

0. Install Kaitai Struct and dependencies for your programming language

«Kaitai Struct is a domain-specific language (DSL) that is designed with one particular task in mind: dealing with arbitrary binary formats... It's a **YAML-based format**, plain and simple.»

Kaitai Struct Cheatsheet



1. Define file structure in Kaitai Struct language

```
meta:
  id: gif
  file-extension: gif
  endian: le
seq:
  - id: header
    type: header
    doc: This is docstring
  - id: logical_screen
    type: logical_screen
types:
  header:
    seq:
      - id: magic
        contents: 'GIF'
      - id: version
        size: 3
  logical_screen:
    seq:
      - id: image_width
        type: u2
      - id: image_height
        type: u2
```

gif.ksy

Specifies the name of the structure / attribute

Checks for "magic" signature

Basic data types

Integers

u1 u2 u4 u8
s1 s2 s4 s8
bX - any width

Floats

f4 f8

Bool

b1

Strings

str strz

Byte arrays

size: XXX
size-eos: true

Enums

type: uX or sX
enum: XXX

Literals

1234, -789,
0xfc08, 0b1101,
123.0,
true, false,
[0x20, 65, 66],
'foo bar',
"baz\nqux",
opcode:: jmp

2. Compile template into a target programming language

```
$ kaitai-struct-compiler -t python gif.ksy
```

Target languages:

all, cpp_stl, csharp, construct, go,
graphviz, html, java, javascript, lua,
nim, perl, php, python, ruby, rust

Auto-generated classes for specified language

gif.py

```
class Gif (KaitaiStruct):  
  ...
```

3. Use generated classes in your parser

```
from gif import Gif
g = Gif.from_file("path/to/some.gif")
print("width = %d" % (g.logical_screen.image_width))
print("height = %d" % (g.logical_screen.image_height))
```

read_gif.py

Variable-length structures

```
seq:
  - id: my_len
    type: u4
  - id: my_str
    type: str
    size: my_len
    encoding: UTF-8
```

```
seq:
  - id: to_the_end_of_file
    type: str
    encoding: UTF-8
    size-eos: true
```

Provides reading to the end of the stream

Data beyond the sequence

```
seq:
  - id: file_name
    type: str
    size: 8 + 3
    encoding: ASCII
  - id: file_offset
    type: u4
  - id: file_size
    type: u4
```

instances:

```
body:
  pos: file_offset
  size: file_size
```

Repeat until end of stream

Repeat for a number of times

Repeat until condition is met

A position in bytes from the beginning of the stream

Repetitions

```
seq:
  - id: numbers
    type: u4
    repeat: eos
```

```
seq:
  - id: num_floats
    type: u4
  - id: floats
    type: f8
    repeat: expr
    repeat-expr: num_floats
```

```
seq:
  - id: numbers
    type: s4
    repeat: until
    repeat-until: _ = -1
```

Typical TLV* implementation

```
seq:
  - id: rec_type
    type: u1
  - id: len
    type: u4
  - id: body
    size: len
    type:
      switch-on: rec_type
      cases:
        1: rec_type_1
        2: rec_type_2
        _: rec_type_unknown
```

default (else) case

* TLV means type-length-value