1.0

[RIFT ADDON API BIBLE]

Providing a comprehensive searchable and easy to read compilation of all information regarding the Rift™ Addon API. This is a community project and as such is free to anyone and everyone.

The core of this document is extracted from the Rift™ Addon API and is subject to any licensing restrictions imposed by Trion. As well many functions documentation is referenced from the LUA 5.0 Manual available at http://www.lua.org/manual/5.1/manual.html and is again subject to the licensing associated with all LUA documentation.

You may download the most recent version as well as the original Word document at:
http://rift.curseforge.com/addons/rift-addon-api-bible/

EDITED BY
Shawn Sagady (Bigmanshawn)

TABLE OF CONTENTS

BASIC GLOBALS

Development.Documentation

Provide documentation on items in the addon environment. Called with no parameters, it returns a table listing all documentation. Can provide both human-readable and computer-readable documentation.

```
documentables = Development.Documentation()

documentation = Development.Documentation(item)

documentation = Development.Documentation(item, parseable = false)

documentationTable = Development.Documentation(item, parseable = true)
```

Parameters

Parsable (Boolean)
Whether to return in a computer-readable format, as opposed to the normal human-readable format.

Item (Variant)
The item to get documentation on. May be either the item itself or a string identifier.

Results

documentables (table)
List of all items that documentation can be retrieved for. In {["itemname"] = true} format.

documentationTable (Table)
Computer-readable documentation for the requested item. Format may change without warning.

documentation (String)
Documentation for the requested item.

UI.CreateContext

Creates a new UI context. A UI context must be created in order to create any frames.

```
context = UI.CreateContext(name)
```

Parameters

name (String)
A descriptive name for this element. Used for error reports and performance information. May be shown to end-users.

Results

context (Context)
A new context. Contexts are guaranteed to have at least the capabilities of a Frame.

UI.CreateFrame

Creates a new frame. Frames are the blocks that all addon UIs are made out of. Since all frames must have a parent, you'll need to create a Context before any frames. See UI.CreateContext.

```
frame = UI.CreateFrame(type, name, parent)
```

Parameters

Type (String)
The type of your new frame. Current supported types: Frame, Text, Texture.

Name (String)
A descriptive name for this element. Used for error reports and performance information. May be shown to end-users.

Parent (Frame)
The new parent for this frame

Results

frame (Frame)
Your new frame.

_VERSION

assert

bit.arshift

Returns the bitwise arithmetic right-shift of its first argument by the number of bits given by the second argument.

Arithmetic right-shift treats the most-significant bit as a sign bit and replicates it. Only the lower 5 bits of the shift count are used (reduces to the range [0..31]).

```
result = bit.arshift(x, n)
```

Examples

```
print(bit.arshift(256, 8))           --> 1

print(bit.arshift(-256, 8))          --> -1

printx(bit.arshift(0x87654321, 12))  --> 0xfff87654
```

bit.band

Returns the bitwise **and** of all its argument.

```
result = bit.band(x[,n...])
```

Examples

```
printx(bit.band(0x12345678, 0xff))        --> 0x00000078
```

bit.bnot

Returns the bitwise **not** of its argument.

```
result = bit.bnot(x)
```

Examples

```
print(bit.bnot(0))             --> -1

printx(bit.bnot(0))            --> 0xffffffff

print(bit.bnot(-1))            --> 0

print(bit.bnot(0xffffffff))    --> 0

printx(bit.bnot(0x12345678))   --> 0xedcba987
```

bit.bor

Returns the bitwise **or** of all of its arguments.

```
result = bit.bor(x, [,n ...])
```

Examples

```
print(bit.bor(1, 2, 4, 8))                  --> 15
```

bit.bswap

Swaps the bytes of its argument and returns it. This can be used to convert little-endian 32 bit numbers to big-endian 32 bit numbers or vice versa.

```
result = bit.bswap(x)
```

Examples

```
printx(bit.bswap(0x12345678)) --> 0x78563412

printx(bit.bswap(0x78563412)) --> 0x12345678
```

bit.bxor

Returns the bitwise **xor** of all of its arguments.

```
results = bit.bxor(x [,n ...])
```

Examples

```
printx(bit.bxor(0xa5a5f0f0, 0xaa55ff00))  --> 0x0ff00ff0
```

bit.lshift

Returns the bitwise logical left-shift of its first argument by the number of bits given by the second argument.

Logical shifts treat the first argument as an unsigned number and shift in 0-bits. Only the lower 5 bits of the shift count are used (reduces to the range [0..31]).

```
result = bit.lshift(x, n)
```

Examples

```
print(bit.rshift(256, 8))            --> 1

print(bit.rshift(-256, 8))           --> 16777215

printx(bit.rshift(0x87654321, 12))   --> 0x00087654
```

bit.rol

Returns the bitwise left rotation of its first argument by the number of bits given by the second argument. Bits shifted out on one side are shifted back in on the other side.

Only the lower 5 bits of the rotate count are use (reduces the range [0..31]).

```
result = bit.rol(x, n)
```

Examples

```
printx(bit.rol(0x12345678, 12))   --> 0x45678123
```

bit.ror

Returns the bitwise right rotation of its first argument by the number of bits given by the second argument. Bits shifted out on one side are shifted back in on the other side.

Only the lower 5 bits of the rotate count are use (reduces the range [0..31]).

```
result = bit.ror(x, n)
```

Examples

```
printx(bit.ror(0x12345678, 12))   --> 0x67812345
```

bit.rshift

Returns the bitwise logical left-shift of its first argument by the number of bits given by the second argument.

Logical shifts treat the first argument as an unsigned number and shift in 0-bits. Only the lower 5 bits of the shift count are used (reduces to the range [0..31]).

```
result = bit.rshift(x, n)
```

Examples

```
print(bit.lshift(1, 0))              --> 1
print(bit.lshift(1, 8))              --> 256
print(bit.lshift(1, 40))             --> 256
printx(bit.lshift(0x87654321, 12))   --> 0x54321000
```

bit.tobit

Normalizes a number to the numeric range for bit operations and returns it. This function is usually not needed since all bit operations already normalize all of their input arguments.

```
result = bit.tobit(x)
```

Examples

```
print(0xffffffff)                --> 4294967295 (*)
print(bit.tobit(0xffffffff))     --> -1
printx(bit.tobit(0xffffffff))    --> 0xffffffff
print(bit.tobit(0xffffffff + 1)) --> 0
print(bit.tobit(2^40 + 1234))    --> 1234
```

bit.tohex

Converts its first argument to a hex string. The number of hex digits is given by the absolute value of the optional second argument. Positive numbers between 1 and 8 generate lowercase hex digits. Negative numbers generate uppercase hex digits. Only the least-significant 4*|n| bits are used. The default is to generate 8 lowercase hex digits.

```
result = bit.tohex(x [,n])
```

Examples

```
print(bit.tohex(1))              --> 00000001
print(bit.tohex(-1))             --> ffffffff
print(bit.tohex(0xffffffff))     --> ffffffff
print(bit.tohex(-1, -8))         --> FFFFFFFF
print(bit.tohex(0x21, 4))        --> 0021
print(bit.tohex(0x87654321, 4))  --> 4321
```

collectgarbage

coroutine.create

coroutine.resume

coroutine.running

coroutine.status

coroutine.wrap

coroutine.yield

debug.traceback

error

gcinfo

getfenv

getmetatable

ipairs

load

loadstring

math.abs

Returns the absolute value of x.

```
result = math.abs(x)
```

math.acos

Returns the arc cosine of x

```
result = math.acos(x)
```

math.asin

Returns the arc sine of a number (in radians)

```
result = math.asin(x)
```

math.atan

Returns the arc tangent of a number (in radians)

```
result = math.atan(x)
```

math.atan2

Returns the arc tangent of x/y (in radians), but uses the signs of both parameters to find the quadrant of the result. (It also handles correctly the case of x being zero.)

```
result = math.atan2(x,y)
```

math.ceil

Returns the smallest integer larger than or equal to x.

```
result = math.ceil(x)
```

math.cos

Returns the cosine of x (assumed to be in radians)

```
result = math.cos(x)
```

math.cosh

Returns the hyperbolic cosine of x.

```
result = math.cosh(x)
```

math.deg

Returns the angle of x (given in radians) in degrees.

```
result = math.deg(x)
```

math.exp

Returns the value e^x.

```
result = math.exp(x)
```

math.floor

Returns the largest integer smaller than or equal to x.

```
result = math.floor(x)
```

math.fmod

Returns the remainder of the division of x by y that rounds the quotient towards zero.

```
result = math.fmod(x,y)
```

math.frexp

Returns m and e such that x = $m2^e$, e is an integer and the absolute value of m is in the range [0.5, 1) (or zero when x is zero)

```
result = math.frexp(x)
```

math.huge

The value HUGE_VAL, a value larger than or equal to any other numerical value.

```
result = math.huge()
```

math.ldexp

Returns $m2^e$ (e should be an integer)

```
result = math.ceil(m,e)
```

math.log

Returns the natural logarithm of x.

```
result = math.log(x)
```

math.log10

Returns the base-10 logarithm of x.

```
result = math.log10(x)
```

math.max

Returns the maximum value among its arguments.

```
result = math.max(x, ...)
```

math.min

Returns the minimum value among its arguments

```
result = math.min(x, ...)
```

math.mod

math.modf

Returns two numbers, the integral part of x and the fractional part of x.

math.pi

The value of pi.

```
result = math.pi()
```

math.pow

Returns x^y (You can also use the expression x^y to compute this value.)

```
result = math.pow(x,y)
```

math.rad

Returns the angle x (given in degrees) in radians.

```
result = math.rad(x)
```

math.random

This function is an interface to the simple psuedo-random generator function rand provided by ANSI C. (No guarantees can be given for its statistical properties.)

When called without arguments, returns a uniform pseudo-random real number in the range [0,1). When called with an integer number m, returns a uniform pseudo-random number in the range [1, m]. When called with two integer numbers m and n, returns a uniform pseudo-random integer in the range [m, n].

```
result = math.random([m [, n]])
```

math.randomseed

Sets x as the “seed” for the pseudo-random generator: equal seeds produce equal sequences of numbers.

```
result = math.randomseed(x)
```

math.sin

Returns the sine of x (assumed to be in radians)

```
result = math.sin(x)
```

math.sinh

Returns the hyperbolic sine of x.

```
result = math.sinh(x)
```

math.sqrt

Returns the square root of x. (You can also use the expression x^0.5 to compute this value.)

```
result = math.sqrt(x)
```

math.tan

Returns the tangent of x (assumed to be in radians).

```
result = math.tan(x)
```

math.tanh

Returns the hyperbolic tangent of x.

```
result = math.tanh(x)
```

newproxy

next

pairs

pcall

print

print_console

rawequal

rawget

rawset

select

setfenv

setmetatable

string.byte

string.char

string.find

string.format

string.gfind

string.gmatch

string.gsub

string.len

string.lower

string.match

string.rep

string.reverse

string.sub

string.upper

table.concat

table.foreach

table.foreachi

table.getn

table.insert

table.maxn

table.remove

table.sort

tonumber

tostring

type

unpack

xpcall

INSPECTORS

Inspect.Ability.Detail

Provides detailed information about abilities.

```
detail = Inspect.Ability.Detail(ability)    -- table <- string

details = Inspect.Ability.Detail(abilities)    -- table <- table
```

Parameters

ability (String)
The identifier of the ability to retrieve detail for.

abilities (Table)
A lookup table of identifiers of abilities to retrieve detail for.

Results

detail (Table)
Detail table for a single ability. May include members:

- name
- icon
- castingTime
- channeled
- continuous
- autoattack
- stealthRequired
- rangeMin
- rangeMax
- cooldown
- currentCooldown
- currentCooldownExpired
- currentCooldownRemaining
- racial
- passive
- positioned
- target
- outOfRange
- unusable
- costPlanarCharge
- costPower
- costMana
- costEnergy
- costCharge
- gainCharge
- weapon

details (Table)
Detail tables for all requested abilities. The key is the ability ID, the value is the ability's detail table.

Inspect.Ability.List

List available abilities.

```
abilities = Inspect.Ability.List()
```

Results

abilities (Table)
A lookup table of IDs of the available abilities.

Inspect.Addon.Cpu

Returns recent CPU usage information. This is calculated using an exponential-falloff method.

```
data = Inspect.Addon.Cpu()
```

Results

data (Table)
Recent CPU usage. This takes the format

```
{ AddonIdentifier = { SubIdentifier = cpu_used_as_a_fraction_of_one } }
```

SubIdentifiers are generated by Rift and the format may change without notice.

Inspect.Addon.Current

Returns the current addon. This information is used internally for counting CPU usage and determining frame ownership.

```
addonIdentifier = Inspect.Addon.Current()
```

Results

addonIdentifier (String)
The addon's identifier, as written in its TOC file.

Inspect.Buff.Detail

Provides detailed information about the buffs on a unit.

```
detail = Inspect.Buff.Detail(unit, buff)

details = Inspect.Buff.Detail(unit, buffs)
```

Parameters

Buff (String)
An identifier for the buff to retrieve detail for.

Buffs (Table)
A lookup table containing buff identifiers to retrieve details for.

Unit (String)
The unit to inspect.

Results

detail (Table)
Detail table for a single buff. May include members: name, buff, debuff, noncancelable, duration, remaining, expired, stack, caster and icon.

details (Table)
Detail tables for all requested buffs. The key is the buff ID, the value is the buff's detail table.

Inspect.Buff.List

List buffs on a unit.

```
buffs = Inspect.Buff.List(unit)
```

Parameters

Unit (String)
The unit to inspect.

Results

buffs (Table)
A lookup table of the IDs of the buffs on the unit.

Inspect.System.Time

A high-resolution timer.

```
time = Inspect.System.Time()
```

Results

time
Time in seconds. Counted from an arbitrary point in the past. Guaranteed to be non-negative.

COMMANDS

Command.Buff.Cancel

Cancels a buff on the player. Not all buffs are cancelable.

```
Command.Buff.Cancel(buff)
```

Parameters

Buff (String)
The ID of the buff to cancel.

Command.Slash.Register

Registers a new chat slash command, inserts a new event table into the Event.Slash hierarchy, and returns that table. If called multiple times with the same slash command, will return the same table each time.

```
eventTable = Command.Slash.Register(slashCommand)
```

Parameters

slashCommand (String)
The name of the slash command to register.

Results

eventTable (Table)
The event table for your slash command. nil if the slash command could not be registered (usually because it conflicts with a built-in slash command.)

EVENTS

Event.Ability.AvailabilityChange

Event.Ability.Cooldown.Begin

Event.Ability.Cooldown.End

Event.Addon.Finalizing

Event.Addon.Load.Begin

Event.Addon.Load.End

Event.Addon.SavedVariables.Load.Begin

Event.Addon.SavedVariables.Load.End

Event.Addon.SavedVariables.Save.Begin

Event.Addon.SavedVariables.Save.End

Event.Addon.Shutdown

Event.Addon.Starting

Event.Slash

Category for dynamically-created events

Event.Slash.dump

1 Handler

Event.System.Error

1 Handler

Event.System.Update.Begin

Event.System.Update.End

1 Handler

UI – LAYOUT

MEMBERS

GetBottom

Retrieves the Y position of the bottom edge of this element.

```
bottom = Layout:GetBottom()
```

Results

bottom (Number)
The Y position of the bottom edge of this element.

GetBounds

Retrieves the complete bounds of this element.

```
left, top, right, bottom = Layout:GetBounds()
```

Results

left (Number)
The X position of the left edge of this element.

top (Number)
The Y position of the top edge of this element.

right (Number)
The X position of the right edge of this element.

bottom (Number)
The Y position of the bottom edge of this element.

GetEventTable

Retrieves the event table of this element. By default, this value is also stored in "this.Event".

```
eventTable = Layout:GetEventTable()
```

Results

eventTable (Table)
The event table of this element.

GetHeight

Retrieves the height of this element.

```
height = Layout:GetHeight()
```

Results

height (Number)
The height of this element.

GetLeft

Retrieves the X position of the left edge of this element.

```
left = Layout:GetLeft()
```

Results

left (Number)
The X position of the left edge of this element.

GetName

Retrieves the name of this element.

```
name = Layout:GetName()
```

Results

name (String)
The name of this element, as provided by the addon that created it.

GetOwner

Retrieves the owner of this element.

```
owner = Layout:GetOwner()
```

Results

owner (String)
The owner of this element. Given as an addon identifier.

GetRight

Retrieves the X position of the right edge of this element.

```
right = Layout:GetRight()
```

Results

right (Number)
The X position of the right edge of this element.

GetTop

Retrieves the Y position of the top edge of this element.

```
top = Layout:GetTop()
```

Results

top (Number)
The Y position of the top edge of this element.

GetWidth

Retrieves the width of this element.

```
width = Layout:GetWidth()
```

Results

width (Number)
The width of this element.

EVENTS

Move

Size

UI – FRAME (INHERITS FROM LAYOUT)

MEMBERS

GetAlpha

Gets the alpha multiplier of this frame.

```
alpha = Frame:GetAlpha()
```

Results

alpha (Number)
The alpha multiplier of this frame. 1 is fully opaque, 0 is fully transparent. This does not include multiplied alphas from this frame's parent - it's the exact value passed to SetAlpha.

GetBackgroundColor

Retrieves the background color of this frame.

```
r, g, b, a = Frame:GetBackgroundColor()
```

Results

r (Number)
Red.

g (Number)
Green.

b (Number)
Blue.

a (Number)
Alpha. 1 is fully opaque, 0 is fully transparent.

GetLayer

Gets the frame's layer order.

```
layer = Frame:GetLayer()
```

Results

layer (Number)
The render layer of this frame. See Frame:SetLayer for details.

GetParent

Gets the parent of this frame.

```
parent = Frame:GetParent()
```

Results

parent (Frame)
The parent element of this frame.

GetVisible

Gets the visibility flag for this frame.

```
visible = Frame:GetVisible()
```

Results

visible (Boolean)
This frame's visibility flag, as set by SetVisible. Does not check the visibility flags of the frame's parents.

SetAllPoints

Pins all the edges of this frame to the edges of a different frame.

```
Frame:SetAllPoints(target)
```

Parameters

target (Layout)
Target Layout to pin this frame's edges to.

SetAlpha

Sets the alpha transparency multiplier for this frame and its children.

```
Frame:SetAlpha(alpha)
```

Parameters

alpha (Number)
The new alpha multiplier. 1 is fully opaque, 0 is fully transparent.

SetBackgroundColor

Sets the background color of this frame.

```
Frame:SetBackgroundColor(r, g, b)

Frame:SetBackgroundColor(r, g, b, a)
```

Parameters

r (Number)
Red.

g (Number)
Green.

b (Number)
Blue.

a (Number)
Alpha. 1 is fully opaque, 0 is fully transparent. Defaults to 1.

SetHeight

Sets the height of this frame. Undefined results if the frame already has two pinned Y coordinates.

```
Frame:SetHeight(height)
```

Parameters

height (Number)
The new height of this frame.

SetLayer

Sets the frame layer for this frame. This can be any number. Frames are drawn in ascending order. If two frames have the same layer number, then the order of those frames is undefined, but stable during a single play session. Frames are always drawn after their parents.

```
Frame:SetLayer(layer)
```

Parameters

layer (Number)
The new layer for this frame.

SetParent

Sets the parent of this frame. Circular dependencies result in undefined behavior.

```
Frame:SetParent(parent)
```

Parameters

parent (Frame)
The new parent for this frame.

SetPoint

Pins a point on this frame to a location on another frame. This is a rather complex function and you should look at examples to see how to use it. Better documentation will be forthcoming.

```
Frame:SetPoint(...)
```

Parameters

...
This function's parameters are complicated. More details will be forthcoming.

SetVisible

Sets the frame's visibility flag. If set to false, then this frame and all its children will not be rendered or accept mouse input.

```
Frame:SetVisible(visible)
```

Parameters

visible (Boolean)
The new visibility flag.

SetWidth

Sets the width of this frame. Undefined results if the frame already has two pinned X coordinates.

```
Frame:SetWidth(width)
```

Parameters

width (Number)
The new width of this frame.

EVENTS

LeftDown

LeftUp

UI – CONTEXT (INHERITS FROM FRAME)

MEMBERS

EVENTS

UI - TEXT (INHERITS FROM FRAME)

MEMBERS

GetFont

Gets the current font used for this element.

```
source, font = Text:GetFont()
```

Results

font (String)
The actual font identifier. Either a resource identifier or a filename.

source (String)
The source of the resource. "Rift" will take the resource from Rift's internal data. Anything else will take the resource from the addon with that identifier.

GetFontColor

Gets the current font color for this element.

```
r, g, b, a = Text:GetFontColor()
```

Results

r (Number)
Red

b (Number)
Blue.

g (Number)
Green.

a (Number)
Alpha. 1 is fully opaque, 0 is fully transparent.

GetFontSize

Gets the font size of the current element.

```
fontsize = Text:GetFontSize()
```

Results

fontsize (Number)
The current font size of this element.

GetFullHeight

Get the height that would be required for this element to display all lines of text.

```
height = Text:GetFullHeight()
```

Results

height (Number)
The element height needed to display all lines of text.

GetFullWidth

Get the width that would be required for this element to avoid word wrapping or truncation.

```
width = Text:GetFullWidth()
```

Results

width (Number)
The element width needed to avoid word wrapping or truncation.

GetText

Get the current text for this element.

```
text = Text:GetText()
```

Results

text (String)
The current text of the element.

GetWordwrap

Gets the wordwrap flag for this element.

```
wordwrap = Text:GetWordwrap()
```

Results

wordwrap (Boolean)
The current wordwrap flag for this element. If false, long lines will be truncated. Defaults to false.

ResizeToText

Sets the element's width and height to display all text without wordwrapping or truncation.

```
Text:ResizeToText()
```

SetFont

Sets the current font used for this element.

```
Text:SetFont(source, font)
```

Parameters

font (String)
The actual font identifier. Either a resource identifier or a filename.

source (String)
The source of the resource. "Rift" will take the resource from Rift's internal data. Anything else will take the resource from the addon with that identifier.

SetFontColor

Sets the current font color for this element.

```
Text:SetFontColor(r, g, b)
Text:SetFontColor(r, g, b, a)
```

Parameters

r (Number)
Red.

g (Number)
Green.

b (Number)
Blue.

a (Number)
Alpha. 1 is fully opaque, 0 is fully transparent. Defaults to 1.

SetFontSize

Sets the current font size of this element.

```
Text:SetFontSize(fontsize)
```

Parameters

fontsize (Number)
The new font size of this element.

SetText

Sets the current text for this element.

```
Text:SetText(text)
```

Parameters

text (String)
The new text for the element.

SetWordwrap

Sets the wordwrap flag for this element.

```
Text:SetWordwrap(wordwrap)
```

Parameters

wordwrap (Boolean)
The new wordwrap flag for this element. If false, long lines will be truncated. Defaults to false.

EVENTS

UI – TEXTURE (INHERITS FROM FRAME)

MEMBERS

GetTexture

Gets the current texture used for this element.

```
source, texture = Texture:GetTexture()
```

Results

source (String)
The source of the resource. "Rift" will take the resource from Rift's internal data. Anything else will take the resource from the addon with that identifier.

texture (String)
The actual texture identifier. Either a resource identifier or a filename.

GetTextureHeight

Returns the actual pixel height of the current texture.

```
height = Texture:GetTextureHeight()
```

Results

height (Number)
The height of the current texture in pixels.

GetTextureWidth

Returns the actual pixel width of the current texture.

```
width = Texture:GetTextureWidth()
```

Results

width (Number)
The width of the current texture in pixels.

ResizeToTexture

Sets the element's width and height to the exact pixel size of the texture.

```
Texture:ResizeToTexture()
```

SetTexture

Sets the current texture used for this element.

```
Texture:SetTexture(source, texture)
```

Parameters

source (String)
The source of the resource. "Rift" will take the resource from Rift's internal data. Anything else will take the resource from the addon with that identifier.

texture (String)
The actual texture identifier. Either a resource identifier or a filename.

EVENTS