



## Contents

- [1 Introduction](#)
- [2 MIF format](#)
  - ◆ [2.1 MIF type 2](#)
  - ◆ [2.2 MIF type 3](#)
  - ◆ [2.3 MIF type 1](#)
- [3 Loading MIF icons](#)
- [4 References](#)

## Introduction

MIF format is a special file where several PNG or SVG-T images may be grouped and used later by your application. It is created using the utility `mifconv.exe`, provided in Nokia SDK. `mifconv.exe` generates a file with several indexes as well, useful for loading specific images from the MIF file. However, if you try to load images from other MIF files, like `avkon2.mif`, it is necessary to understand how MIF files are organized or you need to have the index list.

In this article it will be presented some conclusions about the MIF format, after generating and analyzing several personal MIF files. The results are incomplete and it was not possible to find any formal specification of MIF format. Moreover, the information provided here should not be used for reverse engineering in MIF files that do not belong to you. It is better to check the copyrights of the images first. The contributors of this article or Forum Nokia are not responsible for any illegal action using the information provided here.

Contributions about the MIF specification are welcome and encouraged.

## MIF format

Until this moment, it was possible to identify at least 3 different MIF file headers, denoted as MIF files type 1, 2 and 3. They have slight differences, described below.

## MIF type 2

For MIFs generated by mifconv.exe (MIF type 2), it is possible to see that it has a header (16 bytes long) starting with a magic number mark (B##A) and followed by a number that I guess is the MIF type (2). After, we can see an offset for next section and the amount of images in the MIF file. All these variables are 32bits and the alignment is little endian.

These header fields for MIF file type 2 are below (I am using a personal MIF file):

```
42232334 02000000 10000000 10000000
```

where:

1. magic\_number: 42232334 (B##4)
2. MIF\_type: 02000000 (2)
3. next\_field\_offset: 10000000 (16)
4. image\_count: 10000000 (16, for this example)

For MIF type 2, in the next section we have a table (position 16). This table has (image\_count)x2 entries with two 32 bits numbers. Each entry has a pair of image offset and image size. I believe we have an entry for the image and for its mask.

Ex: entries after offset 16:

```
90000000 971D0000 (image 0)
90000000 971D0000 (image 0 mask)
271E0000 310D0000 (image 1)
271E0000 310D0000 (image 1 mask)
...
```

Where:

- Image 0:
  - ◆ offset from beginning of file for image: 90000000 (144)
  - ◆ image size: 971D0000 (7575)
  - ◆ offset from beginning of file for image mask: 90000000 (144)
  - ◆ image mask size: 971D0000 (7575)
- Image 1:
  - ◆ ...

In this case, images and masks are equal and points to the same location.

Images are appended after this table. Each image has its own header, with 32 bytes. It is easy to see another magic number (C##4) at the beginning of the image header and the amount of bytes in the image, not counting its header. Images may be compressed or not.

Ex: image section header:

```
43232334 01000000 20000000 771D0000
01000000 0B000000 00000000 00000000
```

Where:

MIF type 2

## MIF\_format\_internals

1. magic\_number: 43232334 (C##4)
2. unkown
3. unkown
4. real\_image\_size (discounting header): 771D0000 (7543)
5. unkown
6. unkown (1, 4, 6, and sometimes 7)
7. unkown
8. unkown (I saw 0, 1, 4 here)

I analyzed several image headers and fields 6 and 8 sometimes change. But I could not see any relationship.

We may see non compressed SVG images inside the file, they are plain text. And it possible to extract them easily. I did this for my own MIF file. There is a program for extracting compressed SVG but it not worked for me when I compressed my own MIF file. But it works for avkon2.mif (you need to change the first long of the image, from CE56FA03 to CC56FA03). Of course, this may be illegal and I will not publish the results here or even recommend that you do it. It is better to check the copyrights of the images first.

To summarize, the MIF type 2 sections are below.

```
+-----+
|42232334|02000000|OFFSET  |NUM_IMG | <-- MAGIC, TYPE, NEXT FIELD, IMAGE COUNTER
+-----+
|IMG_OFFS|IMG_SIZE|IMG_OFFS|IMG_SIZE| <-- IMAGE TABLE (IMG_SIZE includes image header)
|IMG_OFFS|IMG_SIZE|IMG_OFFS|IMG_SIZE|
|IMG_OFFS|IMG_SIZE|IMG_OFFS|IMG_SIZE|
|  ...   |  ...   |  ...   |  ...   |
+-----+
|43232334|UNKNOWN |UNKNOWN |IMG_SIZE| <-- IMAGE HEADER, IMG_SIZE (real size)
|UNKNOWN |UNKNOWN |UNKNOWN |UNKNOWN |
+-----+
|                                     | <-- IMAGE
|                                     |
|          IMAGE 1                    |
|                                     |
+-----+
|43232334|UNKNOWN |UNKNOWN |IMG_SIZE| <-- IMAGE HEADER, IMG_SIZE (real size)
|UNKNOWN |UNKNOWN |UNKNOWN |UNKNOWN |
+-----+
|                                     | <-- IMAGE
|                                     |
|          IMAGE 2                    |
|                                     |
+-----+
|43232334|UNKNOWN |UNKNOWN |IMG_SIZE| <-- IMAGE HEADER, IMG_SIZE (real size)
|UNKNOWN |UNKNOWN |UNKNOWN |UNKNOWN |
+-----+
|                                     | <-- IMAGE
|                                     |
|          IMAGE 3                    |
|                                     |
+-----+
```

## MIF type 3

Some MIF files have the number 3 in the second field. MIF type 3 (MIF file in ROM?) does not have the images table section and its header seems to have 5 fields. I am guessing that the last field is a CRC. I tried CRC32 over some parts of the file but the calculated CRC was different.

Images (next section) start after this header at the position indicated in the third field in the header. Since we do not have any image table, it is necessary to read each message sequentially.

Ex avkon2.mif:

```
42232334 03000000 14000000 5A040000 CF272810
```

Where:

1. magic\_number: B##4
2. mif\_type: 03000000 (3)
3. next\_field\_offset: 14000000 (20)
4. image\_count: 5A040000 (1114)
5. CRC: CF272810

The number of images is a mystery. Header says 1114 but I could find only 423 image marks (C##4). Multiple layers/mask images ?

The image headers are equal in size but with some different numbers on it.

## MIF type 1

I saw some files with mif type 1. They were similar to mif type 2.

## Loading MIF icons

The next Python code snippet creates a listbox with all icons in avkon2.mif.

```
# Tested with Python 1.9.5 only (Nokia E71)
#
import struct
from appuifw import *
import os
import e32

mif = u"z:\\resource\\apps\\avkon2.mif"

fd = open(mif, "rb")
# decoding header
hdr = fd.read(16)
(mif_id, mif_type, next_sect_offset, num_imgs) = struct.unpack("<LLLL", hdr)
fd.close()

icons = []
```

## MIF\_format\_internals

```
for n in xrange(num_imgs):
    icons.append((os.path.basename(mif), u"", Icon(mif, n, n)))

app.body = Listbox(icons)
lock = e32.Ao_lock()
lock.wait()
```

## References

- I saw some related code inside [ensymble repository](#).
- You can use [View MBM](#) to load and show MIF file images.
- Discussion board [related thread](#).