Apache POI - HPSF Internals

by Rainer Klute

1. HPSF Internals

1.1. Introduction

A Microsoft Office document is internally organized like a filesystem with directory and files. Microsoft calls these files **streams**. A document can have properties attached to it, like author, title, number of words etc. These metadata are not stored in the main stream of, say, a Word document, but instead in a dedicated stream with a special format. Usually this stream's name is `\005SummaryInformation`, where `\005` represents the character with a decimal value of 5.

A single piece of information in the stream is called a **property**, for example the document title. Each property has an integral **ID** (e.g. 2 for title), a **type** (telling that the title is a string of bytes) and a **value** (what this is should be obvious). A stream containing properties is called a **property set stream**.

This document describes the internal structure of a property set stream, i.e. the **HPSF**. It does not describe how a Microsoft Office document is organized internally and how to retrieve a stream from it. See the POIFS documentation for that kind of stuff.

The HPSF is not only used in the Summary Information stream in the top-level document of a Microsoft Office document. Often there is also a property set stream named `\005DocumentSummaryInformation` with additional properties. Embedded documents may have their own property set streams. You cannot tell by a stream's name whether it is a property set stream or not. Instead you have to open the stream and look at its bytes.

1.2. Data Types

Before delving into the details of the property set stream format we have to have a short look at data types. Integral values are stored in the so-called **little endian** format. In this format the bytes that make out an integral value are stored in the "wrong" order. For example, the decimal value 4660 is 0x1234 in the hexadecimal notation. If you think this should be

represented by a byte 0x12 followed by another byte 0x34, you are right. This is called the **big endian** format. In the little endian format, however, this order is reversed and the low-value byte comes first: 0x3412.

The following table gives an overview about some important data types:

Name	Length	Example (Big Endian)	Example (Little Endian)
Bytes	1 byte	`0x12`	`0x12`
Word	2 bytes	`0x1234`	`0x3412`
DWord	4 bytes	`0x12345678`	`0x78563412`

ClassID
A sequence of one DWord, two Words and eight Bytes
	16 bytes	`0xE0859FF2F94F6810AB9108002B27B3D9`
		resp.
		`E0859FF2-F94F-6810-AB91-08002B27B3D9`
			`0xF29F85E04FF91068AB9108002B27B3D9`
			resp.
			`F29F85E0-4FF9-1068-AB91-08002B27B3D9`

		The ClassID examples are given here in two different notations. The second notation without the "0x" at the beginning and with dashes inside shows the internal grouping into one DWord, two Words and eight Bytes.	*Watch out:* Microsoft documentation and tools show class IDs a little bit differently like `F29F85E0-4FF9-1068-AB91-08002B` However, that representation is (intentionally?) misleading with respect to endianess.

1.3. HPSF Overview

A property set stream consists of three main parts:

1. The **header** and
2. the **section(s)** containing the properties.

1.4. The Header

The first bytes in a property set stream is the **header**. It has a fixed length and looks like this:

Offset	Type	Contents	Remarks
0	Word	`0xFFFE`	If the first four bytes of a stream do not contain these values, the stream is not a property set stream.

2	Word	`0x0000`	
4	DWord	Denotes the operating system and the OS version under which this stream was created. The operating system ID is in the DWord's higher word (after little endian decoding): `0x0000` for Win16, `0x0001` for Macintosh and `0x0002` for Win32 - that's all. The reader is most likely aware of the fact that there are some more operating systems. However, Microsoft does not seem to know.	
8	ClassID	`0x00000000000000000000000000000000`	Most property set streams have this value but this is not required.
24	DWord	`0x01000000` or greater	Section count. This field's value should be equal to 1 or greater. Microsoft claims that this is a "reserved" field, but it seems to tell how many sections (see below) are following in the stream. This would really make sense because otherwise you could not know where and how far you should read section data.

1.5. Section List

Following the header is the section list. This is an array of pairs each consisting of a section format ID and an offset. This array has as many pairs of ClassID and and DWord fields as

the section count field in the header says. The Summary Information stream contains a single section, the Document Summary Information stream contains two.

Type	Contents	Remarks
ClassID	Section format ID	`0xF29F85E04FF91068AB9108002B27B3D9` for the single section in the Summary Information stream. `0xD5CDD5022E9C101B939708002B2CF9AE` for the first section in the Document Summary Information stream.
DWord	Offset	The number of bytes between the beginning of the stream and the beginning of the section within the stream.
ClassID	Section format ID	...
DWord	Offset	...
...	...	...

1.6. Section

A section is divided into three parts: the section header (with the section length and the number of properties in the section), the properties list (with type and offset of each property), and the properties themselves. Here are the details:

	Type	Contents	Remarks
Section header	DWord	Length	The length of the section in bytes.
	DWord	Property count	The number of properties in the section.
Properties list	DWord	Property ID	The property ID tells what the property means. For example, an ID of `0x0002` in the Summary Information stands for the document's title. See the Property IDs chapter below for more

			details.
	DWord	Offset	The number of bytes between the beginning of the section and the property.
	...	...	...
Properties	DWord	Property type ("variant")	This is the property's data type, e.g. an integer value, a byte string or a Unicode string. See the *Property Types* chapter for details!
	Field length depends on the property type ("variant")	Property value	This field's length depends on the property's type. These are the bytes that make out the DWord, the byte string or some other data of fixed or variable length. The property value's length is always stored in an area which is a multiple of 4 in length. If the property is shorter, e.g. a byte string of 13 bytes, the remaining bytes are padded with `0x00` bytes.
	...	...	...

1.7. Property IDs

As seen above, a section holds a property list: an array with property IDs and offsets. The property ID gives each property a meaning. For example, in the Summary Information stream the property ID 2 says that this property is the document's title.

If you want to know a property ID's meaning, it is not sufficient to know the ID itself. You must also know the **section format ID**. For example, in the Document Summary Information stream the property ID 2 means not the document's title but its category. Due to Microsoft's

infinite wisdom the section format ID is not part of the section. Thus if you have only a section without the stream it is in, you cannot make any sense of the properties because you do not know what they mean.

So each section format ID has its own name space of property IDs. Microsoft defined some "well-known" property IDs for the Summary Information and the Document Summary Information streams. You can extend them by your own additional IDs. This will be described below.

1.7.1. Property IDs in The Summary Information Stream

The Summary Information stream has a single section with a section format ID of `0xF29F85E04FF91068AB9108002B27B3D9`. The following table defines the meaning of its property IDs. Each row associates a property ID with a *name* and an *ID string*. (The property *type* is just for informational purposes given here. As we have seen above, the type is always given along with the value.)

The property *name* is a readable string which could be displayed to the user. However, this string is useful only for users who understand English. The property name does not help with other languages.

The property *ID string* is about the same but looks more technically and is nothing a user should bother with. You could the ID string and map it to an appropriate display string in a particular language. Of course you could do that with the property ID as well and with less overhead, but people (including software developers) tend to be better in remembering symbolic constants than remembering numbers.

Property ID	Property Name	Property ID String	Property Type
2	Title	PID_TITLE	VT_LPSTR
3	Subject	PID_SUBJECT	VT_LPSTR
4	Author	PID_AUTHOR	VT_LPSTR
5	Keywords	PID_KEYWORDS	VT_LPSTR
6	Comments	PID_COMMENTS	VT_LPSTR
7	Template	PID_TEMPLATE	VT_LPSTR
8	Last Saved By	PID_LASTAUTHOR	VT_LPSTR
9	Revision Number	PID_REVNUMBER	VT_LPSTR
10	Total Editing Time	PID_EDITTIME	VT_FILETIME

11	Last Printed	PID_LASTPRINTED	VT_FILETIME
12	Create Time/Date	PID_CREATE_DTM	VT_FILETIME
13	Last Saved Time/Date	PID_LASTSAVE_DTM	VT_FILETIME
14	Number of Pages	PID_PAGECOUNT	VT_I4
15	Number of Words	PID_WORDCOUNT	VT_I4
16	Number of Characters	PID_CHARCOUNT	VT_I4
17	Thumbnail	PID_THUMBNAIL	VT_CF
18	Name of Creating Application	PID_APPNAME	VT_LPSTR
19	Security	PID_SECURITY	VT_I4

1.7.2. Property IDs in The Document Summary Information Stream

The Document Summary Information stream has two sections with a section format ID of `0xD5CDD5022E9C101B939708002B2CF9AE` for the first one. The following table defines the meaning of the property IDs in the first section. See the preceeding section for interpreting the table.

Property ID	Property name	Property ID string	VT type
0	Dictionary	PID_DICTIONARY	[Special format]
1	Code page	PID_CODEPAGE	VT_I2
2	Category	PID_CATEGORY	VT_LPSTR
3	PresentationTarget	PID_PRESFORMAT	VT_LPSTR
4	Bytes	PID_BYTECOUNT	VT_I4
5	Lines	PID_LINECOUNT	VT_I4
6	Paragraphs	PID_PARCOUNT	VT_I4
7	Slides	PID_SLIDECOUNT	VT_I4
8	Notes	PID_NOTECOUNT	VT_I4
9	HiddenSlides	PID_HIDDENCOUNT	VT_I4
10	MMClips	PID_MMCLIPCOUNT	VT_I4
11	ScaleCrop	PID_SCALE	VT_BOOL

12	HeadingPairs	PID_HEADINGPAIR	VT_VARIANT | VT_VECTOR
13	TitlesofParts	PID_DOCPARTS	VT_LPSTR | VT_VECTOR
14	Manager	PID_MANAGER	VT_LPSTR
15	Company	PID_COMPANY	VT_LPSTR
16	LinksUpTo Date	PID_LINKSDIRTY	VT_BOOL

1.8. Property Types

A property consists of a DWord *type field* followed by the property value. The property type is an integer value and tells how the data byte following it are to be interpreted. In the Microsoft world it is also known as the *variant*.

The *Usage* column says where a variant type may occur. Not all of them are allowed in a property set but just those marked with a [P]. **[V]** - may appear in a VARIANT, **[T]** - may appear in a TYPEDESC, **[P]** - may appear in an OLE property set, **[S]** - may appear in a Safe Array.

Variant ID	Variant Type	Usage	Description
0	VT_EMPTY	[V] [P]	nothing
1	VT_NULL	[V] [P]	SQL style Null
2	VT_I2	[V] [T] [P] [S]	2 byte signed int
3	VT_I4	[V] [T] [P] [S]	4 byte signed int
4	VT_R4	[V] [T] [P] [S]	4 byte real
5	VT_R8	[V] [T] [P] [S]	8 byte real
6	VT_CY	[V] [T] [P] [S]	currency
7	VT_DATE	[V] [T] [P] [S]	date
8	VT_BSTR	[V] [T] [P] [S]	OLE Automation string
9	VT_DISPATCH	[V] [T] [P] [S]	IDispatch *
10	VT_ERROR	[V] [T] [S]	SCODE
11	VT_BOOL	[V] [T] [P] [S]	True=-1, False=0
12	VT_VARIANT	[V] [T] [P] [S]	VARIANT *

13	VT_UNKNOWN	[V] [T] [S]	IUnknown *
14	VT_DECIMAL	[V] [T] [S]	16 byte fixed point
16	VT_I1	[T]	signed char
17	VT_UI1	[V] [T] [P] [S]	unsigned char
18	VT_UI2	[T] [P]	unsigned short
19	VT_UI4	[T] [P]	unsigned short
20	VT_I8	[T] [P]	signed 64-bit int
21	VT_UI8	[T] [P]	unsigned 64-bit int
22	VT_INT	[T]	signed machine int
23	VT_UINT	[T]	unsigned machine int
24	VT_VOID	[T]	C style void
25	VT_HRESULT	[T]	Standard return type
26	VT_PTR	[T]	pointer type
27	VT_SAFEARRAY	[T]	(use VT_ARRAY in VARIANT)
28	VT_CARRAY	[T]	C style array
29	VT_USERDEFINED	[T]	user defined type
30	VT_LPSTR	[T] [P]	null terminated string
31	VT_LPWSTR	[T] [P]	wide null terminated string
64	VT_FILETIME	[P]	FILETIME
65	VT_BLOB	[P]	Length prefixed bytes
66	VT_STREAM	[P]	Name of the stream follows
67	VT_STORAGE	[P]	Name of the storage follows
68	VT_STREAMED_OBJECT	[P]	Stream contains an object
69	VT_STORED_OBJECT	[P]	Storage contains an

			object
70	VT_BLOB_OBJECT	[P]	Blob contains an object
71	VT_CF	[P]	Clipboard format
72	VT_CLSID	[P]	A Class ID
0x1000	VT_VECTOR	[P]	simple counted array
0x2000	VT_ARRAY	[V]	SAFEARRAY*
0x4000	VT_BYREF	[V]	void* for local use
0x8000	VT_RESERVED		
0xFFFF	VT_ILLEGAL		
0xFFF	VT_ILLEGALMASKED		
0xFFF	VT_TYPEMASK		

1.9. The Dictionary

What a dictionary is good for is explained in the HPSF HOW-TO. This chapter explains how it is organized internally.

The dictionary has a simple header consisting of a single UInt value. It tells how many entries the dictionary comprises:

Name	Data type	Description
nrEntries	UInt	Number of dictionary entries

The dictionary entries follow the header. Each one looks like this:

Name	Data type	Description
key	UInt	The unique number of this property, i.e. the PID
length	UInt	The length of the property name associated with the key
value	String	The property's name, terminated with a 0x00 character

The entries are not aligned, i.e. each one follows its predecessor without any gap or fill

characters.

1.10. References

In order to assemble the HPSF description I used information publically available on the Internet only. The references given below have been very helpful. If you have any amendments or corrections, please let us know! Thank you!

1. In *Understanding OLE documents*, Ken Kyler gives an introduction to OLE2 documents and especially to property sets. He names the property names, types, and IDs of the Summary Information and Document Summary Information stream.
2. The *ActiveX Programmer's Reference* at http://www.dwam.net/docs/oleref/ seems a little outdated, but that's what I have found.
3. An overview of the `VT_` types is in *Variant Type Definitions*.
4. What is a `FILETIME`? The answer can be found under , http://www.vbapi.com/ref/f/filetime.html or http://www.cs.rpi.edu/courses/fall01/os/FILETIME.html. In short: *The FILETIME structure holds a date and time associated with a file. The structure identifies a 64-bit integer specifying the number of 100-nanosecond intervals which have passed since January 1, 1601. This 64-bit value is split into the two dwords stored in the structure.*
5. Microsoft provides some public information in the MSDN Library. Use the search function to try to find what you are looking for, e.g. "codepage" or "document summary information" etc.
6. This documentation origins from the HPSF description available at http://www.rainer-klute.de/~klute/Software/poibrowser/doc/HPSF-Description.html.