Floating-Point Language-Independent Type for YAML™

Working Draft 200?-??-??

Oren Ben-Kiki <oren@ben-kiki.org>
Clark Evans <cce+yaml@clarkevans.com>
Brian Ingerson <ingy@ttul.org>

Copyright © 2001-2004 Oren Ben-KikiClark EvansBrian Ingerson
This document may be freely copied provided it is not modified.

Status

This specification is a release candidate and reflects consensus reached by members of the yaml-core mailing list. Any questions regarding this draft should be raised on this list at http://lists.sourceforge.net/lists/listinfo/yaml-core. With this release of the YAML specification, all further changes will be strictly limited to clarifications, or fixing bugs in productions. At this point, further enhancement or correction of logical flaws will be put off to the next version (1.1) of the YAML specification.

DRAFT

URI: **`tag:yaml.org,2002:float`**

Shorthand: **`!float`**

Kind: Scalar.

Canonical:

```
0
|[-]?0\.([0-9]*[1-9])?e[-+](0|[1-9][0-9]+) (scientific)
|-?\.inf (infinity)
|.nan (not a number)
```

Regexp:

```
[-+]?([0-9][0-9,]*)?\.[0-9.]*([eE][-+][0-9]+)? (base 10)
|[-+]?[0-9][0-9,]*(:[0-5]?[0-9])+\.[0-9,]* (base 60)
|[-+]?\.(inf|Inf|INF) (infinity)
|\.(nan|NaN|NAN) (not a number)
```

Definition: Floating-point approximation to real numbers.

Floating-point numbers are approximations to real numbers, including three special values (positive and negative infinity and “not a number”). Using “`:`” allows expressing the integer part in base 60, which is convenient for time and angle values. Any “`,`” characters in the number are ignored, allowing a readable representation of large values.

This should be loaded to some native float data type. The processor may choose from a range of such native data types according to the size and accuracy of the floating-point

XSL•FO RenderX

value. Note that not all floating-point values can be represented exactly when stored in any native float type, and hence a float value may change by “a small amount” when round-tripped through a native type. The valid range and accuracy depends on the implementation, though 32 bit IEEE floats should be safe.

Example 1. `!float` Examples

```
canonical: 6.8523015e+5
exponentioal: 685.230,15e+03
fixed: 685,230.15
sexagesimal: 190:20:30.15
negative infinity: -.inf
not a number: .NaN
```

DRAFT

XSL•FO
RenderX