Optima SC Inc.

+001-514-965-2177

http://www.optimasc.com

Description explorer magic database

Field name	
Author	Carl Eric Codère
Revision	2005-01-29
Document Reference	SPC-S200401-01
Document status	RELEASE

Table of Contents

Version	*Description of changes*
2004-09-14	**FIRST STABLE RELEASE**. Added version change table. Set status of document to released. Added freq (audio frequency) new reserved keyword. Fixed category numbers.
2004-10-16	Added MIME signature in file characteristics section.
2004-11-07	Small typo fixes. Clarification of ANDed values.
2005-01-29	Document identifier field has now been changed for easier lookup

1 Identification fields

The magic database of file formats is a general format for uniquely identifying the type and contents of files. This database has also limited capabilities for extracting metadata from those file. The magic database contains the following attributes which are assigned to each identified files:

1. Each identified file shall be categorized and sub-categorized
2. Each identified file shall have its extension identified
3. Each identified file shall have its file format version identified
4. Each identified file shall have its ISO/MIME file type
5. Each identified file shall have an identification comment

The following extra information will be retrieved (when possible), from the identified files:

- audio files: number of channel, frequency of sounds
- image files: xresolution, yresolution, bpp, dpi, xsize, ysize
- video files: xrersolution, yresolution, bpp, dpi, size, ysize
- code files: line information, symbol information
- general: the creator and title associated with the file

2 Unique file format identifier (FFID)

Unique file identifiers are used easily retrieve information on a file format from a central database. This is done without worrying if the resource for the specification changes locations. This referencing system is royalty free, and should be used by all new file formats devised throughout the world.

The format of the FID is as follows, and is not case sensitive

```
NNNNNNNNN-XX-IIIIIII
```

`NNNNNNNNN` = 9 digit registration code of the organism that created the standard to this file format. ([0-9]+)

`XX` = 2 digit category of the file format ([0.-9Aa-fF][0-9Aa-Ff])

`IIIIIII` = 7 digit document identifier.([0.-9A-Za-z]+)

2.1 Organization registration code

This will be assigned by the registrar as new applications are received. Organizations can be international associations, companies, or individuals. The registrar must register the maintainer of the file format, or the original creator of the file format if the file format is no longer active.

The following numbers are already assigned and are reserved (all values from 0 through 1000 are reserved for standards organizations)

Organization registration code	*Organization*
00000000000	Unknown
00000000001	ISO

Organization registration code	*Organization*
00000000002	IETF

2.2 Categories of file format

This is a 2 digit hexadecimal numeric value giving information on the category of the file. The possible file categories and their numeric assignments are as follows.

2.2.1 archive

A file that is usually compressed and that is usually composed of one or more files. This is the preferred format to use for exchanging files between computers.

In a more general term, an archive can also be defined as being a directory or folder on a computer, since it contains other files.

String id	*Description*
archive/other	A file that is usually compressed and is composed of one or more other files
archive/code	A compressed executable file

2.2.2 audio

Any file that is used exclusively for producing or reproducing sounds.

String id	*Description*
audio/other	Some files that have a mixture of the types below, or that do not fit in any other category.
audio/notes	A file that contains musical notes without any digitized sounds. This is typical for MIDI files.
audio/digitized	A file type that contains digitized sounds, either music or not.

2.2.3 code

This category comprises several sub categories, and represents executable machine code, or code for a specific virtual machine. The values 0x10-0x1F are reserved for this category. The table summarizes the different categories:

String id	*Description*
code/unknown	A piece of code, either linked or not, that does not fit in any other definition or that is not been fully identified.

String id	*Description*
code/object	A piece of code that must be linked to be usable (still contains some static unresolved symbols)
code/executable	A piece of code that can directly be executed by an operating system or a virtual machine
code/dynlib	A piece of code that is loaded and resolved and runtime and which can then be executed by the operating system or the virtual machine.

2.2.4 database

Any type of file that is used to represent some data in a structured fashion. This includes database, and spreadsheet formats.

String id	*Description*
database/unknown	Any data interchange file that contains any type of data
database/financial	A data file that contains information that is related to finance and economics.

2.2.5 file system

Contains a disk image, or part of a disk image.

2.2.6 font

Any type of file that is used to represent a graphical representation of a character or symbol.

2.2.7 image

Any file that is a visual representation with limited or no animation and no audio components.

String id	*Description*
image/other	A file that contains mixed attributes, or that has not been identified, or does not fit in any category.
image/raster	A file that contains pixels to represent an image
image/vector	A file that contains exclusively x,y coordinates for the drawing
image/metafile	A file that contains a programming language to describe the image (the image is composed of commands)

2.2.8 metadata

Any file that is used exclusively as a source of information for other resources.

2.2.9 model

File that represents a 3d model of one or more objects.

2.2.10 palette

Represents resource that contain palette and color mappings.

2.2.11 personal information management

File representing a contact list, a schedule or a personal to do list.

2.2.12 text

Any type of file that is used to represent a text document, such as from a word processor.

2.2.13 video

Any type of file that is used to represent an animation or video, either with or without audio components.

String id	*Description*
video/complete	A file type that can optionally include audio components
video/simple	A file type that only contains animated sequences, without any audio components

2.3 Document identifier

The document identifier is assigned by the registration authority. In the case of ISO, IEC, IETF standards in that case it will contain the document number according to the numeric scheme of those organizations. In other cases, this value contains 0000ZZZ where ZZZ is the uppercase usual file extension for this file format. If the file format does not have any official file extension, then the value is set to 0.

3 Origin of file format

This is a reference to another database that contains information on the last maintainer of the file format. A maintainer does not necessary mean that the file format must still evolve.

The database for the origin of the file format, is actually a registration code for companies. The following fields are valid for the company information database:

Field name	*Description*
Name	Gives the company or organization name.
Postal Address	Postal Address where comments and requests can be sent.
Telephone	Phone number where this company can be contacted.
E-mail	E-mail address where requests can be sent
URI	This is the URI of this organization, company if available

Field name	*Description*
Country Code	2 character country code identifier for this company (ZZ = unknown)
National reg. number	National Company Registration Number (can be left blank), this is a unique code for the company, organization or individual in question, if it has created a corporation, and this value is assigned by the different governments.

3.1 Specifications

This field can contain three types of information, all depending if the software is still being maintained or not, and if the specification is available:

- If the software is still being maintained, and there is an URL to the online specification, it is given here.
- If the software is still being maintained, and the specification is in the software package, then this field is set to “Documentation”
- If the software is still being maintained, and the specification is in the source package, and this source package is available, then this field is set to “Source”.
- If the software is no longer maintained, and/or there is no official specification, this field gives the URI of the local file containing the specification.

The following identifiers are allowed in this section, within the above identifiers:

- If the file produced does not contain any magic identifier, then the value is set to NM (No magic).
- If the file produced requires complex parsing, then the value is set to MP (Manual parsing).

4 Magic file format

The identification is done by using the information specified in the magic database. The database consists of entries which have at least signatures of 3 bytes to avoid false identification and/or duplicate identification.

In the case where a file is identified internally more than once, only the value with the longest signature is kept, the other identities are discarded.

5 Magic file database

For easy maintenance and updates, the file identifier database will be saved into a normal text file. This text file has a specific format, as described in this section. Updates to the database will be available from the company web site, free of charge. Clones of this database are also possible.

5.1 Overview

The database is separated in three distinct parts, the header, the database itself and the footer.

Section of file	*Requirement*
Header	Required
Database	Required
Footer	Optional

5.2 Header

The header consists of the three following lines, encoded in ASCII characters and terminated by newlines (the newline characters can be any of the formats supported by the different operating system:

The 1st line has the following signature (C style representation) and identifies the file (it is enough to verify this field to determine the magic file database):

```
#\ FILE_ID\ DB
```

The 2nd line contains the date when this database was originally updated by the official source:

```
#\ Date:YYYY-MM-DD
```

The 3rd line contains the source URI of this database, and where new updates can be obtained, the Source field can be empty:

```
#\ Source:http://www.magicdb.org
```

The 4th line is a blank line.

5.3 Database

The database contains lines describing magic numbers which identify particular types of files, each line in the database can contain up to 5 fields, and these fields, are used to identify files. Please see the sample databases for more information.

5.3.1 Character encoding of database

1. All fields must be separated with one or more TAB characters.
2. The first field should be on the first column of the line entry, optionally immediately preceded by a continuation character (& or >)
3. Only ASCII characters should be present in the database file
4. All fields which are numeric accept C-styled decimal, hexadecimal and octal values
5. No space characters should occur at end of the line, it should be terminated by a normal <cr>.
6. Lines starting with the # character are considered as comment lines, and are ignored
7. Octal value should start with 0d in lowercase character for the d
8. All dates (in comments or in fields) follow the full ISO 8601 specification (YYYY-MM-DD format)

5.3.2File start

The first line of the textfile containing the magic database can easily be identified

5.3.3 Fields

5.3.3.1 Operation field

Each line either starts with a byte offset in the file to verify, or with an operation field, in the case of no operation character, the line must be processed as usual. Lines beginning with a **>** or **&** character represent continuation lines to a preceding main entry:

➢	If a match is found on a main entry line, these additional pattern are checked. Any pattern which matches will be used. This may generate additional output; a single blank space separates each line's output (if any output exists for that line).
&	If a match is found on a main entry line, and a following continuation lines begins with this character, that continuation line pattern must also match, or neither line will be used. Output text associated with any line beginning with the & character is ignored.

5.3.3.2 Byte offset in file

The first field is a byte offset in the file; both direct file offsets and indirect file offsets are supported.

5.3.3.2.1 Direct offsets

This type of offset indicates the offset from the start of the file. If the field is preceded by a Z character, it is an offset from the end of the file instead.

Examples:

0x7F 012 255 Z0x7F

5.3.3.2.2 Indirect offsets

An indirect offset has the format `((x.[bslBSL]][+-][y])`, where x is where to read the offset in the file, the format to read being specified by one of the character specifiers.

b, s, l = little endian byte, short or long value

B, S, L = big endian byte, short or long value

The value is read, and this is added to y, and this gives the value to seek to in the file.

5.3.3.3 Type field

The next field is a type field, which indicates a type of the operand to read from the file. The following types are defined:

`byte` :	8-bit unsigned value
`string`:	an array of bytes
`leshort` :	16-bit unsigned little-endian value
`lelong` :	32-bit unsigned little-endian value
`beshort` :	16-bit unsigned big-endian value
`belong` :	32-bit unsigned big-endian value
`string/c` :	case insensitive array of bytes (comparison on ASCII characters is not case sensitive).
`pstring:`	pascal style string, where the first byte contains the length of the following characters. Case sensitive.

All these values (except for strimgs) can be followed by an optional mask (using &) which is bitwise ANDed to the value prior to the comparison.

Example : byte&0x80

5.3.3.4 Value

The next field is a value , preceded by an optional operator. Operators only apply to non-string types. The default operator is = (exact match). The other operators are:

=	equal
!	not equal
>	greater the
<	less than
&	all bits n pattern much match
^	any bits in pattern may match
x	any value matches

If the type specifier is a string or istring, the following additional rules apply:

1. The `x` and > operators can be used only in the case where additional information is requested to be printed. (if the line starts with >)
2. C styled escape sequences are accepted. All control codes should be written using C styled escape sequences. Examples :

   The Tab character : \t

   Furthermore these control sequences must be written using C-styled escape sequences:

   The space character : \

   The < character : \<

5.3.3.5 Comment and identifier field

The rest of the line is the identifier of the file, as well as the extracted attributes of the file. The format of the comment line is as follows:

```
[ "[" special directives "]" ] comment string
```

special directives is optional, and give specific information on the file format, the directives must start before the main comment line, and consists of bracket [ ] characters. The text in brackets consists of an identifier (the identifiers are case sensitive), followed by an equal sign and the actual value. Each identifier value pair is separated from others by a `;` character.

`ext`=<file extesion> : This gives one or more file extensions that this type of file usually has, each file extension separated by the others by a comma.

`fid`=<unique file identifier> : This gives this file format its unique identifier

`title =` <title>: This gives the title of this file.

`creator` = <creator>: This gives the creator of this file

`chn` = <channels> : This gives the number of channels of the audio file

`freq` = <frequency> : Sampling rate in Hz.

`res` = <image attributes>: If available is in the string format: XxYxZbpp

`mime` = <mime signatures>: If available, gives the mime signature for this file type. Only MIME signatures that are registered with IANA are put in this field.

Anything which is not between the [] characters is considered to be the comment of the file, this is used when there is a file match. Note that the contents of this field will be ignored if the line begins with the & continuation character. This field accepts also the following printf() format specifiers to print the magic number , and cannot contain any [and] characters:

`%s` = prints out the magic number as a string value, in the case where the operator is x , then up to 255 characters can be printed or up to a null character.

`%.xs` = where x = numeric value (0-99), prints this number of characters

`%d` = prints out the magic number as a numeric value

`%bh` = prints out the high byte of a BCD coded byte

`%bl` = prints out the low byte of a BCD coded byte

The printf() specification in the next requires a length argument on the string, the string will be read for 32 characters, and then either the full string with the length specified (if specified) will be printed or until a NULL character is discovered.

Depending on the file category, and on the possibility of extracting the information, the format of the comment string returned should be as follows:

[comment string], <file version>

[comment string] : Required field, the actual file format name.

<file version> : If available is in the string format : version %d.%d

<code attributes>: If available is in the string format: w/Line info, w/Symbol info

5.4 Footer

The footer is optional, and if present, should be on the last line of the file. It contains a CRC32 of the data file, excluding the header or the footer.

The format of the footer is as follows:

```
<blank line>
#\ CRC32:NNNNNNNN
```

where NNNNNNNN is an ASCII hexadecimal representation of the CRC32 value.

6 Appendix A: Table of file format codes

Range (Hex.)	***File type***
0x00-0x0D	*unassigned*
0x0E	palette
0x0F	filesystem
0x10-0x1F	code
0x20-0x2F	audio
0x30-0x3F	image
0x40-0x4F	model
0x50-0x5F	archive
0x60-0x6F	text
0x70-0x7F	font
0x80-0x8F	video
0x90-0x9F	database
0xA0-0xAF	metadata
0xB0-0xBF	pim
0xC0-0xDF	*unassigned*
0xF0-0xFF	generic

Format type	***ID (hex.)***
archive/code	0x51
archive/unknown	0x50
audio/music	0x21
audio/unknown	0x20
audio/sampled	0x22
code/dynlib	0x13
code/executable	0x12
code/object	0x11
code/unknown	0x10
database/unknown	0x90
database/financial	0x91
filesystem/unknown	0x0F
font/unknown	0x70

Format type	***ID (hex.)***
`generic/iff`	`0xF0`
`generic/riff`	`0xF1`
`generic/hdf`	`0xF2`
`generic/xml`	`0xF3`
`generic/ole`	`0xF4`
`generic/bento`	`0xF5`
`generic/pico`	`0xF6`
`generic/ddf`	`0xF7`
`generic/efs`	`0xF8`
`image/metafile`	`0x33`
`image/raster`	`0x31`
`image/vector`	`0x32`
`image/unknown`	`0x30`
`metadata/unknown`	`0xA0`
`model/unknown`	`0x40`
`palette/unknown`	`0x0E`
`pim/calendar`	`0xB1`
`pim/contacts`	`0xB2`
`pim/unknown`	`0xB0`
`text/unknown`	`0x60`
`video/unknown`	`0x80`
`video/complete`	`0x81`
`video/simple`	`0x82`