libgdtl documentation

Release latest

Nov 26, 2019

GENERAL

CHAPTER

ONE

LIBGDTL INTRODUCTION

BiDi, shaping and basic text layout for Godot Engine.

1.1 Dependencies

- Godot 3.1+
- C++14 compiler
- Meson build system (for gdnative module build only)
- SCons build system

1.2 Compiling (as builtin module)

1.2.1 Build options

Name	Description	Default value
builtin_runtime	Use the built-in libraries	true
use_graphite2	Enable SIL Graphite 2 complementary shaper	true
use_font_wrapper	Enable Godot font wrapper for default controls	false

If *use_font_wrapper* is enabled, apply *patch_font.diff* from the root of this repository to the Godot engine source first.

1.2.2 Building *libdgtl* module

Clone this repository (without *–recursive* flag) into Godot's *modules* subfolder as *godot_tl*. Rebuild Godot engine as usual.

1.3 Compiling (as gdnative module)

1.3.1 Build options

Name	Description	Default value
godot-cpp-lib-name	godot-cpp static library name (without *.a* or *.lib* extension)	libgodot-cpp
static-lib	Build static library	false
static-runtime	Link libraries statically for better portability	false
builtin-runtime	Use the built-in libraries	false
use-graphite2	Enable SIL Graphite 2 complementary shaper	true

1.3.2 Building *godot-cpp* static library

See https://github.com/GodotNativeTools/godot-cpp/blob/master/README.md#compiling-the-cpp-bindings-library

1.3.3 Building *libdgtl* module

You can compile this module by executing:

```
meson {Targer-Folder} -Dgodot-cpp-lib-name={Godot-CPP-Name} --buildtype=release
ninja -C {Targer-Folder}
```

1.3.4 Exporting projects with *libgdtl* module (iOS)

After exporting Xcode project form the Godot editor, add module static library and its dependencies to the exported project.

Navigate to *Targets* > Exported App Name > *General* > *Frameworks, Libraries, and Embedded Content* and drag-and-drop following files to the list:

- *libgdtl.a*
- *libgodot-cpp.a*
- *libfreetype2.a*
- *libgraphite2.a* (optional)
- *libharfbuzz.a*
- *libicu4c.a*
- *libpng.a*
- *libzlib.a*

1.4 License

- The source code of the **libgdtl** module is released under unlicense.

  For more information, see http://unlicense.org/ or the accompanying UNLICENSE file.

- **Godot** and **GodotNativeTools** are licensed under MIT license.

  For more information, see https://github.com/godotengine/godot/blob/master/LICENSE.txt.

- **HarfBuzz** is licensed under MIT-like License.

  For more information, see https://github.com/harfbuzz/harfbuzz/blob/master/COPYING

- **ICU4C** is licensed under Unicode, Inc. License.

  For more information, see http://www.unicode.org/copyright.html#License

- **FreeType** is licensed under FreeType License (BSD-like) or GNU General Public License (GPL), version 2.

  For more information, see https://www.freetype.org/license.html

- **SIL Graphite engine** is licensed under GNU Lesser General Public License (LGPL), version 2.1+ or GNU General Public License (GPL), version 2 or Mozilla Public License.

  For more information, see https://github.com/silnrsi/graphite/blob/master/COPYING

1.5 Demo data

Montserrat (https://github.com/JulietaUla/Montserrat/), Awami Nastaliq (https://software.sil.org/awami/download/), Comic Neue (http://comicneue.com/) and Noto (https://www.google.com/get/noto/) fonts are published under the SIL Open Font License, Version 1.1 (https://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&id=OFL)

Material Design icons by Google (https://github.com/google/material-design-icons) are published under the Apache License Version 2.0 (https://www.apache.org/licenses/LICENSE-2.0.txt)

Noto Color Emoji font is cut down to single glyph (U+1F604) using glyphhanger (https://github.com/filamentgroup/glyphhanger).

CHAPTER

TWO

INSTALLATION

2.1 Module

Use Godot editor and export templates compiled with the module.

Module is ready to use, no modification to the project is required.

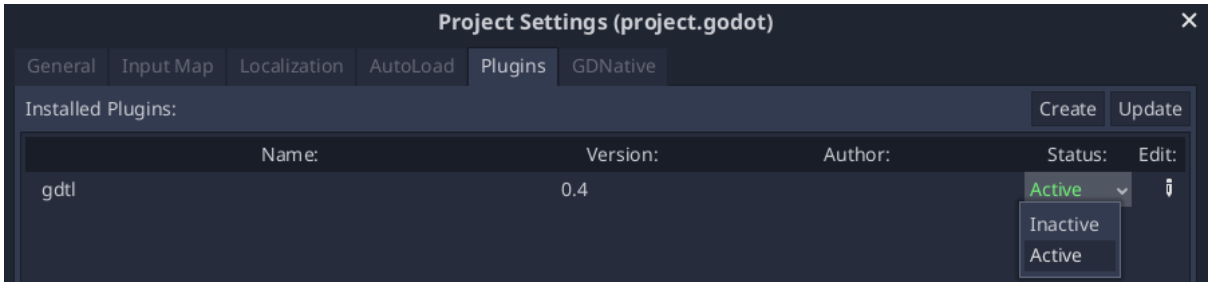
2.2 GDNative

GDNative plugin is intended to be used with official Godot editor and export templates.

To install plugin:

1. Create *addons* folder in the root of your project.
2. Copy the contents of archive to the *addons* folder. (Do not drag-and-drop it into *FileSystem* tab of the editor. Right-click *addons* folder, select *Open in File Manager* and use your OS file manager to copy/extract files.)
3. After installing plugin, your file system should look like this:

4. Go to *Project Settings*, click on the *Plugins* tab and activate *gdtl* plugin.


Project Settings (project.godot)
General
Input Map
Localization
AutoLoad
Plugins
GDNative
Installed Plugins:
Create
Update
Name:
Version:
Author:
Status:
Edit:
gdtl
0.4
Active
Inactive
Active

CHAPTER

THREE

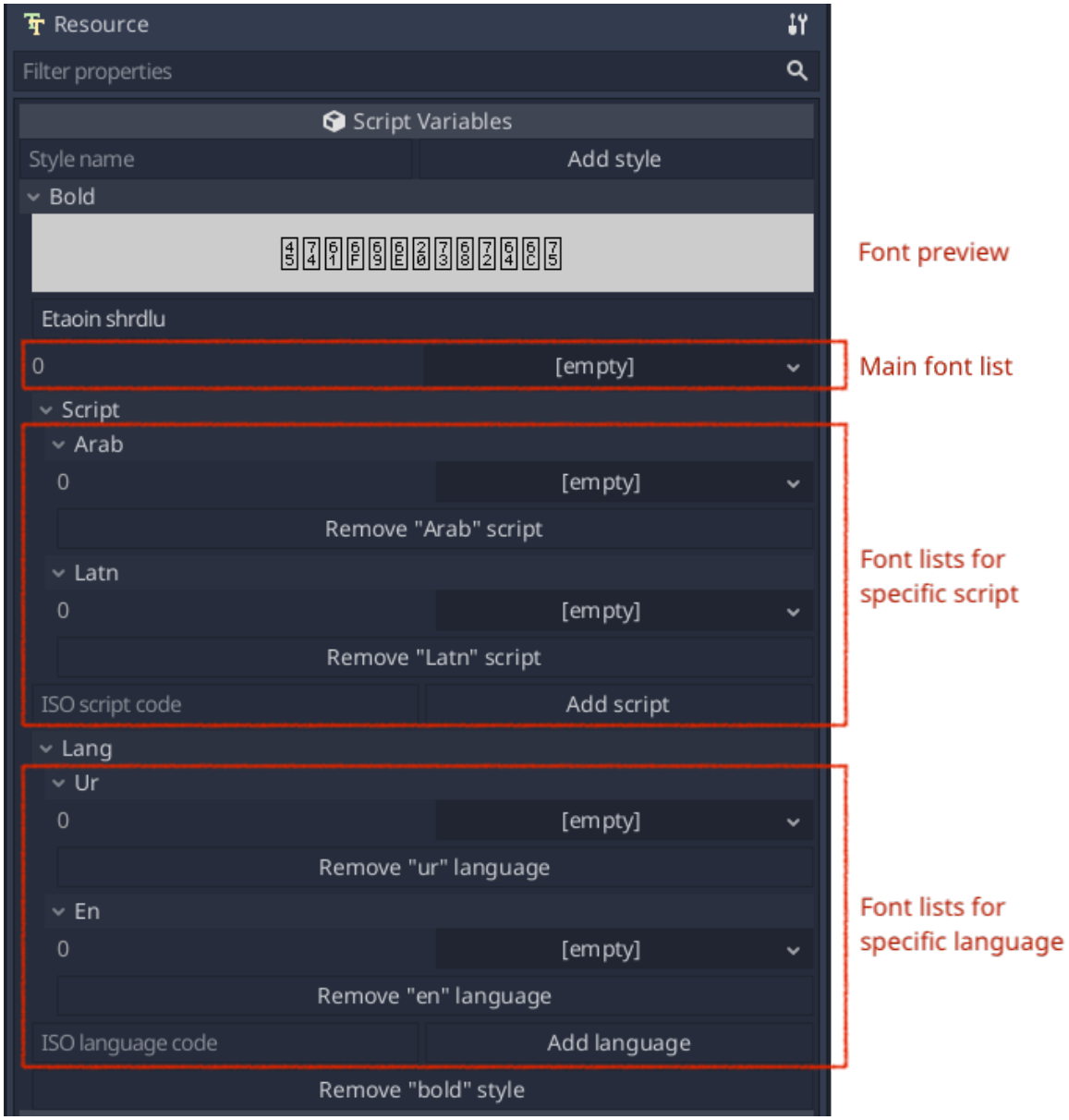
FONT SUBSTITUTION

Font substitution is used to find a replacement for an unavailable character.

In addition to main list of the substitution fonts, you can specify preferred fonts for the script (writing system) and language.

Use ISO 639-1 (https://en.wikipedia.org/wiki/ISO_639-1) codes for the language names, and ISO 15924 (https://en.wikipedia.org/wiki/ISO_15924) for the script names.

Substitution lists have following priority: *Language*, *Script*, *Main List*. Providing script and language information is not required, module will try to detect script (but not the language) automatically, but detection is not always successful and correct. Manual config can improve shaping speed if large number of fonts used.


Resource
Filter properties
Script Variables
Style name
Add style
Bold
Etaoin shrdlu
0
[empty]
Script
Arab
0
[empty]
Remove "Arab" script
Latn
0
[empty]
Remove "Latn" script
ISO script code
Add script
Lang
Ur
0
[empty]
Remove "ur" language
En
0
[empty]
Remove "en" language
ISO language code
Add language
Remove "bold" style
Font preview
Main font list
Font lists for specific script
Font lists for specific language

CHAPTER

FOUR

RICH TEXT TYPESETTING

4.1 Setting formatting for the range of text

1. Select desired attribute and its value in the attribute editor.
2. Enter start and end offsets of the range.
3. Click “Add” button to apply formatting.

Node: This is equivalent of the *add_attribute* function call.

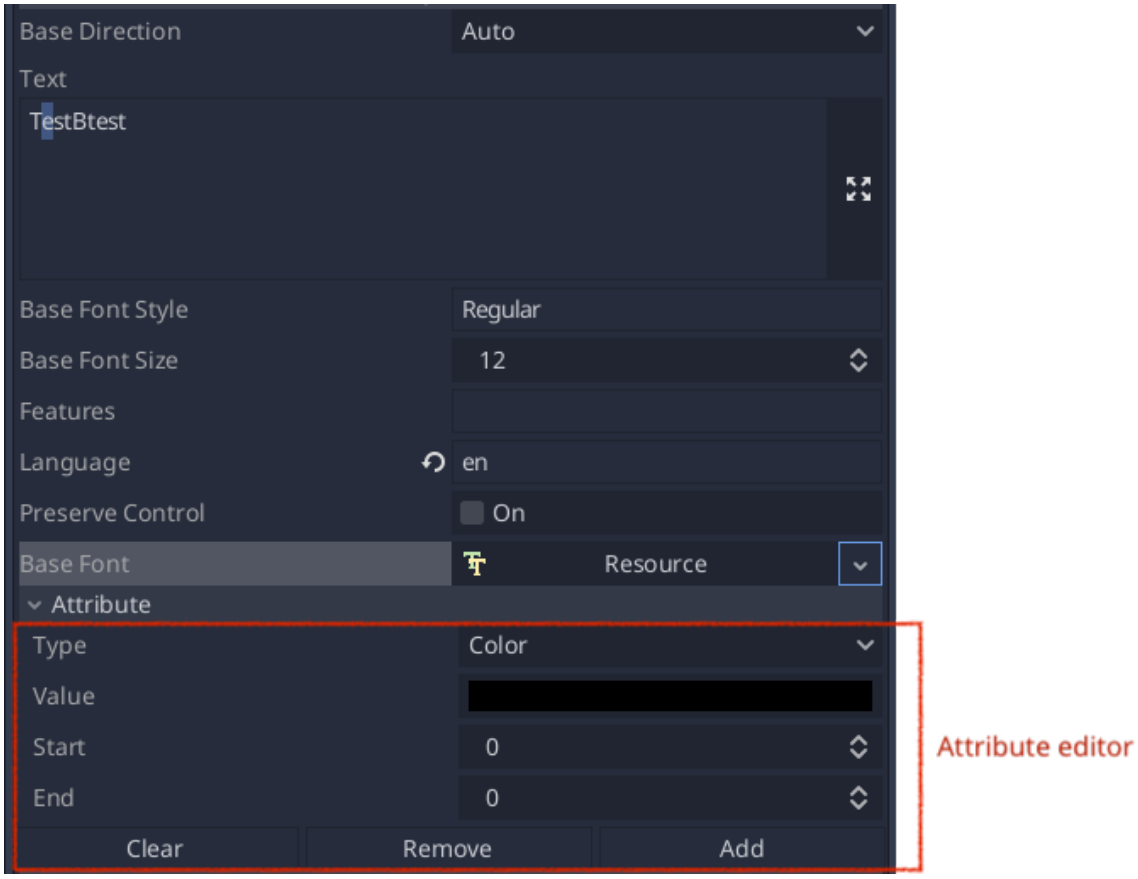
4.2 Removing formatting

1. Enter start and end offsets of the range in the attribute editor.
2. Click “Remove” button to clean all attributes from the range or click “Clean” to remove attributes from entire text.

Node: This is equivalent of the *remove_attributes*/*clear_attributes* function call.

4.3 Embedding child control into the text flow

1. Add control as child of the TLRichTextEdit.
2. Apply “Replacement Rect” attribute with desired size of the control to the single character of the text.
3. Apply “Replacement ID” attribute with the control name to the same character.
4. Use “Replacement VAlign” attribute to set inline alignment of the control.


Base Direction
Auto
Text
TestBtest
Base Font Style
Regular
Base Font Size
12
Features
Language
en
Preserve Control
On
Base Font
Resource
Attribute
Type
Color
Value
Start
0
End
0
Clear
Remove
Add
Attribute editor

CHAPTER

FIVE

LIBGDTL API

5.1 TLBitmapFontFace

Inherits: *TLFontFace*

Category: Core

5.1.1 Brief Description

An AngelCode Bitmap Font Generator bitmap font for drawing text.

5.1.2 Properties

int	texture_flags	**O:** 2048

5.1.3 Description

TLBitmapFontFace have limited shaping support.

TLBitmapFontFace doesn't support OpenType features.

5.2 TLDynamicFontFace

Inherits: *TLFontFace*

Category: Core

5.2.1 Brief Description

A TrueType, OpenType or Graphite font for drawing text.

5.2.2 Properties

bool	*force_autohinter*	false
int	*hinting*	2
float	*oversampling*	1.0
int	texture_flags	**O:** 2048

5.2.3 Methods

bool	*has_graphite* ( ) const

5.2.4 Enumerations

enum **DynamicFaceHinting**:

- **DF_HINTING_NONE = 0** — Disable font hinting (smoother but less crisp)
- **DF_HINTING_LIGHT = 1** — Use the light font hinting mode
- **DF_HINTING_NORMAL = 2** — Use the default font hinting mode (crisper but less smooth)

5.2.5 Property Descriptions

- bool **force_autohinter**

Default	false
Setter	set_force_autohinter(value)
Getter	get_force_autohinter()

If `true`, prefers FreeType auto-hinter over the font's native hinter.

- int **hinting**

Default	2
Setter	set_hinting(value)
Getter	get_hinting()

The font hinting mode used by FreeType auto-hinter.

- float **oversampling**

Default	1.0
Setter	set_oversampling(value)
Getter	get_oversampling()

Font oversampling factor.

5.2.6 Method Descriptions

- bool **has_graphite** ( ) const

Returns `true` if module is built with SIL Graphite 2 shaper support.

5.3 TLFontFace

Inherits:

Inherited By: *TLBitmapFontFace*, *TLDynamicFontFace*

Category: Core

5.3.1 Brief Description

Virtual class

A base font face class.

5.3.2 Properties

String	*font_path*	“”
int	*texture_flags*	0

5.3.3 Methods

void	*draw_glyph* (RID canvas_item, Vector2 pos, int codepoint, Color modulate, int size) const
void	*draw_glyph_outline* (RID canvas_item, Vector2 pos, int codepoint, Color modulate, int size) const
float	*get_ascent* (int size) const
int	*get_base_size* ( ) const
float	*get_descent* (int size) const
Array	*get_glyph_outline* (Vector2 pos, int codepoint, int size) const
float	*get_height* (int size) const
bool	*load* (String resource_path)
Array	*unicode_scripts_supported* ( ) const

5.3.4 Property Descriptions

- String **font_path**

Default	“”
Setter	set_font_path(value)
Getter	get_font_path()

Path of the font file.

- int **texture_flags**

Default	0
Setter	set_texture_flags(value)
Getter	get_texture_flags()

Underlaying textures flags, see Texture class documentation https://docs.godotengine.org/en/latest/classes/class_texture.html#enumerations for more info.

5.3.5 Method Descriptions

- void **draw_glyph** (RID canvas_item, Vector2 pos, int codepoint, Color modulate, int size) const

Draws a single glyph.

- void **draw_glyph_outline** (RID canvas_item, Vector2 pos, int codepoint, Color modulate, int size) const

Draws single glyph outline.

- float **get_ascent** (int size) const

Returns ascent (distance from the baseline to the highest position characters extend to) of the font.

- int **get_base_size** ( ) const

Returns default font size for bitmap fonts or `0` for dynamic fonts.

- float **get_descent** (int size) const

Returns descent (distance from the base line to the lowest point characters extend to) of the font.

- Array **get_glyph_outline** (Vector2 pos, int codepoint, int size) const

Returns array of glyph counture points, if it's supported by font implementation or empty array otherwise.

- float **get_height** (int size) const

Returns height (vertical distance between two consecutive baselines) of the font.

- bool **load** (String resource_path)

Loads font from speified file.

- Array **unicode_scripts_supported** ( ) const

Returns array of supported ISO 15924 script codes.

5.4 TLFontFamily

Inherits:

Category: Core

5.4.1 Brief Description

A set of fonts that make up a font family.

5.4.2 Properties

TLFontFace	*{style_name}/{index}*
TLFontFace	*{style_name}/script/{tag}/{index}*
TLFontFace	*{style_name}/script/{lang}/{index}*

5.4.3 Methods

void	*add_face* (String style, *TLFontFace* ref)
void	*add_face_for_language* (String style, *TLFontFace* ref, String lang)
void	*add_face_for_script* (String style, *TLFontFace* ref, String script)
void	*add_face_unlinked* (String style, *TLFontFace* ref)
void	*add_language* (String style, String language)
void	*add_script* (String style, String script)
void	*add_style* (String style)
TLFontIterator	*get_face* (String style) const
TLFontIterator	*get_face_for_language* (String style, String lang) const
TLFontIterator	*get_face_for_script* (String style, String script) const
bool	*has_style* (String style) const
void	*remove_language* (String style, String language)
void	*remove_script* (String style, String script)
void	*remove_style* (String style)

5.4.4 Property Descriptions

- *TLFontFace* **{style_name}/{index}**

Font faces not linked to the language or script.

- *TLFontFace* **{style_name}/script/{tag}/{index}**

Font faces linked to the script.

- *TLFontFace* **{style_name}/script/{lang}/{index}**

Font faces linked to the language.

5.4.5 Method Descriptions

- void **add_face** (String style, *TLFontFace* ref)

Adds font face to be used for specific style. Supported scripts are detected automatically.

- void **add_face_for_language** (String style, *TLFontFace* ref, String lang)

Adds font face to be used for specific language and style.

- void **add_face_for_script** (String style, *TLFontFace* ref, String script)

Adds font face to be used for specific script and style.

- void **add_face_unlinked** (String style, *TLFontFace* ref)

Adds font face to be used for specific style, without associating it with the script or language.

- void **add_language** (String style, String language)

Creates new font fallback list for the language, list is created automatically when add_face or add_face_for_language function is called.

Use ISO 639-1 (https://en.wikipedia.org/wiki/ISO_639-1) codes for the language names.

- void **add_script** (String style, String script)

Creates new font fallback list for the script, list is created automatically when add_face or add_face_for_script function is called.

Use ISO 15924 (https://en.wikipedia.org/wiki/ISO_15924) for the script names.

- void **add_style** (String style)

Adds new style to the family, style is created automatically when any of the add_face functions is called.

- *TLFontIterator* **get_face** (String style) const

Returns font iterator for the specific style.

- *TLFontIterator* **get_face_for_language** (String style, String lang) const

Returns font iterator for the specific style and supported language.

- *TLFontIterator* **get_face_for_script** (String style, String script) const

Returns font iterator for the specific style and supported script.

- bool **has_style** (String style) const

Returns `true` if font family has style.

- void **remove_language** (String style, String language)

Removes font fallback list for the language.

- void **remove_script** (String style, String script)

Removes font fallback list for the script.

- void **remove_style** (String style)

Removes style from the font family.

5.5 TLFontIterator

Inherits:

Category: Core

5.5.1 Brief Description

5.5.2 Methods

bool	*is_linked* ( ) const
bool	*is_valid* ( ) const
bool	*next* ( )
TLFontFace	*value* ( ) const

5.5.3 Method Descriptions

- bool **is_linked** ( ) const

Return `true` if current iterator value is part of language/script specific fallback list.

- bool **is_valid** ( ) const

Return `true` if current iterator value is valid.

- bool **next** ( )

Advances iterator. Return `true` if current iterator value is valid.

- *TLFontFace* **value** ( ) const

Return current font face.

5.6 TLGDFontWrapper

Inherits:

Category: Core

5.6.1 Brief Description

5.6.2 Properties

TLFontFamily	*base_font*	
int	*base_font_size*	12
String	*base_font_style*	“Regular”
int	*cache_depth*	100

5.6.3 Description

Subclass Font to use with default Godot controls, provides limited shaping support.

Note: This class is only available if module is built with `use_font_wrapper=true` flag.

5.6.4 Property Descriptions

- *TLFontFamily* **base_font**

Setter	set_base_font(value)
Getter	get_base_font()

Base font to wrap.

- int **base_font_size**

Default	12
Setter	set_base_font_size(value)
Getter	get_base_font_size()

Base font size.

- String **base_font_style**

Default	“Regular”
Setter	set_base_font_style(value)
Getter	get_base_font_style()

Base font style.

- int **cache_depth**

Default	100
Setter	set_cache_depth(value)
Getter	get_cache_depth()

Number of shaped strings to keep in cache.

5.7 TLICUDataLoader

Inherits:

Category: Core

5.7.1 Brief Description

Helper class that handles ICU data loading.

5.7.2 Properties

String	*data_path*	“”

5.7.3 Methods

bool	*load* (String resource_path)

5.7.4 Property Descriptions

- String **data_path**

Default	“”
Setter	set_data_path(value)
Getter	get_data_path()

ICU data file path.

5.7.5 Method Descriptions

- bool **load** (String resource_path)

Loads ICU data file, should be done at most once in a process, before the first ICU operation. Returns `true` if function succeeds.

5.8 TLLabel

Inherits:

Category: Core

5.8.1 Brief Description

Displays plain text in a line or wrapped inside a rectangle. For formatted text, use *TLRichTextEdit*.

Copy of Godot Label with the shaping support. TLLabel uses the same theme items as the Label.

5.8.2 Properties

int	*align*	0
bool	*autowrap*	false
TLFontFamily	*base_font*	
int	*base_font_size*	12
String	*base_font_style*	“Regular”
bool	*clip_text*	false
String	*language*	“”
int	*lines_skipped*	0
int	*max_lines_visible*	-1
int	mouse_filter	**O:** 2
String	*ot_features*	“”
float	*percent_visible*	1.0
int	size_flags_vertical	**O:** 4
String	*text*	“”
int	*text_direction*	3
bool	*uppercase*	false
int	*valign*	0
int	*visible_characters*	-1

5.8.3 Methods

int	*get_line_count* ( ) const
int	*get_line_height* ( ) const
int	*get_total_character_count* ( ) const
int	*get_visible_line_count* ( ) const

5.8.4 Enumerations

enum **Align**:

- **ALIGN_LEFT = 0** — Align rows to the left (default).
- **ALIGN_CENTER = 1** — Align rows centered.
- **ALIGN_RIGHT = 2** — Align rows to the right.
- **ALIGN_FILL = 3** — Expand row white spaces to fit the width.

---

enum **VAlign**:

- **VALIGN_TOP = 0** — Align the whole text to the top.
- **VALIGN_CENTER = 1** — Align the whole text to the center.
- **VALIGN_BOTTOM = 2** — Align the whole text to the bottom.
- **VALIGN_FILL = 3** — Align the whole text by spreading the rows.

5.8.5 Description

Label displays plain text on the screen. It gives you control over the horizontal and vertical alignment, and can wrap the text inside the node's bounding rectangle. It doesn't support bold, italics or other formatting. For that, use RichTextLabel instead.

5.8.6 Property Descriptions

- int **align**

Default	0
Setter	set_align(value)
Getter	get_align()

Controls the text's horizontal align. Supports left, center, right, and fill, or justify. Set it to one of the `ALIGN_*` constants.

- bool **autowrap**

Default	false
Setter	set_autowrap(value)
Getter	has_autowrap()

If `true`, wraps the text inside the node's bounding rectangle. If you resize the node, it will change its height automatically to show all the text.

- *TLFontFamily* **base_font**

Setter	set_base_font(value)
Getter	get_base_font()

Base font.

- int **base_font_size**

Default	12
Setter	set_base_font_size(value)
Getter	get_base_font_size()

Base font size.

- String **base_font_style**

Default	"Regular"
Setter	set_base_font_style(value)
Getter	get_base_font_style()

Base font style.

- bool **clip_text**

Default	false
Setter	set_clip_text(value)
Getter	is_clipping_text()

If `true`, the Label only shows the text that fits inside its bounding rectangle. It also lets you scale the node down freely.

- String **language**

Default	""
Setter	set_language(value)
Getter	get_language()

Language code for line-breaking and text shaping algorithms.

- int **lines_skipped**

Default	0
Setter	set_lines_skipped(value)
Getter	get_lines_skipped()

The node ignores the first `lines_skipped` lines before it starts to display text.

- int **max_lines_visible**

Default	-1
Setter	set_max_lines_visible(value)
Getter	get_max_lines_visible()

Limits the lines of text the node shows on screen.

- String **ot_features**

Default	“”
Setter	set_ot_features(value)
Getter	get_ot_features()

Comma separated list of OpenType feature tags.

- float **percent_visible**

Default	1.0
Setter	set_percent_visible(value)
Getter	get_percent_visible()

Limits the count of visible characters. If you set `percent_visible` to 50, only up to half of the text's characters will display on screen. Useful to animate the text in a dialog box.

- String **text**

Default	“”
Setter	set_text(value)
Getter	get_text()

The text to display on screen.

- int **text_direction**

Default	3
Setter	set_text_direction(value)
Getter	get_text_direction()

Base direction of the text.

- bool **uppercase**

Default	false
Setter	set_uppercase(value)
Getter	is_uppercase()

If `true`, all the text displays as UPPERCASE.

- int **valign**

Default	0
Setter	set_valign(value)
Getter	get_valign()

Controls the text's vertical align. Supports top, center, bottom, and fill. Set it to one of the `VALIGN_*` constants.

- int **visible_characters**

Default	-1
Setter	set_visible_characters(value)
Getter	get_visible_characters()

Restricts the number of characters to display. Set to -1 to disable.

5.8.7 Method Descriptions

- int **get_line_count** ( ) const

Returns the amount of lines of text the Label has.

- int **get_line_height** ( ) const

Returns the of the line in pixels.

- int **get_total_character_count** ( ) const

Returns the total number of printable characters in the text.

- int **get_visible_line_count** ( ) const

Returns the number of lines shown. Useful if the Label's height cannot currently display all lines.

5.9 TLLineEdit

Inherits:

Category: Core

5.9.1 Brief Description

Control that provides single-line string editing. For formatted or/and multiline text, use *TLRichTextEdit*.

Copy of Godot LineEdit with the shaping support. TLLineEdit uses the same theme items as the LineEdit.

5.9.2 Properties

int	*align*	0
TLFontFamily	*base_font*	
int	*base_font_size*	12
String	*base_font_style*	“Regular”
bool	*caret_blink*	false
float	*caret_blink_speed*	0.65
int	*caret_position*	0
bool	*clear_button_enabled*	false
bool	*context_menu_enabled*	true
bool	*editable*	true
bool	*expand_to_text_length*	false
int	focus_mode	**O:** 2
String	*language*	“”
int	*max_length*	0
Control.CursorShape	mouse_default_cursor_shape	**O:** 1
String	*ot_features*	“”
float	*placeholder_alpha*	0.6
String	*placeholder_text*	“”
bool	*secret*	false
String	*secret_character*	“*”
String	*text*	“”
int	*text_direction*	3

5.9.3 Methods

void	*append_at_cursor* (String text)
void	*clear* ( )
void	*deselect* ( )
PopupMenu	*get_menu* ( ) const
void	*menu_option* (int option)
void	*select* (int from=0, int to=-1)
void	*select_all* ( )

5.9.4 Signals

- **text_changed** (String new_text)

Emitted when the text changes.

- **text_entered** (String new_text)

Emitted when the user presses `KEY_ENTER` on the `TLLineEdit`.

5.9.5 Enumerations

enum **Align**:

- **ALIGN_LEFT = 0** — Aligns the text on the left-hand side of the `TLLineEdit`.
- **ALIGN_CENTER = 1** — Centers the text in the middle of the `TLLineEdit`.
- **ALIGN_RIGHT = 2** — Aligns the text on the right-hand side of the `TLLineEdit`.

- **ALIGN_FILL = 3 —** Stretches white spaces to fit the `TLLineEdit`'s width.

---

enum **MenuItems**:

- **MENU_CUT = 0 —** Cuts (copies and clears) the selected text.
- **MENU_COPY = 1 —** Copies the selected text.
- **MENU_PASTE = 2 —** Pastes the clipboard text over the selected text (or at the cursor's position).

Non-printable escape characters are automatically stripped from the OS clipboard via `String.strip_escapes`.

- **MENU_CLEAR = 3 —** Erases the whole `TLLineEdit` text.
- **MENU_SELECT_ALL = 4 —** Selects the whole `TLLineEdit` text.
- **MENU_UNDO = 5 —** Undoes the previous action.
- **MENU_REDO = 6 —** Reverse the last undo action.
- **MENU_MAX = 7 —** Represents the size of the *MenuItems* enum.

5.9.6 Description

TLLineEdit provides a single-line string editor, used for text fields. It features many built-in shortcuts which will always be available:

- Ctrl + C: Copy
- Ctrl + X: Cut
- Ctrl + V or Ctrl + Y: Paste/"yank"
- Ctrl + Z: Undo
- Ctrl + Shift + Z: Redo
- Ctrl + U: Delete text from the cursor position to the beginning of the line
- Ctrl + K: Delete text from the cursor position to the end of the line
- Ctrl + A: Select all text
- Ctrl + D: Swap current input direction (primary cursor)
- Up/Down arrow: Move the cursor to the beginning/end of the line

5.9.7 Property Descriptions

- int **align**

Default	0
Setter	set_align(value)
Getter	get_align()

Text alignment as defined in the `ALIGN_*` enum.

- *TLFontFamily* **base_font**

Setter	set_base_font(value)
Getter	get_base_font()

Base font.

- int **base_font_size**

Default	12
Setter	set_base_font_size(value)
Getter	get_base_font_size()

Base font size.

- String **base_font_style**

Default	“Regular”
Setter	set_base_font_style(value)
Getter	get_base_font_style()

Base font style.

- bool **caret_blink**

Default	false
Setter	cursor_set_blink_enabled(value)
Getter	cursor_get_blink_enabled()

If `true`, the caret (visual cursor) blinks.

- float **caret_blink_speed**

Default	0.65
Setter	cursor_set_blink_speed(value)
Getter	cursor_get_blink_speed()

Duration (in seconds) of a caret’s blinking cycle.

- int **caret_position**

Default	0
Setter	set_cursor_position(value)
Getter	get_cursor_position()

The cursor’s position inside the `TLLineEdit`. When set, the text may scroll to accommodate it.

- bool **clear_button_enabled**

Default	false
Setter	set_clear_button_enabled(value)
Getter	is_clear_button_enabled()

If `true`, the `TLLineEdit` will show a clear button if `text` is not empty.

- bool **context_menu_enabled**

Default	true
Setter	set_context_menu_enabled(value)
Getter	is_context_menu_enabled()

If `true`, the context menu will appear when right-clicked.

- bool **editable**

Default	true
Setter	set_editable(value)
Getter	is_editable()

If `false`, existing text cannot be modified and new text cannot be added.

- bool **expand_to_text_length**

Default	false
Setter	set_expand_to_text_length(value)
Getter	get_expand_to_text_length()

If `true`, the `TLLineEdit` width will increase to stay longer than the *text*. It will **not** compress if the *text* is shortened.

- String **language**

Default	""
Setter	set_language(value)
Getter	get_language()

Language code for line-breaking and text shaping algorithms.

- int **max_length**

Default	0
Setter	set_max_length(value)
Getter	get_max_length()

Maximum amount of characters that can be entered inside the LineEdit. If `0`, there is no limit.

- String **ot_features**

Default	""

Continued on next page

Table 47 – continued from previous page

Setter	set_ot_features(value)
Getter	get_ot_features()

Comma separated list of OpenType feature tags.

- float **placeholder_alpha**

Default	0.6
Setter	set_placeholder_alpha(value)
Getter	get_placeholder_alpha()

Opacity of the *placeholder_text*. From `0` to `1`.

- String **placeholder_text**

Default	""
Setter	set_placeholder(value)
Getter	get_placeholder()

Text shown when the `TLLineEdit` is empty. It is **not** the `TLLineEdit`'s default value (see *text*).

- bool **secret**

Default	false
Setter	set_secret(value)
Getter	is_secret()

If `true`, every character is replaced with the secret character (see *secret_character*).

- String **secret_character**

Default	"*"
Setter	set_secret_character(value)
Getter	get_secret_character()

The character to use to mask secret input (defaults to "*"). Only a single character can be used as the secret character.

- String **text**

Default	""
Setter	set_text(value)
Getter	get_text()

String value of the `TLLineEdit`.

- int **text_direction**

Default	3
Setter	set_text_direction(value)
Getter	get_text_direction()

Base direction of the text.

5.9.8 Method Descriptions

- void **append_at_cursor** (String text)

Adds `text` after the cursor. If the resulting value is longer than *max_length*, nothing happens.

- void **clear** ( )

Erases the `TLLineEdit` text.

- void **deselect** ( )

Clears the current selection.

- PopupMenu **get_menu** ( ) const

Returns the PopupMenu of this `TLLineEdit`. By default, this menu is displayed when right-clicking on the `TLLineEdit`.

- void **menu_option** (int option)

Executes a given action as defined in the“MENU_*“ enum.

- void **select** (int from=0, int to=-1)

Selects characters inside `TLLineEdit` between `from` and `to`. By default, `from` is at the beginning and `to` at the end.

```
text = "Welcome"
select()      # Will select "Welcome"
select(4)     # Will select "ome"
select(2, 5) # Will select "lco"
```

- void **select_all** ( )

Selects the whole String.

5.10 TLRichTextEdit

Inherits:

Category: Core

5.10.1 Brief Description

Rich text display and input control.

Rich text can contain custom text, fonts, images and formatting.

5.10.2 Properties

Color	*back_color*	Color(1, 1, 1, 0)
Control.FocusMode	focus_mode	**O:** 2
float	*paragraph_spacing*	3.0
bool	*readonly*	false
bool	*selectable*	true
TLShapedParagraph	*paragraphs/{index}*	

5.10.3 Methods

void	*add_attribute* (*TLRichTextEditSelection* selection, int attribute, Variant value)
void	*clear* ( )
void	*debug_draw* (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)
void	*debug_draw_as_hex* (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)
void	*debug_draw_logical_as_hex* (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)
Vector2	*get_caret_position* ( )
String	*get_cluster_debug_info_hit_test* (Vector2 position)
Array	*get_cluster_glyphs_hit_test* (Vector2 position)
Rect2	*get_cluster_rect_hit_test* (Vector2 position)
TLShapedParagraph	*get_paragraph* (int index) const
int	*get_paragraphs* ( )
TLRichTextEditSelection	*get_selection* ( ) const
int	*insert_paragraph* (*TLShapedParagraph* para, int index)
void	*remove_attribute* (*TLRichTextEditSelection* selection, int attribute)
void	*remove_attributes* (*TLRichTextEditSelection* selection)
void	*remove_paragraph* (int index)
void	*replace_sstring* (*TLRichTextEditSelection* selection, *TLShapedString* text)
void	*replace_text* (*TLRichTextEditSelection* selection, String text)
void	*replace_utf16* (*TLRichTextEditSelection* selection, PoolByteArray text)
void	*replace_utf32* (*TLRichTextEditSelection* selection, PoolByteArray text)
void	*replace_utf8* (*TLRichTextEditSelection* selection, PoolByteArray text)
void	*set_paragraph* (*TLShapedParagraph* para, int index)
void	*set_paragraph_back_color* (*TLRichTextEditSelection* selection, Color bcolor)
void	*set_paragraph_brk_flags* (*TLRichTextEditSelection* selection, int flags)
void	*set_paragraph_halign* (*TLRichTextEditSelection* selection, int halign)
void	*set_paragraph_indent* (*TLRichTextEditSelection* selection, float indent)
void	*set_paragraph_jst_flags* (*TLRichTextEditSelection* selection, int flags)
void	*set_paragraph_line_spacing* (*TLRichTextEditSelection* selection, float line_spacing)
void	*set_paragraph_width* (*TLRichTextEditSelection* selection, float width)
void	*set_selection* (*TLRichTextEditSelection* selection)

5.10.4 Signals

- **cursor_changed ( )**

Emitted when caret moves or selection changes.

- **paragraph_changed ( )**

Emitted when paragraph text or attributes changes.

5.10.5 Property Descriptions

- Color **back_color**

Default	Color(1, 1, 1, 0)
Setter	set_back_color(value)
Getter	get_back_color()

Background color of the control.

- float **paragraph_spacing**

Default	3.0
Setter	set_paragraph_spacing(value)
Getter	get_paragraph_spacing()

Spacing between the paragraphs in pixels.

- bool **readonly**

Default	false
Setter	set_readonly(value)
Getter	get_readonly()

If `true`, existing text cannot be modified and new text cannot be added.

- bool **selectable**

Default	true
Setter	set_selectable(value)
Getter	get_selectable()

If `true`, the control allows text selection.

- *TLShapedParagraph* **paragraphs/{index}**

Paragraphs of the text.

5.10.6 Method Descriptions

- void **add_attribute** (*TLRichTextEditSelection* selection, int attribute, Variant value)

Sets `attribute` attribute to `value` for the selection.

- void **clear** ( )

Clears the attributes and sets text to an empty string.

- void **debug_draw** (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws text with additional debug information.

- void **debug_draw_as_hex** (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws text using "hexbox" fallback font.

- void **debug_draw_logical_as_hex** (RID rid, Vector2 position, Vector2 hit_position, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws logical character codes of the text with additional debug information.

- Vector2 **get_caret_position** ( )

Returns position of the caret, in (paragraph, offset) format.

- String **get_cluster_debug_info_hit_test** (Vector2 position)

`DEBUG:` Returns cluster hit test info.

- Array **get_cluster_glyphs_hit_test** (Vector2 position)

Returns glyph hit test info.

- Rect2 **get_cluster_rect_hit_test** (Vector2 position)

Returns replacement object hit test info.

- *TLShapedParagraph* **get_paragraph** (int index) const

Returns paragraph of text.

- int **get_paragraphs** ( )

Returns number of paragraph in the text.

- *TLRichTextEditSelection* **get_selection** ( ) const

Returns current selection and caret position.

- int **insert_paragraph** (*TLShapedParagraph* para, int index)

Inserts paragraph of text at the index position.

- void **remove_attribute** (*TLRichTextEditSelection* selection, int attribute)

Returns `attribute` attribute value in the selection.

- void **remove_attributes** (*TLRichTextEditSelection* selection)

Removes all attributes in the selection.

- void **remove_paragraph** (int index)

Removes paragraph of the text.

- void **replace_sstring** (*TLRichTextEditSelection* selection, *TLShapedString* text)

Replaces selection with the shaped string.

- void **replace_text** (*TLRichTextEditSelection* selection, String text)

Replaces selection with the plain string.

- void **replace_utf16** (*TLRichTextEditSelection* selection, PoolByteArray text)

Replaces selection with the contents of the raw UTF-16 encoded string.

- void **replace_utf32** (*TLRichTextEditSelection* selection, PoolByteArray text)

Replaces selection with the contents of the raw UTF-32 encoded string.

- void **replace_utf8** (*TLRichTextEditSelection* selection, PoolByteArray text)

Replaces selection with the contents of the raw UTF-8 encoded string.

- void **set_paragraph** (*TLShapedParagraph* para, int index)

Sets content of the specific paragraph.

- void **set_paragraph_back_color** (*TLRichTextEditSelection* selection, Color bcolor)

Sets background color of the paragraph.

- void **set_paragraph_brk_flags** (*TLRichTextEditSelection* selection, int flags)

Sets line breaking flags of the paragraph.

- void **set_paragraph_halign** (*TLRichTextEditSelection* selection, int halign)

Sets horizontal alignment of the paragraph.

- void **set_paragraph_indent** (*TLRichTextEditSelection* selection, float indent)

Sets indentation of the paragraph.

- void **set_paragraph_jst_flags** (*TLRichTextEditSelection* selection, int flags)

Sets justification flags of the paragraph.

- void **set_paragraph_line_spacing** (*TLRichTextEditSelection* selection, float line_spacing)

Sets line spacing of the paragraph.

- void **set_paragraph_width** (*TLRichTextEditSelection* selection, float width)

Sets max width of the paragraph.

- void **set_selection** (*TLRichTextEditSelection* selection)

Sets current selection.

5.11 TLRichTextEditSelection

Inherits:

Category: Core

5.11.1 Brief Description

Selection and caret info of the *TLRichTextEdit*

5.11.2 Properties

int	*caret_offset*	0
int	*caret_para*	0
int	*end_offset*	0
int	*end_para*	0
int	*start_offset*	0
int	*start_para*	0

5.11.3 Signals

- **selection_changed** ( )

Emitted when selection range is changed or caret moved.

5.11.4 Property Descriptions

- int **caret_offset**

Default	0
Setter	set_caret_offset(value)
Getter	get_caret_offset()

Caret position in the paragraph *caret_para*.

- int **caret_para**

Default	0
Setter	set_caret_para(value)
Getter	get_caret_para()

Paragraph where caret currently is in.

- int **end_offset**

Default	0
Setter	set_end_offset(value)
Getter	get_end_offset()

Offset of the end of selection in the paragraph *end_para*.

- int **end_para**

Default	0
Setter	set_end_para(value)
Getter	get_end_para()

Last paragraph in the selection.

- int **start_offset**

Default	0
Setter	set_start_offset(value)
Getter	get_start_offset()

Offset of the start of selection in the paragraph *start_para*.

- int **start_para**

Default	0
Setter	set_start_para(value)
Getter	get_start_para()

First paragraph in the selection.

5.12 TLShapedAttributedString

Inherits: *TLShapedString*

Category: Core

5.12.1 Brief Description

Golds shaped line of text with associated attributes.

5.12.2 Properties

Array	*attributes_dict*
int	*attribute/type*
Variant	*attribute/value*
int	*attribute/start*
int	*attribute/end*

5.12.3 Methods

void	*add_attribute* (int attribute, Variant value, int start, int end)
void	*clear_attributes* ( )
void	*commit_attribute* ( )
Variant	*get_attribute* (int attribute, int index) const
int	*get_attribute_end* (int attribute, int index) const
int	*get_attribute_start* (int attribute, int index) const
Array	*get_embedded_rects* ( )
bool	*has_attribute* (int attribute, int index) const
void	*load_attributes_dict* (Array array)
void	*remove_attribute* (int attribute, int start, int end)
void	*remove_attributes* (int start, int end)
Array	*save_attributes_dict* ( ) const

5.12.4 Enumerations

enum **TextAttribute**:

- **TEXT_ATTRIBUTE_FONT = 1** — Font family. Attribute type: *TLFontFamily*
- **TEXT_ATTRIBUTE_FONT_STYLE = 2** — Font style (Regular, Bold, Italic, Oblique etc.). Attribute type: String
- **TEXT_ATTRIBUTE_FONT_SIZE = 3** — Font size. Attribute type: `int`
- **TEXT_ATTRIBUTE_FONT_FEATURES = 4** — Comma separated list of OpenType feature tags. More info: https://docs.microsoft.com/en-us/typography/opentype/spec/featuretags. Attribute type: String
- **TEXT_ATTRIBUTE_LANGUAGE = 5** — Language code for line-breaking and text shaping algorithms. Attribute type: String
- **TEXT_ATTRIBUTE_REPLACEMENT_IMAGE = 6** — Embedded image. Attribute type: Texture

- **TEXT_ATTRIBUTE_REPLACEMENT_RECT = 7** — Reserved space for custom embedded object. Attribute type: Vector2
- **TEXT_ATTRIBUTE_REPLACEMENT_ID = 8** — Embedded object id key. Attribute type: Variant
- **TEXT_ATTRIBUTE_REPLACEMENT_VALIGN = 9** — Embedded image/object inline alignment. Attribute type: `TEXT_VALIGN_*`
- **TEXT_ATTRIBUTE_COLOR = 31** — Text color. Attribute type: Color
- **TEXT_ATTRIBUTE_OUTLINE_COLOR = 32** — Text outline color. Attribute type: Color
- **TEXT_ATTRIBUTE_UNDERLINE_COLOR = 41** — Underline color. Attribute type: Color
- **TEXT_ATTRIBUTE_UNDERLINE_WIDTH = 42** — Underline width. Attribute type: `int`
- **TEXT_ATTRIBUTE_STRIKETHROUGH_COLOR = 51** — Strike through line color. Attribute type: Color
- **TEXT_ATTRIBUTE_STRIKETHROUGH_WIDTH = 52** — Strike through line width. Attribute type: `int`
- **TEXT_ATTRIBUTE_OVERLINE_COLOR = 61** — Overline color/ Attribute type: Color
- **TEXT_ATTRIBUTE_OVERLINE_WIDTH = 62** — Overline width. Attribute type: `int`
- **TEXT_ATTRIBUTE_HIGHLIGHT_COLOR = 71** — Highlight color. Attribute type: Color
- **TEXT_ATTRIBUTE_META = 100** — User defined data, use `TEXT_ATTRIBUTE_META + x` to define multiple user attributes. Attribute type: Variant

---

enum **TextVAlign**:

- **TEXT_VALIGN_TOP = 0** — Inline vertical top alignment.
- **TEXT_VALIGN_CENTER = 1** — Inline vertical center alignment.
- **TEXT_VALIGN_BOTTOM = 2** — Inline vertical bottom alignment.

5.12.5 Property Descriptions

- Array **attributes_dict**

Setter	load_attributes_dict(value)
Getter	save_attributes_dict()

Array of attribute Dictionary.

- int **attribute/type**

Temporary attribute type, temporary attributes can be committed to the string by calling *commit_attribute*.

- Variant **attribute/value**

Temporary attribute value, temporary attributes can be committed to the string by calling *commit_attribute*.

- int **attribute/start**

Temporary attribute start offset, temporary attributes can be committed to the string by calling *commit_attribute*.

- int **attribute/end**

Temporary attribute end offset, temporary attributes can be committed to the string by calling *commit_attribute*.

5.12.6 Method Descriptions

- void **add_attribute** (int attribute, Variant value, int start, int end)

Sets `attribute` attribute to `value` for specified text range.

- void **clear_attributes** ( )

Removes all attributes.

- void **commit_attribute** ( )

- Variant **get_attribute** (int attribute, int index) const

Returns `attribute` attribute value for specified text position.

- int **get_attribute_end** (int attribute, int index) const

Returns last position of `attribute` attribute run enclosing specified position.

- int **get_attribute_start** (int attribute, int index) const

Returns first position of `attribute` attribute run enclosing specified position.

- Array **get_embedded_rects** ( )

Returns bounding rectangles of embedded objects (`TEXT_ATTRIBUTE_REPLACEMENT_RECT` attributes).

- bool **has_attribute** (int attribute, int index) const

Returns `true` if specified position has `attribute` attribute set.

- void **load_attributes_dict** (Array array)

Loads attributes from Array of Dictionary.

- void **remove_attribute** (int attribute, int start, int end)

Removes `attribute` attribute for specified text range.

- void **remove_attributes** (int start, int end)

Removes all attributes for specified text range.

- Array **save_attributes_dict** ( ) const

Stores string attributes into Array of Dictionary.

5.13 TLShapedParagraph

Inherits:

Category: Core

5.13.1 Brief Description

Class for formatting entire paragraphs of text at once.

5.13.2 Properties

Color	*back_color*	Color(1, 1, 1, 0)
int	*brk_flags*	2
int	*halign*	0
float	*indent*	0.0
int	*jst_flags*	1
float	*line_spacing*	1.0
TLShapedAttributedString	*string*	
float	*width*	-1.0

5.13.3 Methods

void	*copy_properties* (*TLShapedParagraph* source)
TLShapedAttributedString	*get_line* (int index) const
Array	*get_line_bounds* ( ) const
int	*get_lines* ( ) const
Vector2	*get_size* ( ) const
Array	*get_word_bounds* ( ) const

5.13.4 Signals

- **paragraph_changed** ( )

Emitted when text or attributes of the paragraph are changed.

5.13.5 Enumerations

enum **ParaHAlign**:

- **PARA_HALIGN_LEFT** = **0** — Align rows to the left (default).
- **PARA_HALIGN_CENTER** = **1** — Align rows centered.
- **PARA_HALIGN_RIGHT** = **2** — Align rows to the right.
- **PARA_HALIGN_FILL** = **3** — Expand row white spaces to fit the width.

5.13.6 Property Descriptions

- Color **back_color**

Default	Color(1, 1, 1, 0)
Setter	set_back_color(value)
Getter	get_back_color()

Background color of the paragraph.

- int **brk_flags**

Default	2
Setter	set_brk_flags(value)
Getter	get_brk_flags()

Line breaking flags.

- int **halign**

Default	0
Setter	set_halign(value)
Getter	get_halign()

Horizontal alignment.

- float **indent**

Default	0.0
Setter	set_indent(value)
Getter	get_indent()

Indentation.

- int **jst_flags**

Default	1
Setter	set_jst_flags(value)
Getter	get_jst_flags()

Justification flags.

- float **line_spacing**

Default	1.0
Setter	set_line_spacing(value)
Getter	get_line_spacing()

Lien spacing in pixels.

- *TLShapedAttributedString* **string**

Setter	set_string(value)
Getter	get_string()

Text and text attributes of the paragraph.

- float **width**

Default	-1.0
Setter	set_width(value)
Getter	get_width()

Max width of the paragraph.

5.13.7 Method Descriptions

- void **copy_properties** (*TLShapedParagraph* source)

Copies properties of the another paragraph.

- *TLShapedAttributedString* **get_line** (int index) const

Returns line of the pargraphs. NOTE: read only, changes to the line will be discarded on paragraph update!

- Array **get_line_bounds** ( ) const

Returns array of the line boundaries as `Vector2(start, end)`.

- int **get_lines** ( ) const

Returns number of lines in the paragraph.

- Vector2 **get_size** ( ) const

Returns size of the paragraph in pixels.

- Array **get_word_bounds** ( ) const

Returns array of the word boundaries as `Vector2(start, end)`.

5.14 TLShapedString

Inherits:

Inherited By: *TLShapedAttributedString*

Category: Core

5.14.1 Brief Description

Holds shaped line of plain text.

5.14.2 Properties

int	*base_direction*	3
TLFontFamily	*base_font*	
int	*base_font_size*	12
String	*base_font_style*	“Regular”
String	*features*	“”
String	*language*	“en”
bool	*preserve_control*	false
String	*text*	“”

5.14.3 Methods

void	*add_sstring* (*TLShapedString* text)
void	*add_text* (String text)
void	*add_utf16* (PoolByteArray text)
void	*add_utf32* (PoolByteArray text)
void	*add_utf8* (PoolByteArray text)
Array	*break_jst* ( ) const
Array	*break_lines* (float width, int flags) const
Array	*break_words* ( ) const
int	*char_count* ( ) const
int	*clusters* ( ) const
void	*copy_properties* (*TLShapedString* source)
void	*draw* (RID canvas_item, Vector2 position, Color modulate)
void	*draw_as_hex* (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)
Vector2	*draw_cluster* (RID canvas_item, Vector2 position, int index, Color modulate)
void	*draw_dbg* (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)
void	*draw_logical_as_hex* (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)
bool	*empty* ( ) const
float	*extend_to_width* (float width, int flags)
float	*get_ascent* ( ) const
TextDirection	*get_char_direction* (int position) const
float	*get_cluster_ascent* (int index) const
String	*get_cluster_debug_info* (int index) const
float	*get_cluster_descent* (int index) const
int	*get_cluster_end* (int index) const
TLFontFace	*get_cluster_face* (int position) const
float	*get_cluster_face_size* (int position) const
int	*get_cluster_glyph* (int index, int glyph) const
Vector2	*get_cluster_glyph_advance* (int index, int glyph) const
Vector2	*get_cluster_glyph_offset* (int index, int glyph) const
int	*get_cluster_glyphs* (int index) const
float	*get_cluster_height* (int index) const
int	*get_cluster_index* (int position) const
float	*get_cluster_leading_edge* (int index) const
Rect2	*get_cluster_rect* (int index) const
int	*get_cluster_start* (int index) const
float	*get_cluster_trailing_edge* (int index) const
float	*get_cluster_width* (int index) const

Continued on next page

Table 76 – continued from previous page

Array	*get_cursor_positions* (int position, int primary_dir) const
float	*get_descent* ( ) const
float	*get_height* ( ) const
Array	*get_highlight_shapes* (int start, int end) const
int	*get_para_direction* ( ) const
PoolByteArray	*get_utf16* ( ) const
PoolByteArray	*get_utf32* ( ) const
PoolByteArray	*get_utf8* ( ) const
float	*get_width* ( ) const
int	*hit_test* (float position) const
int	*hit_test_cluster* (float position) const
bool	*is_valid* ( ) const
int	*length* ( ) const
int	*next_safe_bound* (int position) const
int	*pos_u16_to_wcs* (int position) const
int	*pos_wcs_to_u16* (int position) const
int	*prev_safe_bound* (int position) const
void	*replace_sstring* (int start, int end, *TLShapedString* text)
void	*replace_text* (int start, int end, String text)
void	*replace_utf16* (int start, int end, PoolByteArray text)
void	*replace_utf32* (int start, int end, PoolByteArray text)
void	*replace_utf8* (int start, int end, PoolByteArray text)
void	*set_utf16* (PoolByteArray data)
void	*set_utf32* (PoolByteArray data)
void	*set_utf8* (PoolByteArray data)
bool	*shape* ( )
TLShapedString	*substr* (int start, int end, int trim) const

5.14.4 Signals

- **string_changed** ( )

Emitted when the text or/and other property of the string changes.

- **string_shaped** ( )

Emitted when the text shaping process is completed.

5.14.5 Enumerations

enum **TextDirection**:

- **TEXT_DIRECTION_LTR = 0** — Left-to-right text writing direction.
- **TEXT_DIRECTION_RTL = 1** — Right-to-left text writing direction.
- **TEXT_DIRECTION_LOCALE = 2** — Text writing direction is derived from the locale's script according to the CLDR metadata.
- **TEXT_DIRECTION_AUTO = 3** — Text writing direction is derived from the first character in the string with BiDi class L, R, or AL or locale's script if text is not strongly directional.
- **TEXT_DIRECTION_INVALID = 4** — Invalid text direction (Used as return value only).

---

enum **TextJustification**:

- **TEXT_JUSTIFICATION_NONE = 0** — No text justification.

- **TEXT_JUSTIFICATION_KASHIDA_AND_WHITESPACE = 1** — Use kashida and white space elongation to justify text.
- **TEXT_JUSTIFICATION_KASHIDA_ONLY = 2** — Use kashida elongation to justify text.
- **TEXT_JUSTIFICATION_WHITESPACE_ONLY = 3** — Use white space elongation to justify text.
- **TEXT_JUSTIFICATION_KASHIDA_AND_WHITESPACE_AND_INTERCHAR = 4** — Use kashida, white space and inter character elongation to justify text.
- **TEXT_JUSTIFICATION_KASHIDA_AND_INTERCHAR = 5** — Use kashida and inter character elongation to justify text.
- **TEXT_JUSTIFICATION_WHITESPACE_AND_INTERCHAR = 6** — Use white space and inter character elongation to justify text.
- **TEXT_JUSTIFICATION_INTERCHAR_ONLY = 7** — Use inter character elongation to justify text.

---

enum **TextBreak**:

- **TEXT_BREAK_NONE = 0** — No line breaking.
- **TEXT_BREAK_MANDATORY = 1** — Break lines only at mandatory break points.
- **TEXT_BREAK_MANDATORY_AND_WORD_BOUND = 2** — Break lines at mandatory break points and word boundaries.
- **TEXT_BREAK_MANDATORY_AND_ANYWHERE = 3** — Break lines at mandatory break points and grapheme cluster boundaries.

---

enum **TextTrimMode**:

- **TEXT_TRIM_NONE = 0** — No substring trimming.
- **TEXT_TRIM_BREAK = 1** — Trim line break characters for substring ends.
- **TEXT_TRIM_BREAK_AND_WHITESPACE = 2** — Trim line break and white space characters for substring ends.

5.14.6 Description

Note 1: Code points, Characters, Clusters and Glyphs

- A code point is a single encoding UTF-16 unit (Unicode character or half of the surrogate pair).
- A character is a full Unicode character.
- A grapheme cluster is the abstract unit of a writing system (a letter, a digit, or punctuation).
- A glyph is a shape used to render a character or a sequence of characters.

In general, code point, characters, clusters and glyphs do not have one-to-one correspondence.

Note 2: Encoding

> TLShapedString uses UTF-16 encoding, all positions accepted and returned by TLShapedString function are measured in UTF-16 code points.

5.14.7 Property Descriptions

- int **base_direction**

Default	3
Setter	set_base_direction(value)
Getter	get_base_direction()

Base text writing direction.

- *TLFontFamily* **base_font**

Setter	set_base_font(value)
Getter	get_base_font()

Base font family reference.

- int **base_font_size**

Default	12
Setter	set_base_font_size(value)
Getter	get_base_font_size()

Font size.

- String **base_font_style**

Default	“Regular”
Setter	set_base_font_style(value)
Getter	get_base_font_style()

Style name (Regular, Bold, Italic, Oblique etc.).

- String **features**

Default	“”
Setter	set_features(value)
Getter	get_features()

Comma separated list of OpenType feature tags. More info: https://docs.microsoft.com/en-us/typography/opentype/spec/featuretags.

- String **language**

Default	“en”
Setter	set_language(value)
Getter	get_language()

Language code for line-breaking and text shaping algorithms.

- bool **preserve_control**

Default	false
Setter	set_preserve_control(value)
Getter	get_preserve_control()

If `true`, displays control character.

- String **text**

Default	“”
Setter	set_text(value)
Getter	get_text()

Text string.

5.14.8 Method Descriptions

- void **add_sstring** (*TLShapedString* text)

Appends shaped string.

- void **add_text** (String text)

Appends plain text string.

- void **add_utf16** (PoolByteArray text)

- void **add_utf32** (PoolByteArray text)

- void **add_utf8** (PoolByteArray text)

- Array **break_jst** ( ) const

- Array **break_lines** (float width, int flags) const

Breaks text into lines that fit within a specified width.

Returns Array of line boundaries.

- Array **break_words** ( ) const

Breaks text into words.

Returns Array of word boundaries.

- int **char_count** ( ) const

Returns number of characters in the string.

- int **clusters** ( ) const

Returns number of grapheme clusters, clusters are indexed in visual order.

- void **copy_properties** (*TLShapedString* source)

- void **draw** (RID canvas_item, Vector2 position, Color modulate)

Draws a string.

- void **draw_as_hex** (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws a string using "hexbox" fallback font.

- Vector2 **draw_cluster** (RID canvas_item, Vector2 position, int index, Color modulate)

Draws single grapheme cluster. Returns advance.

- void **draw_dbg** (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws a string with additional debug information.

- void **draw_logical_as_hex** (RID canvas_item, Vector2 position, Color modulate, bool draw_brk_ops, bool draw_jst_ops)

`DEBUG:` Draws logical character codes of the string with additional debug information.

- bool **empty** ( ) const

Returns `true` if the string is empty.

- float **extend_to_width** (float width, int flags)

Increase text width to the specified. Returns new line width.

- float **get_ascent** ( ) const

Returns ascent of the line.

- *TextDirection* **get_char_direction** (int position) const

Return writing direction of a character writing direction.

- float **get_cluster_ascent** (int index) const

Returns cluster ascent.

- String **get_cluster_debug_info** (int index) const

- float **get_cluster_descent** (int index) const

Returns cluster descent.

- int **get_cluster_end** (int index) const

Returns last character position corresponding cluster.

- *TLFontFace* **get_cluster_face** (int position) const

Returns font face of the cluster.

- float **get_cluster_face_size** (int position) const

Returns font size of the cluster. For non-attributed string it's always equal to base size.

- int **get_cluster_glyph** (int index, int glyph) const

Returns glyph ID.

- Vector2 **get_cluster_glyph_advance** (int index, int glyph) const

Returns glyph advance.

- Vector2 **get_cluster_glyph_offset** (int index, int glyph) const

Returns glyph offset.

- int **get_cluster_glyphs** (int index) const

Returns number of glyphs in cluster.

- float **get_cluster_height** (int index) const

Returns cluster height.

- int **get_cluster_index** (int position) const

Returns cluster index corresponding to a specific character position in string.

- float **get_cluster_leading_edge** (int index) const

Returns cluster leading edge offset in pixels.

- Rect2 **get_cluster_rect** (int index) const

Returns cluster bounding rectangle.

- int **get_cluster_start** (int index) const

Returns first character position corresponding cluster.

- float **get_cluster_trailing_edge** (int index) const

Returns cluster trailing edge offset in pixels.

- float **get_cluster_width** (int index) const

Returns cluster width.

- Array **get_cursor_positions** (int position, int primary_dir) const

Returns an Array of `float` (up to two elements) offsets corresponding to the strong and weak cursor, at the specified character position.

- float **get_descent** ( ) const

Returns descent of the line.

- float **get_height** ( ) const

Returns height of the line.

- Array **get_highlight_shapes** (int start, int end) const

Returns an Array of Rect2 enclosing the selection/highlight in the specified range.

- int **get_para_direction** ( ) const

Returns shaped string paragraph direction.

- PoolByteArray **get_utf16** ( ) const

Returns raw text string in UTF-16 encoding.

- PoolByteArray **get_utf32** ( ) const

Returns raw text string in UTF-32 encoding.

- PoolByteArray **get_utf8** ( ) const

Returns raw text string in UTF-8 encoding.

- float **get_width** ( ) const

Returns width of the line.

- int **hit_test** (float position) const

Returns a cursor position corresponding to the specified pixel offset.

- int **hit_test_cluster** (float position) const

Returns a cluster index corresponding to the specified pixel offset.

- bool **is_valid** ( ) const

Returns `true` if the string is shaped successfully.

- int **length** ( ) const

Returns number of UTF-16 codepoints in the string.

- int **next_safe_bound** (int position) const

Returns next whole character position in the string.

- int **pos_u16_to_wcs** (int position) const

Returns character position (Characters)

- int **pos_wcs_to_u16** (int position) const

Retruns character position (UTF-16 codepoints)

- int **prev_safe_bound** (int position) const

Returns previous whole character position in the string.

- void **replace_sstring** (int start, int end, *TLShapedString* text)

Replaces substring with contents of another shaped string.

- void **replace_text** (int start, int end, String text)

Replaces substring with text.

- void **replace_utf16** (int start, int end, PoolByteArray text)

Replaces substring with contents of the raw UTF-16 encoded string.

- void **replace_utf32** (int start, int end, PoolByteArray text)

Replaces substring with contents of the raw UTF-32 encoded string.

- void **replace_utf8** (int start, int end, PoolByteArray text)

Replaces substring with contents of the raw UTF-8 encoded string.

- void **set_utf16 (** PoolByteArray data **)**

Sets text to the contents of the raw UTF-16 encoded string.

- void **set_utf32 (** PoolByteArray data **)**

Sets text to the contents of the raw UTF-32 encoded string.

- void **set_utf8 (** PoolByteArray data **)**

Sets text to the contents of the raw UTF-8 encoded string.

- bool **shape ( )**

Shapes string and returns `true` if the string is shaped successfully.

- *TLShapedString* **substr (** int start, int end, int trim **)** const

Returns part of the shaped string from the position `start` to `end`, trims it according to `trim` parameter.