YAML Encoding Keys
Language-Independent Type for YAML™ Version 1.1

Working Draft 2005-01-18

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

Copyright © 2001-2005 Oren Ben-Kiki, Clark Evans, Brian Ingerson
This document may be freely copied provided it is not modified.

Status
This specification is a draft reflecting consensus reached by members of the yaml-core mailing list [http://lists.sourceforge.net/lists/listinfo/yaml-core]. Any questions regarding this draft should be raised on this list.

URI: tag:yaml.org,2002:yaml

Shorthand: !!yaml

Kind: Scalar.

Canonical: N/A (single format).

Regexp: ! |& |*

Definition: Keys for encoding YAML in YAML.

YAML encoding keys are used to denote YAML structure information. The in-memory representation of these keys must be different from any value in any other type family. Specifically, these in-memory values must not be implemented as strings. Normally, the encoding keys should not be used in serialized YAML documents; the encoded YAML node is serialized instead.

Encoding is useful when a YAML processor encounters a valid YAML value of an unknown tag. For a schema-specific application, this is not different from encountering any other valid YAML document that does not satisfy the schema. Such an application may safely use a processor that rejects any value of any unknown tag, or discards the tag property with an appropriate warning and parses the value as if the property was not present.

For a schema-independent application (for example, a hypothetical YAML pretty print application), this is not an option. Processors used by such applications should encode the value instead. This may be done by wrapping the value in a mapping containing encoding keys. The “!” key denotes the unsupported tag. In some cases it may be necessary to encode anchors and alias nodes as well. The “&” and “*” keys are used for this purpose.

Encoding should be reversed on output, allowing the schema-independent application to safely round-trip any valid YAML document. In-memory, the encoded data may be accessed and
manipulated in a standard way using the three basic data types (mapping, sequence and scalar), allowing limited processing to be applied to arbitrary YAML data.

Example 1. **!!yaml** Examples

```yaml
encoded YAML node :
!!yaml '!' : '!type'
!!yaml '&' : 12
!!value = : value
```

# The following node should NOT be serialized this way.

```yaml
node : !type &12 value
```