Dissecting media file formats with Kaitai Struct

FOSDEM 2017

Mikhail Yakshin (GreyCat)
Kaitai Project

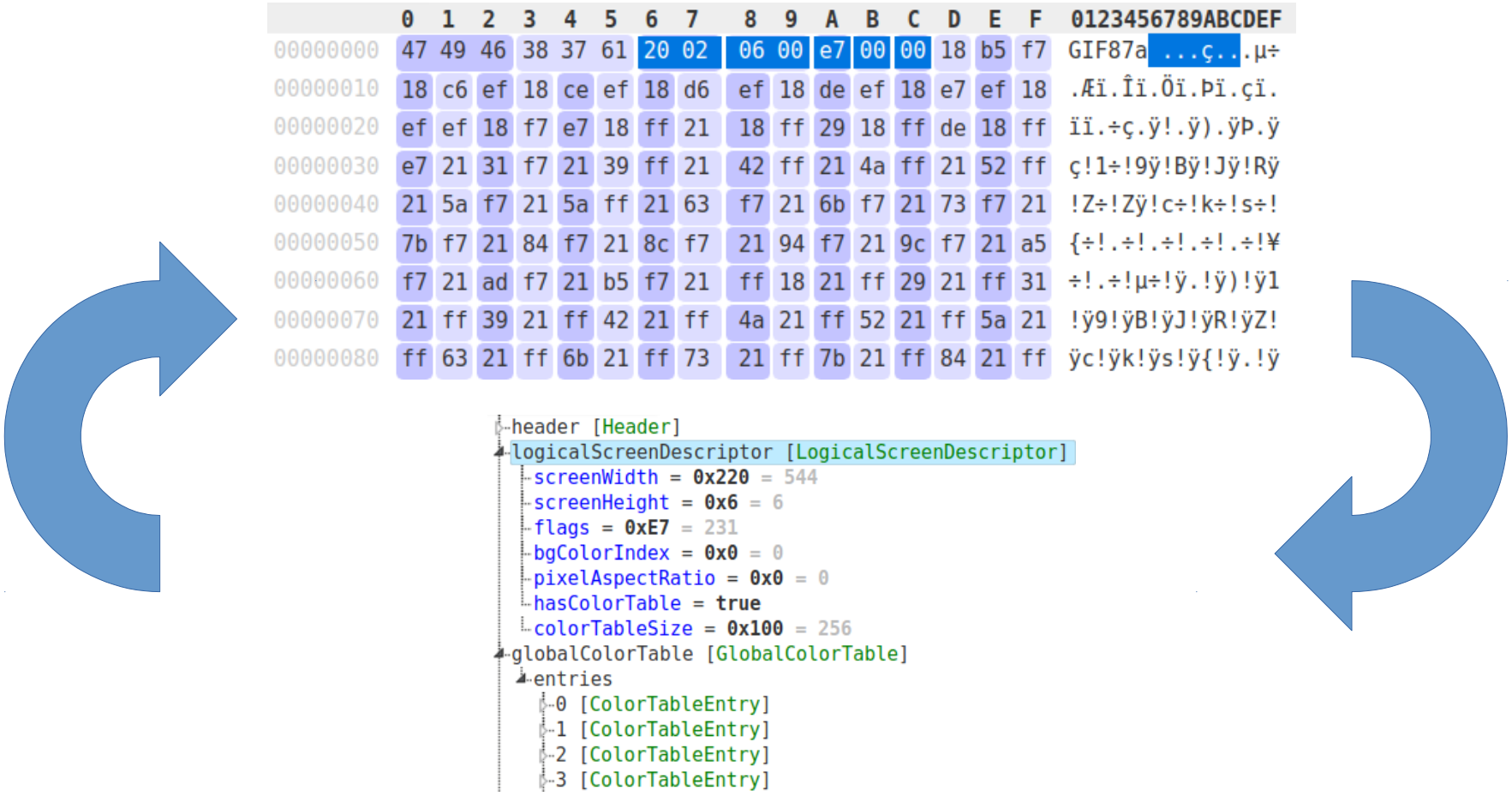
http://kaitai.io/

Twitter: @kaitai_io

File formats: a problem?

- Media software developers have to deal with multitude of different media file formats
- Some of them are proprietary and undocumented → need to be reverse engineered
- Some of them are documented, but still parsing binary files is pain

The mission:
from stream to memory (and back)



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	0123456789ABCDEF
00000000	47	49	46	38	37	61	20	02	06	00	e7	00	00	18	b5	f7	GIF87a ...ç..µ÷
00000010	18	c6	ef	18	ce	ef	18	d6	ef	18	de	ef	18	e7	ef	18	.Æï.Îï.Öï.Þï.çï.
00000020	ef	ef	18	f7	e7	18	ff	21	18	ff	29	18	ff	de	18	ff	ïï.÷ç.ÿ!.ÿ).ÿÞ.ÿ
00000030	e7	21	31	f7	21	39	ff	21	42	ff	21	4a	ff	21	52	ff	ç!1÷!9ÿ!Bÿ!Jÿ!Rÿ
00000040	21	5a	f7	21	5a	ff	21	63	f7	21	6b	f7	21	73	f7	21	!Z÷!Zÿ!c÷!k÷!s÷!
00000050	7b	f7	21	84	f7	21	8c	f7	21	94	f7	21	9c	f7	21	a5	{÷!.÷!.÷!.÷!.÷!¥
00000060	f7	21	ad	f7	21	b5	f7	21	ff	18	21	ff	29	21	ff	31	÷!.÷!µ÷!ÿ.!ÿ)!ÿ1
00000070	21	ff	39	21	ff	42	21	ff	4a	21	ff	52	21	ff	5a	21	!ÿ9!ÿB!ÿJ!ÿR!ÿZ!
00000080	ff	63	21	ff	6b	21	ff	73	21	ff	7b	21	ff	84	21	ff	ÿc!ÿk!ÿs!ÿ{!ÿ.!ÿ

- header [Header]
- logicalScreenDescriptor [LogicalScreenDescriptor]
  - screenWidth = **0x220** = 544
  - screenHeight = **0x6** = 6
  - flags = **0xE7** = 231
  - bgColorIndex = **0x0** = 0
  - pixelAspectRatio = **0x0** = 0
  - hasColorTable = **true**
  - colorTableSize = **0x100** = 256
- globalColorTable [GlobalColorTable]
  - entries
    - 0 [ColorTableEntry]
    - 1 [ColorTableEntry]
    - 2 [ColorTableEntry]
    - 3 [ColorTableEntry]

Typical development workflow

- Write some parsing code in a certain programming language
- Write some extra debugging code (dump to screen, check assertions, etc)
- Debug it till you drop
  - with dumping
  - with debugger
  - with asserts, etc
- Want to support some other programming language? Redo from start.

Almost every media format library has these “dumping” tools

- libpng (PNG)
  - pnginfo, pngcp, pngchunkdesc, pngchunks
- openjpeg2 (JPEG 2000)
  - opj_decompress, opj_compress, opj_dump
- libogg (Ogg)
  - ogginfo
- swftools (Adobe Flash)
  - swfdump, swfextract, swfrender, ...

Errors in file format libraries are devastatingly dangerous

- Almost always remotely exploitable, frequently provide arbitrary code execution, information leaking, DoS
- libpng: since 2010:
  - 22 vulnerabilities
  - 15 DoS
  - 13 overflow / code execution
- libjpeg
  - 4 vulnerabilities
  - 3 infoleaks
  - 1 code execution

File formats description: no single standard

ELF Header

Some object f le control structures can grow, because the ELF header contains their actual sizes. If the object f le format changes, a program may encounter control structures that are larger or smaller than expected. Programs might therefore ignore ‘‘extra’’ information. The treatment of ‘‘missing’’ information depends on context and will be specif ed when and if extensions are def ned.

Figure 1-3: ELF Header

```
#define EI_NIDENT       16

typedef struct {
        unsigned char   e_ident[EI_NIDENT];
        Elf32_Half      e_type;
        Elf32_Half      e_machine;
        Elf32_Word      e_version;
        Elf32_Addr      e_entry;
        Elf32_Off       e_phoff;
        Elf32_Off       e_shoff;
        Elf32_Word      e_flags;
        Elf32_Half      e_ehsize;
        Elf32_Half      e_phentsize;
        Elf32_Half      e_phnum;
        Elf32_Half      e_shentsize;
        Elf32_Half      e_shnum;
        Elf32_Half      e_shstrndx;
} Elf32_Ehdr;
```

`e_ident` The initial bytes mark the f le as an object f le and provide machine-independent data with which to decode and interpret the f le’s contents. Complete descriptions appear below, in ‘‘ELF Identif cation.’’

`e_type` This member identif es the object f le type.

Name	Value	Meaning
ET_NONE	0	No f le type
ET_REL	1	Relocatable f le

File formats description: no single standard

: 768 J. Postel
ISI
28 August 1980

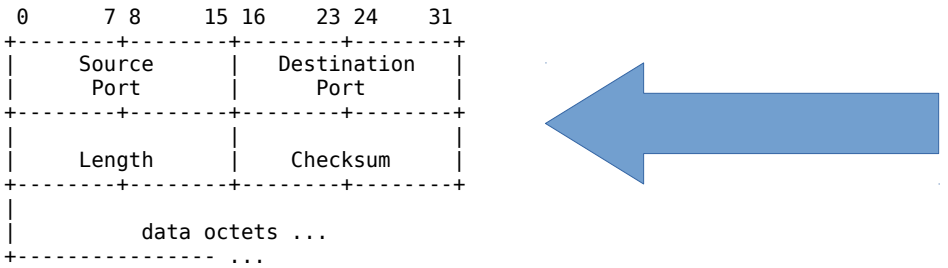
User Datagram Protocol

:roduction

.s User Datagram Protocol (UDP) is defined to make available a :agram mode of packet-switched computer communication in the ‘ironment of an interconnected set of computer networks. This ›tocol assumes that the Internet Protocol (IP) [1] is used as the lerlying protocol.

.s protocol provides a procedure for application programs to senc ;sages to other programs with a minimum of protocol mechanism. The ›tocol is transaction oriented, and delivery and duplicate protectior : not guaranteed. Applications requiring ordered reliable delivery of ‘eams of data should use the Transmission Control Protocol (TCP) [2].

‘mat



User Datagram Header Format

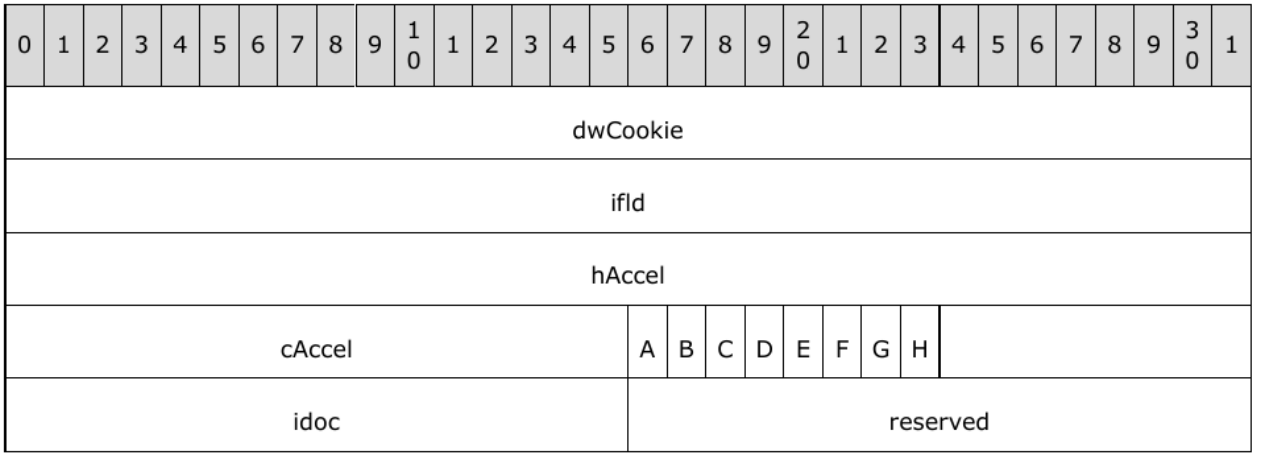
›lds

ırce Port is an optional field, when meaningful, it indicates the port the sending process, and may be assumed to be the port to which a ›ly should be addressed in the absence of any other information. If : used, a value of zero is inserted.

File formats description: no single standard

2.9.161 OcxInfo

The **OcxInfo** structure specifies an **OLE control** (such as a checkbox, radio button, and so on) in the document. The data that is contained in **OcxInfo** structures SHOULD<229> be ignored.



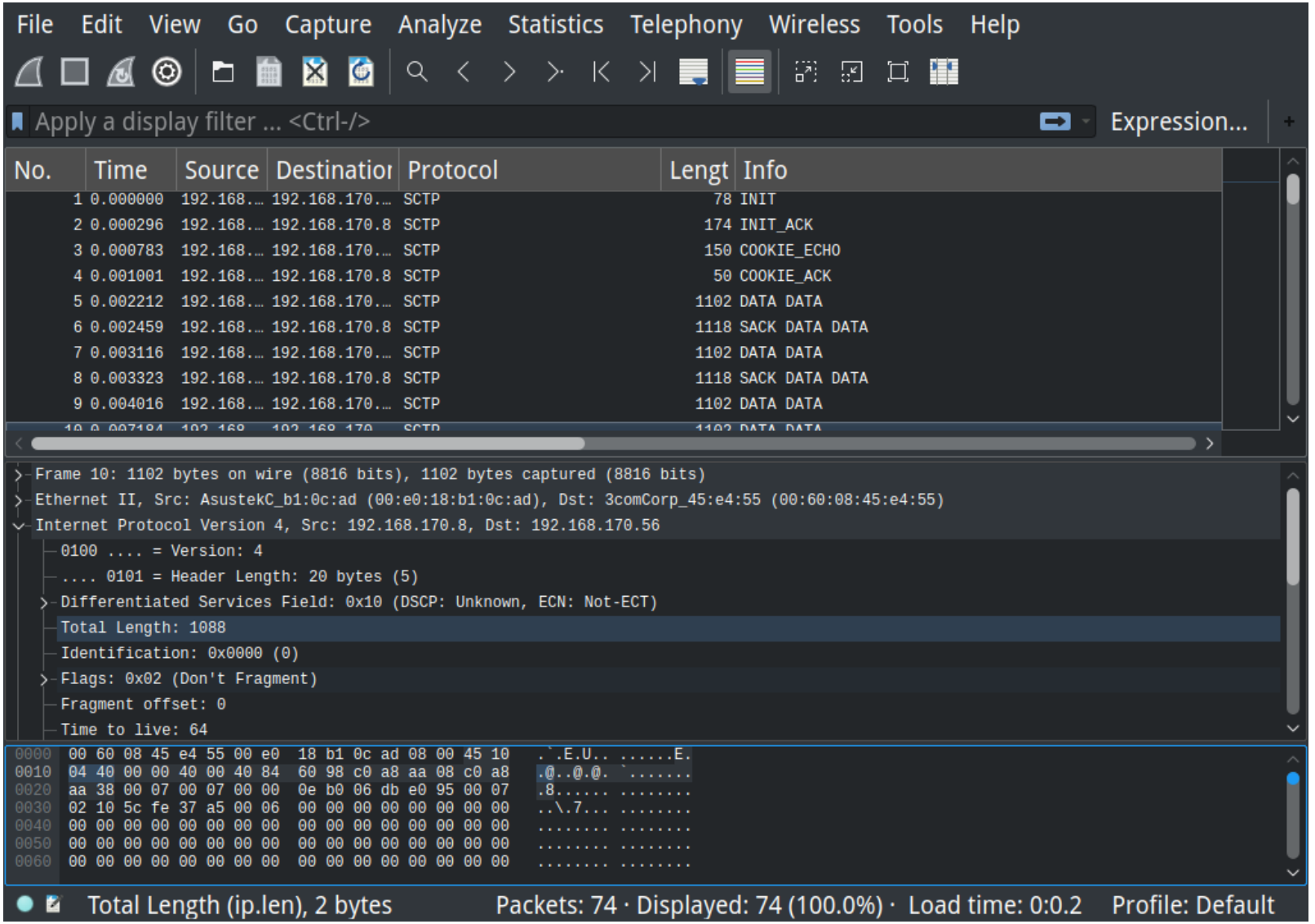
dwCookie (4 bytes): An integer value that specifies the index location of this **OcxInfo** in the **RgxOcxInfo** array. This value MUST be unique for all **OcxInfo** structures in the document.

ifld (4 bytes): An unsigned integer value that specifies an index location in the **PlcFld** structure. The value MUST be a valid **FLD** index in the correct **PlcFld** structure.

The PlcFld that is used is dependent on the value of **idoc**, as specified following.

Value	Location
1	The Main Document (FibRgFcLcb97.**fcPlcfFldMom**).
2	The Header Document (FibRgFcLcb97.**fcPlcfFldHdr**).
3	The Footnote Document (FibRgFcLcb97.**fcPlcfFldFtn**).
4	The Textbox Document (FibRgFcLcb97.**fcPlcfFldTxbx**).
6	The Endnote Document (FibRgFcLcb97.**fcPlcfFldEdn**).
7	The Comment Document (FibRgFcLcb97.**fcPlcfFldAtn**).
8	The Header Textbox Document (FibRgFcLcb97.**fcPlcfHdrtxbxTxt**).

Debugging networking protocols: they've got Wireshark



Enter Kaitai Struct

- Declarative file format specification language (.ksy)
- Compiles into ready-made parsers in many supported target programming languages
- Visualization, dumping and debugging tools
- .ksy is YAML-based → easy to write your own tools
- Free & libre:
  - GPLv3 for compiler
  - MIT/Apache2 for runtime

Supported target languages

- C++ (STL)
- C#
- Java
- JavaScript
- Perl
- PHP
- Python
- Ruby

Bonus: GraphViz support

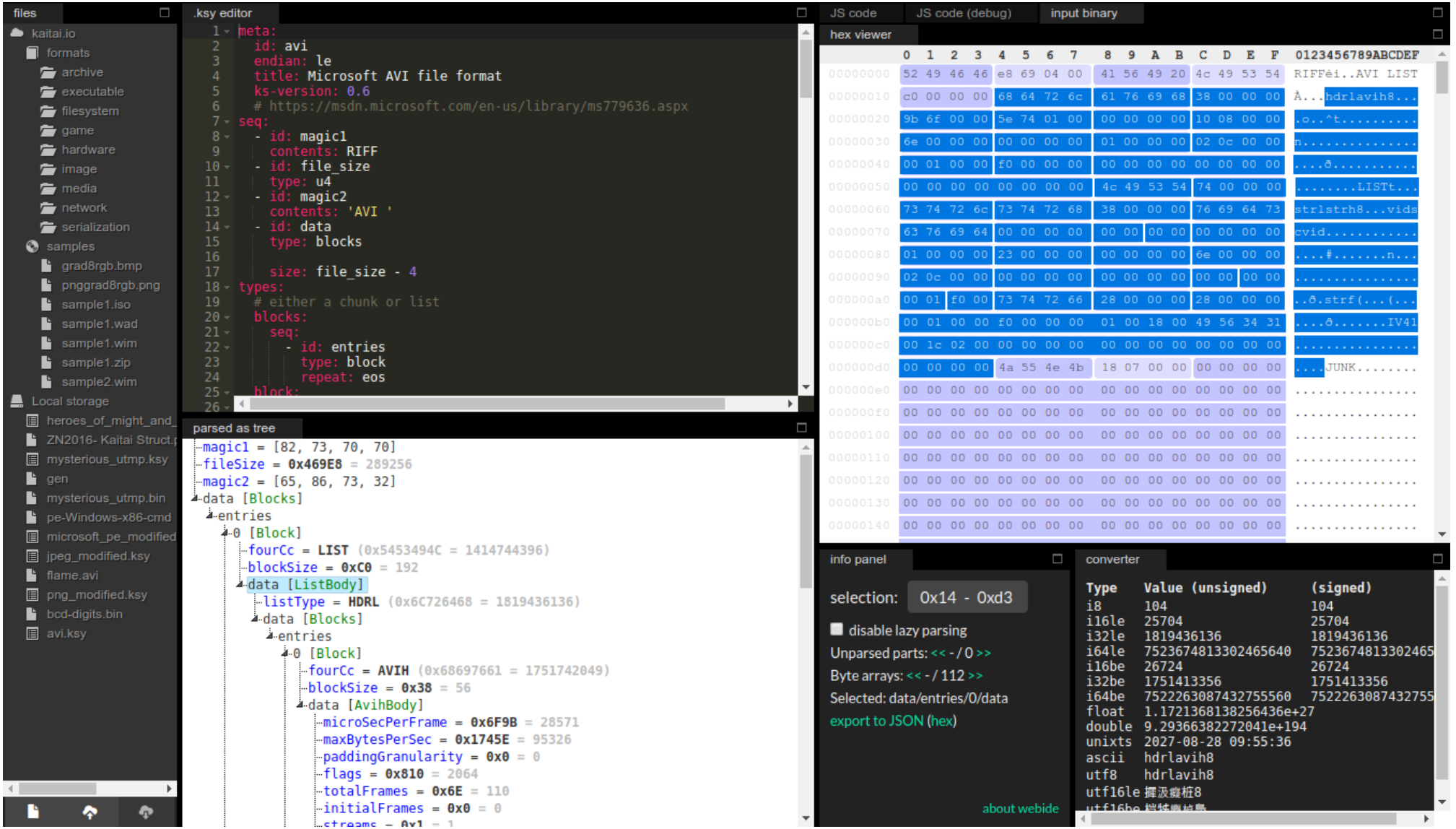
Natural API generated by KS

```
header [Header]
logicalScreenDescriptor [LogicalScreenDescriptor]
  screenWidth = 0x220 = 544
  screenHeight = 0x6 = 6
  flags = 0xE7 = 231
  bgColorIndex = 0x0 = 0
  pixelAspectRatio = 0x0 = 0
  hasColorTable = true
  colorTableSize = 0x100 = 256
globalColorTable [GlobalColorTable]
  entries
    0 [ColorTableEntry]
    1 [ColorTableEntry]
    2 [ColorTableEntry]
    3 [ColorTableEntry]
```

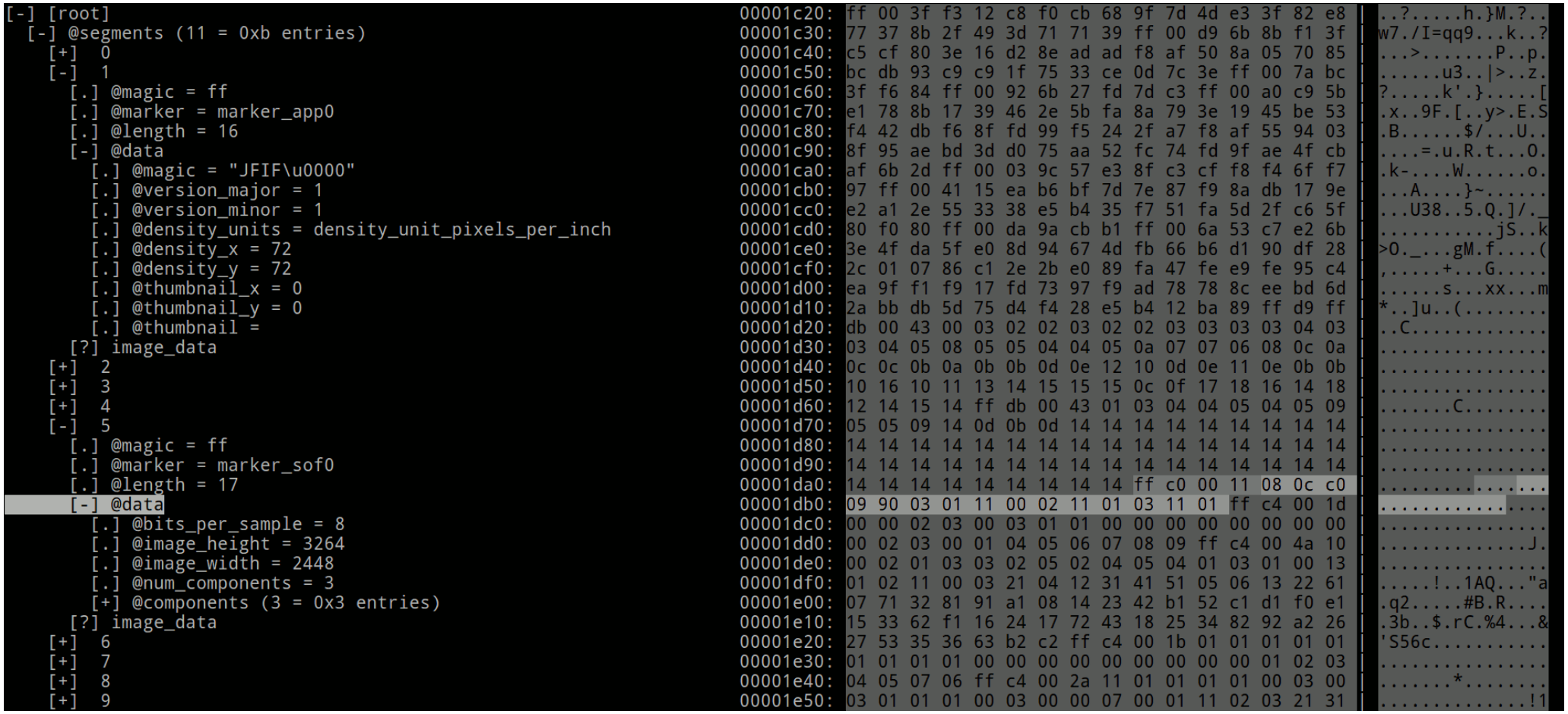
```java
// Java

Gif myFile = Gif.fromFile("/path/to/file.gif");
System.out.println(
    "This file is " +
    myFile.logicalScreenDescriptor.screenWidth +
    " x " +
    myFile.logicalScreenDescriptor.screenHeight
);
```

A picture worth a thousand words: Web IDE



Console visualizer: JPEG



Declarative, not imperative

```
seq:
  - id: title_len
    type: u4
  - id: title
    type: str
    size: title_len
    encoding: UTF-8
  - id: num_elements
    type: u2
  - id: elements
    type: u4
    repeat: eos
```

```
title_len = io.read_uint32();
title = io.read_utf8_string(title_len);
num_elements = io.read_uint16();

elements = new ArrayList();
while (!io.eof()) {
    int tmp = io.read_uint32();
    elements.add(tmp);
}
```

Kaitai Struct data types

- Built-in data types:
  - Integers
  - Floats
  - Unaligned bit integers and bit fields (0.6+)
  - Strings: fixed size, terminator-delimited, up to end of stream
  - Raw byte arrays
  - Enums
- User-defined data types

Kaitia Struct features

- Sequential parsing (“seq”)
- Out-of-order parsing (“instances”)
- Calculated attributes
- Checking for magic signatures (fixed content)
- Conditional parsing (“if”)
- Type switching on a condition (“switch”)
- Repetitions:
  - until the end of stream
  - predefined number of iterations
  - until a condition is met

Expression language to C++

```
seq:
  - id: full_len
    type: u4
  - id: elements
    type: element
    repeat: expr
    repeat-expr: (full_len - 4) / 6
```

```
m_full_len = m__io->read_u4le();
int l_elements = (full_len() - 4) / 6;
m_elements = new std::vector();
m_elements->reserve(l_elements);
for (int i = 0; i < l_elements; i++) {
    m_elements->push_back(new element_t(m__io));
}
```

Expression language to Python

```
seq:
  - id: full_len
    type: u4
  - id: elements
    type: element
    repeat: expr
    repeat-expr: (full_len - 4) / 6
```

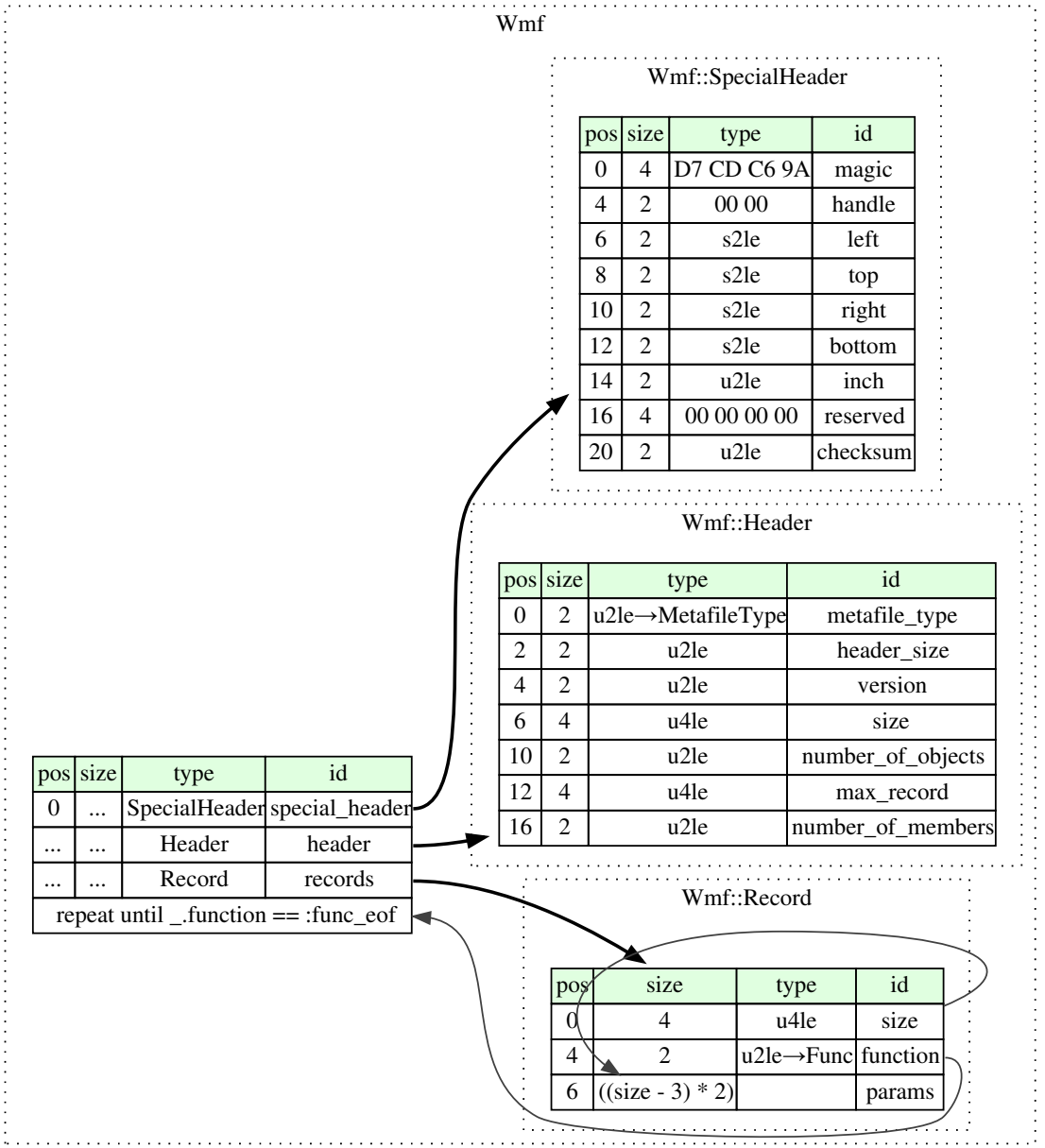
```
self.full_len = self._io.read_u4le()
self.elements = [None] * ((self.full_len - 4) // 6)
for i in range((self.full_len - 4) // 6):
    self.elements[i] = Element(self._io)
```

Expression language to JavaScript

```
seq:
  - id: full_len
    type: u4
  - id: elements
    type: element
    repeat: expr
    repeat-expr: (full_len - 4) / 6
```

```
this.fullLen = this._io.readU4le();
this.elements = new Array(Math.floor((this.fullLen - 4) / 6));
for (var i = 0; i < Math.floor((this.fullLen - 4) / 6); i++) {
  this.elements[i] = new Element(this._io);
}
```

GraphViz visualization: WMF



Is it production-ready?

- We've got a growing repository of formats
- Image files: bmp, cr2, exif, gif, jpeg, pcx, png, tiff, tim (PlayStation), wmf, xwd
- Video files: Microsoft AVI (.avi), QuickTime .mov / MP4 / ISO/IEC 14496-14:2003
- Audio files: Standard MIDI (.mid), RIFF (.wav), ID3 tags, Amiga .mod modules
- More media files: Blender's .blend, 3D Systems Stereolithography (.stl)

And more...

- Archives: .lzh, .zip
- Documents: Microsoft's Compound File Binary (CFB, AKA OLE)
- Executables: DOS MZ, Windows PE, ELF, Mach-O, Python bytecode (.pyc), Java classes (.class), Adobe Flash (.swf)
- Filesystems: cramfs, ext2, iso9660, MBR partition tables, VirtualBox disk images (.vdi)
- Networking

Thanks for your attention!
Questions?

http://kaitai.io/

GitHub: http://github.com/kaitai-io/kaitai_struct/
Twitter: @kaitai_io
Gitter: https://gitter.im/kaitai_struct/