

# 复合文档文件格式研究

作者: Agstick

Date: 2007-3-20

Email: agstick@126.com

版权声明: 本文仅供个人学习研究, 如需复制、转载请注明出处, 未经作者许可, 不得作任何商业用途。

## 前言

复合文档 (Compound Document) 是一种不仅包含文本而且包括图形、电子表格数据、声音、视频图象以及其它信息的文档。可以把复合文档想象成一个所有者, 它装着文本、图形以及多媒体信息 如声音和图象。目前建立复合文档的趋势是使用面向对象技术, 在这里, 非标准信息如图像和声音可以作为独立的、自包含式对象包含在文档中。Microsoft Windows 就是使用这种技术, 叫做“OLE2 storage file format”或“Microsoft Office compatible storage file format”。当然 Excel、Word 等都是用这种格式存储的。本文主要研究复合文档的二进制结构, 本文的目的就是为另一篇文章《Excel 文件格式研究》(创作中) 做准备的。

本文是在英文资料的基础上完成的, 文中某些术语的翻译可能不够标准, 但都会给出英文原名。关于这方面的中文资料网上几乎找不到, 如果您有的话, 请发到我的邮箱, 谢谢。由于本人水平有限, 文中错误难免, 欢迎大家批评指正。

## 目录

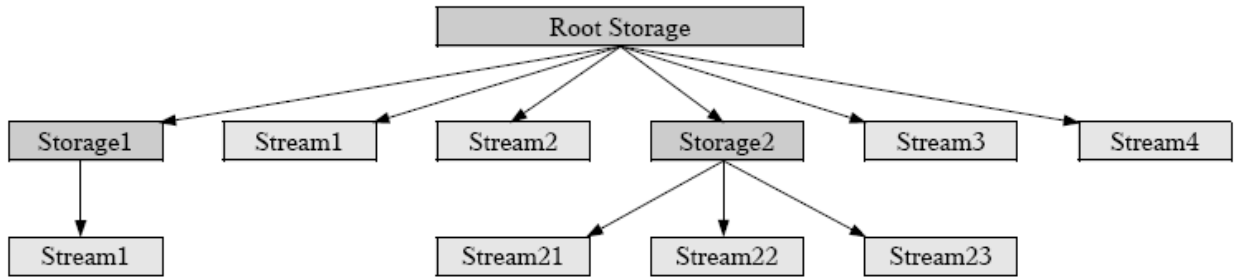
- 第一章 仓库与流 (Storages and Streams)
- 第二章 扇区与扇区链 (Sectors and Sector Chains)
- 第三章 复合文档头 (Compound Document Header)
- 第四章 扇区配置 (Sector Allocation)
- 第五章 短流 (Short-Streams)
- 第六章 目录 (Directory)
- 第七章 Excel 文件实例剖析

## 第一章 仓库与流

复合文档的原理就像一个文件系统 (文件系统: 如 FAT 与 NTFS)。复合文档将数据分成许多流 (Streams), 这些流又存储在不同的仓库 (Storages) 里。将复合文档想象成你的 D 盘, D 盘用的是 NTFS (NT File System) 格式, 流就相当于 D 盘里的文件, 仓库就相当于 D 盘里的文件夹。

流和仓库的命名规则与文件系统相似, 同一个仓库下的流及仓库不能重名, 不同仓库下可以有同名的流。每个复合文档都有一个根仓库 (root storage)。

**例:**



## 第二章 扇区与扇区链

### 2.1 扇区与扇区标识

所有的流又分成更小的数据块，叫做数据扇区（sectors）。Sectors 可能包含控制数据或用户数据。

整个文件由一个头（Header）结构以及其后的所有 Sectors 组成。Sectors 的大小在头中确定，且每个 Sectors 的大小都相同。

以下为示意图：

```

HEADER
SECTOR 0
SECTOR 1
SECTOR 2
SECTOR 3
SECTOR 4
SECTOR 5
SECTOR 6
  ⋮
  
```

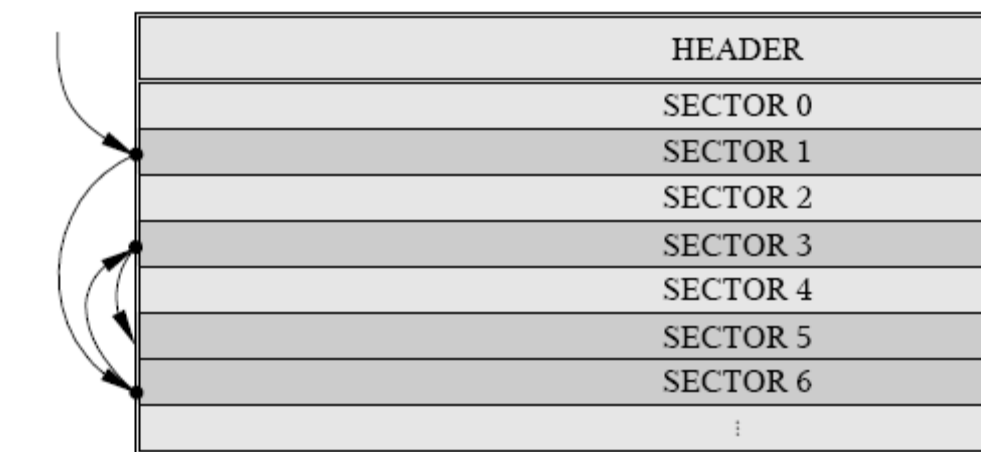
Sectors 简单的以其在文件中的顺序列举，一个扇区的索引（从 0 开始）叫做扇区标识（SID: sector identifier）。SID 是一个有符号的 32 位的整型值。如果一个 SID 的值非负，就表示真正存在的那个 Sector；如果为负，就表示特殊的含义。下表给出有效的特殊 SID：

SID	Name	Meaning
- 1	Free SID	空闲 sector，可存在于文件中，但不是任何流的组成部分。
- 2	End Of Chain SID	SID 链的结束标记（见 2.2 节）
- 3	SAT SID	此 Sector 用于存放扇区配置表（SAT）（见 4.2 节）
- 4	MSAT SID	此 Sector 用于存放主扇区配置表（MSAT）（见 4.1 节）

### 2.2 扇区链与扇区标识链

用于存储流数据的所有 Sectors 的列表叫做扇区链（Sector Chain）。这些 Sectors 可以是无序的。因此用于指定一个流的 Sectors 的顺序的 SID 数组就称为 SID chain。一个 SID chain 总是以 End Of Chain SID（-2）为结束标记。

**例：**一个流由 4 个 Sector 组成，其 SID 链为[1, 6, 3, 5, -2]。



流的 SID 链是通过扇区配置表构建的（见 4.2 节），但短流和以下两种内部流除外：

1. 主扇区配置表，其从自身构建 SID 链（每个扇区包含下一个扇区的 SID）。
2. 扇区配置表，其通过主扇区配置表构建 SID 链。

## 第三章 复合文档头

### 3.1 复合文档头的内容

复合文档头在文件的开始，且其大小必定为 512 字节。这意味着第一个 Sector 的开始相对文件的偏移量为 512 字节。

复合文档头的结构如下：

Offset	Size	Contents
0	8	复合文档文件标识：D0 <sub>H</sub> CF <sub>H</sub> 11 <sub>H</sub> E0 <sub>H</sub> A1 <sub>H</sub> B1 <sub>H</sub> 1A <sub>H</sub> E1 <sub>H</sub>
8	16	此文件的唯一标识(不重要, 可全部为0)
24	2	文件格式修订号 (一般为003E <sub>H</sub> )
26	2	文件格式版本号(一般为0003 <sub>H</sub> )
28	2	字节顺序规则标识(见3.2): FE <sub>H</sub> FF <sub>H</sub> = Little-Endian FF <sub>H</sub> FE <sub>H</sub> = Big-Endian
30	2	复合文档中sector的大小(ssz), 以2的幂形式存储, sector实际大小为s_size = 2 <sup>ssz</sup> 字节(一般为9即512字节, 最小值为7即128字节)
32	2	short sector的大小(见5.1), 以2的幂形式存储, short sector实际大小为s_s_size = 2 <sup>sss</sup> 字节(一般为6即64字节, 最大为sector的大小)
34	10	Not used
44	4	用于存放扇区配置表 (SAT) 的sector总数
48	4	用于存放目录流的第一个sector的SID (见6)
52	4	Not used
56	4	标准流的最小大小(一般为4096 bytes), 小于此值的流即为短流。
60	4	用于存放短扇区配置表 (SSAT) 的第一个sector的SID (见5.2), 或为-2 (End Of Chain SID)如不存在。
64	4	用于存放短扇区配置表 (SSAT) 的sector总数

68	4	用于存放主扇区配置表 (MSAT) 的第一个 sector 的 SID (见4.1), 或为-2 ( <i>End Of Chain SID</i> ) 若无附加的 sectors。
72	4	用于存放主扇区配置表 (MSAT) 的 sector 总数
76	436	存放主扇区配置表 (MSAT) 的第一部分, 包含109个SID。

---

### 3.2 字节顺序 (Byte Order)

文件数据的二进制存储有两种方法 Little-Endian 和 Big-Endian, 但实际应用中只使用 Little-Endian 方法即: 低位 8 字节存放在地址的低位, 高位 8 字节存放在地址的高位。

**例:** 一个 32 位的整数 13579BDF<sub>H</sub>(转为十进制即 324508639), 以 Little-Endian 存放为 DF<sub>H</sub> 9B<sub>H</sub> 57<sub>H</sub> 13<sub>H</sub>, 以 Big-Endian 存放为 13<sub>H</sub> 57<sub>H</sub> 9B<sub>H</sub> DF<sub>H</sub>。(H 下标表示十六进制)

### 3.3 扇区偏移量

从头中的信息可以计算出一个 sector 的偏移量 (offset), 公式为:

$$\text{sec\_pos}(\text{SID}) = 512 + \text{SID} \cdot \text{s\_size} = 512 + \text{SID} \cdot 2^{\text{ssz}}$$

**例:** ssz = 10 and SID = 5:

$$\text{sec\_pos}(\text{SID}) = 512 + \text{SID} \cdot 2^{\text{ssz}} = 512 + 5 \cdot 2^{10} = 512 + 5 \cdot 1024 = 5632.$$

## 第四章 扇区配置

### 4.1 主扇区配置表

主扇区配置表 (MSAT: master sector allocation table) 是一个 SID 数组, 指明了所有用于存放扇区配置表 (SAT: sector allocation table) 的 sector 的 SID。MSAT 的大小 (SID 个数) 就等于存放 SAT 的 sector 数, 在头中指明。

MSAT 的前 109 个 SID 也存放于头中, 如果一个 MSAT 的 SID 数多余 109 个, 那么多出来的 SID 将存放于 sector 中, 头中已经指明了用于存放 MSAT 的第一个 sector 的 SID。在用于存放 MSAT 的 sector 中的最后一个 SID 指向下一个用于存放 MSAT 的 sector, 如果没有下一个则为 End Of Chain SID (-2)。

存放 MSAT 的 sector 的内容: (s\_size 表示 sector 的大小)

Offset	Size	Contents
0	s_size-4	MSAT 的(s_size-4) / 4 个 SID 的数组
s_size-4	4	下一个用于存放 MSAT 的 sector 的 SID, 或-2 (已为最后一个)

---

最后一个存放 MSAT 的 sector 可能未被完全填满, 空闲的地方将被填上 Free SID(-1)。

**例:** 一个复合文档需要 300 个 sector 用于存放 SAT, 头中指定 sector 的大小为 512 字节, 这说明一个 sector 可存放 128 个 SID。MAST 有 300 个 SID, 前 109 个放于头中, 其余的 191 个将要占用 2 个 sector 来存放。此例假定第一个存放

MSAT 的 sector 为 sector 1, 则 sector 1 包含 127 个 SID。第 128 个 SID 指向一个用于存放 MSAT 的 sector, 假定为 sector 6, 则 sector 6 包含剩下的 64 个 SID (最后一个 SID 为 -2, 其他的值为 -1)。

## 4.2 扇区配置表

扇区配置表 (SAT: sector allocation table) 是一个 SID 数组, 包含所有用户流 (短流除外) 和内部控制流 (the short-stream container stream, 见 5.1, the short sector allocation table, 见 5.2, and the directory, 见 7) 的 SID 链。SAT 的大小 (SID 个数) 就等于复合文档中所存在的 sector 的个数。

SAT 的建立就是通过按顺序读取 MSAT 中指定的 sector 中的内容。

存放 SAT 的 sector 的内容: (s\_size 表示 sector 的大小)

Offset	Size	Contents
0	s_size	SAT 的 s_size / 4 个 SID 的数组

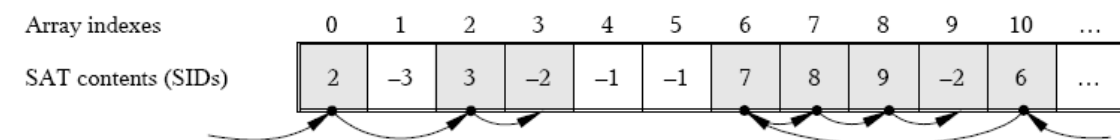
当通过 SAT 为一个流创建 SID 链时, SAT 数组的当前位置 (array index) 表示的就是当前的 sector, 而该位置存放的 SID 则指向下一个 sector。

SAT 可能在任意位置包含 Free SID (-1), 这些 sector 将不被流使用。如果该位置包含 End Of Chain SID (-2) 表示一个流的结束。如果 sector 用于存放 SAT 则为 SAT SID (-3), 同样用于存放 MSAT 则为 MSAT SID (-4)。

一个 SID 链的起点从用户流的目录入口 (directory entry, 见 6.2 节) 或头 (内部控制流) 或目录流本身获得。

**例:** 一个复合文档包含一个用于存放 SAT 的 sector (sector 1) 和 2 个流。

Sector 1 的内容如下图:



在位置 1 其值为 -3, 表明 Sector 1 是 SAT 的一部分。

其中一个流为内部目录流, 假定头中指定其开始为 Sector 0, SAT 中位置 0 的值为 2, 位置 2 的值为 3, 位置 3 的值为 -2。因此目录流的 SID 链为 [0, 2, 3, -2], 即此目录流存放于 3 个 sector 中。

目录中包含一个用户流的入口假定为 sector 10, 从图中可看出此流的 SID 链为 [10, 6, 7, 8, 9, -2]。

## 第五章 短流

### 5.1 短流存放流

当一个流的大小小于指定的值 (在头中指定), 就称为短流 (short-stream)。短流并不是直接使用 sector 存放数据, 而是内含在一种特殊的内部控制流——短流存放流 (short-stream container stream) 中。

短流存放流象其他的用户流一样: 先从目录中的根仓库入口 (root storage entry) 获得第一个使用的 sector, 其 SID 链从 SAT 中获得。然后此流将其所占

用的 sectors 分成 short-sector，以便用来存放短流。此处也许较难理解，我们来打个比方：既然流组成符合文档，而短流组成短流存放流，这两者是相似的。把短流存放流当作复合文档，那么短流对应流，short-sector 对应 sector，唯一的不同是复合文档有一个头结构，而短流存放流没有。short-sector 的大小在头中已经指定，因此可根据 SID 计算 short-sector 相对于短流存放流的偏移量 (offset)。公式为：

$$\text{short\_s\_pos}(\text{SID}) = \text{SID} \cdot \text{short\_s\_size} = \text{SID} \cdot 2^{\text{SSSZ}}$$

**例：** sssz = 6 and SID = 5:

$$\text{short\_s\_pos}(\text{SID}) = \text{SID} \cdot 2^{\text{SSSZ}} = 5 \cdot 2^6 = 5 \cdot 64 = 320.$$

## 5.2 短扇区配置表

短扇区配置表 (SSAT: short-sector allocation table) 是一个 SID 数组，包含所有短流的 SID 链。与 SAT 很相似。

用于存放 SSAT 的第一个 sector 的 SID 在头中指定，其余的 SID 链从 SAT 中获得。

存放 SSAT 的 sector 的内容：(s\_size 表示 sector 的大小)

Offset	Size	Contents
0	s_size	SSAT 的 s_size / 4 个 SID 的数组

SSAT 的用法与 SAT 类似，不同的是其 SID 链引用的是 short-sector。

# 第六章 目录

## 6.1 目录结构

目录(directory)是一种内部控制流，由一系列目录入口(directory entry)组成。每一个目录入口都指向复合文档的一个仓库或流。目录入口以其在目录流中出现的顺序被列举，一个以 0 开始的目录入口索引称为目录入口标识(DID: directory entry identifier)。

如下图所示：

```
DIRECTORY ENTRY 0
DIRECTORY ENTRY 1
DIRECTORY ENTRY 2
DIRECTORY ENTRY 3
:
```

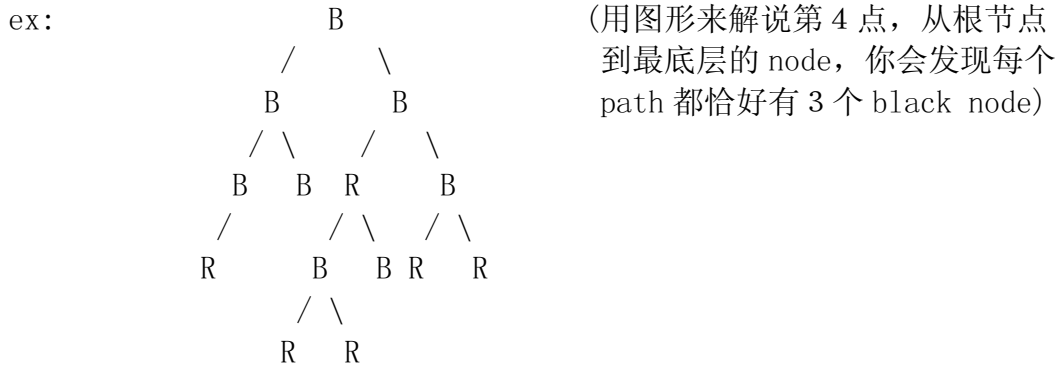
目录入口的位置不因其指向的仓库或流的存在与否而改变。如果一个仓库或流被删除了，其相应的目录入口就标记为空。在目录的开始有一个特殊的目录入口，叫做根仓库入口 (root storage entry)，其指向根仓库。

目录将每个仓库的直接成员 (仓库或流) 放在一个独立的红黑树 (red-black tree) 中。红黑树是一种树状的数据结构，本文仅简单介绍一下，详细情况请参

考有关资料。

建构一个 Red-Black tree 的规则：

1. 每个节点 (node) 的颜色属性不是红就是黑。
2. 根节点一定是黑的。
3. 如果某个节点是红的，那它的子节点一定是黑的。
4. 从根节点到每个叶节点的路径 (path) 必须有相同数目的黑节点。



注意并不总是执行上述规则。安全的方法是忽略节点的颜色。

**例：**以第一章中的图为例

1. 根仓库入口描述根仓库，它不是任何仓库入口的成员，因此无需构建红黑树。
2. 根仓库的所有直接成员 (“Storage1”, “Storage2”, “Stream1”, “Stream2”, “Stream3”, 和 “Stream4”) 将组成一棵红黑树，其根节点的 DID 存放于根仓库入口中。
3. 仓库 Storage1 只有一个成员 Stream1, Stream1 构成一棵红黑树，此树只有一个节点。Storage1 的目录入口包含 Stream1 的 DID。
4. 仓库 Storage2 包含 3 个成员 “Stream21”, “Stream22”, 和 “Stream23”。这 3 个成员将构建一棵红黑树，其根节点的 DID 存放于 Storage2 的目录入口中。

这种存放规则将导致每个目录入口都包含 3 个 DID：

1. 在包含此目录入口的红黑树中，此目录入口的左节点的 DID。
2. 在包含此目录入口的红黑树中，此目录入口的右节点的 DID。
3. 若此目录入口表示一个仓库，则还包含此仓库的直接成员所组成的另一颗红黑树的根节点的 DID。

在构建红黑树的过程中，一个节点究竟作为左还是右，是通过比较其名字来判断的。一个节点比另一个小是指其名字的长度更短，如长度一样，则逐字符比较。

规定：左节点 < 根节点 < 右节点。

## 6.2 目录入口

一个目录入口的大小严格地为 128 字节，计算其相对目录流的偏移量的公式为： $\text{dir\_entry\_pos(DID)} = \text{DID} \cdot 128$ 。

目录入口的内容：

Offset	Size	Contents
0	64	此入口的名字 (字符数组)，一般为 16 位的 Unicode 字符，以 0 结束。(因此最大长度为 31 个字符)
64	2	用于存放名字的区域的大小，包括结尾的 0。 (如：一个名字右 5 个字符则此值为 $(5+1) \cdot 2 = 12$ )

66	1	入口类型: 00H = Empty 01H = User storage 02H = User stream	03H = LockBytes (unknown) 04H = Property (unknown) 05H = Root storage
67	1	此入口的节点颜色: 00H = Red	01H = Black
68	4	其左节点的DID (若此入口为一个user storage or stream), 若没有左节点就为-1。	
72	4	其右节点的DID (若此入口为一个user storage or stream), 若没有右节点就为-1。	
76	4	其成员红黑树的根节点的DID (若此入口为storage), 其他为-1。	
80	16	唯一标识符 (若为storage) (不重要, 可能全为0)	
96	4	用户标记(不重要, 可能全为0)	
100	8	创建此入口的时间标记。大多数情况都不写。	
108	8	最后修改此入口的时间标记。大多数情况都不写。	
116	4	若此为流的入口, 指定流的第一个sector或short-sector的SID, 若此为根仓库入口, 指定短流存放流的第一个sector的SID, 其他情况, 为0。	
120	4	若此为流的入口, 指定流的大小 (字节) 若此为根仓库入口, 指定短流存放流的大小 (字节) 其他情况, 为0。	
124	4	Not used	

时间标记(time stamp) 是一个符号的64位的整数, 表示从1601-01-01 00:00:00开始的时间值。此值的单位为 $10^{-7}$ 秒。

当计算时间标记是要注意闰年。

**例:** 时间标记值为 01AE408B10149C00<sub>H</sub>

计算步骤	公式	结果
转为十进制		$t_0 = 121,105,206,000,000,000$
化成秒的余数	$r_{frac} = t_0 \bmod 10^7$	$r_{frac} = 0$
化成秒的整数	$t_1 = t_0 / 10^7$	$t_1 = 12,110,520,600$
化成分的余数	$r_{sec} = t_1 \bmod 60$	$r_{sec} = 0$
化成秒的整数	$t_2 = t_1 / 60$	$t_2 = 201,842,010$
化成小时的余数	$r_{min} = t_2 \bmod 60$	$r_{min} = 30$
化成小时的整数	$t_3 = t_2 / 60$	$t_3 = 3,364,033$
化成天的余数	$r_{hour} = t_3 \bmod 24$	$r_{hour} = 1$
化成天的整数	$t_4 = t_3 / 24$	$t_4 = 140,168$
距1601-01-01的整年到1984年还剩的天数	$r_{year} = 1601 + t_4$ 含的年 $t_5 = t_4 - (1601-01-01$ 到1984-01-01的天数)	$r_{year} = 1601 + 383 = 1984$ $t_5 = 140,168 - 139,887 = 281$
距1984-01-01的月数到10月还剩的天数	$r_{month} = 1 + t_5$ 含的月数 $t_6 = t_5 - (1984-01-01$ 到1984-10-01的天数)	$r_{month} = 1 + 9 = 10$ $t_6 = 281 - 274 = 7$
10月最终天数	$r_{day} = 1 + t_6$	$r_{day} = 1 + 7 = 8$

**最后的结果为1984-10-8 01:30:00。猜猜看这是什么日子.....**

## 第七章 Excel文件实例剖析

这章我们以一个Excel文件作为实例来分析其二进制结构。看实例永远是最好的学习方法，呵呵。

### 1.复合文档头

首先，读取此文件头，假定此Excel文件的头（512字节）内容如下：

```
00000000h DO CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00
00000010h 00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00
00000020h 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00
00000030h 0A 00 00 00 00 00 00 00 00 10 00 00 02 00 00 00
00000040h 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00
00000050h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000060h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000070h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000080h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000090h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000A0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000B0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000C0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000D0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000E0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000000F0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000100h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000110h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000120h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000130h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000140h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000150h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000160h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000170h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000180h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000190h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001A0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001B0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001C0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001D0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001E0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000001F0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```

1) 前8个字节是固定的标识，表示这是一个复合文档文件。

```
00000000h DO CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00
```

2) 接着16个字节是唯一标识, 其后的4个字节表示修订号和版本号, 可不用管它。

00000000<sub>h</sub> D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00

00000010<sub>h</sub> 00 00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00

3) 接下来的2个字节是字节顺序 (Byte Order) 标识符, 总是FE<sub>H</sub> FF<sub>H</sub>。

00000010<sub>h</sub> 00 00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00

4) 接着的2个字节表示sector的大小, 再2个字节表示short sector的大小。分别是2<sup>9</sup>=512字节和2<sup>6</sup>=64字节。

00000010<sub>h</sub> 00 00 00 00 00 00 00 00 00 3B 00 03 00 FE FF 09 00

00000020<sub>h</sub> 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00

5) 接着的10个字节无有效数据, 可忽略。

00000020<sub>h</sub> 06 00 00 00 00 00 00 00 00 00 00 01 00 00 00

6) 接着的4个字节表示用于存放扇区配置表 (SAT) 的sector总数, 此例为1个。

00000020<sub>h</sub> 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00

7) 接着的4个字节表示用于存放目录流的第一个sector的SID, 这里为sector 10。

00000030<sub>h</sub> 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

8) 接着的4个字节无有效数据, 可忽略。

00000030<sub>h</sub> 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

9) 接着的4个字节表示标准流的最小大小, 这里为00001000<sub>H</sub> = 4096字节。

00000030<sub>h</sub> 0A 00 00 00 00 00 00 00 00 10 00 00 02 00 00 00

10) 接着的4个字节表示用于存放短扇区配置表 (SSAT) 的第一个sector的SID, 其后4个字节表示用于存放短扇区配置表 (SSAT) 的sector总数, 这里SSAT从sector 2开始, 并只占用1个sector。

00000030<sub>h</sub> 0A 00 00 00 00 00 00 00 10 00 00 02 00 00 00

00000040<sub>h</sub> 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

11) 接着的4个字节表示用于存放主扇区配置表 (MSAT) 的第一个sector的SID, 其后的4个字节表示用于存放主扇区配置表 (MSAT) 的sector总数, 这里SID为-2, 说明没有附加的sector用于存放MSAT。

00000040<sub>h</sub> 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

12) 最后的436个字节包含MSAT的前109个SID。只有第一个SID有效, 因为上面已经说了SAT只占用1个sector。从这里可看出为sector 0。其他的SID标记为Free SID值为-1。

00000040<sub>h</sub> 01 00 00 00 FE FF FF FF 00 00 00 00 00 00 00 00

00000050<sub>h</sub> FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

00000060<sub>h</sub> FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

.....

## 2. 主扇区配置表

头中已经包含整个MSAT, 其SID链为[0, -2]。

## 3. 扇区配置表

在此例中扇区配置表仅占用sector 0, 它开始于文件偏移量为00000200<sub>H</sub> = 512字节处。假定内容如下:

```

00000200h FD FF FF FF FF FF FF FE FF FF FF 04 00 00 00
00000210h 05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00
00000220h 09 00 00 00 FE FF FF FF 0B 00 00 00 FE FF FF FF
00000230h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000240h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000250h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000260h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000270h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000280h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000290h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002A0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002B0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002C0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002D0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002E0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000002F0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000300h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000310h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000320h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000330h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000340h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000350h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000360h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000370h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000380h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000390h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003A0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003B0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003C0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003D0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003E0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000003F0h FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

```

因此可构建SAT的SID数组为：

Array index	0	1	2	3	4	5	6	7	8	9	10	11	12	...
SID array	-3	-1	-2	4	5	6	7	8	9	-2	11	-2	-1	...

可看出sector 0被标记为SAT SID（-3），sector 1和sector 12及其后的所有sector都未被使用。

#### 4. 短扇区配置表

在头中我们知道SSAT从sector 2开始只占用一个sector。从SAT中可看出位置2的值为-2，表示结束，故SSAT的SID链为[2, -2]。

开始于文件偏移量为00000600<sub>H</sub> = 1536字节处。  
假定内容如下：

```

00000600H 01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00
00000610H 05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00
00000620H 09 00 00 00 0A 00 00 00 0B 00 00 00 0C 00 00 00
00000630H 0D 00 00 00 0E 00 00 00 0F 00 00 00 10 00 00 00
00000640H 11 00 00 00 12 00 00 00 13 00 00 00 14 00 00 00
00000650H 15 00 00 00 16 00 00 00 17 00 00 00 18 00 00 00
00000660H 19 00 00 00 1A 00 00 00 1B 00 00 00 1C 00 00 00
00000670H 1D 00 00 00 1E 00 00 00 1F 00 00 00 20 00 00 00
00000680H 21 00 00 00 22 00 00 00 23 00 00 00 24 00 00 00
00000690H 25 00 00 00 26 00 00 00 27 00 00 00 28 00 00 00
000006A0H 29 00 00 00 2A 00 00 00 2B 00 00 00 2C 00 00 00
000006B0H 2D 00 00 00 FE FF FF FF 2F 00 00 00 FE FF FF FF
000006C0H FE FF FF FF 32 00 00 00 33 00 00 00 34 00 00 00
000006D0H 35 00 00 00 FE FF FF FF FF FF FF FF FF FF FF FF
000006E0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000006F0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000700H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000710H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000720H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000730H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000740H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000750H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000760H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000770H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000780H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
00000790H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007A0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007B0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007C0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007D0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007E0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
000007F0H FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF

```

因此可构建SSAT的SID数组为：

Array indexes	0	1	2	3	4	5	6	7	8	9	10	11	...	41	42	43	44	45	46	47	48	49	50	51	52	53	54	...
SID array	1	2	3	4	5	6	7	8	9	10	11	12	...	42	43	44	45	-2	47	-2	-2	50	51	52	53	-2	-1	...

从sector 54开始所有的short-sectors都未被使用。

## 5. 目录

头中指明用于存放目录流的第一个sector为sector 10。目录总是存放于sector中，而不是short sector中。故从SAT构建SID链为[10, 11, -2]。sector 10的偏移量为00001600H = 5632, sector 11的偏移量为00001800H = 6144。此例中sector的大小为512字节，因此每个sector包含4个目录入口（一个目录入口128字节），故此目录共包含8个入口。

### <1>根仓库入口（Root Storage Entry）

第一个目录入口总是根目录入口，假定其内容如下：

```
00001600H 52 00 6F 00 6F 00 74 00 20 00 45 00 6E 00 74 00
00001610H 72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00
00001620H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001630H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
00001650H 10 08 02 00 00 00 00 00 C0 00 00 00 00 00 00 46
00001660H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001670H 00 00 00 00 03 00 00 00 80 0D 00 00 00 00 00 00
```

1) 前64个字节是此入口名字的字符数组（为16位的字符，以第一个00结束），可看出此入口的名字是“Root Entry”。

```
00001600H 52 00 6F 00 6F 00 74 00 20 00 45 00 6E 00 74 00
00001610H 72 00 79 00 00 00 00 00 00 00 00 00 00 00 00 00
00001620H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001630H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

2) 接着2个字节表示用于存放上述字符数组的有效区域的大小，这里是22字节，只有10个有效字符。

```
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
```

3) 接着1个字节表示入口类型，根仓库入口故为05H。

```
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
```

4) 接着1个字节表示入口节点颜色，这里为红色，打破了根仓库入口为黑的规定。

```
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
```

5) 接着4个字节表示其左节点的DID，再接着4个字节表示右节点的DID。对根仓库入口来说都是-1。

```
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
```

6) 接着4个字节表示根仓库入口的成员构建的红黑树的根节点的DID，此例为1。

```
00001640H 16 00 05 00 FF FF FF FF FF FF FF FF 01 00 00 00
```

7) 接着16个字节表示唯一标识符，说明这是一个仓库，其后4字节表示用户标识，再接着是两个时间标记，各8字节，表明此仓库的创建时间和最后修改时间。这些数据都可忽略。

```
00001650H 10 08 02 00 00 00 00 00 C0 00 00 00 00 00 00 46
00001660H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00001670H 00 00 00 00 03 00 00 00 80 0D 00 00 00 00 00 00
```

8) 接着4个字节表示短流存放流的第一个sector的SID，其后4字节表示短流存放流的大小。此例中第一个sector为sector 3，其大小为00000D80H = 3456字节。

00001670H 00 00 00 00 03 00 00 00 80 0D 00 00 00 00 00 00

9) 最后4个字节无有效数据，可忽略。

00001670H 00 00 00 00 03 00 00 00 80 0D 00 00 00 00 00 00

<1>第二个目录入口

第二个目录入口（DID 1）假定其内容如下：

00001680H 57 00 6F 00 72 00 6B 00 62 00 6F 00 6F 00 6B 00

00001690H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000016A0H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000016B0H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000016C0H 12 00 02 00 02 00 00 00 04 00 00 00 FF FF FF FF

000016D0H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000016E0H 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

000016F0H 00 00 00 00 00 00 00 00 51 0B 00 00 00 00 00 00

用颜色标记的为重要数据。入口的名字是“Workbook”，它表示一个流。其左节点的DID是2，右节点的DID是4。其第一个sector的SID是0，其大小是00000B51H = 2897字节，小于4096字节，故其存放于短流存放流中。

<3>剩下的目录入口

剩下的目录入口可按上面的方法读取，下面给出目录：

DID	Name	Type	左节点的DID	右节点的DID	第一个成员的DID	第一个sector的SID	流大小	配置表
0	Root Entry	root	none	none	1	3	3456	SAT
1	Workbook	stream	2	4	none	0	2897	SSAT
2	<01H>CompObj	stream	3	none	none	46	73	SSAT
3	<01H>Ole	stream	none	none	none	48	20	SSAT
4	<05H>SummaryInformation	stream	none	none	none	49	312	SSAT
5	empty							
6	empty							
7	empty							

从根仓库入口的第一个成员的DID（此为1）可以找到根仓库的所有成员。入口1有2个子节点，DID 2 和 DID 4，DID 2又有一个子节点DID 3，DID 3和DID 4都没有子节点了。因此根仓库包含DID 1、2、3、4这4个成员。

<4>流的SID链

短流存放流是存储在sector中的，小于4096字节的用户流（为短流）存储在短流存放流中，并使用SSAT构建SID链。

下表给出所有流的情况:

<b>DID</b>	<b>流名</b>	<b>配置表</b>	<b>SID 链</b>
0	Root Entry (短流存放流)	SAT	[3, 4, 5, 6, 7, 8, 9, - 2]
1	Workbook	SSAT	[0, 1, 2, 3, 4, 5, ..., 43, 44, 45, - 2]
2	<01 <sub>H</sub> >CompObj	SSAT	[46, 47, - 2]
3	<01 <sub>H</sub> >Ole	SSAT	[48, - 2]
4	<05 <sub>H</sub> >SummaryInformation	SSAT	[49, 50, 51, 52, 53, - 2]

#### <5>流的读取

短流存放流读取其SID链中的所有sector, 此例中将按顺序读取sector 3, 4, 5, 6, 7, 8, 9, 故此流的大小为 $512 \times 7 = 3584$ 字节。但是只有前3456字节有效(根仓库入口中指定的)。这3456字节被分成大小为64字节的short sector, 一共54个。

现在我们来读取流“<01<sub>H</sub>>CompObj”, 其SID链为[46, 47, - 2], 此流是一个短流。其数据存放在short sector 46和short-sector 47中。short sector 46在短流存放流中的偏移量为2944字节, short-sector 47的偏移量为3008字节。

#### 参考文献:

Daniel Rentz. Microsoft Compound Document File Format. 2006-Dec 21.