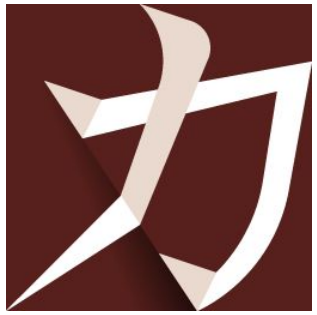


Kaitai Struct

a tool for dealing with binary formats

Petr Pucil

Mikhail Yakshin

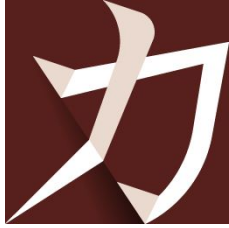
About us

Petr Pucil

- admin of Kaitai Struct since 2020
- Master’s degree in Security and Network Engineering, University of Amsterdam (2025)

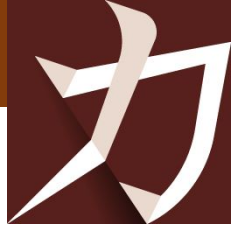
Mikhail Yakshin

- original creator of Kaitai Struct (2014)
- principal software engineering manager in Observability in Microsoft Ireland Research

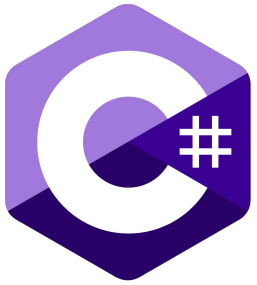


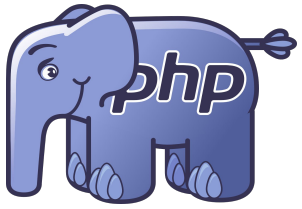
What is Kaitai Struct?

What is Kaitai Struct?



- tool for dealing with binary formats
- declarative language (.ksy) for specifying arbitrary binary formats
- parser generator for 12 programming languages
- serializer generator for 2 programming languages

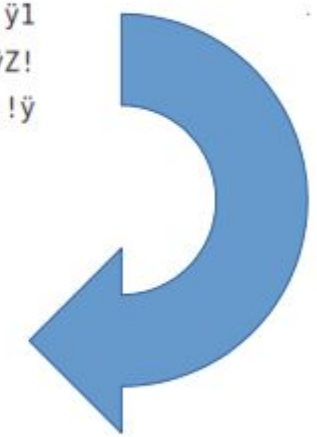



What is Kaitai Struct?

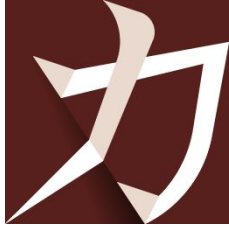
serialization
(Java and
Python only)

	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]
```

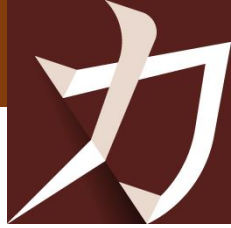


parsing



What can you use Kaitai Struct for?

What can you use Kaitai Struct for?



- Reverse engineering unknown binary file formats or protocols
  - Iterative process, try and fail, rinse and repeat
  - Once it scales, you want to automate it
- Creation of separate reference parser (to challenge and test correctness of existing implementation)
  - Existing library: hand-written? generated from spec?
  - Safe implementation
  - Guidance for fuzzing
- Porting specifications of formats between languages
  - Got support in Python → want support in JavaScript, etc.

Kaitai workflow

Kaitai workflow

1. Compilation

hello_world.**ksy**

```
meta:
  id: hello_world
seq:
  - id: one
    type: u1
```

kaitai-struct-compiler

hello_world.**py**

```
class HelloWorld(KaitaiStruct):
    # ...
    def _read(self):
        self.one = self._io.read_u1()
```

2. Parsing

input binary file

sample.bin

```
          0  1  2  3  4  5  6  7   01234567
00000000  ff 01                    ÿ.
```

hello_world.**py**

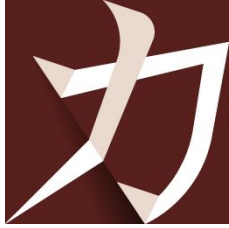
```
class HelloWorld(KaitaiStruct):
    # ...
    def _read(self):
        self.one = self._io.read_u1()
```

parsed data

```
└─ one = 0xFF = 255
```

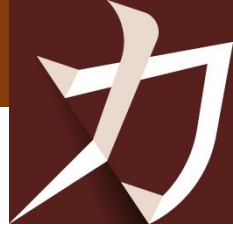
kaitaistruct.py (runtime library)

```
class KaitaiStream:
    # ...
    def read_u1(self):
        return struct.unpack('B', self.read_bytes(1))[0]
```

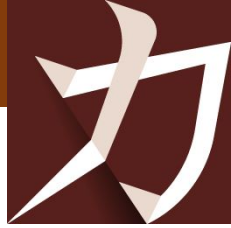


Why Kaitai

Why Kaitai



- write once, use everywhere
- standard way to describe binary formats
- library of format specifications
- GraphViz diagrams
- visualization and dumping tools
  - **Web IDE**
  - console visualizer (ksv)
  - ksdump



→ write once, use everywhere

1 .ksy spec = 12 parsers

.ksy spec

```
meta:
  id: hello_world
seq:
  - id: one
    type: u1
```

```
public class HelloWorld extends KaitaiStruct {
    // ...
    private void _read() {
        this.one = this._io.readU1();
    }
```

Java

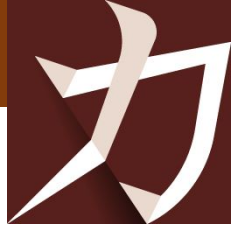
```
class HelloWorld(KaitaiStruct):
    # ...
    def _read(self):
        self.one = self._io.read_u1()
```

Python

```
class HelloWorld < Kaitai::Struct::Struct
  # ...
  def _read
    @one = @_io.read_u1
  end
```

Ruby

and others (C++, C#, Go, JavaScript, Lua, Nim, Perl, PHP…)



→ standard way to describe binary formats

no single standard

GIF

18. Logical Screen Descriptor.

a. Description. The Logical Screen Descriptor contains the parameters necessary to define the area of the display device within which the images will be rendered. The coordinates in this block are given with respect to the top-left corner of the virtual screen; they do not necessarily refer to absolute coordinates on the display device. This implies that they could refer to window coordinates in a window-based environment or printer coordinates when a printer is used.

This block is REQUIRED; exactly one Logical Screen Descriptor must be present per Data Stream.

b. Required Version. Not applicable. This block is not subject to a version number. This block must appear immediately after the Header.

c. Syntax.

	7 6 5 4 3 2 1 0	Field Name	Type
0	| |	Logical Screen Width	Unsigned
1	| |		
2	| |	Logical Screen Height	Unsigned
3	| |		
4	| | | | |	<Packed Fields>	See below
5	| |	Background Color Index	Byte
6	| |	Pixel Aspect Ratio	Byte

Microsoft Word .doc

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.

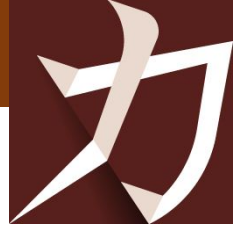
0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
dwCookie																															
ifld																															
hAccel																															
cAccel											A	B	C	D	E	F	G	H													
idoc											reserved																				

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**).



→ library of .ksy format specifications

185 specifications

formats.kaitai.io

3D Models

gltf_binary , quake_mdl

Archive Files

android_bootldr_asus , android_bootldr_huawei , android_bootldr_qcom , android_dto , android_img , android_sparse , chrome_pak , cpio_old_le , gzip , lzh , mozilla_mar , phar_without_stub , rar , rpm , xar , zip , zisofs

Commonly Used Data Types

bcd , bytes_with_io , dos_datetime , riff , utf8_string , vlq_base128_be , vlq_base128_le

DOS-specific

dos_datetime , dos_mz , mbr_partition_table , vfat

Filesystems

android_super , apm_partition_table , apple_single_double , btrfs_stream , cramfs , ext2 , gpt_partition_table , iso9660 , luks , lvm2 , mbr_partition_table , tr_dos_image , vdi , vfat , vmware_vmdk , zisofs , zx_spectrum_tap

Fonts

grub2_font , pcf_font , ttf

Android-specific

android_bootldr_asus , android_bootldr_huawei , android_bootldr_qcom , android_img , android_nanoapp_header , android_opengl_shaders_cache , android_sparse , android_super , dex

CAD

monomakh_sapr_chg

Databases

dbf , gettext_mo , sqlite3 , tsm

Executables and Byte-code

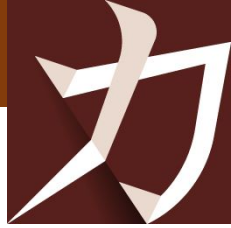
android_nanoapp_header , dex , dos_mz , elf , java_class , mach_o , mach_o_fat , microsoft_pe , python_pyc_27 , swf , uefi_te

Firmware

andes_firmware , broadcom_trx , ines , uefi_te , uimage

Game Data Files

allegro_dat , doom_wad , dune_2_pak , fallout2_dat , fallout_dat , ftl_dat , gran_turismo_vol , heaps_pak , heroes_of_might_and_magic_agg , heroes_of_might_and_magic_bmp , minecraft_nbt , quake_mdl , quake_pak , renderware_binary_stream , saints_row_2_vpp_pc , warcraft_2_pud



➔ library of .ksy format specifications

185 specifications

formats.kaitai.io

Geospatial (Maps)
shapefile_index , shapefile_main

Image Files
bmp , dicom , exif , gif , gimp_brush , icc_4 , ico , jpeg , nitf , pcx , pcx_dcx , png , psx_tim , tga , wmf , xwd

Logs
aix_utmp , glibc_utmp , hashcat_restore , mcap , sudoers_ts , systemd_journal , windows_evt_log

macOS-specific
apm_partition_table , apple_single_double , compressed_resource , dcmp_0 , dcmp_1 , dcmp_2 , dcmp_variable_length_integer , ds_store , mac_os_resource_snd , resource_fork

Networking Protocols
bitcoin_transaction , dime_message , dns_packet , ethernet_frame , hccap , hccapx , icmp_packet , ipv4_packet , ipv6_packet , microsoft_network_monitor_v2 , packet_ppi , pcap , protocol_body , rtcp_payload , rtp_packet , rtpdump , some_ip , some_ip_container , some_ip_sd , some_ip_sd_entries , some_ip_sd_options , tcp_segment , tls_client_hello , udp_datagram , websocket

Security
efivar_signature_list , openpgp_message , ssh_public_key

Windows-specific
avi , bmp , ico , microsoft_pe , regf , wav , windows_evt_log , windows_lnk_file , windows_minidump , windows_resource_file , windows_shell_items , windows_systemtime , wmf

Hardware Protocols
dtb , edid , mifare_classic

GNU/Linux-specific
btrfs_stream , cramfs , dtb , elf , ext2 , gettext_mo , glibc_utmp , luks , lvm2 , sudoers_ts , systemd_journal

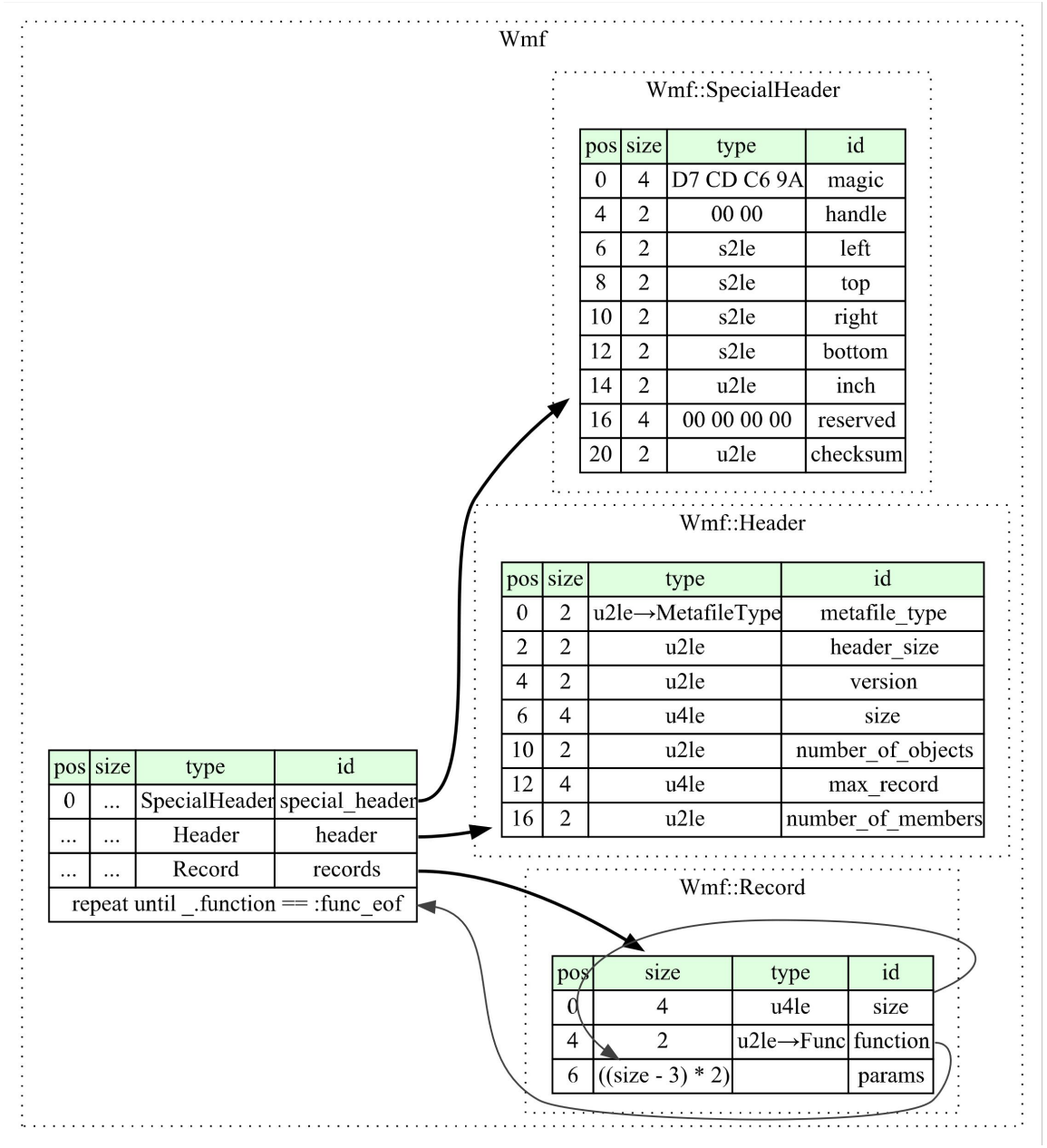
CPU / Machine Code Disassembly
code_6502

Multimedia Files
android_opengl_shaders_cache , au , avi , blender_blend , creative_voice_file , fasttracker_xm_module , genmidi_op2 , id3v1_1 , id3v2_3 , id3v2_4 , magicavoxel_vox , ogg , quicktime_mov , s3m , standard_midi_file , stl , swf , vp8_ivf , wav

Scientific Applications
avantes_roh60 , nt_mdt , nt_mdt_pal , specpr

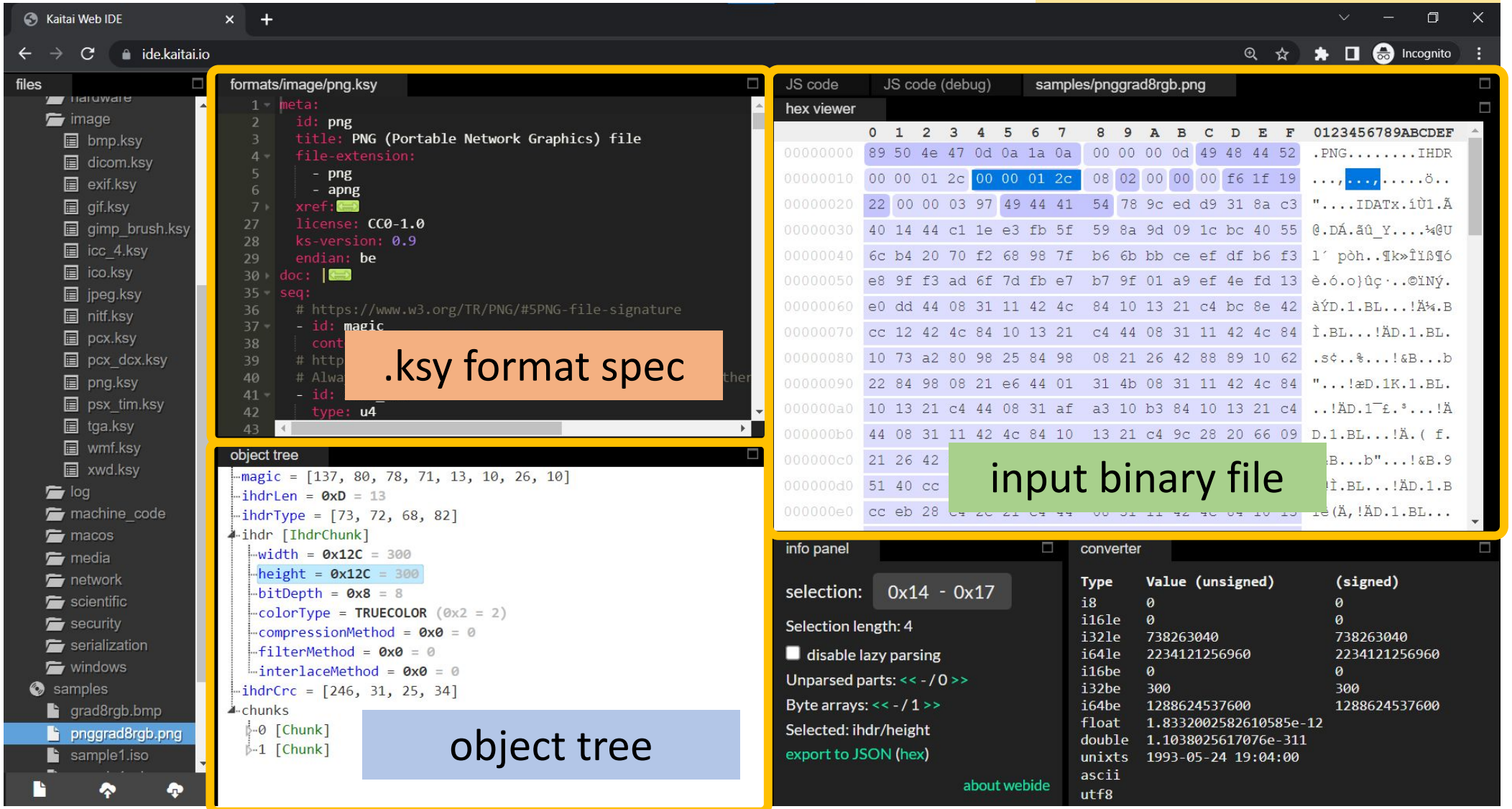
Serialization Protocols
asn1_der , bson , chrome_pak , dtb , google_protobuf , microsoft_cfb , minecraft_nbt , msgpack , php_serialized_value , python_pickle , ruby_marshal

→ GraphViz diagrams



→ visualization and dumping tools

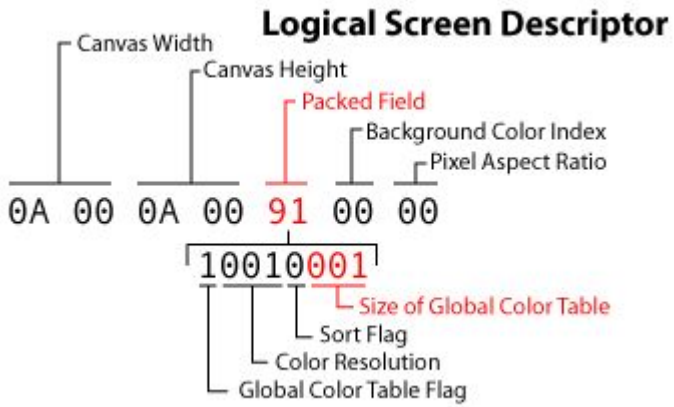
Web IDE (ide.kaitai.io)

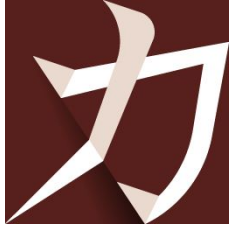


GIF format structure

```
     0001 0203 0405 0607 0809 0A0B 0C0D 0E0F  0123456789ABCDEF
00   4749 4638 3961 0100 0100 8000 00FF 8000  GIF89a....€..ÿ€.
10   FFFF FF2C 0000 0000 0100 0100 0002 0244  ÿÿÿ,...........D
20   0100 3B                                  ..;
```

	Offset (hex)	Byte sequence (hex)	Meaning
Header	00	47 49 46	ASCII string ‘GIF’ – “magic number” (GIF signature)
	03	38 39 61	ASCII string ‘89a’ – GIF version
Logical Screen Descriptor	06	01 00	Logical screen width: 1 pixel
	08	01 00	Logical screen height: 1 pixel
	0A	80	Packed field: enable Global Color Table
	0B	00	Background color index
	0C	00	Pixel aspect ratio
Global Color Table	0D	FF 80 00	First color in the palette: orange (#ff8000)
	10	FF FF FF	Second color in the palette: white (#ffffff)





Thanks!

https://kaitai.io/

https://github.com/kaitai-io

https://gitter.im/kaitai_struct/Lobby

@kaitai_io