

# POI 2.0 Vision Document

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

## 1. Jakarta POI -- Vision -- ##

#####POI#2.0#####

#####POI#####[Apache](#)

[Cocoon](#)#####POI#Jakarta#####

### 2. 1. #####

#### 2.1. 1.1 #####

#####POI#####POI#####

Serializer#HSSF#####POIFS#####

- HSSF Serializer  
##Cocoon2#####Serializer#####Microsoft Excel  
'97#####Java#####
- HSSF##### Pure Java ####Microsoft Excel  
97#####
- POIFS#####Microsoft OLE2  
#####100%Java#####

#####POI#####HSSF Generator##HDF#####

- HSSF Generator ##HSSF#####XLS (Excel  
97)#####SAX#####HSSF Generator##Apache  
Cocoon2#####
- HDF#####Microsoft Word 97#####Pure  
Java#####

#### 2.2. 1.2 #####

POI#####

- #####API#####POIFS#input#output#####

- #####API#####HSSF#####
- HSSF#####formula#####
- HSSF#####
- HSSF Serializer#####HSSF##Cocoon 2 Generator#####
- Microsoft Word#DOC#####(HDF)#####
- HSSFSerializer#####POIFSSerializer#HSSFSerializer#####
- #####

3. 2. #####

3.1. 2.1 #####

XML#UNIX#Java#####Office#####Microsoft#####OLE#####

1. #####HSSF  
Serializer#####XML#####
2. #####HSSF#####HSSF  
serializer#####XML###Java###Excel#####Serializer##
3. #####POIFS#####HSSF  
#HDF#####Java#####OLE2#####
4. #####HSSF  
generator#####Excel#####XML#####
5. #####HDF#####HDF  
Serializer#####.DOC#####

3.2. 2.2. #####

#####Java#####OS#####XML#####

3.3. 2.3. #####

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.

### 3.4. 2.4. ##

####Excel#####[##](#)#####...#####...

### 4. 3. #####

### 4.1. 3.1. #####

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. #####

Java#####XML#####XML####Microsoft Excel  
#####OLE2#####Microsoft  
Windows#####XML#Excel#Word#####J  
API# #####

### 4.3. 3.3. #####

Apache Cocoon 2####HSSF Serializer

##/####	#####
Ability to serialize styles from XML spreadsheets.	HSSFSerialzier#####

Ability to read and write formulas in XLS files.	HSSF#####formula#####
Ability to output in MS Word on any platform using Java.	100% Java #####Word#####API#####
Enhance performance for reading and writing XLS files.	HSSF#####HSSF###XLS##### CDF #####API#####
Ability to generate XML from XLS files	#####HSSF Generator#####
The ability to generate charts	HSSF#####API##### #####

**4.4. 3.4. #####**

- 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 HDF 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. #####**

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.

## POI 2.0 Vision Document

- 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 HDF 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. #####

#### 6.1. 5.1. #####

###Java#####100% pure Java#####

#### 6.2. 5.2. #####

POIFS API #####

- 64 Mbyte###
- Java 2 ##
- Pentium#####

HSSF API#####

- 64 Mbyte###
- Java 2 ##
- Pentium#####
- POIFS API

HDF API#####

- 64 Mbyte###
- Java 2 ##
- Pentium#####
- POIFS API

HSSF Serializer#####

- 64 Mbyte###
- Java 2 ##
- Pentium#####
- Cocoon 2

- HSSF API
- POI API

**6.3. 5.3. #####**

#####(##### Cocoon2/Tomcat/apache  
 ###)#####

**6.4. 5.4. #####**

#####HSSF#####XML#####

**7. 6. #####**

**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 HDF API**

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

#### **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.