Kaitai Struct

A new way to develop parsers for binary structures.

~ Jarosław Wieczorek

Table of contents

What is the Kaitai Struct ?

What is Kaitai Struct?

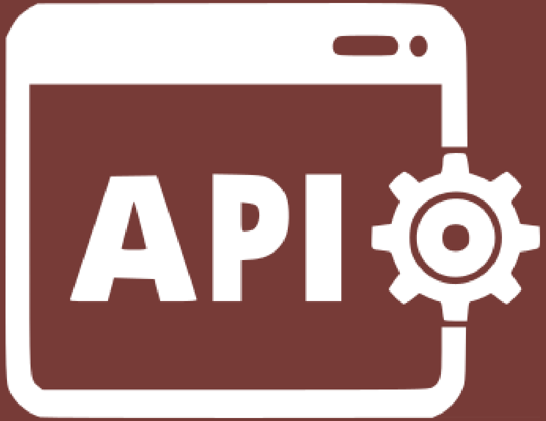
Kaitai Struct is a **declarative language** used to describe various binary data structures, laid out in files or in memory: i.e. binary file formats, network stream packet formats, etc.

What is Kaitai Struct?

The main idea is that a particular format is described in Kaitai Struct language (.ksy files are simple YAML) and then can be compiled with ksc (Kaitai Struct Compiler) into source files in one of the supported programming languages.

What is Kaitai Struct?

These modules will include a generated code for a parser that can read described data structure from a file / stream and give access to it in a nice, easy-to-comprehend API.



Why use Kaitai Struct ?

Declarative

Describe the very structure of the data, not how you read or write it.

Language-neutral

Write once, use in all supported languages:

- **C++/STL**
- **C#**
- **Go (*)**
- **Java**
- **JavaScript**

- **Lua**
- **Perl**
- **PHP**
- **Python**
- **Ruby**

Packed with tools and samples

Includes:

- compiler
- IDE
- visualizer
- massive library of popular formats

Free & open source

Feel free to use, modify and join the project.

KS Tools:

- GNU GPL 3

KS Runtime libraries:

- MIT or Apache v2

Kaitai Struct syntax

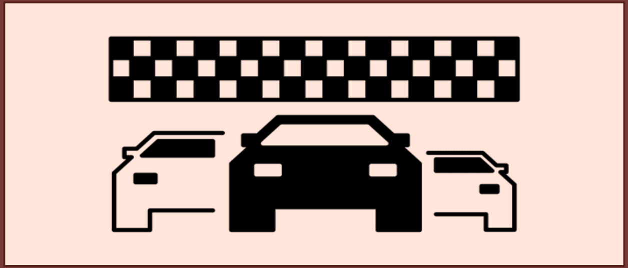
Syntax

User-defined type specification is an essential component of KSY specification. It declares a single user-defined type, which may include:

- **meta —** meta-information
- **doc** — doc-strings
- **seq —** sequence of attributes
- **instances**
- **enums**
- **types** — declaration of subtypes

meta tag - meta information

Meta key is a map of string to objects that provides meta-information relevant the current user-defined type or KSY file in whole. It also can be used to assign some defaults and provide some configuration options for compiler.



```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
    - fooarc
    - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
    - common/archive_header
    - common/compressed_file
  encoding: UTF-8
  endian: le
```

id tag

Contents: a string that follows rules for all identifiers

Purpose: identifier for a primary structure described in top-level map

Influences: it would be converted to suit general formatting rules of a language and used as the name of class

Mandatory: yes

```
meta:
  id: foo_arc
```

title tag

Contents: a string

Purpose: free-form text string that is a longer title of this .ksy file

Influences: nothing

Mandatory: no

```
meta:
  id: foo_arc
  title: Foo Archive
```

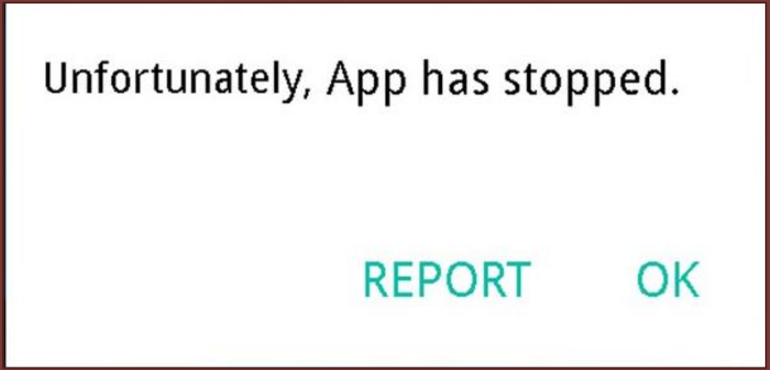
application tag

Contents: a string

Purpose: free-form text string that describes application that’s associated with this particular format, if it’s a format used by single application

Influences: nothing

Mandatory: no


Unfortunately, App has stopped.

```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
```

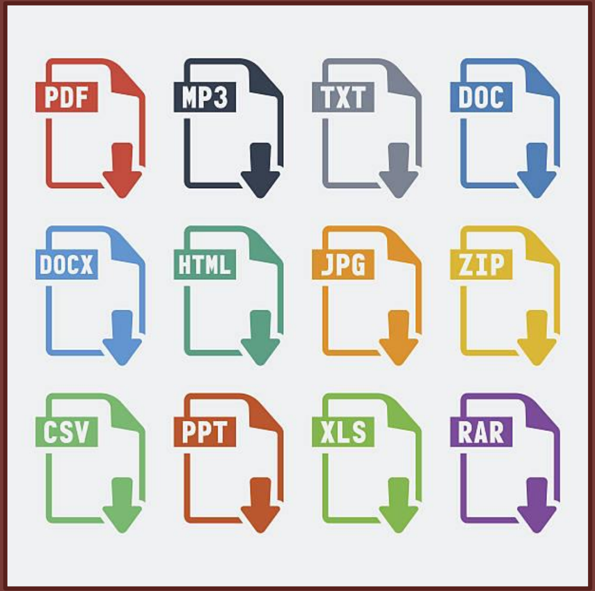
file-extension tag

Contents: a string or an array of strings

Purpose: roughly identify which files can be parsed with this format by filename extension

Influences: may be used for navigation purposes by browsing applications

Mandatory: no



```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
```

license tag

Contents: a string which matches one of the identifiers within the SPDX license list:

https://spdx.org/licenses/

Purpose: identify the copyright license of this .ksy file

Influences: nothing

Mandatory: no

```yaml
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
```

ks-version tag

Contents: a string which contains a Kaitai Struct version number

Purpose: sets the minimum version of **Kaitai Struct Compiler (KSC)** required to interpret this .ksy file

Influences: prevents this .ksy file from being read by older versions of KSC which may not understand newer syntax of this .ksy file

Mandatory: no

```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
```

imports tag

Contents: sequence of strings which contain valid filesystem characters (generally A-Z, a-z, 0-9, _, - and /) corresponding to a **relative** or **absolute path** to another **.ksy** file **(without the .ksy extension)**

Purpose: identify one or more .ksy files which will be imported

Influences: allows types defined within the imported .ksy files to be used in the current context

Mandatory: no

```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
      - common/archive_header
      - common/compressed_file
```

encoding tag

Contents: a string which is a user-defined encoding scheme, for example:

> ASCII, UTF-8, UTF-16LE, UTF-16BE, UTF-32LE, UTF-32BE or a Name from the IANA character sets registry.

Purpose: sets a default string encoding for this file

Influences: if set, str and strz data types will have their encoding by default set to this value

Mandatory: no

```yaml
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
      - common/archive_header
      - common/compressed_file
  encoding: UTF-8
```

endian tag

Contents: le (for little-endian) or be (for big-endian)

Purpose: sets a default endianness for this type and all nested subtypes

Influences: if set, primitive data types like u4 would be treated as aliases to u4le / u4be (depending on the setting); if not set, attempt to use abbreviated types like u4 (i.e. without full endianness qualifier) will **yield compile-time error.**

Mandatory: no

```yaml
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
      - common/archive_header
      - common/compressed_file
  encoding: UTF-8
  endian: le
```

ks-debug tag

Contents: true or false (default)

Purpose: advise the Kaitai Struct Compiler (KSC) to use debug mode

Influences: when set to true, KSC will generate classes as if --debug mode was specified in the command line

Mandatory: no

```
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
      - common/archive_header
      - common/compressed_file
  encoding: UTF-8
  endian: le
  ks-debug: true
```

ks-opaque-types tag

Contents: true or false (default)

Purpose: advise the Kaitai Struct Compiler (KSC) to ignore missing types in the .ksy file, and assume that these types are already provided externally by the environment the classes are generated for.

Influences: when set to true, KSC will generate classes as if --opaque-types=true mode was specified in the command line

Mandatory: no

```yaml
meta:
  id: foo_arc
  title: Foo Archive
  application: Foo Archiver v1.23
  file-extension:
      - fooarc
      - fooarcz
  license: CC0-1.0
  ks-version: 9.9
  imports:
      - common/archive_header
      - common/compressed_file
  encoding: UTF-8
  endian: le
  ks-debug: true
  ks-opaque-types: true
```

doc tag

doc - used to give a more detailed description of a user-defined type.

In most target languages, it will be used as docstring (i.e. a special comment which is exported as part of code documentation), compatible with tools like Javadoc, Doxygen, JSDoc, .NET XML documentation comments, etc..

```
doc: |
  A variable-length unsigned integer using base128 encoding.
  1-byte groups consists of 1-bit flag of continuation and 7-bit value,
  and are ordered "most significant group first", i.e. in "big-endian"
  manner.

  This particular encoding is specified and used in:

  * Standard MIDI file format
  * ASN.1 BER encoding
```

Documentation reference

doc-ref element can be used to provide reference to original documentation, if your KSY file is actually an implementation of some documented format.

```
doc-ref: 'http://example.org/file-format-spec/1.0#header'
doc-ref: ECMA-119 standard, section 4.18 "Volume Set"
doc-ref: http://example.org/some-spec Header section
```

doc-ref tag

Contents: one of:
URL as text , Arbitrary string, URL as text + space + arbitrary string

Purpose: provide reference to original documentation (either in HTML form, available to be referenced by certain URL, or just a free-form reference that can be used to address printed manuals, etc)

Influences: generated docstring comments, usually in a form of "see also". If only text is provided, it will be rendered as neutral text.

If an URL is provided, it will be rendered an active hyperlink, if possible.
If both URL and text is provided, it will create an active hyperlink that leads to URL, with a visible caption equal to provided text.

Mandatory: no

Attributes

Every attribute **MUST BE** a map that maps certain keys to values. Some of these keys are common to every possible attribute specification, some are only valid for certain types.

seq tag - Sequence of attributes

Contents: a sequence of attribute specification elements

Purpose: identifier for a primary structure described in top-level map

Influences: would be translated into parsing method in a target class

Mandatory: no

```yaml
meta:
  id: tcp_segment
  endian: be
seq:
  - id: src_port
    type: u2
  - id: dst_port
    type: u2
  - id: seq_num
    type: u4
  - id: ack_num
    type: u4
```

Instances tag

Instance specification is very close to attribute sec (and inherits all its properties), but it specifies an attribute that lies beyond regular parsing sequence.

Typically, each **instance** is compiled into a **lazy reader function/method** that will parse (or calculate) requested data on demand, cache the result and return whatever’s been parsed previously on subsequent calls.

Instances tag

Contents: map of strings to Instance spec

Purpose: description of data that lies outside of normal sequential parsing flow (for example, that requires seeking somewhere in the file) or just needs to be loaded only by special request

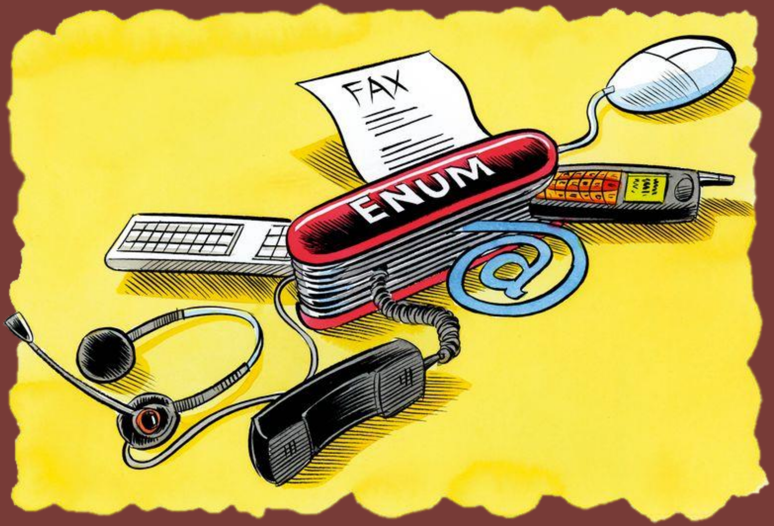
Influences: would be translated into distinct methods (that read desired data on demand) in current class

Mandatory: no

```
seq:
  - id: value_as_type1
    type: type1
    size: 16
instances:
  value_with_type2:
    io: value_with_type1._io
    pos: 0
    size: 2
types:
  type1 ...
```

Enums tag

Enum specification allows to set up a enum (or closest equivalent) construct in target language source file, which can then be referenced in attribute specs using enum key.



Enums tag

Contents: map of strings to Enum specification

Purpose: allow to set up named enums: essentially a mapping between integer constants to some symbolic names; these enums can be used in integer attributes using enum key, thus converting it from simple integer attribute into a proper enum constant

Influences: would be represented as enum-like construct
(or closest equivalent, if target language doesn’t support enums), nested or namespaced in current type/class

Mandatory: no

```
enums:
  ip_protocol:
      1: icmp
      6: tcp
      0x11: udp
  port:
      22: ssh
      25: smtp
      80: http
```

```
seq:
  - id: src_port
    type: u2
    enum: port

          ...

seq:
  - id: http_version
      type: u1
      if: src_port == port::http
```

Types tag - declaration of subtypes

Contents: map of strings to User-defined type specification.

Purpose: declare types for sub-structures that could be referenced in attribute specification in any seq or instances element.

Influences: would be translated into distinct classes
(usually nested into main one, if target language allows it).

Mandatory: no.

```
meta
  id: top_level
seq:
  - id: foo
    type: header
  - id: bar
    type: body1
  - id: baz
    type: body2
types:
  header: # ...
  body1: # ...
  body2: # ...
```

Common keys

id - key

Contents: a string that matches **/^[a-z][a-z0-9_]*$/** — i.e. starts with lowercase letter and then may contain lowercase letters, numbers and underscore

Purpose: identify attribute among others

Influences: used as variable / field name in target programming language

Mandatory:

- yes (for attributes in a seq — sequence of attributes)
- forbidden (for attributes in instances)

Type - declaration of subtype

Contents: one of primitive data types or a name of User-defined type spec

Purpose: define a data type for an attribute

Influences: how much bytes would be read, data type and contents of a variable in target programming language

Mandatory: no — if type is not specified, then attribute is considered [a generic byte sequence](#no-type-specified)

Type - declaration of subtype

If type is used to reference a User-defined type specification, then the following algorithm it used to find which type is referred to, given the name:

1) It tries to find a given type by name in current type’s types — declaration of subtypes map.

2) If that fails, it checks if current type actually has that name and if it does, uses current type recursively.

3) If that fails too, it goes one level up in the hierarchy of nested types and tries to resolve it there.

Contents

Contents: one of:

- a string in UTF-8 encoding
- an array of:
- bytes in decimal representation
- bytes in hexadecimal representation, starting with **0x**
- strings in UTF-8 encoding

Contents

Purpose: specify fixed contents that should be encountered by parser at this point.

Influences: parser checks if specified content exists at a given point in stream; if everything matches, then parsing continues; if content in the stream doesn’t match bytes specified in given contents, it will trigger a parsing exception, thus signalling that something went terribly wrong and it’s meaningless to continue parsing.

Mandatory: no.

Contents

Examples:

foo — expect bytes 66 6f 6f

[foo, 0, A, 0xa, 42] — expect bytes 66 6f 6f 00 41 0a 2a

[1, 0x55, '▒,3', 3] — expect bytes 01 55 e2 96 92 2c 33 03

Repeat

Contents: expr or eos or until

Purpose: designate repeated attribute in a structure;

Influences: attribute would be read as array / list / sequence, executing parsing code multiple times.

Mandatory: no

Repeat

if repeat: **expr** is used, then attribute is repeated the number of times specified in repeat-expr key;

if repeat: **eos** is used, then attribute is repeated until the end of current stream

if repeat: **until** is used, then attribute is repeated until given expression becomes true (one may use a reference to last parsed element in such expression)

Repeat-expr

Contents: expression, expected to be of integer type

Purpose: specify number of repetitions for repeated attribute

Influences: number of times attribute is parsed

Mandatory: yes, if repeat: expr

Repeat-until

Contents: expression, expected to be of boolean type

Purpose: specify expression that would be checked each time after an element of requested type is parsed; while expression is false (i.e. until it becomes true), more elements would be parsed and added to resulting array; one can use _ in expression as a special variable that references last read element

Influences: number of times attribute is parsed

Mandatory: yes, if repeat: until

If, else

Contents: expression, expected to be of boolean type

Purpose: mark the attribute as optional

Influences: attribute would be parsed only if condition specified in if key evaluates (in runtime) to true

Mandatory: no

size and size-eos

Specify amount of bytes to read in size key.

One can specify an integer constant or an [[**expression|expressions**]] in this field (for example, if the number of bytes to read depends on some other attribute).

Set **size-eos**: **true**, thus ordering to read all the bytes till the end of current stream.

pos, io, value

pos

Specifies position in a stream from which the value should be parsed.

io

Specifies an IO stream from which a value should be parsed.

value

Overrides any reading & parsing. Instead, just calculates function specified in value and returns the result as this instance. Can be used for multitude of purposes, such as data conversion while reading, etc.

Terminator

Contents: integer that represents terminating byte

Purpose: string reading will stop when this byte will be encountered

Influences: field data type becomes given enum

Mandatory: no, default is 0

Consume

Contents: boolean

Purpose: specify if terminator byte should be "consumed" when reading - that is:

If consume is **true**, stream pointer will point to the byte after the terminator byte.

If consume is **false**, stream pointer will point to the terminator byte itself.

Influences: stream position after reading of string.

Mandatory: no, default is **true**.

Include

Contents: boolean.

Purpose: specify if terminator byte should be considered a part of string read and thus appended to it.

Influences: string parsed: if true, then resulting string would be 1 byte longer and that byte would be terminator byte.

Mandatory: no, default is false.

Kaitai Struct IDE