

# RDoc Documentation

This is the API documentation for 'RDoc Documentation'.

# Classes/Modules

## [CIGUIERR](#)

 [CIGUIERR::CannotCreateWindow](#)

 [CIGUIERR::CantInterpretCommand](#)

 [CIGUIERR::CantReadNumber](#)

 [CIGUIERR::CantReadString](#)

 [CIGUIERR::CantStart](#)

 [CIGUIERR::WrongWindowIndex](#)

## [CIGUI](#)

 [Color](#)

 [Rect](#)

 [Spr3](#)

 [Text](#)

 [Tone](#)

 [Win3](#)

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# module CIGUI

Основной модуль, обеспечивающий работу Cigui.  
Для передачи команд используйте массив \$do, например:

- \$do<<“команда”
- \$do.push(“команда”)

Оба варианта имеют один и тот же результат.  
<br> Перед запуском модуля вызовите метод ::setup.  
<br> Для исполнения команд вызовите метод ::update.<br>

## In Files

-  cigui.rb
-  localize.rb

## Constants

### СМВ

Хэш-таблица всех возможных сочетаний слов из VOCAB и их положения относительно друг друга в тексте.  
<br> Именно по этим сочетаниям производится поиск команд Cigui в тексте.  
Редактировать без понимания правил составления регулярных выражений не рекомендуется.

### VOCAB

Специальный словарь, содержащий все используемые команды Cigui. Предоставляет возможности не только внесения новых слов, но и добавление локализации (перевода) имеющихся (см. `update_by_user`).  
Для удобства поиска разбит по категориям:

-  `common` - общие команды, не имеющие категории;
-  `cigui` - управление интерпретатором;
-  `event` - событиями на карте;
-  `map` - параметрами карты;
-  `picture` - изображениями, используемыми через команды событий;
-  `sprite` - самостоятельными изображениями;
-  `text` - текстом и шрифтами
-  `window` - окнами.

Русификацию этого словаря Вы можете найти в файле `localize.rb` по адресу: [<github.com/deadelf79/CIGUI/>](https://github.com/deadelf79/CIGUI/), если этот файл не приложен к демонстрационной версии проекта, который у Вас есть.

## Attributes

### `last_log`<sup>[R]</sup>

Хранит массив всех произведенных действий (все записи `last`).  
Действия записываются только если параметр `::logging` имеет значение `true`.  
Результаты работы `decimal`, `fraction`, `boolean`, `substring` и их расширенных аналогов не логируются.  
Открыт только для чтения.

### **logging**<sup>[RW]</sup>

Флаг включения/выключения логгирования действий. Если имеет значение true, то записи всех произведенных действий с момента применения значения будут записываться в массив [::last\\_log](#)

### **sprites**<sup>[R]</sup>

Внутренний массив для вывода информации обо всех созданных спрайтах.<br> Открыт только для чтения.

### **windows**<sup>[R]</sup>

Внутренний массив для вывода информации обо всех созданных окнах.<br> Открыт только для чтения.

## Public Class Methods

### **bool(source\_string, prefix="", postfix=")**

Данный метод работает по аналогии с boolean, но производит поиск в строке с учетом указанных префикса (текста перед подстрокой) и постфикса (после подстроки).<br> prefix и postfix могут содержать символы, используемые в регулярных выражениях. Если булево значение в строке не обнаружено, по умолчанию возвращает false.

### **boolean(source\_string)**

Данный метод производит поиск булевого значения (true или false) в строке и возвращает его. Если булево значение в строке не

обнаружено, по умолчанию возвращает false.<br> Слова true и false берутся из словаря VOCAB, что значит, что их локализованные версии также могут быть успешно найдены при поиске.

### **dec(source\_string, prefix="", postfix="", std\_conversion=true)**

Данный метод работает по аналогии с decimal, но производит поиск в строке с учетом указанных префикса (текста перед числом) и постфикса (после числа).<br> Метод не требует обязательного указания символов квадратных и круглых скобок, а также одинарных и двойных кавычек вокруг числа.<br> prefix и postfix могут содержать символы, используемые в регулярных выражениях для более точного поиска.

```
dec('x=1cm', 'x=', 'cm') # => 1  
dec('y=120 m', '[xy]=', '[\s]*(?:cm|m|km)') # => 120
```

В отличие от frac, возвращает целое число. <br> Метод работает вне зависимости от работы модуля - нет необходимости запускать для вычисления setup и update.

### **decimal(source\_string, std\_conversion=true)**

Данный метод возвращает первое попавшееся целое число, найденное в строке source\_string.<br> Производит поиск только в том случае, если число записано:

- в скобки, например (10);
- в квадратные скобки, например [23];
- в кавычки(апострофы), например '45';

 в двойные кавычки, например “8765”.

Также, вернет всю целую часть числа записанную:

 до точки, так здесь [1.35] вернет 1;

 до запятой, так здесь (103,81) вернет 103;

 до первого пробела (при стандартной конвертации в целое число), так здесь “816 586,64” вернет только 816;

 через символ подчеркивания, так здесь '1\_000\_0\_0\_0,143' вернет ровно один миллион (1000000).

Если присвоить `std_conversion` значение `false`, то это отменит стандартную конвертацию строки, встроенную в Ruby, и метод попытается найти число, игнорируя пробелы, табуляцию и знаки подчеркивания. Выключение `std_conversion` может привести к неожиданным последствиям.

```
decimal( '[10,25]' ) # => 10  
decimal( '[1 0 2]', false ) # => 102  
decimal( '[1_234_5678 89]', false ) # => 123456789
```

<br> Метод работает вне зависимости от работы модуля - нет необходимости запускать для вычисления `setup` и `update`.

### **frac(source\_string, prefix="", postfix="", std\_conversion=true)**

Данный метод работает по аналогии с `fraction`, но производит поиск в строке с учетом указанных префикса (текста перед числом) и постфикса (после числа).<br> Метод не требует обязательного указания символов квадратных и круглых скобок, а также одинарных и двойных кавычек вокруг числа.<br> `prefix` и `postfix` могут содержать символы, используемые в регулярных

выражениях для более точного поиска.

```
frac('x=31.2mm', 'x=', 'mm') # => 31.2  
frac('y=987,67 m', '[xy]=' , '[\s]*(?:cm|m|km)') # =>
```

В отличие от `dec`, возвращает рациональное число. <br> Метод работает вне зависимости от работы модуля - нет необходимости запускать для вычисления `setup` и `update`.

### **`fraction(source_string, std_conversion=true)`**

Данный метод работает по аналогии с `decimal`, но возвращает рациональное число (число с плавающей запятой или точкой).<br> Имеется пара замечаний к правилам использования в дополнение к упомянутым в `decimal`:

- Все цифры после запятой или точки считаются дробной частью и также могут содержать помимо цифр символ подчеркивания;
- При наличии между цифрами в дробной части пробела вернет ноль (в стандартной конвертации в целое или дробное число).

Если присвоить `std_conversion` значение `false`, то это отменит стандартную конвертацию строки, встроенную в `Ruby`, и метод попыбует найти число, игнорируя пробелы, табуляцию и знаки подчеркивания. Выключение `std_conversion` может привести к неожиданным последствиям.

```
fraction('(109,86)') # => 109.86  
fraction('(1 0 9 , 8 6)', false) # => 109.86
```

<br> Метод работает вне зависимости от работы модуля - нет необходимости запускать для

вычисления setup и update.

### last()

Возвращает сообщение о последнем произведенном действии или классе последнего использованного объекта, используя метод Kernel.inspect.<br> **Пример:**

```
CIGUI.setup
puts CIGUI.last # => 'CIGUI started'
```

### rect(source\_string)

Возвращает массив из четырех значений для передачи в качестве параметра в объекты класса [Rect](#). Массив в строке должен быть помещен в квадратные скобки, а значения в нем должны разделяться **точкой с запятой**.<br> **Пример:**

```
rect('[1;2,0;3.5;4.0_5]') # => [ 1, 2.0, 3.5, 4.0]
```

### setup()

Требуется выполнить этот метод перед началом работы с [CIGUI](#).<br> Инициализирует массив \$do, если он еще не был создан. В этот массив пользователь подает команды для исполнения при следующем запуске метода update.<br> Если даже массив \$do был инициализирован ранее, то исполняет команду *cigui start* прежде всего.<br>

**Пример:**

```
begin
  CIGUI.setup
  #~~~ some code fragment ~~~
  CIGUI.update
  #~~~ some other code fragment ~~~
end
```

---

### 📦 **substr(source\_string, prefix="", postfix=")**

Данный метод работает по аналогии с `substring`, но производит поиск в строке с учетом указанных префикса (текста перед подстрокой) и постфикса (после подстроки).  
Указание квадратных или круглый скобок, а также экранированных одинарных или двойных кавычек в строке после префикса обязательно. `prefix` и `postfix` могут содержать символы, используемые в регулярных выражениях для более точного поиска.

**Пример:**

```
puts 'who can make me strong?'
someone = substring("Make[ me ]invincible", 'Make')
puts 'Only'+someone # => 'Only me'
```

Метод работает вне зависимости от работы модуля - нет необходимости запускать для вычисления `setup` и `update`.

### 📦 **substring(source\_string)**

Данный метод производит поиск подстроки, используемой в качестве параметра.  
Строка должна быть заключена в одинарные или двойные кавычки или же в круглые или квадратные скобки. **Пример:**

```
substring('[Hello cruel world!}') # => Hello cruel world!
substring("set window label='SomeSome' and no more") # => SomeSome
```

### 📦 **update(clear\_after\_update=true)**

Вызывает все методы обработки команд, содержащиеся в массиве `$do`.  
Вызовет

исключение [CIGUIERR::CantInterpretCommand](#) в том случае, если после выполнения *cigui finish* в массиве \$do будут находиться новые команды для обработки.<br> По умолчанию очищает массив \$do после обработки всех команд. Если `clear_after_update` поставить значение `false`, то все команды из массива \$do будут выполнены повторно при следующем запуске этого метода.<br> Помимо приватных методов обработки вызывает также метод `update_by_user`, который может быть модифицирован пользователем (подробнее смотри в описании метода).<br>

### **update\_by\_user(string)**

Метод обработки текста, созданный для пользовательских модификаций, не влияющих на работу встроенных обработчиков.<br> Используйте *alias* этого метода при добавлении обработки собственных команд.<br> **Пример:**

```
alias my_update update_by_user
def update_by_user
  # add new word
  VOCAB[:window][:throw]='throw'
  # add 'window throw' combination
  CMB[:window_throw]="(?:#{VOCAB[:window
  # call method
  window_throw? line
end
```

### **update\_internal\_objects()**

Вызывает обновление всех объектов из внутренних массивов [::windows](#) и [::sprites](#).<br> Вызывается автоматически по окончании обработки команд из массива \$do в методе `update`.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# module CIGUIERR

Модуль, содержащий данные обо всех возможных ошибках, которые может выдать [CIGUI](#) при некорректной работе.<br>Включает в себя:

- [CIGUIERR::CantStart](#)
- [CIGUIERR::CantReadNumber](#)
- [CIGUIERR::CantReadString](#)
- [CIGUIERR::CantInterpretCommand](#)
- [CIGUIERR::CannotCreateWindow](#)
- [CIGUIERR::WrongWindowIndex](#)

## In Files

 [cigui.rb](#)

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class CIGUIERR::CannotCreateWin

Ошибка создания окна.

## In Files

 cigui.rb

## Parent

StandardError

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class CIGUIERR::CantInterpretCor

Ошибка, которая появляется при попытке работать с Cigui после вызова команды *cigui finish*.

## In Files

 cigui.rb

## Parent

StandardError

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

# class

## CIGUIERR::CantReadNumbe

Ошибка, которая появляется, если в строке не было обнаружено числовое значение.<br>Правила оформления строки указаны для каждого типа значений отдельно:

- целые числа - [CIGUI.decimal](#)
- дробные числа - [CIGUI.fraction](#)

### In Files

 [cigui.rb](#)

### Parent

[StandardError](#)

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class

## CIGUIERR::CantReadString

Ошибка, которая появляется, если в строке не было обнаружено строчное значение.<br>Правила оформления строки и примеры использования указаны в описании к этому методу: CIGUI.string

### In Files

 cigui.rb

### Parent

StandardError

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class CIGUIERR::CantStart

Ошибка, которая появляется при неудачной попытке запуска интерпретатора Cigui

## In Files

 cigui.rb

## Parent

StandardError

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

# class CIGUIERR::WrongWindowIn

Ошибка, которая появляется при попытке обращения к несуществующему индексу в массиве *windows*  
Вызывается при некорректном вводе пользователя

## In Files

 cigui.rb

## Parent

StandardError

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class Color

Класс, хранящий данные о цвете в формате RGBA (красный, зеленый, синий и прозрачность). Каждое значение является рациональным числом (число с плавающей точкой) и имеет значение от 0.0 до 255.0. Все значения, выходящие за указанный интервал, корректируются автоматически.

## In Files

 cigui.rb

## Parent

Object

## Public Class Methods

### **new(r,g,b,a=255.0)**

Создает экземпляр класса.

 r, g, b - задает изначальные значения красного, зеленого и синего цветов

 a - задает прозрачность, по умолчанию имеет значение 255.0 (полностью непрозрачный цвет)

## Public Instance Methods

### **alpha()**

Возвращает значение прозрачности

### **blue()**

Возвращает значение синего цвета

### **green()**

Возвращает значение зеленого цвета

### **red()**

Возвращает значение красного цвета

### **set(\*args)**

Задаёт новые значения цвета и прозрачности.<br>

**Варианты параметров:**

 `set(Color)` - в качестве параметра задан другой экземпляр класса [Color](#)

Все значения цвета и прозрачности будут скопированы из него.

 `set(red, green, blue)` - задаёт новые значения цвета.

Прозрачность по умолчанию становится равна 255.0

 `set(red, green, blue, alpha)` - задаёт новые значения цвета и прозрачности.

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class Rect

Класс абстрактного (невизуального) прямоугольника. Хранит значения о положении и размере прямоугольника

## In Files

 cigui.rb

## Parent

Object

## Attributes

 **height**<sup>[RW]</sup>  
Высота прямоугольника

 **width**<sup>[RW]</sup>  
Ширина прямоугольника

 **x**<sup>[RW]</sup>  
Координата X

 **y**<sup>[RW]</sup>  
Координата Y

## Public Class Methods

### **new(x,y,width,height)**

Создание прямоугольника

 x, y - назначает положение прямоугольника в пространстве

 width, height - устанавливает ширину и высоту прямоугольника

## Public Instance Methods

### **empty()**

Устанавливает все параметры прямоугольника равными нулю.

### **set(\*args)**

Задаёт все параметры одновременно. Может принимать значения:

 [Rect](#) - другой экземпляр класса [Rect](#), все значения копируются из него

 x, y, width, height - позиция и размер прямоугольника

```
# Оба варианта работают аналогично:  
set(Rect.new(0, 0, 192, 64))  
set(0, 0, 192, 64)
```

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class Spr3

Класс спрайта со всеми параметрами, доступными в RGSS3. Пока пустой, ожидается обновление во время работы над спрайтами (ветка work-with-sprites в github).

## In Files

 cigui.rb

## Parent

Object

## Public Class Methods

 **new()**  
Создает новый спрайт

## Public Instance Methods

 **dispose()**  
Удаляет спрайт

 **refresh()**  
Производит повторную отрисовку содержимого спрайта

## **update()**

Производит обновление спрайта

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class Text

Класс, хранящий данные о тексте в окне.  
Создается для каждого окна отдельно, имеет индивидуальные настройки.

## In Files

 cigui.rb

## Parent

Object

## Attributes

 **bold**<sup>[RW]</sup>

Устанавливает жирность шрифта для **всего** текста. `<br>` Игнорирует тег `< b >` в тексте

 **colorset**<sup>[RW]</sup>

Массив цветов для отрисовки текста, по умолчанию содержит 32 цвета

 **font**<sup>[RW]</sup>

Гарнитура (название) шрифта. По умолчанию - Таhоmа

 **italic**<sup>[RW]</sup>

Устанавливает наклон шрифта (курсив) для **всего** текста.<br> Игнорирует тег < i > в тексте

 **out\_color**<sup>[RW]</sup>

Переменная класса Color, содержит цвет обводки текста.

 **outline**<sup>[RW]</sup>

Булевая переменная (принимает только значения true или false), которая отвечает за отрисовку обводки текста. По умолчанию, обводка включена (outline=true)

 **shadow**<sup>[RW]</sup>

Булевая переменная (принимает только значения true или false), которая отвечает за отрисовку тени от текста. По умолчанию, тень выключена (shadow=false)

 **size**<sup>[RW]</sup>

Размер шрифта. По умолчанию - 20

 **string**<sup>[RW]</sup>

Строка текста, которая будет отображена при использовании экземпляра класса.

 **underline**<sup>[RW]</sup>

Устанавливает подчеркивание шрифта для **всего** текста.<br> Игнорирует тег < u > в тексте

## **windowskin<sup>[RW]</sup>**

Название файла изображения, из которого загружаются данные для отрисовки окон.<br> По умолчанию задан путь 'GraphicsSystemWindow.png'.

## Public Class Methods

### **new(string, font\_family=['Tahoma'], font\_size=20, bold=false, italic=false, underline=false)**

Создает экземпляр класса.<br> **Параметры:**

 string - строка текста

 font\_family - массив названий (гарнитур) шрифта, по умолчанию

имеет только “Tahoma”. При выборе гарнитуры шрифта, убедитесь в том, что символы, используемые в тексте, корректно отображаются при использовании данного шрифта.

 font\_size - размер шрифта, по умолчанию равен 20 пунктам

 bold - **жирный шрифт** (по умолчанию - false)

 italic - *курсив* (по умолчанию - false)

 underline - подчеркнутый шрифт (по умолчанию - false)

## Public Instance Methods

### **default\_colorset()**

Восстанавливает первоначальные значения цвета. По возможности, эти данные загружаются из файла

### **empty()**

Сбрасывает все данные, кроме colorset, на значения по умолчанию

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

# class Tone

Класс, хранящий данные о тонировании в виде четырех значений: красный, зеленый, синий и насыщенность. Последнее значение влияет на цветовую насыщенность, чем ниже его значение, тем сильнее цветовые оттенки заменяются на оттенки серого. Каждое значение, кроме последнего, является рациональным числом (число с плавающей точкой) и имеет значение от -255.0 до 255.0. Значение насыщенности лежит в пределах от 0 до 255. Все значения, выходящие за указанные интервалы, корректируются автоматически.

## In Files

 cigui.rb

## Parent

Object

## Public Class Methods

 **new(r,g,b,gs=0.0)**

Создает экземпляр класса.

 r, g, b - задает изначальные значения

красного, зеленого и синего цветов

 gs - задает насыщенность, по умолчанию имеет значение 0

## Public Instance Methods

### **blue()**

Возвращает значение синего цвета

### **gray()**

Возвращает значение насыщенности

### **green()**

Возвращает значение зеленого цвета

### **red()**

Возвращает значение красного цвета

### **set(\*args)**

Задаёт новые значения цвета и насыщенности.

<br> **Варианты параметров:**

 set(Tone) - в качестве параметра задан другой экземпляр класса Tone

Все значения цвета и прозрачности будут скопированы из него.

 set(red, green, blue) - задает новые значения цвета.

Прозрачность по умолчанию становится равна 255.0

 `set(red, green, blue, greyscale)` - задает  
новые значения цвета и насыщенности.

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

# class Win3

Класс окна с реализацией всех возможностей, доступных при помощи Cigui.  
<br> Реализация выполнена для RGSS3.

## In Files

 cigui.rb

## Parent

Window\_Selectable

## Attributes

 **back\_opacity**<sup>[RW]</sup>

Прозрачность фона окна. Может принимать значения от 0 до 255

 **label**<sup>[R]</sup>

Метка окна. Строка, по которой происходит поиск экземпляра в массиве [CIGUI.windows](#) при выборе окна по метке (select by label)

 **opacity**<sup>[RW]</sup>

Прозрачность окна. Может принимать значения от 0 до 255

 **speed**<sup>[RW]</sup>  
Скорость перемещения

## Public Class Methods

 **new(x=0,y=0,w=192,h=64)**  
Создает окно. По умолчанию задается размер 192x64 и помещается в координаты 0, 0

## Public Instance Methods

 **add\_item(command,procname,enabled=true, text\_only=false)**

Этот метод добавляет команду во внутренний массив *items*. Команды используются для отображения кнопок.<br>

 `command` - отображаемый текст кнопки

 `procname` - название вызываемого метода (String или Symbol)

По умолчанию значение `enable` равно `true`, что значит, что кнопка включена и может быть нажата.

 **add\_text(text)**

Этот метод позволяет добавить текст в окно.<br>Принимает в качестве параметра значение класса [Text](#)

 **delete\_item(indexORcomORproc)**

Метод удаляет команду из внутреннего массива

*items*.<br> В качестве аргумента принимаются:

- 📄 индекс команды
- 📄 имя команды
- 📄 имя вызываемой процедуры
- 📄 регулярное выражение для продвинутого поиска по именам команд и процедур

### 📦 **disable\_item(commandORindex)**

Выключает кнопку.<br> В параметр *commandORindex* помещается либо строковое значение, являющееся названием кнопки, либо целое число - индекс кнопки во внутреннем массиве *items*.

```
disable_item(0) # => @items[0].enabled set 'false'  
disable_item('New game') # => @items[0].enabled s
```

Выключение происходит только если кнопка не имеет тип *text\_only* (устанавливается при добавлении с помощью метода [add\\_text](#)).

### 📦 **dispose()**

Удаляет окно и все связанные с ним ресурсы

### 📦 **draw\_items(ignore\_disabled=false)**

С помощью этого метода производится полная отрисовка всех элементов из массива *items*.<br> Параметр *ignore\_disabled* отвечает за отображение выключенных команд из массива *items*. Если его значение равно *true*, то отрисовка выключенных команд производиться не будет.

## **enable\_item(commandORindex)**

Включает кнопку.<br> В параметр `commandORindex` помещается либо строковое значение, являющееся названием кнопки, либо целое число - индекс кнопки во внутреннем массиве `items`.

```
@items.clear
add_item('New game', :new_game, false)
enable_item(0) # => @items[0].enabled set 'true'
enable_item('New game') # => @items[0].enabled se
enable_item(/[Nn][Ee][Ww][\s_]*[Gg][Aa][Mm][Ee]/)
```

Включение происходит только если кнопка не имеет тип `text_only` (устанавливается при добавлении с помощью метода [add\\_text](#)).

## **inspect()**

Возвращает полную информацию обо всех параметрах в формате строки

## **label=(string)**

Задаёт метку окну, проверяя ее на правильность перед этим:

-  удаляет круглые и квадратные скобки
-  удаляет кавычки
-  заменяет пробелы и табуляцию на символы подчеркивания
-  заменяет символы “больше” и “меньше” на символы подчеркивания

## **refresh()**

Обновляет окно. В отличие от [update](#), влияет

только на содержимое окна (производит повторную отрисовку).

### **resize(new\_width, new\_height)**

Устанавливает новые размеры окна дополнительно изменяя также и размеры содержимого (contents).<br> Все части содержимого, которые не помещаются в новые размеры, удаляются безвозвратно.

### **update()**

Обновляет окно. Влияет только на положение курсора (параметр cursor\_rect), прозрачность и цветовой тон окна.

Generated by [RDoc 3.12.2](#).

Generated with the [Darkfish Rdoc Generator 3](#).

# RDoc Documentation

This is the API documentation for 'RDoc Documentation'.

# Files

 [LICENSE](#)

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE

SOFTWARE.

For more information, please refer to  
<[unlicense.org](http://unlicense.org)>

Generated by [RDoc](#) 3.12.2.

Generated with the [Darkfish Rdoc Generator](#) 3.