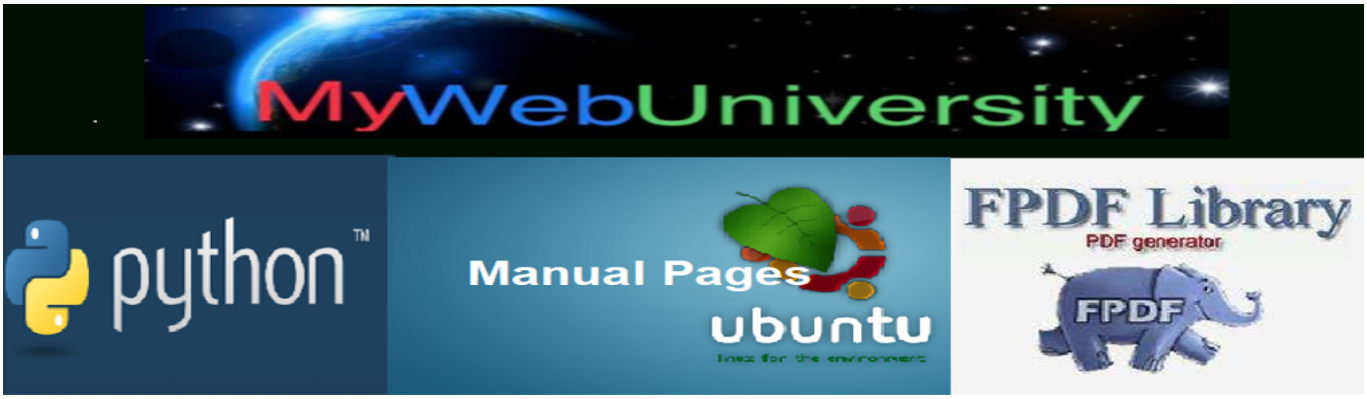

Rocky Enterprise Linux 9.2 Manual Pages on command 'Glib::Variant.3pm'

$ man Glib::Variant.3pm

Glib::Variant(3pm) User Contributed Perl Documentation Glib::Variant(3pm)

NAME

Glib::Variant - strongly typed value datatype

SYNOPSIS

```
my $v = Glib::Variant->new ('as', ['GTK+', 'Perl']);
my $aref = $v->get ('as');
```

DESCRIPTION

There are two sets of APIs for creating and dealing with "Glib::Variant"s: the low-level API described below under "METHODS", and the convenience API described in this section.

CONVENIENCE API

variant = Glib::Variant->new ($format_string, $value)

(variant1, ...) = Glib::Variant->new ($format_string, $value1, ...)

Constructs a variant from $format_string and $value. Also supports constructing multiple variants when the format string is a concatenation of multiple types.

value = $variant->get ($format_string)

Deconstructs $variant according to $format_string.

The following symbols are currently supported in format strings:

Symbol	Meaning
b, y, n, q, i, u, x, t, h, d	Boolean, byte and numeric types
s, o, g	String types
v	Variant types
a	Arrays
m	Maybe types
()	Tuples
{}	Dictionary entries

Note that if a format string specifies an array, a tuple or a dictionary entry ("a", "()" or "{}"), then array references are expected by "new" and produced by "get". For arrays of dictionary entries ("a{}"), hash references are also supported by "new" and handled as you would expect.

For a complete specification, see the documentation at

<https://developer.gnome.org/glib/stable/glib-GVariantType.html>
<https://developer.gnome.org/glib/stable/glib-GVariant.html>
<https://developer.gnome.org/glib/stable/gvariant-format-strings.html>
<https://developer.gnome.org/glib/stable/gvariant-text.html>

HIERARCHY

Glib::Variant

METHODS

variant = Glib::Variant->new_array ($child_type, $children)

? $child_type (Glib::VariantType)

? $children (scalar)

variant = Glib::Variant->new_boolean ($value)

? $value (boolean)

variant = Glib::Variant->new_byte ($value)

? $value (Glib::UChar)

variant = Glib::Variant->new_bytestring ($string)

? $string (byte string)

Since: glib 2.26

variant = Glib::Variant->new_dict_entry ($key, $value)

? $key (Glib::Variant)

? $value (Glib::Variant)

variant = Glib::Variant->new_double ($value)

? $value (double)

variant = Glib::Variant->new_handle ($value)

? $value (integer)

variant = Glib::Variant->new_int16 ($value)

? $value (integer)

variant = Glib::Variant->new_int32 ($value)

? $value (integer)

variant = Glib::Variant->new_int64 ($value)

? $value (64 bit integer)

variant = Glib::Variant->new_maybe ($child_type, $child)

? $child_type (Glib::VariantType)

? $child (Glib::Variant)

variant = Glib::Variant->new_object_path ($object_path)

? $object_path (string)

variant = Glib::Variant->new_signature ($signature)

? $signature (string)

variant = Glib::Variant->new_string ($string)

? $string (string)

variant = Glib::Variant->new_tuple ($children)

? $children (scalar)

variant = Glib::Variant->new_uint16 ($value)

? $value (unsigned)

variant = Glib::Variant->new_uint32 ($value)

? $value (unsigned)

variant = Glib::Variant->new_uint64 ($value)

? $value (64 bit unsigned)

variant = Glib::Variant->new_variant ($value)

? $value (Glib::Variant)

boolean = $value->get_boolean

uchar = $value->get_byte

string = $value->get_bytestring

Since: glib 2.26

variant = $value->byteswap

variant = $value->get_child_value ($index_)

? $index_ (unsigned)

string = $value->classify

integer = $one->compare ($two)

? $two (Glib::Variant)

Since: glib 2.26

double = $value->get_double

boolean = $one->equal ($two)

? $two (Glib::Variant)

integer = $value->get_handle

integer = $value->hash

integer = $value->get_int16

integer = $value->get_int32

64 bit integer = $value->get_int64

boolean = $value->is_container

boolean = $value->is_normal_form

boolean = $string->is_object_path

boolean = $value->is_of_type ($type)

? $type (Glib::VariantType)

boolean = $string->is_signature

variant = $dictionary->lookup_value ($key, $expected_type)

? $key (string)

? $expected_type (Glib::VariantType)

Since: glib 2.28

variant = $value->get_maybe

unsigned = $value->n_children

variant = $value->get_normal_form

variant = Glib::Variant::parse ($type, $text)

? $type (Glib::VariantType)

? $text (string)

May croak with a Glib::Error in $@ on failure.

string = $value->print ($type_annotate)

? $type_annotate (boolean)

unsigned = $value->get_size

string = $value->get_string

varianttype = $value->get_type

string = $value->get_type_string

unsigned = $value->get_uint16

unsigned = $value->get_uint32

64 bit unsigned = $value->get_uint64

variant = $value->get_variant

SEE ALSO

Glib, Glib::VariantType, Glib::VariantDict

COPYRIGHT