POI-HMEF - Java API To Access Microsoft Transport Neutral Encoding Files (TNEF)

Overview

by Nick Burch

1. Overview

HMEF is the POI Project's pure Java implementation of Microsoft's TNEF (Transport Neurtral Encoding Format), aka winmail.dat, which is used by Outlook and Exchange in some situations.

Currently, HMEF provides a read-only api for accessing common message and attachment attributes, including the message body and attachment files. In addition, it's possible to have read-only access to all of the underlying TNEF and MAPI attributes of the message and attachments.

HMEF also provides a command line tool for extracting out the message body and attachment files from a TNEF (winmail.dat) file.

Note:

This code currently lives the scratchpad area of the POI SVN repository. Ensure that you have the scratchpad jar or the scratchpad build area in your classpath before experimenting with this code.

Note:

This code is a new POI feature, and the first release that will contain it will be POI 3.8 beta 2. Until then, you will need to build your own jars from a svn checkout.

2. Using HMEF to access TNEF (winmail.dat) files

2.1. Easy extraction of message body and attachment files

The class *org.apache.poi.hmef.extractor.HMEFContentsExtractor* provides both command line and Java extraction. It allows the saving of the message body (an RTF file), and all of the attachment files, to a single directory as specified.

From the command line, simply call the class specifying the TNEF file to extract, and the directory to place the extracted files into, eg:

```
java -classpath poi-3.8-FINAL.jar:poi-scratchpad-3.8-FINAL.jar org.apache
```

From Java, there are two method calls on the class, one to extract the message body RTF to a file, and the other to extract all the attachments to a directory. A typical use would be:

```
public void extract(String winmailFilename, String directoryName) throws Exception {
   HMEFContentsExtractor ext = new HMEFContentsExtractor(new File(winmailFilename));

   File dir = new File(directoryName);
   File rtf = new File(dir, "message.rtf");
   if(! dir.exists()) {
       throw new FileNotFoundException("Output directory " + dir.getName() + " not foun
   }

   System.out.println("Extracting...");
   ext.extractMessageBody(rtf);
   ext.extractAttachments(dir);
   System.out.println("Extraction completed");
}
```

2.2. Attachment attributes and contents

To get at your attachments, simply call the *getAttachments()* method on a *HMEFMessage* instance, and you'll receive a list of all the attachments.

When you have a *org.apache.poi.hmef.Attachment* object, there are several helper methods available. These will all return the value of the appropriate underlying attachment attributes, or null if for some reason the attribute isn't present in your file.

- *getFilename()* - returns the name of the attachment file, possibly in 8.3 format
- *getLongFilename()* - returns the full name of the attachment file
- *getExtension()* - returns the extension of the attachment file, including the "."
- *getModifiedDate()* - returns the date that the attachment file was last edited on
- *getContents()* - returns a byte array of the contents of the attached file
- *getRenderedMetaFile()* - returns a byte array of a windows meta file representation of the attached file

2.3. Message attributes and message body

A *org.apache.poi.hmef.HMEFMessage* instance is created from an *InputStream* of the underlying TNEF (winmail.dat) file.

From a *HMEFMessage*, there are three main methods of interest to call:

- *getBody()* - returns a String containing the RTF contents of the message body.
- *getSubject()* - returns the message subject
- *getAttachments()* - returns the list of *Attachment* objects for the message

2.4. Low level attribute access

Both Messages and Attachments contain two kinds of attributes. These are *TNEFAttribute* and *MAPIAttribute*.

TNEFAttribute is specific to TNEF files in terms of the available types and properties. In general, Attachments have a few more useful ones of these then Messages.

MAPIAttributes hold standard MAPI properties and values, and work in a similar way to HSMF (Outlook) does. There are typically many of these on both Messages and Attachments. *Note - see limitations*

Both *HMEFMessage* and *Attachment* supports support two different ways of getting to attributes of interest. Firstly, they support list getters, to return all attributes (either TNEF or MAPI). Secondly, they support specific getters by TNEF or MAPI property.

```
HMEFMessage msg = new HMEFMessage(new FileInputStream(file));
for(TNEFAttribute attr : msg.getMessageAttributes) {
   System.out.println("TNEF : " + attr);
}
for(MAPIAttribute attr : msg.getMessageMAPIAttributes) {
   System.out.println("MAPI : " + attr);
}
System.out.println("Subject is " + msg.getMessageMAPIAttribute(MAPIProperty.CONVERSATIO

for(Attachment attach : msg.getAttachments()) {
   for(TNEFAttribute attr : attach.getAttributes) {
      System.out.println("A.TNEF : " + attr);
   }
   for(MAPIAttribute attr : attach.getMAPIAttributes) {
      System.out.println("A.MAPI : " + attr);
   }
   System.out.println("Filename is " + attach.getAttribute(TNEFProperty.CID_ATTACHTITLE
   System.out.println("Extension is " + attach.getMAPIAttribute(MAPIProperty.ATTACH_EXT
}
```

3. Investigating a TNEF file

To get a feel for the contents of a file, and to track down where data of interest is stored, HMEF comes with HMEFDumper to print out the contents of the file.

4. Limitations

HMEF is currently a work-in-progress, and not everything works yet. The current limitations are:

- Non-standard MAPI properties from the range 0x8000 to 0x8fff may not be being quite correctly turned into attributes. The values show up, but the name and type may not always be correct.
- All testing so far has been performed on a small number of English documents. We think we're correctly turning bytes into Java unicode strings, but we need a few non-English sample files in the test suite to verify this!