Nikon Electronic File (NEF) file format description

Version 0.31 (May 22th, 2009)
lclevy@free.fr with "NEF format" as subject.

This document is a work in progress, if you want to help, send me a mail!

Changelog

- 22may2009 : commented dcraw NEF decompression (when compression tag == 34713), see section 4.2.

---

1. Overview

The NEF RAW format from Nikon is used to store digital picture produced by their digital camera.

This format is based on TIFF format and usually has 2 subIFDs, the first one to store the full image in lossy jpeg, the second one for the full RAW image lossless compressed. The D100 and the D1x have only one subIFD, for the RAW image.
The Makernote has the NikonImagePreview tag (0x0011) which contains a thumbnail image (in lossy jpeg).
IFD#0 also contains a thumbnail image in uncompressed TIFF, size is 160x120.

In Makernote, starting with version 200, the ColorBalance tag (0x0097) is encrypted. This can be decrypted using content of tags 0x1d (serial number) and 0x00a7 (shutter count), and some hardcoded values. See extract of the Dcraw code below.

All information in this page has been gathered without hurting any proprietary software, using public information like Dcraw or ExifTool and as a test for my CR2 file parser.

2. NEF header

the NEF header is a standard TIFF header.

Offset	Length	Type	Description	Value
0x0000	1	short	byte order	Usually 0x4D4D / "MM", except for E5700 (0x4949 / "II")
0x0002	1	short	TIFF magic value	0x2a
0x0004	1	long	TIFF offset	8
0x0008		IFD	first IFD	

3. IFD#0

See Exif tags for Phil Harvey for more details.

Tag value	Name	Type	Length	Description
0x00fe / 254	SubfileType	4	1	1=Reduced-resolution image
0x0100 / 256	imageWidth	4	1	160
0x0101 / 257	ImageHeight	4	1	120
0x0102 / 258	BitsPerSample	3	3	[8, 8, 8]
0x0103 / 259	Compression	3	1	1=uncompressed
...	...			...

0x014a / 330	**SubIFD tag**	4	2	[JpegImageOffset, RawOffset] : offsets to the 2 child IFDs
0x0214 / 34665	ReferenceBlackWhite	5=rational	6	
0x8769 / 34665	**EXIF**	4	1	offset to the EXIF IFD. the EXIF IFD contains a pointer to the **Makernote** IFD
0x9286 / 37510	UserComment	7	variable	
...	...			...

4. SubIFDs

The 0x014a tag is the subIFD tag. See tech note from Adobe. It contains the list of child IFDs.

4.1 IFD#0, subIFD#0

This subIFD stores the full image in lossy jpeg. It does not exists within D100 and D1x files.

Tag value	Name	Type	Length	Description
0x00fe / 254	SubfileType	4	1	1=Reduced-resolution image
0x0103 / 259	Compression	3	1	6=old/jpeg
0x011a / 282	XResolution	5=rational	1	300
0x011b / 283	YResolution	5	1	300
0x0128 / 296	ResolutionUnit	3	1	2=pixel_per_inch
0x0201 / 513	JpgFromRawStart	4	1	offset to image data
0x0202 / 514	JpgFromRawLength	4	1	image data length
0x0213 / 531	YCbCrPositioning	3	1	2=co_sited

4.2 IFD#0, subIFD#1

this subIFD stores the full image in lossless compression.

Tag value	Name	Type	Length	Description
0x00fe / 254	SubfileType	4	1	0=Full-resolution Image
0x0100 / 256	ImageWidth	4	1	3904 for the D60
0x0101 / 257	ImageHeight	4	1	2616 for the D60
0x0102 / 258	BitsPerSample	1	1	12bits for the D60
0x0103 / 259	Compression	3	1	1=uncompressed, 34713=Nikon NEF Compressed [**Details**, Values]
0x0106 / 262	PhotometricInterpretation	3	1	32803=Color Filter Array
0x0111 / 273	JpgFromRawStart	4	1	offset to the image data
0x0115 / 277	SamplesPerPixel	3	1	1
0x0116 / 278	RowsPerStrip	3	1	2616 for the D60
0x0117 / 279	JpgFromRawLength	4	1	image data lenght
0x011a / 282	XResolution	5=rational	1	300
0x011b / 283	YResolution	5	1	300
0x011c / 284	PlanarConfiguration	3	1	1 = Chunky
0x0128 / 296	ResolutionUnit	3	1	2=pixel_per_inch
0x828d / 33421	CFARepeatPatternDim	3	2	[2, 2] = 2x2
0x828e / 33422	CFAPattern2	1	4	[1, 2, 0, 1] = [G, B, R, G] for the D60

0x9217 / 37399	SensingMethod	3	1	2 = One-chip color area (D60)

5. Makernote

the Nikon Makernote has its own format

5.1 Makernote header

Offset	Length	Type	Description	Value
0x0000	6	string	magic value	"Nikon", zero terminated
0x0006	1	short	version ?	0x0210
0x0008	1	short	?	0x0000
0x000a	1	short	byte order	Usually 0x4D4D / "MM", except for E5700 (0x4949 / "II")
0x000c	1	short	TIFF magic value	0x2a
0x00e	1	long	TIFF offset	8
0x0012		IFD	first IFD	

5.2 Makernote tags

Here follows some tags. See Nikon tags for Phil Harvey for more details.

Tag value	Name	Type	Length	Description
0x0001 / 1	MakerNoteVersion	7	4	version="0210" for the D60
0x0002 / 2	ISO	3	2	
0x000e / 14	ExposureDifference	7	4	
0x0011 / 17	**NikonPreview**	4	1	offset to this IFD
0x0012 / 18	FlashExposureComp	7	4	
0x001d / 27	SerialNumber	3	7	Serial numner (used for Color Balance decryption)
0x0093 / 147	NEF Compression	3		1=lossy type1 3=lossless 4=lossy type 2 (d90, d3, d700, d300)
0x0096 / 150	Linearization table	bytes		see format below
0x0097 / 151	Color Balance	7	572 for D60	Color Balance (see decryption code below for versionn >= 200)
0x00a7 / 167	ShutterCount	4	1	shutter count. used to decrypt While Balance tag
0x00bb / 187	**?**	7	6	

Nikon compression types

```
0x96 (linearization table) tag format

offset how_many   type   name
----+-----------+------+-------------------------------------------------------------------------------------------------
0    1           byte   version0
1    1           byte   version1
                         ver0=0x44, ver1=0x20 for 12bits and 14bits lossy (d300)
                         0x44, 0x20 : lossy (d300, d90 and d5000)
                         0x46, 0x30 for 12bits and 14 lossless (d300 and d700)
                         0x46, 0x30 : d3x/12b/lossless
                         0x46, 0x30. with d300 lossless. and d700/14b/lossless
                         0x44, 0x10 : with D100/D200/D2X/D40/D80/D60 12bits/lossy
                         tag 0x93 = 3 for lossless (0x46/0x30).
                         tag 0x93 = 4 for lossy type 2 (0x44/0x20)
                         tag 0x93 = 1 for lossy type 1 (0x44/0x10)
2    4           shorts vpred[2][2] (when ver0 == 0x49 || ver1 == 0x58, fseek (ifp, 2110, SEEK_CUR) before)
```

```
0x0a 1          short  curve_size.
                        32 with d3x/12bits/lossless, d300/12bits/lossless
                        34 with 14bits/lossless (d300 and d700)
                        257 with d300/12+14b/lossy.
                        257 with 12b/lossy for d90
                        567 with D100/12bits/lossy.
                        683 with 12b/lossy for d200,d2x,d40x,d40,d80,d60
0x0c curve_size  shorts curve[]
                        for lossy type 2, if curve_size == 257 (d90 and d300), end of curve table is 1+257*2 = 526
562  1          short  split_value (for 0x44/0x20 only (lossy type 2), d90 and d300)
----+-----------+------+-------------------------------------------------------------------------------------------

lossy type 2
- is reading a incomplete table from the NEF file and interpolation is required.
- is using a split value and a 2nd huffman table with some rows. See Dcraw code below.
```

Details

The base offset for Makernote is the first M of Makernote's "MM".

5.3 Makernote Nikon preview tag (0x0011)

Tag value	Name	Type	Length	Description
0x0103 / 259	Compression	3	1	6=old/jpeg
0x011a / 282	XResolution	5=rational	1	300
0x011b / 283	YResolution	5	1	300
0x0128 / 296	ResolutionUnit	3	1	2=pixel_per_inch
0x0201 / 513	JpgFromRawStart	4	1	offset to image data
0x0202 / 514	JpgFromRawLength	4	1	image data length
0x0213 / 531	YCbCrPositioning	3	1	2=co_sited

6. Parser outputs (experimental)

- Motorola byte order: D40, D40x, D60, D80, D90, D100, D200, D700, D1x, D2x, D3.
- Intel byte order: e5700.txt.

7. References

- DCRaw, the reference open source software to decode RAW formats, by Dave Coffin.
- EXIF organisation.
- Jpeg file format, ITU-t81 and ISO/IEC IS 10918-1 standard.
- TIFF resources. Adobe.
- Exiftool Nikon tags.
- DeNEF - Nikon D1 NEF image file decoder. local
- Formats d'image - RAW. C. Rousseau (French)
- Open source photography, NEF
- Is the Nikon D70 NEF (RAW) format truly lossless?

8. Samples

- to get RAW samples: raw.fotosite.pl, www.rawsamples.ch, imaging-resource.

9. White Balance encryption

- Nikon's photo encryption reported broken

See ExifTool Nikon tags : 0x1d (serial), 0xa7 (shuttercount), 0x97 (colorbalance)

```
(from DCraw v8.89)

ushort CLASS sget2 (uchar *s)
{
  if (order == 0x4949)          /* "II" means little-endian */
    return s[0] | s[1] << 8;
```

```
  else                          /* "MM" means big-endian */
    return s[0] << 8 | s[1];
}

void CLASS parse_makernote (int base, int uptag) {
  static const uchar xlat[2][256] = {
  { 0xc1,0xbf,0x6d,0x0d,0x59,0xc5,0x13,0x9d,0x83,0x61,0x6b,0x4f,0xc7,0x7f,0x3d,0x3d,
    0x53,0x59,0xe3,0xc7,0xe9,0x2f,0x95,0xa7,0x95,0x1f,0xdf,0x7f,0x2b,0x29,0xc7,0x0d,
    0xdf,0x07,0xef,0x71,0x89,0x3d,0x13,0x3d,0x3b,0x13,0xfb,0x0d,0x89,0xc1,0x65,0x1f,
    0xb3,0x0d,0x6b,0x29,0xe3,0xfb,0xef,0xa3,0x6b,0x47,0x7f,0x95,0x35,0xa7,0x47,0x4f,
    0xc7,0xf1,0x59,0x95,0x35,0x11,0x29,0x61,0xf1,0x3d,0xb3,0x2b,0x0d,0x43,0x89,0xc1,
    0x9d,0x9d,0x89,0x65,0xf1,0xe9,0xdf,0xbf,0x3d,0x7f,0x53,0x97,0xe5,0xe9,0x95,0x17,
    0x1d,0x3d,0x8b,0xfb,0xc7,0xe3,0x67,0xa7,0x07,0xf1,0x71,0xa7,0x53,0xb5,0x29,0x89,
    0xe5,0x2b,0xa7,0x17,0x29,0xe9,0x4f,0xc5,0x65,0x6d,0x6b,0xef,0x0d,0x89,0x49,0x2f,
    0xb3,0x43,0x53,0x65,0x1d,0x49,0xa3,0x13,0x89,0x59,0xef,0x6b,0xef,0x65,0x1d,0x0b,
    0x59,0x13,0xe3,0x4f,0x9d,0xb3,0x29,0x43,0x2b,0x07,0x1d,0x95,0x59,0x59,0x47,0xfb,
    0xe5,0xe9,0x61,0x47,0x2f,0x35,0x7f,0x17,0x7f,0xef,0x7f,0x95,0x95,0x71,0xd3,0xa3,
    0x0b,0x71,0xa3,0xad,0x0b,0x3b,0xb5,0xfb,0xa3,0xbf,0x4f,0x83,0x1d,0xad,0xe9,0x2f,
    0x71,0x65,0xa3,0xe5,0x07,0x35,0x3d,0x0d,0xb5,0xe9,0xe5,0x47,0x3b,0x9d,0xef,0x35,
    0xa3,0xbf,0xb3,0xdf,0x53,0xd3,0x97,0x53,0x49,0x71,0x07,0x35,0x61,0x71,0x2f,0x43,
    0x2f,0x11,0xdf,0x17,0x97,0xfb,0x95,0x3b,0x7f,0x6b,0xd3,0x25,0xbf,0xad,0xc7,0xc5,
    0xc5,0xb5,0x8b,0xef,0x2f,0xd3,0x07,0x6b,0x25,0x49,0x95,0x25,0x49,0x6d,0x71,0xc7 },

  { 0xa7,0xbc,0xc9,0xad,0x91,0xdf,0x85,0xe5,0xd4,0x78,0xd5,0x17,0x46,0x7c,0x29,0x4c,
    0x4d,0x03,0xe9,0x25,0x68,0x11,0x86,0xb3,0xbd,0xf7,0x6f,0x61,0x22,0xa2,0x26,0x34,
    0x2a,0xbe,0x1e,0x46,0x14,0x68,0x9d,0x44,0x18,0xc2,0x40,0xf4,0x7e,0x5f,0x1b,0xad,
    0x0b,0x94,0xb6,0x67,0xb4,0x0b,0xe1,0xea,0x95,0x9c,0x66,0xdc,0xe7,0x5d,0x6c,0x05,
    0xda,0xd5,0xdf,0x7a,0xef,0xf6,0xdb,0x1f,0x82,0x4c,0xc0,0x68,0x47,0xa1,0xbd,0xee,
    0x39,0x50,0x56,0x4a,0xdd,0xdf,0xa5,0xf8,0xc6,0xda,0xca,0x90,0xca,0x01,0x42,0x9d,
    0x8b,0x0c,0x73,0x43,0x75,0x05,0x94,0xde,0x24,0xb3,0x80,0x34,0xe5,0x2c,0xdc,0x9b,
    0x3f,0xca,0x33,0x45,0xd0,0xdb,0x5f,0xf5,0x52,0xc3,0x21,0xda,0xe2,0x22,0x72,0x6b,
    0x3e,0xd0,0x5b,0xa8,0x87,0x8c,0x06,0x5d,0x0f,0xdd,0x09,0x19,0x93,0xd0,0xb9,0xfc,
    0x8b,0x0f,0x84,0x60,0x33,0x1c,0x9b,0x45,0xf1,0xf0,0xa3,0x94,0x3a,0x12,0x77,0x33,
    0x4d,0x44,0x78,0x28,0x3c,0x9e,0xfd,0x65,0x57,0x16,0x94,0x6b,0xfb,0x59,0xd0,0xc8,
    0x22,0x36,0xdb,0xd2,0x63,0x98,0x43,0xa1,0x04,0x87,0x86,0xf7,0xa6,0x26,0xbb,0xd6,
    0x59,0x4d,0xbf,0x6a,0x2e,0xaa,0x2b,0xef,0xe6,0x78,0xb6,0x4e,0xe0,0x2f,0xdc,0x7c,
    0xbe,0x57,0x19,0x32,0x7e,0x2a,0xd0,0xb8,0xba,0x29,0x00,0x3c,0x52,0x7d,0xa8,0x49,
    0x3b,0x2d,0xeb,0x25,0x49,0xfa,0xa3,0xaa,0x39,0xa7,0xc5,0xa7,0x50,0x11,0x36,0xfb,
    0xc6,0x67,0x4a,0xf5,0xa5,0x12,0x65,0x7e,0xb0,0xdf,0xaf,0x4e,0xb3,0x61,0x7f,0x2f } };

  unsigned ver97=0, serial=0, i, wbi=0, wb[4]={0,0,0,0};
  uchar buf97[324], ci, cj, ck;

    if (tag == 0x97) {
      for (i=0; i < 4; i++)
        ver97 = ver97 * 10 + fgetc(ifp)-'0';
      switch (ver97) {
      case 100: // version < 200 is not encrypted
              fseek (ifp, 68, SEEK_CUR);
              FORC4 cam_mul[(c >> 1) | ((c & 1) << 1)] = get2();
              break;
      case 102:
              fseek (ifp, 6, SEEK_CUR);
              goto get2_rggb;
      case 103:
              fseek (ifp, 16, SEEK_CUR);
              FORC4 cam_mul[c] = get2();
      }
      if (ver97 >= 200) { // encrypted
              if (ver97 != 205) fseek (ifp, 280, SEEK_CUR);
              fread (buf97, 324, 1, ifp);
      }
    }

    if (tag == 0x1d) // first key (serialnumber)
      while ((c = fgetc(ifp)) && c != EOF)
        serial = serial*10 + (isdigit(c) ? c - '0' : c % 10);

    if (tag == 0xa7 && (unsigned) (ver97-200) < 12 && !cam_mul[0]) { // second key (shuttercount)
      ci = xlat[0][serial & 0xff];
      cj = xlat[1][fgetc(ifp)^fgetc(ifp)^fgetc(ifp)^fgetc(ifp)];
      ck = 0x60;
      for (i=0; i < 324; i++)
        buf97[i] ^= (cj += ci * ck++); // decryption
      i = "66666>666;6A"[ver97-200] - '0'; // offset depending on version : 6, 14 ('>'-'0'), 11 (';'-'0') or 17 ('A'-'0')
```

```
      FORC4 cam_mul[c ^ (c >> 1) ^ (i & 1)] =
              sget2 (buf97 + (i & -2) + c*2);
    }
}

...

get2_rggb:
      FORC4 cam_mul[c ^ (c >> 1)] = get2();
      fseek (ifp, 22, SEEK_CUR);
      FORC4 sraw_mul[c ^ (c >> 1)] = get2();

}
```