Lotus 123 - .wk4 format

Contents

Copyright and Disclaimer

This document describes the Lotus 123R4 (.wk4) spreadsheet format. I don't have access to any documentation for this format so I've had to reverse engineer it. Consequently, there will be errors and omissions in this document. This document is provided in the hope that it may be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

This is a work in progress. I've described what I've found to date. I've found many bytes which always have the same value. These may be for future expansion, or are set by a feature I've never used. If you write an input filter from this document, you should check all fixed-value bytes and output a message if you read a different value, otherwise important information could be lost without warning.

Summary of file format

The format of a Lotus 123R4 file is a series of variable-length binary records. Each record consist of a four byte header and a variable length body. The first two bytes of the header contain the record type and bytes three and four hold the length of the body (which may be zero). As the x86 architecture is little-endian, all numbers stored in the file are stored least significant byte first.

For example: record type 0x1B with a length of 0x138 would be stored as: 1B 00 38 01

Some simple code to process records in the file might look like this:

```
unsigned char hdr[4];
unsigned char rec[65536];

while (read(0, hdr, 4) == 4) {
        int code = hdr[0] + hdr[1]*256;
        int len = hdr[2] + hdr[3]*256;
        printf("Code %2X, len %5d\n", code, len);

        int byt = read(0, rec, len);
        if (byt != len) {
                printf("error: attempted to read %d bytes, got %d\n", len, byt);
                break;
        }

        switch (code) {
        ...
        }
}
```

Description of record types

Before starting the description, some terms will be defined:

workbook	All the data stored in a .wk4 file. A workbook may consist of several sheets.
sheet	Sometimes referred to as a table or page.
cell	A location on a sheet, referred to by column and row e.g. B52.
record	A piece of data in a .wk4 file. Each record has a type and a length.

All record types and other codes should be assumed to be hexadecimal unless stated otherwise. Record lengths are decimal.

The .wk4 format appears to be in three sections. The first section begins with record type 00 and ends with record type 01. It contains all the data in all the sheets in the workbook, including style and formatting information. The second section also begins with record type 00 and ends with record type 01. It contains a number of records with types between 80 and FF. I don't know what any of these are for. The third section doesn't have a start or end record. It appears to contain general workbook information e.g. author, last revision time etc. The final record in this section doesn't have a header. This section is described in Description of document info fields. Note that some records in this section have the same record type as some in section 1, but have a different meaning.

A number of records contain a cell reference. These are stored in four bytes as: <row_lo> <row_hi> <sheet> <column>. Numbering is from 0. For example, cell A1 on sheet 1 would be stored as 00 00 00 00; cell AA345 on sheet 3 would be stored as: 58 01 02 1A. You will observe from this that Lotus 123R4 supports at most 256 sheets with 256 columns and 65536 rows (although some versions of Lotus are further limited to 8192

rows).

There now follows a list of record types in numerical order. Note that some types contain subtypes - the first one or two bytes in the record body further classify the data in the record (e.g. record type 1B).

00	Start Section
01	End Section
07	column width
13	Cells to Format
16	Text
17	long double
18	encoded number
19	Formula
1A	Formula result if text
1B	Styles etc.
1B - subtype B0 36	sheet name
1B - subtype D7 07	row height
1B - subtype E6 0F	background style
1B - subtype E7 32	named style
1B - subtype F0 0F	text style

---

00 - Start Section

Length: 26 bytes

This record appears twice in a file (it may appear a third time - see Description of document info fields - but has a different meaning in that case). The first time this record appears, bytes[12,13] contain the number of occasions on which the workbook was edited. Somewhere in this record is probably a Lotus version number. I find bytes[0..3] always contain: 02 10 04 00.

---

01 - End Section

Length: 0 bytes

This record appears twice in a file (it may appear a third time - see Description of document info fields - but has a different meaning in that case). This record has no body.

07 - column width

Length: 4 bytes + column widths * 2

There is one of these records for each sheet containing columns whose widths have been adjusted. This record consists of a four byte header and sub-records of two bytes. In the header, byte[0] indicates the sheet, byte[1] contains 00 and bytes[2,3] contain ??

In the sub-records, the first byte indicates the column affected and the second byte indicates the column width. The column width is measured in "characters". A character-width appears to be 8 pixels - it doesn't depend on the font size used in a cell. There are as many sub-records as columns whose widths have been adjusted in that sheet.

13 - Cells to Format

Length: variable

This record appears once for every row containing a cell that **doesn't** use the default format (i.e. no borders, background colour, bold, italic etc.). This record consists of a four byte header, and sub-records of length four or five bytes. In the header, byte[0] indicates the sheet, byte[1] contains 00; and bytes[2,3] contain the row affected.

Note: there are also records where byte[1] contains 01 or 02. I don't know what these are for yet.

The format of the sub-records is (think runlength encoding):

Byte Number	Byte Description
0	numeric format (see below)
1-3	style to use (see 1B subtype E6 0F - background style) style = (byte[1] + (byte[2] << 8) + ((byte[3] & 0x3F) << 16)) >> 6
3	type of sub-record (byte[3] & 0xC0): 00 - default style for this cell 40 - apply style in bytes[2-3] to this cell 80 - default style for this cell and next nn C0 - apply style in bytes[2-3] to this cell and next nn
4	only present for sub-record types 80 and C0 type 80 - number of columns to skip type C0 - number of cells following this one with the same format

The lowest 6 bits of byte[1] are always zero.

The coding of the numeric format:

Code	Description
8x	Fixed (e.g. 1.23): x = number of decimal places
9x	Scientific (e.g. 1.23E+00): x = number of decimal places
Ax	Currency (e.g. $1.23): x = number of decimal places
Bx	Percentage (e.g. 1.23%): x = number of decimal places
Cx	Comma (e.g. 1,000,000.23): x = number of decimal places
F2	Date: 30-Dec-69
F3	Date: 30-Dec
F4	Date: Dec-69
F7	Time: 12:23:45 PM
F8	Time: 12:23 PM
F9	Date: 30/12/69
FA	Date: 30/12
FB	Time: 12:23:45
FC	Time: 12:23
FF	default format

As an example, consider cells B50..E50 have default style with fixed format numbers of 2dp and K50 uses style 0x16 and default numeric format:

Bytes	Value	Description
0-3	00 00 31 00	50 = 0x32, but row numbers as stored are zero-based
4-7	FF 00 00 00	always start at column A. No special formatting for A50
8-12	82 00 00 C0 03	numeric style 82; C0 03 indicates following 3 cells have same format
13-17	FF 00 00 80 03	80 03 indicates skip this cell and next 3
18-21	FF 80 05 40	style = 580 >> 6; 40 indicates only this cell affected

---

16 - Text

Length: 5 + length of string + '\0'

Byte Number	Byte Description
0-3	cell reference
4	alignment: ' - left; ^ - centre; " - right

5-end	zero-terminated string

17 - long double

Length: 14 bytes

Byte Number	Byte Description
0-3	cell reference
4-13	10 byte floating point number (GNU long double)

18 - encoded number

Length: 6 bytes

Byte Number	Byte Description
0-3	cell reference
4-5	encoded number - see code below

The idea of this record is to represent common numbers in a more compact form than long double.

```
long double decode_num(unsigned char* bytes)
{
        short h = (bytes[5] << 8) | bytes[4];
        if (!(h & 1) return h/2;

        long double ld = h >> 4;
        switch (h & 0x0F) {
        case 0x1: return ld * 5000;
        case 0x3: return ld * 500;
        case 0x5: return ld / 20;
        case 0x7: return ld / 200;
        case 0x9: return ld / 2000;
        case 0xB: return ld / 20000;
        case 0xD: return ld / 16;
        case 0xF: return ld / 64;
        }
        // not reached
}
```

19 - Formula

Length: variable

Byte Number	Byte Description
0-3	cell reference
4-13	10 byte floating point number (GNU long double). This is the result of evaluating the formula. If the result is not numeric, this will be NAN and the answer will be in a record of type 1A
14-end	formula in reverse Polish notation

Formula codes:

Code	Description
00	floating point number: 10 byte (long double)
01	cell reference: a one byte code followed by 4 byte cell reference. The one byte code means: 00 or 04 - fixed row and column e.g. A1 05 - fixed row e.g. A$1 06 - fixed column e.g. $A1 07 - normal e.g. A1
02	range: a one byte code followed by two 4 byte cell references. The one byte describes which columns/rows are fixed: bits0-2 - fixed/floating for start cell (encoded as above) bits3-5 - fixed/floating for end cell (encoded as above) bit6 - 0 bit7 - 1
03	end of formula
04	parentheses (reverse Polish doesn't need parantheses - these are just to recreate the original formula
05	a 2 byte encoded number
06	null terminated string
0E	- (unary)
0F	+ (binary)
10	- (binary)
11	*
12	/
13	^ (exponentiation)

14	= (equal to - not assignment)
15	<> (not equal to)
16	<=
17	>=
18	<
19	>
1A	AND
1B	OR
1C	NOT
1D	+ (unary)
1F	NA
20	ERR
26	PI
27	SIN
28	COS
29	TAN
30	CHOOSE (followed by one byte = number of arguments)
31	ISNA
32	ISERR
33	FALSE
34	TRUE
3B	IF
44	ISNUMBER
45	ISSTRING
46	LENGTH
47	VALUE
48	STRING
49	MID
4A	CHAR
4B	CODE
4C	FIND

50	SUM (followed by one byte = number of arguments)
53	MIN (followed by one byte = number of arguments)
54	MAX (followed by one byte = number of arguments)
55	VLOOKUP
5A	HLOOKUP
62	INDEX (followed by one byte = number of arguments)
65	REPEAT
66	UPPER
67	LOWER
68	LEFT
69	RIGHT
6A	REPLACE
6B	PROPER
6D	TRIM
6E	CLEAN
6F	S
70	N
73	@@
7A	An extended function. After the code 7A there is: one byte - number of arguments to function four bytes - purpose unknown nine bytes - "@<<@123>>" variable bytes - function name one byte - '(' one byte - 00 (null-terminator for string)
7E	ISRANGE
8B	ISAPP (followed by one byte = number of arguments)
8C	ISAAF (followed by one byte = number of arguments)

Obviously this list is incomplete.

As an example, consider the formula: 7 * -E2. First convert to reverse Polish: 7 E2 - *. This will be encoded as follows:

Bytes	**Description**
05 0E 00	encoded number (7)

01 07 01 00 00 04	cell reference (E2)
0E	unary -
11	*
03	end of formula

1A - Formula result if text

Length: variable

If we have a formula e.g. @if(a>b, "greater", "less") where the result is a string instead of a number, then the result of evaluating the formula is stored in this record.

Byte Number	Byte Description
0-3	cell reference
4-end	formula result as zero-terminated string

1B - Styles etc.

Length: variable

This record type is further subdivided. The first two bytes in the record indicate the type of data.

1B - subtype B0 36 - sheet name

Byte Number	Byte Description
0-1	B0 36 - sheet name subtype
2	sheet number (0 = first sheet)
3	00
4-end	sheet name: zero-terminated string

1B - subtype D7 07 - row height

Length: 4 bytes + row heights * 8

There is one of these records for each sheet containing rows whose heights have been adjusted. This record consists of a four byte header and sub-records of eight bytes.

Byte Number	Byte Description
0-1	D7 07 - row height subtype
2	sheet number (0 = first sheet)
3	00
4-end	sub-records of 8 bytes

There is one sub-record for each row whose height has been adjusted:

Byte Number	Byte Description
0-1	row number
2-3	height of row. Divide this by 32 to get height in points (lower 5 bits always 0). Add approx 15% to this number to get height in pixels.
4-7	some bit fields (I usually get 60 00 00 00)

1B - subtype E6 0F - background style

Length: 12 bytes + styles * 15

A note about styles: when the user changes the font, text colour, number format, background, border etc. Lotus 123 generates none, one or two styles. If the only change is the number format e.g. Fixed 2dp then this is encoded in a format record (type 13). If the user changes "background attributes" e.g. background colour/pattern, borders etc. then these are stored in a background style record. If the user changes a text attribute e.g. font, point size, text colour etc. then a text style record and a background style record are created. This is because the format record contains a link to a background style; each background style record then contains a "pointer" to an associated text style.

Byte Number	Byte Description
0-1	E6 0F - background style subtype
2-9	10, 00, 00, 01, 10, 00, 0E, 00 There may be some bit flags in here
10	high byte of background style number
11	00
12-end	sub-records of 15 bytes

There is one sub-record for each background style:

Byte Number	Byte Description
0	background style number, used by format record
1-2	associated foreground style (byte[1] + (byte[2] & 0x1F) * 256)
2	bit5 (0x20) indicates underline
3	pattern colour (see table)
4	background colour (see table)
5	bit0 (0x01) indicates background colour is used. If this is 0 then background colour will be 00 and the worksheet default background colour will be used (usually white). bit1 (0x02) indicates pattern colour is used. If this is 0 then pattern colour will be 00 and then worksheet default will be used (usually black). bits2-7 indicate pattern to use. Pattern 0 is solid background colour, 1 is solid pattern colour (seems pointless), 2 is unused?. Patterns 3-63 are various stripes, checkerboards, blobs etc. Each pattern is composed of two colours: the background colour and the pattern colour.
6	alignment: 0x04 = left, 0x08 = right, 0x0C =centre (default 0x00)
7-12	border types and colour. The bit layout is too horrible to describe, so see the explanation and code fragments below.
13	currency - see table. If the format of a cell is to display a currency this byte contains the currency to use. The number of decimal places is stored in the format record.
14	??

Table of colours for background, pattern and text. There are 256 possible colours (with a couple of duplicates). I have indicated below only the 16 standard Windows colours. If someone would like to decode the other 240 that would be great :-)

colour number	RGB value	description
FF	000000	black
00	FFFFFF	white
15	FFFFFF	white again
81	FF0000	red
44	00FF00	green
A8	0000FF	blue
32	FFFF00	yellow
7C	FF00FF	magenta
46	00FFFF	cyan

AE	800000	dark red
84	008000	dark green
BA	000080	dark blue
83	808000	dark yellow
AC	800080	dark magenta
76	008080	dark cyan
5F	C0C0C0	light grey (RGB may be wrong)
9F	808080	dark grey (RGB may be wrong)

Table of currencies

number	**currency**	**description**
01	ARS	Argentine Peso
02	AUD	Australian Dollar
03	ATS	Austrian Schilling
04	BEF	Belgian Franc
05	BRC	Brazilian Cruzeiro
06	GBP	United Kingdom Pound
07	CAD	Canadian Dollar
08	CNY	Chinese Yuan
09	CSK	Czech Koruna
0A	DKK	Danish Krone
0B	ECU	European Currency Unit (obsolete)
0C	FIM	Finnish Markka
0D	FRF	French Franc
0E	DEM	German Mark
0F	GRD	Greek Drachma
10	HKD	Hong Kong Dollar
11	HUF	Hungarian Forint
12	INR	Indian Rupee
13	IDR	Indonesian Rupiah
14	IEP	Irish Punt

15	ITL	Italian Lira
16	JPY	Japanese Yen
17	LUF	Luxembourg Franc
18	MYR	Malaysian Ringgit
19	MXP	Mexican Peso
1A	NLG	Netherlands Guilder
1B	NZD	New Zealand Dollar
1C	NOK	Norwegian Kroner
1D	PLZ	Polish Zloty
1E	PTE	Portuguese Escudo
1F	ROL	Romanian Lei
20	RUB	Russian Rouble
21	SGD	Singapore Dollar
22	SKK	Slovakian Koruna
23	SIT	Slovenian Tholar
24	ZAR	South African Rand
25	KRW	South Korean Won
26	ESP	Spanish Peseta
27	SEK	Swedish Krona
28	CHF	Swiss Franc
29	TWD	Taiwan Dollar
2A	THB	Thai Baht
2B	USD	United States Dollar
2C	--	Other (custom currency)

There are four possible border positions: left, top, right, bottom. For each border, four bits are allocated for the border colour and four bits for the border type. As the encoding is a little odd, it is shown as a C expression in the table below.

position	**type**	**colour**
left	byte[12] & 0x0F	((byte[10] & 0x01) << 3) | (byte[9] >> 5)
top	byte[11] & 0x0F	(byte[8] >> 2) & 0x0F
right	byte[11] >> 4	byte[9] & 0x0F
bottom	(byte[10] >> 2) & 0x0F	((byte[8] & 0x01) << 3) | (byte[7] >> 5)

The possible border types are shown below. I have shown binary patterns for some of them - a 1 indicates a pixel is set.

border type	description
0	none
1	line
2	double line
3	double thickness line
4	short dashes (0011 0011)
5	short dashes separated (0000 0011)
6	long dashes (0011 1111)
7	dot dash (0010 0111)
8	dot dot dash (0101 0111)

The possible border colours are shown below:

border colour	RGB value	description
0	000000	black
1	FFFFFF	white
2	FF0000	red
3	00FF00	green
4	0000FF	blue
5	FFFF00	yellow
6	FF00FF	magenta
7	00FFFF	cyan
8	800000	dark red
9	008000	dark green
A	000080	dark blue
B	808000	dark yellow
C	800080	dark magenta
D	008080	dark cyan
E	C0C0C0	light grey (RGB may be wrong)
F	808080	dark grey (RGB may be wrong)

1B - subtype E7 32 - named style

Length: variable

The user can format a cell then create a style based on the formatting in that cell. This style can then be applied to other cells. There will always be 16 of these records stored in the file, though usually they will contain only default values.

Byte Number	Byte Description
0-1	F0 0F - text style subtype
2	style number (01 - 10)
3	00
4-7	number format (see format sub-records). byte[7]=40
8-13	text (not zero-terminated). Don't know what it's for. It contains "Normal" for undefined styles and "STYLn" for defined styles (n = 1-??)
14-23	unknown
24-end	name of style (zero-terminated string). The name is entered by the user, or "Undefined" for undefined styles.

1B - subtype F0 0F - text style

Length: variable

Byte Number	Byte Description
0-1	F0 0F - text style subtype
2-3	style number
4	font size (in points)
5	00
6	usually same as byte 4
7	00
8-15	unknown
16	bit0 (0x01) indicates bold. bit1 (0x02) indicates italic. Note: if the style number is 08 or 18 then the style will be bold even if this byte is 00; if the style number is 10 or 18 then the style will be italic even if this byte is 00.
17-19	unknown

20	text colour (see table)
21-end	font name

Description of document info fields

The doument info fields are stored in the third section of the file. Each record contains one piece of information. Some record types are the same as used in the first section of the file. Document info fields which contain single strings are **not** zero-termintated (use the record length).

The following records contain (non-zero-terminated) strings.

Record Type	Description
00	Comments
01	Title
02	Subject
03	Keywords
04	Revisions
05	Author or Last Revisor
09	Author or Last Revisor

Other record types:

Record Type	Length	Description
06	4	number of minutes document has been edited for (stored as 4 byte int)
07	10	document creation time. See time format below
0A	10	time document last revised. See time format below
0C	2	Number of times document saved

After Record 0C, there are some zero-terminated strings and then some random(?) bytes. Note that there is no header for this information.

Time format:

Byte Number	Byte Description
0-3	number of days. 1 = 31/12/1899. According to Lotus, 29/2/1900 is a valid date, so be careful of out-by-one errors before/after this date. To convert to a Unix time_t: time_t t = (days - (365*70 + 19)) * 24*60*60;
4-5	number of hours

6-7	number of minutes
8-9	number of seconds

TODO

There are a number of areas where this document is incomplete. I don't know all of the things that are missing, but I've listed below some of them.

- Table of RGB values for all 256 colours used as backgrounds and text colours
- Pictures of the 61 background patterns
- Complete list of formula codes

The following Lotus spreadsheet features are not decoded at all

- Charts
- Drawing elements e.g. lines and rectangles
- "Fancy borders"
- Workbook/sheet defaults e.g. default font, column width, number format etc.
- Printing info e.g. printer, paper type, print range, header, footer, margins