Scripting - Variants

Application Note

Introduction

Within Lua Scripting (as with other scripting languages) it is possible to store data within a named location (variable). Lua typically doesn't differentiate between the contents of a variable (unlike some programming languages) and the type (integer, string, boolean) of the variable can change at any time.
Pharos has added an object to the scripting environment called a Variant, which can be used to contain the data with an assignment as to the type of data that is contained. This means that a single Variant can be utilised and handled differently depending on the data that is contained and how it is being used.

Usage

```
Variant(value, range)
```

Defining a Variant

Within your Lua script you can create a Variant with the following syntax:

```
var = Variant() -- where var is the name of the variant.
```

Variant Types

Integer

An integer variant can be used to store a whole number

```
var = Variant() -- where var is the name of the variant.
var.integer = 123 -- Set var to an integer value of 123
log(var.integer) -- get the integer value stored in var
log(var.real) -- get the integer value stored in var and convert it to a float
log(var.string) -- get the integer value stored in var and convert it to a string
```

.integer can be used to either Get or Set the value of var as an integer (whole number).

```
var:is_integer() -- returns a boolean if the variant contains an integer
```

Range

An integer can be stored with an optional range parameter

```
var = Variant() -- where var is the name of the variant.
var.integer = 123 -- Set var to an integer value of 123
var.range = 255 -- Set the range of var to be 255
```

This can be used to calculate fractions and/or to define that an Variant is a 0-1, 0-100 or 0-255 value.

The range of the variant should be set if you intend to use the variant to set an intensity or colour value.

Real

A real variant can be used to store a floating point (decimal) number.

```
var = Variant() -- where var is the name of the variant.
var.real = 12.3 -- Set var to an integer value of 12.3
log(var.real) -- get the integer value stored in var
```

String

A string variant can be used to store a string of ASCII characters

```
var = Variant() -- where var is the name of the variant
var.string = "example" -- Set var to a string value of "example"
log(var.string) -- get the string value stored in var
```

.string can be used to either Get or Set the value of var as a string

```
var:is_string() -- returns a boolean if the variant contains a string
```

IP Address

```
var = Variant() -- where var is the name of the variant
var.ip_address = "192.168.1.23" -- Set var to the IP Address 192.168.1.23 or -1062731497
log(var) -- get the stored data ("192.168.1.23")
log(var.ip_address) -- get the stored IP Address (-1062731497)
log(var.string) -- get the  stored IP Address and convert it to a string ("192.168.1.23")
log(var.integer) -- get the  stored IP Address and convert it to an integer (-1062731497)
```

.ip_address can be used to either Get or Set the value of var as an IP Address.
As a setter, you can pass a dotted decimal string (e.g. "192.168.1.23" or the integer representation -1062731497)

```
var:is_ip_address() -- returns a boolean if the variant contains a IP Address
```

Shorthand

Variants can also be defined using a shorthand:

```
var = Variant(128,255) -- create variable var as an integer (128) with range 0-255
var = Variant(128) -- create variable var as a real number (128.0)
var = Variant(12.3) -- create variable var as a real number (12.3)
var = Variant("text") -- create variable var as a string ("text")
```

Note: There isn't a shorthand for IP Addresses