ability to read and write user-defined styles.
- Create a Cocoon 2 Generator for HSSF using the same tags as the HSSF Serializer.
- Create a new library (HWPF) for reading and writing Microsoft Word DOC format.
- Refactor the HSSFSerializer into a separate extensible POIFSSerializer and HSSFSerializer
- Providing the create excel charts. (write only)

**3. 2. User Description**

## 3.1. 2.1 User/Market Demographics

There are a number of enthusiastic users of XML, UNIX and Java technology. Furthermore, the Microsoft solution for outputting Office Document formats often involves actually manipulating the software as an OLE Server. This method provides extremely low performance, extremely high overhead and is only capable of handing one document at a time.

1. Our intended audience for the HSSF Serializer portion of this project are developers writing reports or data extracts in XML format.
2. Our intended audience for the HSSF library portion of this project is ourselves as we are developing the HSSF serializer and anyone who needs to read and write Excel spreadsheets in a non-XML Java environment, or who has specific needs not addressed by the Serializer
3. Our intended audience for the POIFS library is ourselves as we are developing the HSSF and HWPF libraries and anyone wishing to provide other libraries for reading/writing other file formats utilizing the OLE 2 Compound Document Format in Java.

4. Our intended audience for the HSSF generator are developers who need to export Excel spreadsheets to XML in a non-proprietary environment.
5. Our intended audience for the HWPF library is ourselves, as we will be developing a HWPF Serializer in a later release, and anyone wishing to add .DOC file processing and creation to their projects.

## 3.2. 2.2. User environment

The users of this software shall be developers in a Java environment on any operating system, or power users who are capable of XML document generation/deployment.

## 3.3. 2.3. Key User Needs

The HSSF library currently requires a full object representation to be created before reading values. This results in very high memory utilization. We need to reduce this substantially for reading. It would be preferable to do this for writing, but it may not be possible due to the constraints imposed by the file format itself. Memory utilization during read is our top user complaint.

The POIFS library currently requires a full object representation to be created before reading values. This results in very high memory utilization. We need to reduce this substantially for reading.

The HSSF library currently ignores formula cells and identifies them as "UnknownRecord" at the lower level of the API. We must provide a way to read and write formulas. This is now the top requested feature.

The HSSF library currently does not support charts. This is a key requirement of some users who wish to use HSSF in a reporting engine.

The HSSF Serializer currently does not provide serialization for cell styling. User's will want stylish spreadsheets to result from their XML.

There is currently no way to generate the XML from an XLS that is consistent with the format used by the HSSF Serializer.

There should be a way to read and write the DOC file format using pure Java.

### 4. 3. Project Overview

## 4.1. 3.1. Project Perspective

The produced code shall be licensed by the Apache License as used by the Cocoon 2 project

(APL 1.1) and maintained on at http://poi.sourceforge.net and http://sourcefoge.net/projects/poi. It is our hope to at some point integrate with the various Apache projects (xml.apache.org and jakarta.apache.org), at which point we'd turn the copyright over to them.

## 4.2. 3.2. Project Position Statement

For developers on a Java and/or XML environment this project will provide all the tools necessary for outputting XML data in the Microsoft Excel format. This project seeks to make the use of Microsoft Windows based servers unnecessary for file format considerations and to fully document the OLE 2 Compound Document format. The project aims not only to provide the tools for serializing XML to Excel and Word file formats and the tools for writing to those file formats from Java, but also to provide the tools for later projects to convert other OLE 2 Compound Document formats to pure Java APIs.

## 4.3. 3.3. Summary of Capabilities

HSSF Serializer for Apache Cocoon 2

| Benefit | Supporting Features |
|---|---|
| Ability to serialize styles from XML spreadsheets. | HSSFSerialzier will support styles. |
| Ability to read and write formulas in XLS files. | HSSF will support reading/writing formulas. |
| Ability to output in MS Word on any platform using Java. | The project will develop an API that outputs in Word format using pure Java. |
| Enhance performance for reading and writing XLS files. | HSSF will undergo a number of performance enhancements. HSSF will include a new event-based API for reading XLS files. POIFS will support a new event-based API for reading OLE2 CDF files. |
| Ability to generate XML from XLS files | The project will develop an HSSF Generator. |
| The ability to generate charts | HSSF will provide low level support for chart records as well as high level API support for generating charts. The ability to read chart information will not initially be provided. |

## 4.4. 3.4. Assumptions and Dependencies

- The HSSF Serializer and Generator will support the Gnumeric 1.0 XML tag language.

- The HSSF Generator and HSSF Serializer will be mutually validating. It should be possible to have an XLS file created by the Serializer run through the Generator and the output back through the Serializer (via the Cocoon pipeline) and get the same file or a reasonable facimille (no one cares if it differs by the order of the binary records in some minor but non-visually recognizable manner).
- The HSSF Generator will run on any Java 2 supporting platform with Apache Cocoon 2 installed along with the HSSF and POIFS APIs.
- The HSSF Serializer will run on any Java 2 supporting platform with Apache Cocoon 2 installed along with the HSSF and POIFS APIs.
- The HWPF API requires a Java 2 implementation and the POIFS API.
- The HSSF API requires a Java 2 implementation and the POIFS API.
- The POIFS API requires a Java 2 implementation.

### 5. 4. Project Features

Enhancements to the POIFS API will include:

- An event driven API for reading POIFS Filesystems.
- A low-level API for creating/manipulating POI filesystems.
- Code improvements supporting greater separation between read and write structures.

Enhancements to the HSSF API will include:

- An event driven API for reading XLS files.
- Performance improvements.
- Formula support (read/write)
- Support for user-defined data formats
- Better documentation of the file format and structure.
- An API for creation of charts.

The HSSF Generator will include:

- A set of classes supporting the Cocoon 2 Generator interfaces providing a method for reading XLS files and outputting SAX events.
- The same tag format used by the HSSFSerializer in any given release.

The HWPF API will include:

- An event driven API for reading DOC files.
- A set of high and low level APIs for reading and writing DOC files.
- Documentation of the DOC file format or enhancements to existing documentation.

### 6. 5. Other Product Requirements

## 6.1. 5.1. Applicable Standards

All Java code will be 100% pure Java.

## 6.2. 5.2. System Requirements

The minimum system requirements for the POIFS API are:

- 64 Mbytes memory
- Java 2 environment
- Pentium or better processor (or equivalent on other platforms)

The minimum system requirements for the the HSSF API are:

- 64 Mbytes memory
- Java 2 environment
- Pentium or better processor (or equivalent on other platforms)
- POIFS API

The minimum system requirements for the the HWPF API are:

- 64 Mbytes memory
- Java 2 environment
- Pentium or better processor (or equivalent on other platforms)
- POIFS API

The minimum system requirements for the HSSF Serializer are:

- 64 Mbytes memory
- Java 2 environment
- Pentium or better processor (or equivalent on other platforms)
- Cocoon 2
- HSSF API
- POI API

## 6.3. 5.3. Performance Requirements

All components must perform well enough to be practical for use in a webserver environment (especially the "killer trio": Cocoon2/Tomcat/Apache combo)

## 6.4. 5.4. Environmental Requirements

The software will run primarily in developer environments. We should make some allowances for not-highly-technical users to write XML documents for the HSSF Serializer. All other components will assume intermediate Java 2 knowledge. No XML knowledge will be required except for using the HSSF Serializer. As much documentation as is practical shall be required for all components as XML is relatively new, and the concepts introduced

for writing spreadsheets and to POI filesystems will be brand new to Java and many Java developers.

### 7. 6. Documentation Requirements

## 7.1. 6.1 POI Filesystem

The filesystem as read and written by POI shall be fully documented and explained so that the average Java developer can understand it.

## 7.2. 6.2. POI API

The POI API will be fully documented through Javadoc. A walkthrough of using the high level POI API shall be provided. No documentation outside of the Javadoc shall be provided for the low-level POI APIs.

## 7.3. 6.3. HSSF File Format

The HSSF File Format as implemented by the HSSF API will be fully documented. No documentation will be provided for features that are not supported by HSSF API that are supported by the Excel 97 File Format. Care will be taken not to infringe on any "legal stuff". Additionally, we are collaborating with the fine folks at OpenOffice.org on *free* documentation of the format.

## 7.4. 6.4. HSSF API

The HSSF API will be documented by javadoc. A walkthrough of using the high level HSSF API shall be provided. No documentation outside of the Javadoc shall be provided for the low level HSSF APIs.

## 7.5. 6.5 HWPF API

The HWPF API will be documented by javadoc. A walkthrough of using the high level HWPF API shall be provided. No documentation outside of the Javadoc shall be provided for the low level HWPF APIs.

## 7.6. 6.6 HSSF Serializer

The HSSF Serializer will be documented by javadoc.

## 7.7. 6.7 HSSF Generator

The HSSF Generator will be documented by javadoc.

## 7.8. 6.8 HSSF Serializer Tag language

The XML tag language along with function and usage shall be fully documented. Examples will be provided as well.

### 8. 7. Terminology

## 8.1. 7.1 Filesystem

filesystem shall refer only to the POI formatted archive.

## 8.2. 7.2 File

file shall refer to the embedded data stream within a POI filesystem. This will be the actual embedded document.