

Graphiteng

Generated by Doxygen 1.9.1



<b>1</b>	<b>Deprecated List</b>	<b>1</b>
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Class Documentation</b>	<b>7</b>
4.1	gr_face_ops Struct Reference . . . . .	7
4.1.1	Detailed Description . . . . .	7
4.1.2	Member Data Documentation . . . . .	7
4.1.2.1	get_table . . . . .	7
4.1.2.2	release_table . . . . .	8
4.1.2.3	size . . . . .	8
4.2	gr_faceinfo Struct Reference . . . . .	8
4.2.1	Detailed Description . . . . .	9
4.2.2	Member Enumeration Documentation . . . . .	9
4.2.2.1	gr_space_contextuals . . . . .	9
4.2.3	Member Data Documentation . . . . .	10
4.2.3.1	extra_ascent . . . . .	10
4.2.3.2	extra_descent . . . . .	10
4.2.3.3	has_bidi_pass . . . . .	10
4.2.3.4	justifies . . . . .	10
4.2.3.5	line_ends . . . . .	11
4.2.3.6	space_contextuals . . . . .	11
4.2.3.7	upem . . . . .	11
4.3	gr_font_ops Struct Reference . . . . .	11
4.3.1	Detailed Description . . . . .	11
4.3.2	Member Data Documentation . . . . .	12
4.3.2.1	glyph_advance_x . . . . .	12
4.3.2.2	glyph_advance_y . . . . .	12
4.3.2.3	size . . . . .	12
<b>5</b>	<b>File Documentation</b>	<b>13</b>
5.1	Font.h File Reference . . . . .	13
5.1.1	Macro Definition Documentation . . . . .	15
5.1.1.1	GR2_VERSION_BUGFIX . . . . .	15
5.1.1.2	GR2_VERSION_MAJOR . . . . .	15
5.1.1.3	GR2_VERSION_MINOR . . . . .	15
5.1.2	Typedef Documentation . . . . .	16
5.1.2.1	gr_advance_fn . . . . .	16
5.1.2.2	gr_face . . . . .	16
5.1.2.3	gr_feature_ref . . . . .	16

---

5.1.2.4	<a href="#">gr_feature_val</a>	16
5.1.2.5	<a href="#">gr_font</a>	16
5.1.2.6	<a href="#">gr_get_table_fn</a>	16
5.1.2.7	<a href="#">gr_release_table_fn</a>	17
5.1.3	<a href="#">Enumeration Type Documentation</a>	17
5.1.3.1	<a href="#">gr_face_options</a>	17
5.1.4	<a href="#">Function Documentation</a>	17
5.1.4.1	<a href="#">gr_engine_version()</a>	18
5.1.4.2	<a href="#">gr_face_destroy()</a>	18
5.1.4.3	<a href="#">gr_face_featureval_for_lang()</a>	18
5.1.4.4	<a href="#">gr_face_find_fref()</a>	18
5.1.4.5	<a href="#">gr_face_fref()</a>	19
5.1.4.6	<a href="#">gr_face_info()</a>	19
5.1.4.7	<a href="#">gr_face_is_char_supported()</a>	19
5.1.4.8	<a href="#">gr_face_lang_by_index()</a>	20
5.1.4.9	<a href="#">gr_face_n_fref()</a>	20
5.1.4.10	<a href="#">gr_face_n_glyphs()</a>	20
5.1.4.11	<a href="#">gr_face_n_languages()</a>	20
5.1.4.12	<a href="#">gr_featureval_clone()</a>	20
5.1.4.13	<a href="#">gr_featureval_destroy()</a>	21
5.1.4.14	<a href="#">gr_font_destroy()</a>	21
5.1.4.15	<a href="#">gr_fref_feature_value()</a>	21
5.1.4.16	<a href="#">gr_fref_id()</a>	21
5.1.4.17	<a href="#">gr_fref_label()</a>	22
5.1.4.18	<a href="#">gr_fref_n_values()</a>	22
5.1.4.19	<a href="#">gr_fref_set_feature_value()</a>	22
5.1.4.20	<a href="#">gr_fref_value()</a>	23
5.1.4.21	<a href="#">gr_fref_value_label()</a>	23
5.1.4.22	<a href="#">gr_label_destroy()</a>	24
5.1.4.23	<a href="#">gr_make_face()</a>	24
5.1.4.24	<a href="#">gr_make_face_with_ops()</a>	24
5.1.4.25	<a href="#">gr_make_face_with_seg_cache()</a>	25
5.1.4.26	<a href="#">gr_make_face_with_seg_cache_and_ops()</a>	25
5.1.4.27	<a href="#">gr_make_file_face()</a>	26
5.1.4.28	<a href="#">gr_make_file_face_with_seg_cache()</a>	26
5.1.4.29	<a href="#">gr_make_font()</a>	27
5.1.4.30	<a href="#">gr_make_font_with_advance_fn()</a>	27
5.1.4.31	<a href="#">gr_make_font_with_ops()</a>	28
5.1.4.32	<a href="#">gr_str_to_tag()</a>	28
5.1.4.33	<a href="#">gr_tag_to_str()</a>	28
5.2	<a href="#">Log.h File Reference</a>	30
5.2.1	<a href="#">Enumeration Type Documentation</a>	30

5.2.1.1 GrLogMask . . . . .	30
5.2.2 Function Documentation . . . . .	31
5.2.2.1 gr_start_logging() . . . . .	31
5.2.2.2 gr_stop_logging() . . . . .	31
5.2.2.3 graphite_start_logging() . . . . .	31
5.2.2.4 graphite_stop_logging() . . . . .	32
5.3 Segment.h File Reference . . . . .	32
5.3.1 Typedef Documentation . . . . .	34
5.3.1.1 gr_char_info . . . . .	34
5.3.1.2 gr_segment . . . . .	34
5.3.1.3 gr_slot . . . . .	34
5.3.2 Enumeration Type Documentation . . . . .	34
5.3.2.1 gr_attrCode . . . . .	34
5.3.2.2 gr_bidirtl . . . . .	36
5.3.2.3 gr_break_weight . . . . .	36
5.3.2.4 gr_justFlags . . . . .	37
5.3.3 Function Documentation . . . . .	37
5.3.3.1 gr_cinfo_after() . . . . .	37
5.3.3.2 gr_cinfo_base() . . . . .	37
5.3.3.3 gr_cinfo_before() . . . . .	38
5.3.3.4 gr_cinfo_break_weight() . . . . .	38
5.3.3.5 gr_cinfo_unicode_char() . . . . .	39
5.3.3.6 gr_count_unicode_characters() . . . . .	39
5.3.3.7 gr_make_seg() . . . . .	39
5.3.3.8 gr_seg_advance_X() . . . . .	40
5.3.3.9 gr_seg_advance_Y() . . . . .	40
5.3.3.10 gr_seg_cinfo() . . . . .	40
5.3.3.11 gr_seg_destroy() . . . . .	40
5.3.3.12 gr_seg_first_slot() . . . . .	41
5.3.3.13 gr_seg_justify() . . . . .	41
5.3.3.14 gr_seg_last_slot() . . . . .	42
5.3.3.15 gr_seg_n_cinfo() . . . . .	42
5.3.3.16 gr_seg_n_slots() . . . . .	42
5.3.3.17 gr_slot_advance_X() . . . . .	42
5.3.3.18 gr_slot_advance_Y() . . . . .	43
5.3.3.19 gr_slot_after() . . . . .	43
5.3.3.20 gr_slot_attached_to() . . . . .	43
5.3.3.21 gr_slot_attr() . . . . .	43
5.3.3.22 gr_slot_before() . . . . .	43
5.3.3.23 gr_slot_can_insert_before() . . . . .	44
5.3.3.24 gr_slot_first_attachment() . . . . .	44
5.3.3.25 gr_slot_gid() . . . . .	44

---

5.3.3.26	<a href="#">gr_slot_index()</a>	44
5.3.3.27	<a href="#">gr_slot_linebreak_before()</a>	44
5.3.3.28	<a href="#">gr_slot_next_in_segment()</a>	45
5.3.3.29	<a href="#">gr_slot_next_sibling_attachment()</a>	45
5.3.3.30	<a href="#">gr_slot_origin_X()</a>	45
5.3.3.31	<a href="#">gr_slot_origin_Y()</a>	45
5.3.3.32	<a href="#">gr_slot_original()</a>	45
5.3.3.33	<a href="#">gr_slot_prev_in_segment()</a>	46
5.4	<a href="#">Types.h File Reference</a>	46
5.4.1	<a href="#">Macro Definition Documentation</a>	46
5.4.1.1	<a href="#">GR2_API</a>	46
5.4.1.2	<a href="#">GR2_DEPRECATED_API</a>	47
5.4.2	<a href="#">Typedef Documentation</a>	47
5.4.2.1	<a href="#">gr_byte</a>	47
5.4.2.2	<a href="#">gr_int16</a>	47
5.4.2.3	<a href="#">gr_int32</a>	47
5.4.2.4	<a href="#">gr_int8</a>	47
5.4.2.5	<a href="#">gr_uint16</a>	47
5.4.2.6	<a href="#">gr_uint32</a>	47
5.4.2.7	<a href="#">gr_uint8</a>	47
5.4.3	<a href="#">Enumeration Type Documentation</a>	47
5.4.3.1	<a href="#">gr_encform</a>	47
<b>Index</b>		<b>49</b>

# Chapter 1

## Deprecated List

**Member [gr\\_face\\_dumbRendering](#)**

Since 1.311

**Member [gr\\_make\\_face](#) (const void \*appFaceHandle, gr\_get\_table\_fn getTable, unsigned int faceOptions)**

Since v1.2.0 in favour of [gr\\_make\\_face\\_with\\_ops](#).

**Member [gr\\_make\\_face\\_with\\_seg\\_cache](#) (const void \*appFaceHandle, gr\_get\_table\_fn getTable, unsigned int segCacheMaxSize, unsigned int faceOptions)**

Since 1.3.7 this function is now an alias for [gr\\_make\\_face\(\)](#).

**Member [gr\\_make\\_face\\_with\\_seg\\_cache\\_and\\_ops](#) (const void \*appFaceHandle, const [gr\\_face\\_ops](#) \*face\_ops, unsigned int segCacheMaxSize, unsigned int faceOptions)**

Since 1.3.7 this function is now an alias for [gr\\_make\\_face\\_with\\_ops\(\)](#).

**Member [gr\\_make\\_file\\_face\\_with\\_seg\\_cache](#) (const char \*filename, unsigned int segCacheMaxSize, unsigned int faceOptions)**

Since 1.3.7.





## Chapter 2

# Class Index

### 2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">gr_face_ops</a>	Struct housing function pointers to manage font table buffers for the graphite engine . . . . .	7
<a href="#">gr_faceinfo</a>	Holds information about a particular Graphite silf table that has been loaded . . . . .	8
<a href="#">gr_font_ops</a>	Struct housing function pointers to manage font hinted metrics for the graphite engine . . . . .	11



# Chapter 3

## File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">Font.h</a>	13
<a href="#">Log.h</a>	30
<a href="#">Segment.h</a>	32
<a href="#">Types.h</a>	46



# Chapter 4

## Class Documentation

### 4.1 gr\_face\_ops Struct Reference

struct housing function pointers to manage font table buffers for the graphite engine.

```
#include <Font.h>
```

#### Public Attributes

- [size\\_t size](#)  
*size in bytes of this structure*
- [gr\\_get\\_table\\_fn get\\_table](#)  
*a pointer to a function to request a table from the client.*
- [gr\\_release\\_table\\_fn release\\_table](#)  
*is a pointer to a function to notify the client the a table can be released.*

#### 4.1.1 Detailed Description

struct housing function pointers to manage font table buffers for the graphite engine.

#### 4.1.2 Member Data Documentation

##### 4.1.2.1 get\_table

[gr\\_get\\_table\\_fn](#) get\_table

a pointer to a function to request a table from the client.

#### 4.1.2.2 release\_table

```
gr_release_table_fn release_table
```

is a pointer to a function to notify the client the a table can be released.

This can be NULL to signify that the client does not wish to do any release handling.

#### 4.1.2.3 size

```
size_t size
```

size in bytes of this structure

The documentation for this struct was generated from the following file:

- [Font.h](#)

## 4.2 gr\_faceinfo Struct Reference

Holds information about a particular Graphite silf table that has been loaded.

```
#include <Font.h>
```

### Public Types

- enum [gr\\_space\\_contextuals](#) {  
[gr\\_space\\_unknown](#) = 0 , [gr\\_space\\_none](#) = 1 , [gr\\_space\\_left\\_only](#) = 2 , [gr\\_space\\_right\\_only](#) = 3 ,  
[gr\\_space\\_either\\_only](#) = 4 , [gr\\_space\\_both](#) = 5 , [gr\\_space\\_cross](#) = 6 }

### Public Attributes

- [gr\\_uint16 extra\\_ascent](#)  
*The extra\_ascent in the GDL, in design units.*
- [gr\\_uint16 extra\\_descent](#)  
*The extra\_descent in the GDL, in design units.*
- [gr\\_uint16 upem](#)  
*The design units for the font.*
- enum [gr\\_faceinfo::gr\\_space\\_contextuals](#) [space\\_contextuals](#)
- unsigned int [has\\_bidi\\_pass](#): 1  
*the table specifies that a bidirectional pass should run*
- unsigned int [line\\_ends](#): 1  
*there are line end contextuals somewhere*
- unsigned int [justifies](#): 1  
*there are .justify properties set somewhere on some glyphs*

### 4.2.1 Detailed Description

Holds information about a particular Graphite silf table that has been loaded.

### 4.2.2 Member Enumeration Documentation

#### 4.2.2.1 gr\_space\_contextuals

```
enum gr\_space\_contextuals
```

### Enumerator

<code>gr_space_unknown</code>	no information is known.
<code>gr_space_none</code>	the space character never occurs in any rules.
<code>gr_space_left_only</code>	the space character only occurs as the first element in a rule.
<code>gr_space_right_only</code>	the space character only occurs as the last element in a rule.
<code>gr_space_either_only</code>	the space character only occurs as the only element in a rule.
<code>gr_space_both</code>	the space character may occur as the first or last element of a rule.
<code>gr_space_cross</code>	the space character occurs in a rule not as a first or last element.

## 4.2.3 Member Data Documentation

### 4.2.3.1 `extra_ascent`

`gr_uint16` `extra_ascent`

The `extra_ascent` in the GDL, in design units.

### 4.2.3.2 `extra_descent`

`gr_uint16` `extra_descent`

The `extra_descent` in the GDL, in design units.

### 4.2.3.3 `has_bidi_pass`

`unsigned int` `has_bidi_pass`

the table specifies that a bidirectional pass should run

### 4.2.3.4 `justifies`

`unsigned int` `justifies`

there are `.justify` properties set somewhere on some glyphs



#### 4.2.3.5 line\_ends

```
unsigned int line_ends
```

there are line end contextualls somewhere

#### 4.2.3.6 space\_contextualls

```
enum gr_faceinfo::gr_space_contextualls space_contextualls
```

#### 4.2.3.7 upem

```
gr_uint16 upem
```

The design units for the font.

The documentation for this struct was generated from the following file:

- [Font.h](#)

## 4.3 gr\_font\_ops Struct Reference

struct housing function pointers to manage font hinted metrics for the graphite engine.

```
#include <Font.h>
```

### Public Attributes

- [size\\_t size](#)  
*size of the structure in bytes to allow for future extensibility*
- [gr\\_advance\\_fn glyph\\_advance\\_x](#)  
*a pointer to a function to retrieve the hinted advance width of a glyph which the font cannot provide without client assistance.*
- [gr\\_advance\\_fn glyph\\_advance\\_y](#)  
*a pointer to a function to retrieve the hinted advance height of a glyph which the font cannot provide without client assistance.*

### 4.3.1 Detailed Description

struct housing function pointers to manage font hinted metrics for the graphite engine.

## 4.3.2 Member Data Documentation

### 4.3.2.1 `glyph_advance_x`

`gr_advance_fn` `glyph_advance_x`

a pointer to a function to retrieve the hinted advance width of a glyph which the font cannot provide without client assistance.

This can be NULL to signify no horizontal hinted metrics are necessary.

### 4.3.2.2 `glyph_advance_y`

`gr_advance_fn` `glyph_advance_y`

a pointer to a function to retrieve the hinted advance height of a glyph which the font cannot provide without client assistance.

This can be NULL to signify no horizontal hinted metrics are necessary.

### 4.3.2.3 `size`

`size_t` `size`

size of the structure in bytes to allow for future extensibility

The documentation for this struct was generated from the following file:

- [Font.h](#)

# Chapter 5

## File Documentation

### 5.1 Font.h File Reference

```
#include "graphite2/Types.h"
```

#### Classes

- struct [gr\\_faceinfo](#)  
*Holds information about a particular Graphite silf table that has been loaded.*
- struct [gr\\_face\\_ops](#)  
*struct housing function pointers to manage font table buffers for the graphite engine.*
- struct [gr\\_font\\_ops](#)  
*struct housing function pointers to manage font hinted metrics for the graphite engine.*

#### Macros

- #define [GR2\\_VERSION\\_MAJOR](#) 1
- #define [GR2\\_VERSION\\_MINOR](#) 3
- #define [GR2\\_VERSION\\_BUGFIX](#) 14

#### Typedefs

- typedef struct [gr\\_face](#) [gr\\_face](#)
- typedef struct [gr\\_font](#) [gr\\_font](#)
- typedef struct [gr\\_feature\\_ref](#) [gr\\_feature\\_ref](#)
- typedef struct [gr\\_feature\\_val](#) [gr\\_feature\\_val](#)
- typedef const void \*(\* [gr\\_get\\_table\\_fn](#)) (const void \*appFaceHandle, unsigned int name, size\_t \*len)  
*type describing function to retrieve font table information*
- typedef void(\* [gr\\_release\\_table\\_fn](#)) (const void \*appFaceHandle, const void \*table\_buffer)  
*type describing function to release any resources allocated by the above get\_table table function*
- typedef float(\* [gr\\_advance\\_fn](#)) (const void \*appFontHandle, [gr\\_uint16](#) glyphid)  
*query function to find the hinted advance of a glyph*

## Enumerations

- enum `gr_face_options` {  
`gr_face_default` = 0 , `gr_face_dumbRendering` = 1 , `gr_face_preloadGlyphs` = 2 , `gr_face_cacheCmap` = 4 ,  
`gr_face_preloadAll` = `gr_face_preloadGlyphs` | `gr_face_cacheCmap` }

*The Face Options allow the application to require that certain tables are read during face construction.*

## Functions

- GR2\_API void `gr_engine_version` (int \*nMajor, int \*nMinor, int \*nBugFix)  
*Returns version information on this engine.*
- GR2\_API `gr_face` \* `gr_make_face_with_ops` (const void \*appFaceHandle, const `gr_face_ops` \*face\_ops, unsigned int faceOptions)  
*Create a `gr_face` object given application information and a table functions.*
- GR2\_DEPRECATED\_API `gr_face` \* `gr_make_face` (const void \*appFaceHandle, `gr_get_table_fn` getTable, unsigned int faceOptions)
- GR2\_DEPRECATED\_API `gr_face` \* `gr_make_face_with_seg_cache_and_ops` (const void \*appFaceHandle, const `gr_face_ops` \*face\_ops, unsigned int segCacheMaxSize, unsigned int faceOptions)
- GR2\_DEPRECATED\_API `gr_face` \* `gr_make_face_with_seg_cache` (const void \*appFaceHandle, `gr_get_table_fn` getTable, unsigned int segCacheMaxSize, unsigned int faceOptions)
- GR2\_API `gr_uint32` `gr_str_to_tag` (const char \*str)  
*Convert a tag in a string into a `gr_uint32`.*
- GR2\_API void `gr_tag_to_str` (`gr_uint32` tag, char \*str)  
*Convert a `gr_uint32` tag into a string.*
- GR2\_API `gr_feature_val` \* `gr_face_featureval_for_lang` (const `gr_face` \*pFace, `gr_uint32` langname)  
*Get feature values for a given language or default.*
- GR2\_API const `gr_feature_ref` \* `gr_face_find_fref` (const `gr_face` \*pFace, `gr_uint32` featId)  
*Get feature reference for a given feature id from a face.*
- GR2\_API `gr_uint16` `gr_face_n_fref` (const `gr_face` \*pFace)  
*Returns number of feature references in a face.*
- GR2\_API const `gr_feature_ref` \* `gr_face_fref` (const `gr_face` \*pFace, `gr_uint16` i)  
*Returns feature reference at given index in face.*
- GR2\_API unsigned short `gr_face_n_languages` (const `gr_face` \*pFace)  
*Return number of languages the face knows about.*
- GR2\_API `gr_uint32` `gr_face_lang_by_index` (const `gr_face` \*pFace, `gr_uint16` i)  
*Returns a language id corresponding to a language of given index in the face.*
- GR2\_API void `gr_face_destroy` (`gr_face` \*face)  
*Destroy the given face and free its memory.*
- GR2\_API unsigned short `gr_face_n_glyphs` (const `gr_face` \*pFace)  
*Returns the number of glyphs in the face.*
- GR2\_API const `gr_faceinfo` \* `gr_face_info` (const `gr_face` \*pFace, `gr_uint32` script)  
*Returns a `faceinfo` for the face and script.*
- GR2\_API int `gr_face_is_char_supported` (const `gr_face` \*pFace, `gr_uint32` usv, `gr_uint32` script)  
*Returns whether the font supports a given Unicode character.*
- GR2\_API `gr_face` \* `gr_make_file_face` (const char \*filename, unsigned int faceOptions)  
*Create `gr_face` from a font file.*
- GR2\_DEPRECATED\_API `gr_face` \* `gr_make_file_face_with_seg_cache` (const char \*filename, unsigned int segCacheMaxSize, unsigned int faceOptions)
- GR2\_API `gr_font` \* `gr_make_font` (float ppm, const `gr_face` \*face)  
*Create a font from a face.*
- GR2\_API `gr_font` \* `gr_make_font_with_ops` (float ppm, const void \*appFontHandle, const `gr_font_ops` \*font\_ops, const `gr_face` \*face)

- Creates a font with hinted advance width query functions.*

  - [GR2\\_API](#) [gr\\_font](#) \* [gr\\_make\\_font\\_with\\_advance\\_fn](#) (float ppm, const void \*appFontHandle, [gr\\_advance\\_fn](#) getAdvance, const [gr\\_face](#) \*face)

*Creates a font with hinted advance width query function.*

  - [GR2\\_API](#) void [gr\\_font\\_destroy](#) ([gr\\_font](#) \*font)

*Free a font.*

  - [GR2\\_API](#) [gr\\_uint16](#) [gr\\_fref\\_feature\\_value](#) (const [gr\\_feature\\_ref](#) \*pfeatureref, const [gr\\_feature\\_val](#) \*feats)

*get a feature value*

  - [GR2\\_API](#) int [gr\\_fref\\_set\\_feature\\_value](#) (const [gr\\_feature\\_ref](#) \*pfeatureref, [gr\\_uint16](#) val, [gr\\_feature\\_val](#) \*p← Dest)

*set a feature value*

  - [GR2\\_API](#) [gr\\_uint32](#) [gr\\_fref\\_id](#) (const [gr\\_feature\\_ref](#) \*pfeatureref)

*Returns the id tag for a gr\_feature\_ref.*

  - [GR2\\_API](#) [gr\\_uint16](#) [gr\\_fref\\_n\\_values](#) (const [gr\\_feature\\_ref](#) \*pfeatureref)

*Returns number of values a feature may take, given a gr\_feature\_ref.*

  - [GR2\\_API](#) [gr\\_int16](#) [gr\\_fref\\_value](#) (const [gr\\_feature\\_ref](#) \*pfeatureref, [gr\\_uint16](#) settingno)

*Returns the value associated with a particular value in a feature.*

  - [GR2\\_API](#) void \* [gr\\_fref\\_label](#) (const [gr\\_feature\\_ref](#) \*pfeatureref, [gr\\_uint16](#) \*langId, enum [gr\\_encform](#) utf, [gr\\_uint32](#) \*length)

*Returns a string of the UI name of a feature.*

  - [GR2\\_API](#) void \* [gr\\_fref\\_value\\_label](#) (const [gr\\_feature\\_ref](#) \*pfeatureref, [gr\\_uint16](#) settingno, [gr\\_uint16](#) \*langId, enum [gr\\_encform](#) utf, [gr\\_uint32](#) \*length)

*Return a UI string for a possible value of a feature.*

  - [GR2\\_API](#) void [gr\\_label\\_destroy](#) (void \*label)

*Destroy a previously returned label string.*

  - [GR2\\_API](#) [gr\\_feature\\_val](#) \* [gr\\_featureval\\_clone](#) (const [gr\\_feature\\_val](#) \*pfeatures)

*Copies a gr\_feature\_val.*

  - [GR2\\_API](#) void [gr\\_featureval\\_destroy](#) ([gr\\_feature\\_val](#) \*pfeatures)

*Destroys a gr\_feature\_val.*

## 5.1.1 Macro Definition Documentation

### 5.1.1.1 GR2\_VERSION\_BUGFIX

```
#define GR2_VERSION_BUGFIX 14
```

### 5.1.1.2 GR2\_VERSION\_MAJOR

```
#define GR2_VERSION_MAJOR 1
```

### 5.1.1.3 GR2\_VERSION\_MINOR

```
#define GR2_VERSION_MINOR 3
```

## 5.1.2 Typedef Documentation

### 5.1.2.1 gr\_advance\_fn

```
typedef float(* gr_advance_fn) (const void *appFontHandle, gr_uint16 glyphid)
```

query function to find the hinted advance of a glyph

#### Parameters

<i>appFontHandle</i>	is the unique information passed to gr_make_font_with_advance()
<i>glyphid</i>	is the glyph to retrieve the hinted advance for.

### 5.1.2.2 gr\_face

```
typedef struct gr_face gr_face
```

### 5.1.2.3 gr\_feature\_ref

```
typedef struct gr_feature_ref gr_feature_ref
```

### 5.1.2.4 gr\_feature\_val

```
typedef struct gr_feature_val gr_feature_val
```

### 5.1.2.5 gr\_font

```
typedef struct gr_font gr_font
```

### 5.1.2.6 gr\_get\_table\_fn

```
typedef const void*(* gr_get_table_fn) (const void *appFaceHandle, unsigned int name, size_t *len)
```

type describing function to retrieve font table information

#### Returns

a pointer to the table in memory. The pointed to memory must exist as long as the gr\_face which makes the call.

## Parameters

<i>appFaceHandle</i>	is the unique information passed to <a href="#">gr_make_face()</a>
<i>name</i>	is a 32bit tag to the table name.
<i>len</i>	returned by this function to say how long the table is in memory.

## 5.1.2.7 gr\_release\_table\_fn

```
typedef void(* gr_release_table_fn) (const void *appFaceHandle, const void *table_buffer)
```

type describing function to release any resources allocated by the above `get_table` table function

## Parameters

<i>appFaceHandle</i>	is the unique information passed to <a href="#">gr_make_face()</a>
<i>pointer</i>	to table memory returned by <code>get_table</code> .

## 5.1.3 Enumeration Type Documentation

## 5.1.3.1 gr\_face\_options

```
enum gr_face_options
```

The Face Options allow the application to require that certain tables are read during face construction.

This may be of concern if the `appFaceHandle` used in the `gr_get_table_fn` may change. The values can be combined

## Enumerator

<code>gr_face_default</code>	No preload, no cmap caching, fail if the graphite tables are invalid.
<code>gr_face_dumbRendering</code>	Dumb rendering will be enabled if the graphite tables are invalid. <b>Deprecated</b> Since 1.311
<code>gr_face_preloadGlyphs</code>	preload glyphs at construction time
<code>gr_face_cacheCmap</code>	Cache the lookup from code point to glyph ID at construction time.
<code>gr_face_preloadAll</code>	Preload everything.

## 5.1.4 Function Documentation

#### 5.1.4.1 gr\_engine\_version()

```
GR2_API void gr_engine_version (
    int * nMajor,
    int * nMinor,
    int * nBugFix )
```

Returns version information on this engine.

#### 5.1.4.2 gr\_face\_destroy()

```
GR2_API void gr_face_destroy (
    gr_face * face )
```

Destroy the given face and free its memory.

#### 5.1.4.3 gr\_face\_featureval\_for\_lang()

```
GR2_API gr_feature_val* gr_face_featureval_for_lang (
    const gr_face * pFace,
    gr_uint32 langname )
```

Get feature values for a given language or default.

##### Returns

a copy of the default feature values for a given language. The application must call [gr\\_featureval\\_destroy\(\)](#) to free this object when done.

##### Parameters

<i>pFace</i>	The font face to get feature values from
<i>langname</i>	The language tag to get feature values for. If there is no such language or langname is 0, the default feature values for the font are returned. langname is right 0 padded and assumes lowercase. Thus the en language would be 0x656E0000. Langname may also be space padded, thus 0x656E2020.

#### 5.1.4.4 gr\_face\_find\_fref()

```
GR2_API const gr_feature_ref* gr_face_find_fref (
    const gr_face * pFace,
    gr_uint32 featId )
```

Get feature reference for a given feature id from a face.



**Returns**

a feature reference corresponding to the given id. This data is part of the `gr_face` and will be freed when the face is destroyed.

**Parameters**

<i>pFace</i>	Font face to get information on.
<i>featId</i>	Feature id tag to get reference to.

**5.1.4.5 gr\_face\_fref()**

```
GR2_API const gr_feature_ref* gr_face_fref (
    const gr_face * pFace,
    gr_uint16 i )
```

Returns feature reference at given index in face.

**5.1.4.6 gr\_face\_info()**

```
GR2_API const gr_faceinfo* gr_face_info (
    const gr_face * pFace,
    gr_uint32 script )
```

Returns a faceinfo for the face and script.

**5.1.4.7 gr\_face\_is\_char\_supported()**

```
GR2_API int gr_face_is_char_supported (
    const gr_face * pFace,
    gr_uint32 usv,
    gr_uint32 script )
```

Returns whether the font supports a given Unicode character.

**Returns**

true if the character is supported.

**Parameters**

<i>pFace</i>	face to test within
<i>usv</i>	Unicode Scalar Value of character to test
<i>script</i>	Tag of script for selecting which set of pseudo glyphs to test. May be NULL.

#### 5.1.4.8 gr\_face\_lang\_by\_index()

```
GR2_API gr_uint32 gr_face_lang_by_index (
    const gr_face * pFace,
    gr_uint16 i )
```

Returns a language id corresponding to a language of given index in the face.

#### 5.1.4.9 gr\_face\_n\_fref()

```
GR2_API gr_uint16 gr_face_n_fref (
    const gr_face * pFace )
```

Returns number of feature references in a face.

#### 5.1.4.10 gr\_face\_n\_glyphs()

```
GR2_API unsigned short gr_face_n_glyphs (
    const gr_face * pFace )
```

Returns the number of glyphs in the face.

#### 5.1.4.11 gr\_face\_n\_languages()

```
GR2_API unsigned short gr_face_n_languages (
    const gr_face * pFace )
```

Return number of languages the face knows about.

#### 5.1.4.12 gr\_featureval\_clone()

```
GR2_API gr_feature_val* gr_featureval_clone (
    const gr_feature_val * pfeatures )
```

Copies a gr\_feature\_val.

#### 5.1.4.13 gr\_featureval\_destroy()

```
GR2_API void gr_featureval_destroy (
    gr_feature_val * pfeatures )
```

Destroys a gr\_feature\_val.

#### 5.1.4.14 gr\_font\_destroy()

```
GR2_API void gr_font_destroy (
    gr_font * font )
```

Free a font.

#### 5.1.4.15 gr\_fref\_feature\_value()

```
GR2_API gr_uint16 gr_fref_feature_value (
    const gr_feature_ref * pfeatureref,
    const gr_feature_val * feats )
```

get a feature value

##### Returns

value of specific feature or 0 if any problems.

##### Parameters

<i>pfeatureref</i>	gr_feature_ref to the feature
<i>feats</i>	gr_feature_val containing all the values

#### 5.1.4.16 gr\_fref\_id()

```
GR2_API gr_uint32 gr_fref_id (
    const gr_feature_ref * pfeatureref )
```

Returns the id tag for a gr\_feature\_ref.

### 5.1.4.17 gr\_fref\_label()

```
GR2_API void* gr_fref_label (
    const gr_feature_ref * pfeatureref,
    gr_uint16 * langId,
    enum gr_encform utf,
    gr_uint32 * length )
```

Returns a string of the UI name of a feature.

#### Returns

string of the UI name, in the encoding form requested. Call [gr\\_label\\_destroy\(\)](#) after use.

#### Parameters

<i>pfeatureref</i>	gr_feature_ref of the feature
<i>langId</i>	This is a pointer since the face may not support a string in the requested language. The actual language of the string is returned in langId
<i>utf</i>	Encoding form for the string
<i>length</i>	Used to return the length of the string returned in bytes.

### 5.1.4.18 gr\_fref\_n\_values()

```
GR2_API gr_uint16 gr_fref_n_values (
    const gr_feature_ref * pfeatureref )
```

Returns number of values a feature may take, given a gr\_feature\_ref.

### 5.1.4.19 gr\_fref\_set\_feature\_value()

```
GR2_API int gr_fref_set_feature_value (
    const gr_feature_ref * pfeatureref,
    gr_uint16 val,
    gr_feature_val * pDest )
```

set a feature value

#### Returns

false if there were any problems (value out of range, etc.)

#### Parameters

<i>pfeatureref</i>	gr_feature_ref to the feature
<i>val</i>	value to set the feature to
<i>pDest</i>	the gr_feature_val containing all the values for all the features

#### 5.1.4.20 gr\_fref\_value()

```
GR2_API gr_int16 gr_fref_value (
    const gr_feature_ref * pfeatureref,
    gr_uint16 settingno )
```

Returns the value associated with a particular value in a feature.

##### Returns

value

##### Parameters

<i>pfeatureref</i>	gr_feature_ref of the feature of interest
<i>settingno</i>	Index up to the return value of <a href="#">gr_fref_n_values()</a> of the value

#### 5.1.4.21 gr\_fref\_value\_label()

```
GR2_API void* gr_fref_value_label (
    const gr_feature_ref * pfeatureref,
    gr_uint16 settingno,
    gr_uint16 * langId,
    enum gr_encform utf,
    gr_uint32 * length )
```

Return a UI string for a possible value of a feature.

##### Returns

string of the UI name, in the encoding form requested. nul terminated. Call [gr\\_label\\_destroy\(\)](#) after use.

##### Parameters

<i>pfeatureref</i>	gr_feature_ref of the feature
<i>settingno</i>	Value setting index
<i>langId</i>	This is a pointer to the requested language. The requested language id is replaced by the actual language id of the string returned.
<i>utf</i>	Encoding form for the string
<i>length</i>	Returns the length of the string returned in bytes.

#### 5.1.4.22 `gr_label_destroy()`

```
GR2_API void gr_label_destroy (
    void * label )
```

Destroy a previously returned label string.

#### 5.1.4.23 `gr_make_face()`

```
GR2_DEPRECATED_API gr_face* gr_make_face (
    const void * appFaceHandle,
    gr_get_table_fn getTable,
    unsigned int faceOptions )
```

**Deprecated** Since v1.2.0 in favour of `gr_make_face_with_ops`.

Create a `gr_face` object given application information and a `getTable` function.

##### Returns

`gr_face` or NULL if the font fails to load for some reason.

##### Parameters

<i>appFaceHandle</i>	This is application specific information that is passed to the <code>getTable</code> function. The <code>appFaceHandle</code> must stay alive as long as the <code>gr_face</code> is alive.
<i>getTable</i>	Callback function to get table data.
<i>faceOptions</i>	Bitfield describing various options. See enum <code>gr_face_options</code> for details.

#### 5.1.4.24 `gr_make_face_with_ops()`

```
GR2_API gr_face* gr_make_face_with_ops (
    const void * appFaceHandle,
    const gr_face_ops * face_ops,
    unsigned int faceOptions )
```

Create a `gr_face` object given application information and a table functions.

##### Returns

`gr_face` or NULL if the font fails to load for some reason.

## Parameters

<i>appFaceHandle</i>	This is application specific information that is passed to the <code>getTable</code> function. The <code>appFaceHandle</code> must stay alive as long as the <code>gr_face</code> is alive.
<i>face_ops</i>	Pointer to face specific callback structure for table management. Must stay alive for the duration of the call only.
<i>faceOptions</i>	Bitfield describing various options. See enum <code>gr_face_options</code> for details.

5.1.4.25 `gr_make_face_with_seg_cache()`

```
GR2_DEPRECATED_API gr_face* gr_make_face_with_seg_cache (
    const void * appFaceHandle,
    gr_get_table_fn getTable,
    unsigned int segCacheMaxSize,
    unsigned int faceOptions )
```

**Deprecated** Since 1.3.7 this function is now an alias for `gr_make_face()`.

Create a `gr_face` object given application information, with subsegmental caching support. This function is deprecated as of v1.2.0 in favour of `gr_make_face_with_seg_cache_and_ops`.

## Returns

`gr_face` or NULL if the font fails to load.

## Parameters

<i>appFaceHandle</i>	is a pointer to application specific information that is passed to <code>getTable</code> . This may not be NULL and must stay alive as long as the <code>gr_face</code> is alive.
<i>getTable</i>	The function graphite calls to access font table data
<i>segCacheMaxSize</i>	How large the segment cache is.
<i>faceOptions</i>	Bitfield of values from enum <code>gr_face_options</code>

5.1.4.26 `gr_make_face_with_seg_cache_and_ops()`

```
GR2_DEPRECATED_API gr_face* gr_make_face_with_seg_cache_and_ops (
    const void * appFaceHandle,
    const gr_face_ops * face_ops,
    unsigned int segCacheMaxSize,
    unsigned int faceOptions )
```

**Deprecated** Since 1.3.7 this function is now an alias for `gr_make_face_with_ops()`.

Create a `gr_face` object given application information, with subsegmental caching support

**Returns**

gr\_face or NULL if the font fails to load.

**Parameters**

<i>appFaceHandle</i>	is a pointer to application specific information that is passed to getTable. This may not be NULL and must stay alive as long as the gr_face is alive.
<i>face_ops</i>	Pointer to face specific callback structure for table management. Must stay alive for the duration of the call only.
<i>segCacheMaxSize</i>	Unused.
<i>faceOptions</i>	Bitfield of values from enum gr_face_options

**5.1.4.27 gr\_make\_file\_face()**

```
GR2_API gr_face* gr_make_file_face (
    const char * filename,
    unsigned int faceOptions )
```

Create gr\_face from a font file.

**Returns**

gr\_face that accesses a font file directly. Returns NULL on failure.

**Parameters**

<i>filename</i>	Full path and filename to font file
<i>faceOptions</i>	Bitfile from enum gr_face_options to control face options.

**5.1.4.28 gr\_make\_file\_face\_with\_seg\_cache()**

```
GR2_DEPRECATED_API gr_face* gr_make_file_face_with_seg_cache (
    const char * filename,
    unsigned int segCacheMaxSize,
    unsigned int faceOptions )
```

**Deprecated** Since 1.3.7.

This function is now an alias for [gr\\_make\\_file\\_face\(\)](#).

Create gr\_face from a font file, with subsegment caching support.

**Returns**

gr\_face that accesses a font file directly. Returns NULL on failure.



## Parameters

<i>filename</i>	Full path and filename to font file
<i>segCacheMaxSize</i>	Specifies how big to make the cache in segments.
<i>faceOptions</i>	Bitfield from enum <code>gr_face_options</code> to control face options.

5.1.4.29 `gr_make_font()`

```
GR2_API gr_font* gr_make_font (
    float ppm,
    const gr_face * face )
```

Create a font from a face.

## Returns

`gr_font` Call `font_destroy` to free this font

## Parameters

<i>ppm</i>	Resolution of the font in pixels per em
<i>face</i>	Face this font corresponds to. This must stay alive as long as the font is alive.

5.1.4.30 `gr_make_font_with_advance_fn()`

```
GR2_API gr_font* gr_make_font_with_advance_fn (
    float ppm,
    const void * appFontHandle,
    gr_advance_fn getAdvance,
    const gr_face * face )
```

Creates a font with hinted advance width query function.

This function is deprecated. Use `gr_make_font_with_ops` instead.

## Returns

`gr_font` to be destroyed via `font_destroy`

## Parameters

<i>ppm</i>	size of font in pixels per em
<i>appFontHandle</i>	font specific information that must stay alive as long as the font does
<i>getAdvance</i>	callback function reference that returns horizontal advance in pixels for a glyph.
<i>face</i>	the face this font corresponds to. Must stay alive as long as the font does.

### 5.1.4.31 gr\_make\_font\_with\_ops()

```
GR2_API gr_font* gr_make_font_with_ops (
    float ppm,
    const void * appFontHandle,
    const gr_font_ops * font_ops,
    const gr_face * face )
```

Creates a font with hinted advance width query functions.

#### Returns

gr\_font to be destroyed via font\_destroy

#### Parameters

<i>ppm</i>	size of font in pixels per em
<i>appFontHandle</i>	font specific information that must stay alive as long as the font does
<i>font_ops</i>	pointer font specific callback structure for hinted metrics. Need only stay alive for the duration of the call.
<i>face</i>	the face this font corresponds to. Must stay alive as long as the font does.

### 5.1.4.32 gr\_str\_to\_tag()

```
GR2_API gr_uint32 gr_str_to_tag (
    const char * str )
```

Convert a tag in a string into a gr\_uint32.

#### Returns

gr\_uint32 tag, zero padded

#### Parameters

<i>str</i>	a nul terminated string of which at most the first 4 characters are read
------------	--

### 5.1.4.33 gr\_tag\_to\_str()

```
GR2_API void gr_tag_to_str (
    gr_uint32 tag,
    char * str )
```

Convert a `gr_uint32` tag into a string.

## Parameters

<i>tag</i>	contains the tag to convert
<i>str</i>	is a pointer to a char array of at least size 4 bytes. The first 4 bytes of this array will be overwritten by this function. No nul is appended.

## 5.2 Log.h File Reference

```
#include <graphite2/Types.h>
#include <graphite2/Font.h>
#include <stdio.h>
```

### Enumerations

- enum [GrLogMask](#) {  
[GRLOG\\_NONE](#) = 0x0 , [GRLOG\\_FACE](#) = 0x01 , [GRLOG\\_SEGMENT](#) = 0x02 , [GRLOG\\_PASS](#) = 0x04 ,  
[GRLOG\\_CACHE](#) = 0x08 , [GRLOG\\_OPCODE](#) = 0x80 , [GRLOG\\_ALL](#) = 0xFF }  
*deprecated mechanism that doesn't do anything now.*

### Functions

- [GR2\\_API](#) bool [gr\\_start\\_logging](#) ([gr\\_face](#) \*face, const char \*log\_path)  
*Start logging all segment creation and updates on the provided face.*
- [GR2\\_API](#) void [gr\\_stop\\_logging](#) ([gr\\_face](#) \*face)  
*Stop logging on the given face.*
- [GR2\\_API](#) bool [graphite\\_start\\_logging](#) (FILE \*logFile, [GrLogMask](#) mask)  
*Start logging to a FILE object.*
- [GR2\\_API](#) void [graphite\\_stop\\_logging](#) ()  
*Stop logging to a FILE object.*

### 5.2.1 Enumeration Type Documentation

#### 5.2.1.1 GrLogMask

```
enum GrLogMask
```

*deprecated mechanism that doesn't do anything now.*

#### Enumerator

GRLOG_NONE	
GRLOG_FACE	
GRLOG_SEGMENT	
GRLOG_PASS	
GRLOG_CACHE	
GRLOG_OPCODE	
GRLOG_ALL	

## 5.2.2 Function Documentation

### 5.2.2.1 gr\_start\_logging()

```
GR2_API bool gr_start_logging (
    gr_face * face,
    const char * log_path )
```

Start logging all segment creation and updates on the provided face.

This is logged to a JSON file, see "Segment JSON Schema.txt" for a precise definition of the file

#### Returns

true if the file was successfully created and logging is correctly initialised.

#### Parameters

<i>face</i>	the <i>gr_face</i> whose segments you want to log to the given file
<i>log_path</i>	a utf8 encoded file name and path to log to.

### 5.2.2.2 gr\_stop\_logging()

```
GR2_API void gr_stop_logging (
    gr_face * face )
```

Stop logging on the given face.

This will close the log file created by `gr_start_logging`.

#### Parameters

<i>face</i>	the <i>gr_face</i> whose segments you want to stop logging
-------------	--

### 5.2.2.3 graphite\_start\_logging()

```
GR2_API bool graphite_start_logging (
    FILE * logFile,
    GrLogMask mask )
```

Start logging to a FILE object.

This function is deprecated as of 1.2.0, use the `_face` versions instead.

**Returns**

True on success

**Parameters**

<i>logfile</i>	FILE reference to output logging to
<i>mask</i>	What aspects of logging to report (ignored)

**5.2.2.4 graphite\_stop\_logging()**

```
GR2_API void graphite_stop_logging ( )
```

Stop logging to a FILE object.

This function is deprecated as of 1.2.0, use the `_face` versions instead.

**5.3 Segment.h File Reference**

```
#include "graphite2/Types.h"
#include "graphite2/Font.h"
```

**Typedefs**

- typedef struct [gr\\_char\\_info](#) [gr\\_char\\_info](#)
- typedef struct [gr\\_segment](#) [gr\\_segment](#)
- typedef struct [gr\\_slot](#) [gr\\_slot](#)

**Enumerations**

- enum [gr\\_break\\_weight](#) {  
[gr\\_breakNone](#) = 0 , [gr\\_breakWhitespace](#) = 10 , [gr\\_breakWord](#) = 15 , [gr\\_breakIntra](#) = 20 ,  
[gr\\_breakLetter](#) = 30 , [gr\\_breakClip](#) = 40 , [gr\\_breakBeforeWhitespace](#) = -10 , [gr\\_breakBeforeWord](#) = -15 ,  
[gr\\_breakBeforeIntra](#) = -20 , [gr\\_breakBeforeLetter](#) = -30 , [gr\\_breakBeforeClip](#) = -40 }
  - enum [gr\\_justFlags](#) { [gr\\_justCompleteLine](#) = 0 , [gr\\_justStartInline](#) = 1 , [gr\\_justEndInline](#) = 2 }
  - enum [gr\\_attrCode](#) {  
[gr\\_slatAdvX](#) = 0 , [gr\\_slatAdvY](#) , [gr\\_slatAttTo](#) , [gr\\_slatAttX](#) ,  
[gr\\_slatAttY](#) , [gr\\_slatAttGpt](#) , [gr\\_slatAttXOff](#) , [gr\\_slatAttYOff](#) ,  
[gr\\_slatAttWithX](#) , [gr\\_slatAttWithY](#) , [gr\\_slatWithGpt](#) , [gr\\_slatAttWithXOff](#) ,  
[gr\\_slatAttWithYOff](#) , [gr\\_slatAttLevel](#) , [gr\\_slatBreak](#) , [gr\\_slatCompRef](#) ,  
[gr\\_slatDir](#) , [gr\\_slatInsert](#) , [gr\\_slatPosX](#) , [gr\\_slatPosY](#) ,  
[gr\\_slatShiftX](#) , [gr\\_slatShiftY](#) , [gr\\_slatUserDefnV1](#) , [gr\\_slatMeasureSol](#) ,  
[gr\\_slatMeasureEol](#) , [gr\\_slatJStretch](#) , [gr\\_slatJShrink](#) , [gr\\_slatJStep](#) ,  
[gr\\_slatJWeight](#) , [gr\\_slatJWidth](#) = 29 , [gr\\_slatSegSplit](#) = [gr\\_slatJStretch](#) + 29 , [gr\\_slatUserDefn](#) ,  
[gr\\_slatBidilLevel](#) = 56 , [gr\\_slatColFlags](#) , [gr\\_slatColLimitbx](#) , [gr\\_slatColLimitbly](#) ,  
[gr\\_slatColLimitrx](#) , [gr\\_slatColLimitry](#) , [gr\\_slatColShiftx](#) , [gr\\_slatColShifty](#) ,  
[gr\\_slatColMargin](#) , [gr\\_slatColMarginWt](#) , [gr\\_slatColExclGlyph](#) , [gr\\_slatColExclOffx](#) ,  
[gr\\_slatColExclOffy](#) , [gr\\_slatSeqClass](#) , [gr\\_slatSeqProxClass](#) , [gr\\_slatSeqOrder](#) ,  
[gr\\_slatSeqAboveXoff](#) , [gr\\_slatSeqAboveWt](#) , [gr\\_slatSeqBelowXlim](#) , [gr\\_slatSeqBelowWt](#) ,  
[gr\\_slatSeqValignHt](#) , [gr\\_slatSeqValignWt](#) , [gr\\_slatMax](#) , [gr\\_slatNoEffect](#) = [gr\\_slatMax](#) + 1 }
- Used for looking up slot attributes.*
- enum [gr\\_bidirrtl](#) { [gr\\_rtl](#) = 1 , [gr\\_nobidi](#) = 2 , [gr\\_nomirror](#) = 4 }

## Functions

- [GR2\\_API](#) unsigned int [gr\\_cinfo\\_unicode\\_char](#) (const [gr\\_char\\_info](#) \*p)  
*Returns Unicode character for a charinfo.*
- [GR2\\_API](#) int [gr\\_cinfo\\_break\\_weight](#) (const [gr\\_char\\_info](#) \*p)  
*Returns breakweight for a charinfo.*
- [GR2\\_API](#) int [gr\\_cinfo\\_after](#) (const [gr\\_char\\_info](#) \*p)  
*Returns the slot index that after this character is after in the slot stream.*
- [GR2\\_API](#) int [gr\\_cinfo\\_before](#) (const [gr\\_char\\_info](#) \*p)  
*Returns the slot index that before this character is before in the slot stream.*
- [GR2\\_API](#) size\_t [gr\\_cinfo\\_base](#) (const [gr\\_char\\_info](#) \*p)  
*Returns the code unit index of this character in the input string.*
- [GR2\\_API](#) size\_t [gr\\_count\\_unicode\\_characters](#) (enum [gr\\_encform](#) enc, const void \*buffer\_begin, const void \*buffer\_end, const void \*\*pError)  
*Returns the number of unicode characters in a string.*
- [GR2\\_API](#) [gr\\_segment](#) \* [gr\\_make\\_seg](#) (const [gr\\_font](#) \*font, const [gr\\_face](#) \*face, [gr\\_uint32](#) script, const [gr\\_feature\\_val](#) \*pFeats, enum [gr\\_encform](#) enc, const void \*pStart, size\_t nChars, int dir)  
*Creates and returns a segment.*
- [GR2\\_API](#) void [gr\\_seg\\_destroy](#) ([gr\\_segment](#) \*p)  
*Destroys a segment, freeing the memory.*
- [GR2\\_API](#) float [gr\\_seg\\_advance\\_X](#) (const [gr\\_segment](#) \*pSeg)  
*Returns the advance for the whole segment.*
- [GR2\\_API](#) float [gr\\_seg\\_advance\\_Y](#) (const [gr\\_segment](#) \*pSeg)  
*Returns the height advance for the segment.*
- [GR2\\_API](#) unsigned int [gr\\_seg\\_n\\_cinfo](#) (const [gr\\_segment](#) \*pSeg)  
*Returns the number of gr\_char\_infos in the segment.*
- [GR2\\_API](#) const [gr\\_char\\_info](#) \* [gr\\_seg\\_cinfo](#) (const [gr\\_segment](#) \*pSeg, unsigned int index)  
*Returns a gr\_char\_info at a given index in the segment.*
- [GR2\\_API](#) unsigned int [gr\\_seg\\_n\\_slots](#) (const [gr\\_segment](#) \*pSeg)  
*Returns the number of glyph gr\_slots in the segment.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_seg\\_first\\_slot](#) ([gr\\_segment](#) \*pSeg)  
*Returns the first gr\_slot in the segment.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_seg\\_last\\_slot](#) ([gr\\_segment](#) \*pSeg)  
*Returns the last gr\_slot in the segment.*
- [GR2\\_API](#) float [gr\\_seg\\_justify](#) ([gr\\_segment](#) \*pSeg, const [gr\\_slot](#) \*pStart, const [gr\\_font](#) \*pFont, double width, enum [gr\\_justFlags](#) flags, const [gr\\_slot](#) \*pFirst, const [gr\\_slot](#) \*pLast)  
*Justifies a linked list of slots for a line to a given width.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_slot\\_next\\_in\\_segment](#) (const [gr\\_slot](#) \*p)  
*Returns the next slot along in the segment.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_slot\\_prev\\_in\\_segment](#) (const [gr\\_slot](#) \*p)  
*Returns the previous slot along in the segment.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_slot\\_attached\\_to](#) (const [gr\\_slot](#) \*p)  
*Returns the attachment parent slot of this slot.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_slot\\_first\\_attachment](#) (const [gr\\_slot](#) \*p)  
*Returns the first slot attached to this slot.*
- [GR2\\_API](#) const [gr\\_slot](#) \* [gr\\_slot\\_next\\_sibling\\_attachment](#) (const [gr\\_slot](#) \*p)  
*Returns the next slot attached to our attachment parent.*
- [GR2\\_API](#) unsigned short [gr\\_slot\\_gid](#) (const [gr\\_slot](#) \*p)  
*Returns glyph id of the slot.*
- [GR2\\_API](#) float [gr\\_slot\\_origin\\_X](#) (const [gr\\_slot](#) \*p)  
*Returns X offset of glyph from start of segment.*

- **GR2\_API** float `gr_slot_origin_Y` (const `gr_slot` \*p)  
*Returns Y offset of glyph from start of segment.*
- **GR2\_API** float `gr_slot_advance_X` (const `gr_slot` \*p, const `gr_face` \*face, const `gr_font` \*font)  
*Returns the glyph advance for this glyph as adjusted for kerning.*
- **GR2\_API** float `gr_slot_advance_Y` (const `gr_slot` \*p, const `gr_face` \*face, const `gr_font` \*font)  
*Returns the vertical advance for the glyph in the slot adjusted for kerning.*
- **GR2\_API** int `gr_slot_before` (const `gr_slot` \*p)  
*Returns the `gr_char_info` index before us.*
- **GR2\_API** int `gr_slot_after` (const `gr_slot` \*p)  
*Returns the `gr_char_info` index after us.*
- **GR2\_API** unsigned int `gr_slot_index` (const `gr_slot` \*p)  
*Returns the index of this slot in the segment.*
- **GR2\_API** int `gr_slot_attr` (const `gr_slot` \*p, const `gr_segment` \*pSeg, enum `gr_attrCode` index, `gr_uint8` subindex)  
*Return a slot attribute value.*
- **GR2\_API** int `gr_slot_can_insert_before` (const `gr_slot` \*p)  
*Returns whether text may be inserted before this glyph.*
- **GR2\_API** int `gr_slot_original` (const `gr_slot` \*p)  
*Returns the original `gr_char_info` index this slot refers to.*
- **GR2\_API** void `gr_slot_linebreak_before` (`gr_slot` \*p)  
*Breaks a segment into lines.*

## 5.3.1 Typedef Documentation

### 5.3.1.1 `gr_char_info`

```
typedef struct gr_char_info gr_char_info
```

### 5.3.1.2 `gr_segment`

```
typedef struct gr_segment gr_segment
```

### 5.3.1.3 `gr_slot`

```
typedef struct gr_slot gr_slot
```

## 5.3.2 Enumeration Type Documentation

### 5.3.2.1 `gr_attrCode`

```
enum gr_attrCode
```

Used for looking up slot attributes.

Most are already available in other functions



## Enumerator

gr_slatAdvX	adjusted glyph advance in x direction in design units
gr_slatAdvY	adjusted glyph advance in y direction (usually 0) in design units
gr_slatAttTo	returns 0. Deprecated.
gr_slatAttX	This slot attaches to its parent at the given design units in the x direction.
gr_slatAttY	This slot attaches to its parent at the given design units in the y direction.
gr_slatAttGpt	This slot attaches to its parent at the given glyph point (not implemented)
gr_slatAttXOff	x-direction adjustment from the given glyph point (not implemented)
gr_slatAttYOff	y-direction adjustment from the given glyph point (not implemented)
gr_slatAttWithX	Where on this glyph should align with the attachment point on the parent glyph in the x-direction.
gr_slatAttWithY	Where on this glyph should align with the attachment point on the parent glyph in the y-direction.
gr_slatWithGpt	Which glyph point on this glyph should align with the attachment point on the parent glyph (not implemented).
gr_slatAttWithXOff	Adjustment to gr_slatWithGpt in x-direction (not implemented)
gr_slatAttWithYOff	Adjustment to gr_slatWithGpt in y-direction (not implemented)
gr_slatAttLevel	Attach at given nesting level (not implemented)
gr_slatBreak	Line break breakweight for this glyph.
gr_slatCompRef	Ligature component reference (not implemented)
gr_slatDir	bidi directionality of this glyph (not implemented)
gr_slatInsert	Whether insertion is allowed before this glyph.
gr_slatPosX	Final positioned position of this glyph relative to its parent in x-direction in pixels.
gr_slatPosY	Final positioned position of this glyph relative to its parent in y-direction in pixels.
gr_slatShiftX	Amount to shift glyph by in x-direction design units.
gr_slatShiftY	Amount to shift glyph by in y-direction design units.
gr_slatUserDefnV1	attribute user1
gr_slatMeasureSol	not implemented
gr_slatMeasureEol	not implemented
gr_slatJStretch	Amount this slot can stretch (not implemented)
gr_slatJShrink	Amount this slot can shrink (not implemented)
gr_slatJStep	Granularity by which this slot can stretch or shrink (not implemented)
gr_slatJWeight	Justification weight for this glyph (not implemented)
gr_slatJWidth	Amount this slot must shrink or stretch in design units.
gr_slatSegSplit	SubSegment split point.
gr_slatUserDefn	User defined attribute, see subattr for user attr number.
gr_slatBidiLevel	Bidi level.
gr_slatColFlags	Collision flags.
gr_slatColLimitblx	Collision constraint rectangle left (bl.x)
gr_slatColLimitbly	Collision constraint rectangle lower (bl.y)
gr_slatColLimittrx	Collision constraint rectangle right (tr.x)
gr_slatColLimittry	Collision constraint rectangle upper (tr.y)
gr_slatColShiftx	Collision shift x.
gr_slatColShifty	Collision shift y.
gr_slatColMargin	Collision margin.
gr_slatColMarginWt	Margin cost weight.
gr_slatColExclGlyph	
gr_slatColExclOffx	

## Enumerator

gr_slatColExclOffy	
gr_slatSeqClass	
gr_slatSeqProxClass	
gr_slatSeqOrder	
gr_slatSeqAboveXoff	
gr_slatSeqAboveWt	
gr_slatSeqBelowXlim	
gr_slatSeqBelowWt	
gr_slatSeqValignHt	
gr_slatSeqValignWt	
gr_slatMax	not implemented
gr_slatNoEffect	not implemented

## 5.3.2.2 gr\_bidirtl

```
enum gr_bidirtl
```

## Enumerator

gr_rtl	Underlying paragraph direction is RTL.
gr_nobidi	Set this to not run the bidi pass internally, even if the font asks for it. This presumes that the segment is in a single direction. Most of the time this bit should be set unless you know you are passing full paragraphs of text.
gr_nomirror	Disable auto mirroring for rtl text.

## 5.3.2.3 gr\_break\_weight

```
enum gr_break_weight
```

## Enumerator

gr_breakNone	
gr_breakWhitespace	
gr_breakWord	
gr_breakIntra	
gr_breakLetter	
gr_breakClip	
gr_breakBeforeWhitespace	
gr_breakBeforeWord	
gr_breakBeforeIntra	
gr_breakBeforeLetter	
gr_breakBeforeClip	

### 5.3.2.4 gr\_justFlags

enum `gr_justFlags`

Enumerator

<code>gr_justCompleteLine</code>	Indicates that this segment is a complete line.
<code>gr_justStartInline</code>	Indicates that the start of the slot list is not at the start of a line.
<code>gr_justEndInline</code>	Indicates that the end of the slot list is not at the end of a line.

## 5.3.3 Function Documentation

### 5.3.3.1 gr\_cinfo\_after()

```
GR2_API int gr_cinfo_after (
    const gr_char_info * p )
```

Returns the slot index that after this character is after in the slot stream.

In effect each character is associated with a set of slots and this returns the index of the last slot in the segment this character is associated with.

Returns

after slot index between 0 and `gr_seg_n_slots()`

Parameters

<code>p</code>	Pointer to charinfo to return information on.
----------------	---

### 5.3.3.2 gr\_cinfo\_base()

```
GR2_API size_t gr_cinfo_base (
    const gr_char_info * p )
```

Returns the code unit index of this character in the input string.

Returns

code unit index between 0 and the end of the string

## Parameters

<i>p</i>	Pointer to charinfo to return information on.
----------	---

### 5.3.3.3 gr\_cinfo\_before()

```
GR2_API int gr_cinfo_before (
    const gr_char_info * p )
```

Returns the slot index that before this character is before in the slot stream.

In effect each character is associated with a set of slots and this returns the index of the first slot in the segment this character is associated with.

## Returns

before slot index between 0 and [gr\\_seg\\_n\\_slots\(\)](#)

## Parameters

<i>p</i>	Pointer to charinfo to return information on.
----------	---

### 5.3.3.4 gr\_cinfo\_break\_weight()

```
GR2_API int gr_cinfo_break_weight (
    const gr_char_info * p )
```

Returns breakweight for a charinfo.

## Returns

Breakweight is a number between -50 and 50 indicating the cost of a break before or after this character. If the value < 0, the absolute value is this character's contribution to the overall breakweight before it. If the value

0, then the value is this character's contribution to the overall breakweight after it.

The overall breakweight between two characters is the maximum of the breakweight contributions from the characters either side of it. If a character makes no contribution to the breakweight on one side of it, the contribution is considered to be 0.

## Parameters

<i>p</i>	Pointer to charinfo to return information on.
----------	---

### 5.3.3.5 gr\_cinfo\_unicode\_char()

```
GR2_API unsigned int gr_cinfo_unicode_char (
    const gr_char_info * p )
```

Returns Unicode character for a charinfo.

#### Parameters

<i>p</i>	Pointer to charinfo to return information on.
----------	---

### 5.3.3.6 gr\_count\_unicode\_characters()

```
GR2_API size_t gr_count_unicode_characters (
    enum gr_encform enc,
    const void * buffer_begin,
    const void * buffer_end,
    const void ** pError )
```

Returns the number of unicode characters in a string.

#### Returns

number of characters in the string

#### Parameters

<i>enc</i>	Specifies the type of data in the string: utf8, utf16, utf32
<i>buffer_begin</i>	The start of the string
<i>buffer_end</i>	Measure up to the first nul or when end is reached, whichever is earliest. This parameter may be NULL.
<i>pError</i>	If there is a structural fault in the string, the location is returned in this variable. If no error occurs, pError will contain NULL. NULL may be passed for pError if no such information is required.

### 5.3.3.7 gr\_make\_seg()

```
GR2_API gr_segment* gr_make_seg (
    const gr_font * font,
    const gr_face * face,
    gr_uint32 script,
    const gr_feature_val * pFeats,
    enum gr_encform enc,
    const void * pStart,
    size_t nChars,
    int dir )
```

Creates and returns a segment.

**Returns**

a segment that needs `seg_destroy` called on it. May return NULL if bad problems in segment processing.

**Parameters**

<i>font</i>	Gives the size of the font in pixels per em for final positioning. If NULL, positions are returned in design units, i.e. at a ppm of the upem of the face.
<i>face</i>	The face containing all the non-size dependent information.
<i>script</i>	This is a tag containing a script identifier that is used to choose which graphite table within the font to use. Maybe 0. Tag may be 4 chars NULL padded in LSBs or space padded in LSBs.
<i>pFeats</i>	Pointer to a feature values to be used for the segment. Only one feature values may be used for a segment. If NULL the default features for the font will be used.
<i>enc</i>	Specifies what encoding form the string is in (utf8, utf16, utf32)
<i>pStart</i>	Start of the string
<i>nChars</i>	Number of unicode characters to process in the string. The string will be processed either up to the first NULL or until nChars have been processed. nChars is also used to initialise the internal memory allocations of the segment. So it is wise not to make nChars too much greater than the actual number of characters being processed.
<i>dir</i>	Specifies whether the segment is processed right to left (1) or left to right (0) and whether to run the internal bidi pass, if a font requests it. See enum <code>gr_bidir</code> for details.

**5.3.3.8 gr\_seg\_advance\_X()**

```
GR2_API float gr_seg_advance_X (
    const gr_segment * pSeg )
```

Returns the advance for the whole segment.

Returns the width of the segment up to the next glyph origin after the segment

**5.3.3.9 gr\_seg\_advance\_Y()**

```
GR2_API float gr_seg_advance_Y (
    const gr_segment * pSeg )
```

Returns the height advance for the segment.

**5.3.3.10 gr\_seg\_cinfo()**

```
GR2_API const gr_char_info* gr_seg_cinfo (
    const gr_segment * pSeg,
    unsigned int index )
```

Returns a `gr_char_info` at a given index in the segment.

**5.3.3.11 gr\_seg\_destroy()**

```
GR2_API void gr_seg_destroy (
    gr_segment * p )
```

Destroys a segment, freeing the memory.

## Parameters

<i>p</i>	The segment to destroy
----------	------------------------

**5.3.3.12 gr\_seg\_first\_slot()**

```
GR2_API const gr_slot* gr_seg_first_slot (
    gr_segment * pSeg )
```

Returns the first `gr_slot` in the segment.

The first slot in a segment has a `gr_slot_prev_in_segment()` of NULL. Slots are owned by their segment and are destroyed along with the segment.

**5.3.3.13 gr\_seg\_justify()**

```
GR2_API float gr_seg_justify (
    gr_segment * pSeg,
    const gr_slot * pStart,
    const gr_font * pFont,
    double width,
    enum gr_justFlags flags,
    const gr_slot * pFirst,
    const gr_slot * pLast )
```

Justifies a linked list of slots for a line to a given width.

Passed a pointer to the start of a linked list of slots corresponding to a line, as set up by `gr_slot_linebreak_before`, this function will position the glyphs in the line to take up the given width. It is possible to specify a subrange within the line to process. This allows skipping of line initial or final whitespace, for example. While this will ensure that the subrange fits width, the line will still be positioned with the first glyph of the line at 0. So the resulting positions may be beyond width.

## Returns

float The resulting width of the range of slots justified.

## Parameters

<i>pSeg</i>	Pointer to the segment
<i>pStart</i>	Pointer to the start of the line linked list (including skipped characters)
<i>pFont</i>	Font to use for positioning
<i>width</i>	Width in pixels in which to fit the line. If < 0. don't adjust natural width, just run justification passes to handle line end contextuials, if there are any.
<i>flags</i>	Indicates line ending types. Default is linked list is a full line
<i>pFirst</i>	If not NULL, the first slot in the list to be considered part of the line (so can skip)
<i>pLast</i>	If not NULL, the last slot to process in the line (allow say trailing whitespace to be skipped)

### 5.3.3.14 gr\_seg\_last\_slot()

```
GR2_API const gr_slot* gr_seg_last_slot (
    gr_segment * pSeg )
```

Returns the last gr\_slot in the segment.

The last slot in a segment has a [gr\\_slot\\_next\\_in\\_segment\(\)](#) of NULL

### 5.3.3.15 gr\_seg\_n\_cinfo()

```
GR2_API unsigned int gr_seg_n_cinfo (
    const gr_segment * pSeg )
```

Returns the number of gr\_char\_infos in the segment.

### 5.3.3.16 gr\_seg\_n\_slots()

```
GR2_API unsigned int gr_seg_n_slots (
    const gr_segment * pSeg )
```

Returns the number of glyph gr\_slots in the segment.

### 5.3.3.17 gr\_slot\_advance\_X()

```
GR2_API float gr_slot_advance_X (
    const gr_slot * p,
    const gr_face * face,
    const gr_font * font )
```

Returns the glyph advance for this glyph as adjusted for kerning.

#### Parameters

<i>p</i>	Slot to give results for
<i>face</i>	gr_face of the glyphs. May be NULL if unhinted advances used
<i>font</i>	gr_font to scale for pixel results. If NULL returns design units advance. If not NULL then returns pixel advance based on hinted or scaled glyph advances in the font. face must be passed for hinted advances to be used.



**5.3.3.18 gr\_slot\_advance\_Y()**

```
GR2_API float gr_slot_advance_Y (
    const gr_slot * p,
    const gr_face * face,
    const gr_font * font )
```

Returns the vertical advance for the glyph in the slot adjusted for kerning.

Returns design units unless font is not NULL in which case the pixel value is returned scaled for the given font

**5.3.3.19 gr\_slot\_after()**

```
GR2_API int gr_slot_after (
    const gr_slot * p )
```

Returns the gr\_char\_info index after us.

Returns the index of the gr\_char\_info that a cursor after this slot would put an underlying cursor after. This may also be interpreted as each slot holding a set of char\_infos that it is associated with and this function returning the index of the char\_info with the highest index, from this set.

**5.3.3.20 gr\_slot\_attached\_to()**

```
GR2_API const gr_slot* gr_slot_attached_to (
    const gr_slot * p )
```

Returns the attachment parent slot of this slot.

Attached slots form a tree. This returns the parent of this slot in that tree. A base glyph which is not attached to another glyph, always returns NULL.

**5.3.3.21 gr\_slot\_attr()**

```
GR2_API int gr_slot_attr (
    const gr_slot * p,
    const gr_segment * pSeg,
    enum gr_attrCode index,
    gr_uint8 subindex )
```

Return a slot attribute value.

Given a slot and an attribute along with a possible subattribute, return the corresponding value in the slot. See enum gr\_attrCode for details of each attribute.

**5.3.3.22 gr\_slot\_before()**

```
GR2_API int gr_slot_before (
    const gr_slot * p )
```

Returns the gr\_char\_info index before us.

Returns the index of the gr\_char\_info that a cursor before this slot, would put an underlying cursor before. This may also be interpreted as each slot holding a set of char\_infos that it is associated with and this function returning the index of the char\_info with lowest index, from this set.

**5.3.3.23 gr\_slot\_can\_insert\_before()**

```
GR2_API int gr_slot_can_insert_before (
    const gr_slot * p )
```

Returns whether text may be inserted before this glyph.

This indicates whether a cursor can be put before this slot. It applies to base glyphs that have no parent as well as attached glyphs that have the `.insert` attribute explicitly set to true. This is the primary mechanism for identifying contiguous sequences of base plus diacritics.

**5.3.3.24 gr\_slot\_first\_attachment()**

```
GR2_API const gr_slot* gr_slot_first_attachment (
    const gr_slot * p )
```

Returns the first slot attached to this slot.

Attached slots form a singly linked list from the parent. This returns the first slot in that list. Note that this is a reference to another slot that is also in the main segment doubly linked list.

if `gr_slot_first_attachment(p) != NULL` then `gr_slot_attached_to(gr_slot_first_attachment(p)) == p`.

**5.3.3.25 gr\_slot\_gid()**

```
GR2_API unsigned short gr_slot_gid (
    const gr_slot * p )
```

Returns glyph id of the slot.

Each slot has a glyhid which is rendered at the position given by the slot. This glyhid is the real glyph to be rendered and never a pseudo glyph.

**5.3.3.26 gr\_slot\_index()**

```
GR2_API unsigned int gr_slot_index (
    const gr_slot * p )
```

Returns the index of this slot in the segment.

Returns the index given to this slot during final positioning. This corresponds to the value returned by `gr_cinfo_before()` and `gr_cinfo_after()`

**5.3.3.27 gr\_slot\_linebreak\_before()**

```
GR2_API void gr_slot_linebreak_before (
    gr_slot * p )
```

Breaks a segment into lines.

Breaks the slot linked list at the given point in the linked list. It is up to the application to keep track of the first slot on each line.

### 5.3.3.28 `gr_slot_next_in_segment()`

```
GR2_API const gr_slot* gr_slot_next_in_segment (
    const gr_slot * p )
```

Returns the next slot along in the segment.

Slots are held in a linked list. This returns the next in the linked list. The slot may or may not be attached to another slot. Returns NULL at the end of the segment.

### 5.3.3.29 `gr_slot_next_sibling_attachment()`

```
GR2_API const gr_slot* gr_slot_next_sibling_attachment (
    const gr_slot * p )
```

Returns the next slot attached to our attachment parent.

This returns the next slot in the singly linked list of slots attached to this slot's parent. If there are no more such slots, NULL is returned. If there is no parent, i.e. the passed slot is a cluster base, then the next cluster base in graphical order (ltr, even for rtl text) is returned.

if `gr_slot_next_sibling_attachment(p) != NULL` then `gr_slot_attached_to(gr_slot_next_sibling_attachment(p)) == gr_slot_attached_to(p)`.

### 5.3.3.30 `gr_slot_origin_X()`

```
GR2_API float gr_slot_origin_X (
    const gr_slot * p )
```

Returns X offset of glyph from start of segment.

### 5.3.3.31 `gr_slot_origin_Y()`

```
GR2_API float gr_slot_origin_Y (
    const gr_slot * p )
```

Returns Y offset of glyph from start of segment.

### 5.3.3.32 `gr_slot_original()`

```
GR2_API int gr_slot_original (
    const gr_slot * p )
```

Returns the original `gr_char_info` index this slot refers to.

Each Slot has a `gr_char_info` that it originates from. This is that `gr_char_info`. The index is passed to `gr_seg_cinfo()`. This information is useful for testing.

### 5.3.3.33 gr\_slot\_prev\_in\_segment()

```
GR2_API const gr_slot* gr_slot_prev_in_segment (
    const gr_slot * p )
```

Returns the previous slot along in the segment.

Slots are held in a doubly linked list. This returns the previous slot in the linked list. This slot may or may not be attached to it. Returns NULL at the start of the segment.

## 5.4 Types.h File Reference

```
#include <stddef.h>
```

### Macros

- #define [GR2\\_API](#) \_gr2\_tag\_fn(\_gr2\_import\_flag)
- #define [GR2\\_DEPRECATED\\_API](#) \_gr2\_tag\_fn(\_gr2\_deprecated\_flag \_gr2\_and \_gr2\_import\_flag)

### Typedefs

- typedef unsigned char [gr\\_uint8](#)
- typedef [gr\\_uint8](#) [gr\\_byte](#)
- typedef signed char [gr\\_int8](#)
- typedef unsigned short [gr\\_uint16](#)
- typedef short [gr\\_int16](#)
- typedef unsigned int [gr\\_uint32](#)
- typedef int [gr\\_int32](#)

### Enumerations

- enum [gr\\_encform](#) { [gr\\_utf8](#) = 1 , [gr\\_utf16](#) = 2 , [gr\\_utf32](#) = 4 }

## 5.4.1 Macro Definition Documentation

### 5.4.1.1 GR2\_API

```
#define GR2_API _gr2_tag_fn(_gr2_import_flag)
```

### 5.4.1.2 GR2\_DEPRECATED\_API

```
#define GR2_DEPRECATED_API _gr2_tag_fn(_gr2_deprecated_flag _gr2_and _gr2_import_flag)
```

## 5.4.2 Typedef Documentation

### 5.4.2.1 gr\_byte

```
typedef gr_uint8 gr_byte
```

### 5.4.2.2 gr\_int16

```
typedef short gr_int16
```

### 5.4.2.3 gr\_int32

```
typedef int gr_int32
```

### 5.4.2.4 gr\_int8

```
typedef signed char gr_int8
```

### 5.4.2.5 gr\_uint16

```
typedef unsigned short gr_uint16
```

### 5.4.2.6 gr\_uint32

```
typedef unsigned int gr_uint32
```

### 5.4.2.7 gr\_uint8

```
typedef unsigned char gr_uint8
```

## 5.4.3 Enumeration Type Documentation

### 5.4.3.1 gr\_encform

```
enum gr_encform
```

## Enumerator

gr_utf8	
gr_utf16	
gr_utf32	

# Index

- extra\_ascent
  - gr\_faceinfo, 10
- extra\_descent
  - gr\_faceinfo, 10
- Font.h, 13
  - GR2\_VERSION\_BUGFIX, 15
  - GR2\_VERSION\_MAJOR, 15
  - GR2\_VERSION\_MINOR, 15
  - gr\_advance\_fn, 16
  - gr\_engine\_version, 17
  - gr\_face, 16
  - gr\_face\_cacheCmap, 17
  - gr\_face\_default, 17
  - gr\_face\_destroy, 18
  - gr\_face\_dumbRendering, 17
  - gr\_face\_featureval\_for\_lang, 18
  - gr\_face\_find\_fref, 18
  - gr\_face\_fref, 19
  - gr\_face\_info, 19
  - gr\_face\_is\_char\_supported, 19
  - gr\_face\_lang\_by\_index, 20
  - gr\_face\_n\_fref, 20
  - gr\_face\_n\_glyphs, 20
  - gr\_face\_n\_languages, 20
  - gr\_face\_options, 17
  - gr\_face\_preloadAll, 17
  - gr\_face\_preloadGlyphs, 17
  - gr\_feature\_ref, 16
  - gr\_feature\_val, 16
  - gr\_featureval\_clone, 20
  - gr\_featureval\_destroy, 20
  - gr\_font, 16
  - gr\_font\_destroy, 21
  - gr\_fref\_feature\_value, 21
  - gr\_fref\_id, 21
  - gr\_fref\_label, 21
  - gr\_fref\_n\_values, 22
  - gr\_fref\_set\_feature\_value, 22
  - gr\_fref\_value, 23
  - gr\_fref\_value\_label, 23
  - gr\_get\_table\_fn, 16
  - gr\_label\_destroy, 23
  - gr\_make\_face, 24
  - gr\_make\_face\_with\_ops, 24
  - gr\_make\_face\_with\_seg\_cache, 25
  - gr\_make\_face\_with\_seg\_cache\_and\_ops, 25
  - gr\_make\_file\_face, 26
  - gr\_make\_file\_face\_with\_seg\_cache, 26
  - gr\_make\_font, 27
  - gr\_make\_font\_with\_advance\_fn, 27
  - gr\_make\_font\_with\_ops, 28
  - gr\_release\_table\_fn, 17
  - gr\_str\_to\_tag, 28
  - gr\_tag\_to\_str, 28
- get\_table
  - gr\_face\_ops, 7
- glyph\_advance\_x
  - gr\_font\_ops, 12
- glyph\_advance\_y
  - gr\_font\_ops, 12
- GR2\_API
  - Types.h, 46
- GR2\_DEPRECATED\_API
  - Types.h, 46
- GR2\_VERSION\_BUGFIX
  - Font.h, 15
- GR2\_VERSION\_MAJOR
  - Font.h, 15
- GR2\_VERSION\_MINOR
  - Font.h, 15
- gr\_advance\_fn
  - Font.h, 16
- gr\_attrCode
  - Segment.h, 34
- gr\_bidirtl
  - Segment.h, 36
- gr\_break\_weight
  - Segment.h, 36
- gr\_breakBeforeClip
  - Segment.h, 36
- gr\_breakBeforeIntra
  - Segment.h, 36
- gr\_breakBeforeLetter
  - Segment.h, 36
- gr\_breakBeforeWhitespace
  - Segment.h, 36
- gr\_breakBeforeWord
  - Segment.h, 36
- gr\_breakClip
  - Segment.h, 36
- gr\_breakIntra
  - Segment.h, 36
- gr\_breakLetter
  - Segment.h, 36
- gr\_breakNone
  - Segment.h, 36
- gr\_breakWhitespace
  - Segment.h, 36

gr\_breakWord  
  Segment.h, 36

gr\_byte  
  Types.h, 47

gr\_char\_info  
  Segment.h, 34

gr\_cinfo\_after  
  Segment.h, 37

gr\_cinfo\_base  
  Segment.h, 37

gr\_cinfo\_before  
  Segment.h, 38

gr\_cinfo\_break\_weight  
  Segment.h, 38

gr\_cinfo\_unicode\_char  
  Segment.h, 38

gr\_count\_unicode\_characters  
  Segment.h, 39

gr\_encform  
  Types.h, 47

gr\_engine\_version  
  Font.h, 17

gr\_face  
  Font.h, 16

gr\_face\_cacheCmap  
  Font.h, 17

gr\_face\_default  
  Font.h, 17

gr\_face\_destroy  
  Font.h, 18

gr\_face\_dumbRendering  
  Font.h, 17

gr\_face\_featureval\_for\_lang  
  Font.h, 18

gr\_face\_find\_fref  
  Font.h, 18

gr\_face\_fref  
  Font.h, 19

gr\_face\_info  
  Font.h, 19

gr\_face\_is\_char\_supported  
  Font.h, 19

gr\_face\_lang\_by\_index  
  Font.h, 20

gr\_face\_n\_fref  
  Font.h, 20

gr\_face\_n\_glyphs  
  Font.h, 20

gr\_face\_n\_languages  
  Font.h, 20

gr\_face\_ops, 7  
  get\_table, 7  
  release\_table, 7  
  size, 8

gr\_face\_options  
  Font.h, 17

gr\_face\_preloadAll  
  Font.h, 17

gr\_face\_preloadGlyphs  
  Font.h, 17

gr\_faceinfo, 8  
  extra\_ascent, 10  
  extra\_descent, 10  
  gr\_space\_both, 10  
  gr\_space\_contextuals, 9  
  gr\_space\_cross, 10  
  gr\_space\_either\_only, 10  
  gr\_space\_left\_only, 10  
  gr\_space\_none, 10  
  gr\_space\_right\_only, 10  
  gr\_space\_unknown, 10  
  has\_bidi\_pass, 10  
  justifies, 10  
  line\_ends, 10  
  space\_contextuals, 11  
  upem, 11

gr\_feature\_ref  
  Font.h, 16

gr\_feature\_val  
  Font.h, 16

gr\_featureval\_clone  
  Font.h, 20

gr\_featureval\_destroy  
  Font.h, 20

gr\_font  
  Font.h, 16

gr\_font\_destroy  
  Font.h, 21

gr\_font\_ops, 11  
  glyph\_advance\_x, 12  
  glyph\_advance\_y, 12  
  size, 12

gr\_fref\_feature\_value  
  Font.h, 21

gr\_fref\_id  
  Font.h, 21

gr\_fref\_label  
  Font.h, 21

gr\_fref\_n\_values  
  Font.h, 22

gr\_fref\_set\_feature\_value  
  Font.h, 22

gr\_fref\_value  
  Font.h, 23

gr\_fref\_value\_label  
  Font.h, 23

gr\_get\_table\_fn  
  Font.h, 16

gr\_int16  
  Types.h, 47

gr\_int32  
  Types.h, 47

gr\_int8  
  Types.h, 47

gr\_justCompleteLine  
  Segment.h, 37



`gr_justEndInline`  
Segment.h, 37

`gr_justFlags`  
Segment.h, 37

`gr_justStartInline`  
Segment.h, 37

`gr_label_destroy`  
Font.h, 23

`gr_make_face`  
Font.h, 24

`gr_make_face_with_ops`  
Font.h, 24

`gr_make_face_with_seg_cache`  
Font.h, 25

`gr_make_face_with_seg_cache_and_ops`  
Font.h, 25

`gr_make_file_face`  
Font.h, 26

`gr_make_file_face_with_seg_cache`  
Font.h, 26

`gr_make_font`  
Font.h, 27

`gr_make_font_with_advance_fn`  
Font.h, 27

`gr_make_font_with_ops`  
Font.h, 28

`gr_make_seg`  
Segment.h, 39

`gr_nobidi`  
Segment.h, 36

`gr_nomirror`  
Segment.h, 36

`gr_release_table_fn`  
Font.h, 17

`gr_rtl`  
Segment.h, 36

`gr_seg_advance_X`  
Segment.h, 40

`gr_seg_advance_Y`  
Segment.h, 40

`gr_seg_cinfo`  
Segment.h, 40

`gr_seg_destroy`  
Segment.h, 40

`gr_seg_first_slot`  
Segment.h, 41

`gr_seg_justify`  
Segment.h, 41

`gr_seg_last_slot`  
Segment.h, 42

`gr_seg_n_cinfo`  
Segment.h, 42

`gr_seg_n_slots`  
Segment.h, 42

`gr_segment`  
Segment.h, 34

`gr_slatAdvX`  
Segment.h, 35

`gr_slatAdvY`  
Segment.h, 35

`gr_slatAttGpt`  
Segment.h, 35

`gr_slatAttLevel`  
Segment.h, 35

`gr_slatAttTo`  
Segment.h, 35

`gr_slatAttWithX`  
Segment.h, 35

`gr_slatAttWithXOff`  
Segment.h, 35

`gr_slatAttWithY`  
Segment.h, 35

`gr_slatAttWithYOff`  
Segment.h, 35

`gr_slatAttX`  
Segment.h, 35

`gr_slatAttXOff`  
Segment.h, 35

`gr_slatAttY`  
Segment.h, 35

`gr_slatAttYOff`  
Segment.h, 35

`gr_slatBidiLevel`  
Segment.h, 35

`gr_slatBreak`  
Segment.h, 35

`gr_slatColExclGlyph`  
Segment.h, 35

`gr_slatColExclOffx`  
Segment.h, 35

`gr_slatColExclOffy`  
Segment.h, 36

`gr_slatColFlags`  
Segment.h, 35

`gr_slatColLimitblx`  
Segment.h, 35

`gr_slatColLimitbly`  
Segment.h, 35

`gr_slatColLimittrx`  
Segment.h, 35

`gr_slatColLimittry`  
Segment.h, 35

`gr_slatColMargin`  
Segment.h, 35

`gr_slatColMarginWt`  
Segment.h, 35

`gr_slatColShiftx`  
Segment.h, 35

`gr_slatColShifty`  
Segment.h, 35

`gr_slatCompRef`  
Segment.h, 35

`gr_slatDir`  
Segment.h, 35

`gr_slatInsert`  
Segment.h, 35

[gr\\_slatJShrink](#)  
   Segment.h, [35](#)

[gr\\_slatJStep](#)  
   Segment.h, [35](#)

[gr\\_slatJStretch](#)  
   Segment.h, [35](#)

[gr\\_slatJWeight](#)  
   Segment.h, [35](#)

[gr\\_slatJWidth](#)  
   Segment.h, [35](#)

[gr\\_slatMax](#)  
   Segment.h, [36](#)

[gr\\_slatMeasureEol](#)  
   Segment.h, [35](#)

[gr\\_slatMeasureSol](#)  
   Segment.h, [35](#)

[gr\\_slatNoEffect](#)  
   Segment.h, [36](#)

[gr\\_slatPosX](#)  
   Segment.h, [35](#)

[gr\\_slatPosY](#)  
   Segment.h, [35](#)

[gr\\_slatSegSplit](#)  
   Segment.h, [35](#)

[gr\\_slatSeqAboveWt](#)  
   Segment.h, [36](#)

[gr\\_slatSeqAboveXoff](#)  
   Segment.h, [36](#)

[gr\\_slatSeqBelowWt](#)  
   Segment.h, [36](#)

[gr\\_slatSeqBelowXlim](#)  
   Segment.h, [36](#)

[gr\\_slatSeqClass](#)  
   Segment.h, [36](#)

[gr\\_slatSeqOrder](#)  
   Segment.h, [36](#)

[gr\\_slatSeqProxClass](#)  
   Segment.h, [36](#)

[gr\\_slatSeqValignHt](#)  
   Segment.h, [36](#)

[gr\\_slatSeqValignWt](#)  
   Segment.h, [36](#)

[gr\\_slatShiftX](#)  
   Segment.h, [35](#)

[gr\\_slatShiftY](#)  
   Segment.h, [35](#)

[gr\\_slatUserDefn](#)  
   Segment.h, [35](#)

[gr\\_slatUserDefnV1](#)  
   Segment.h, [35](#)

[gr\\_slatWithGpt](#)  
   Segment.h, [35](#)

[gr\\_slot](#)  
   Segment.h, [34](#)

[gr\\_slot\\_advance\\_X](#)  
   Segment.h, [42](#)

[gr\\_slot\\_advance\\_Y](#)  
   Segment.h, [42](#)

[gr\\_slot\\_after](#)  
   Segment.h, [43](#)

[gr\\_slot\\_attached\\_to](#)  
   Segment.h, [43](#)

[gr\\_slot\\_attr](#)  
   Segment.h, [43](#)

[gr\\_slot\\_before](#)  
   Segment.h, [43](#)

[gr\\_slot\\_can\\_insert\\_before](#)  
   Segment.h, [43](#)

[gr\\_slot\\_first\\_attachment](#)  
   Segment.h, [44](#)

[gr\\_slot\\_gid](#)  
   Segment.h, [44](#)

[gr\\_slot\\_index](#)  
   Segment.h, [44](#)

[gr\\_slot\\_linebreak\\_before](#)  
   Segment.h, [44](#)

[gr\\_slot\\_next\\_in\\_segment](#)  
   Segment.h, [44](#)

[gr\\_slot\\_next\\_sibling\\_attachment](#)  
   Segment.h, [45](#)

[gr\\_slot\\_origin\\_X](#)  
   Segment.h, [45](#)

[gr\\_slot\\_origin\\_Y](#)  
   Segment.h, [45](#)

[gr\\_slot\\_original](#)  
   Segment.h, [45](#)

[gr\\_slot\\_prev\\_in\\_segment](#)  
   Segment.h, [45](#)

[gr\\_space\\_both](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_contextuals](#)  
   gr\_faceinfo, [9](#)

[gr\\_space\\_cross](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_either\\_only](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_left\\_only](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_none](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_right\\_only](#)  
   gr\_faceinfo, [10](#)

[gr\\_space\\_unknown](#)  
   gr\_faceinfo, [10](#)

[gr\\_start\\_logging](#)  
   Log.h, [31](#)

[gr\\_stop\\_logging](#)  
   Log.h, [31](#)

[gr\\_str\\_to\\_tag](#)  
   Font.h, [28](#)

[gr\\_tag\\_to\\_str](#)  
   Font.h, [28](#)

[gr\\_uint16](#)  
   Types.h, [47](#)

[gr\\_uint32](#)  
   Types.h, [47](#)

gr\_uint8  
     Types.h, 47  
 gr\_utf16  
     Types.h, 48  
 gr\_utf32  
     Types.h, 48  
 gr\_utf8  
     Types.h, 48  
 graphite\_start\_logging  
     Log.h, 31  
 graphite\_stop\_logging  
     Log.h, 32  
 GRLOG\_ALL  
     Log.h, 30  
 GRLOG\_CACHE  
     Log.h, 30  
 GRLOG\_FACE  
     Log.h, 30  
 GRLOG\_NONE  
     Log.h, 30  
 GRLOG\_OPCODE  
     Log.h, 30  
 GRLOG\_PASS  
     Log.h, 30  
 GRLOG\_SEGMENT  
     Log.h, 30  
 GrLogMask  
     Log.h, 30  
  
 has\_bidi\_pass  
     gr\_faceinfo, 10  
  
 justifies  
     gr\_faceinfo, 10  
  
 line\_ends  
     gr\_faceinfo, 10  
 Log.h, 30  
     gr\_start\_logging, 31  
     gr\_stop\_logging, 31  
     graphite\_start\_logging, 31  
     graphite\_stop\_logging, 32  
     GRLOG\_ALL, 30  
     GRLOG\_CACHE, 30  
     GRLOG\_FACE, 30  
     GRLOG\_NONE, 30  
     GRLOG\_OPCODE, 30  
     GRLOG\_PASS, 30  
     GRLOG\_SEGMENT, 30  
     GrLogMask, 30  
  
 release\_table  
     gr\_face\_ops, 7  
  
 Segment.h, 32  
     gr\_attrCode, 34  
     gr\_bidirtl, 36  
     gr\_break\_weight, 36  
     gr\_breakBeforeClip, 36  
     gr\_breakBeforeIntra, 36  
     gr\_breakBeforeLetter, 36  
     gr\_breakBeforeWhitespace, 36  
     gr\_breakBeforeWord, 36  
     gr\_breakClip, 36  
     gr\_breakIntra, 36  
     gr\_breakLetter, 36  
     gr\_breakNone, 36  
     gr\_breakWhitespace, 36  
     gr\_breakWord, 36  
     gr\_char\_info, 34  
     gr\_cinfo\_after, 37  
     gr\_cinfo\_base, 37  
     gr\_cinfo\_before, 38  
     gr\_cinfo\_break\_weight, 38  
     gr\_cinfo\_unicode\_char, 38  
     gr\_count\_unicode\_characters, 39  
     gr\_justCompleteLine, 37  
     gr\_justEndInline, 37  
     gr\_justFlags, 37  
     gr\_justStartInline, 37  
     gr\_make\_seg, 39  
     gr\_nobidi, 36  
     gr\_nomirror, 36  
     gr\_rtl, 36  
     gr\_seg\_advance\_X, 40  
     gr\_seg\_advance\_Y, 40  
     gr\_seg\_cinfo, 40  
     gr\_seg\_destroy, 40  
     gr\_seg\_first\_slot, 41  
     gr\_seg\_justify, 41  
     gr\_seg\_last\_slot, 42  
     gr\_seg\_n\_cinfo, 42  
     gr\_seg\_n\_slots, 42  
     gr\_segment, 34  
     gr\_slatAdvX, 35  
     gr\_slatAdvY, 35  
     gr\_slatAttGpt, 35  
     gr\_slatAttLevel, 35  
     gr\_slatAttTo, 35  
     gr\_slatAttWithX, 35  
     gr\_slatAttWithXOff, 35  
     gr\_slatAttWithY, 35  
     gr\_slatAttWithYOff, 35  
     gr\_slatAttX, 35  
     gr\_slatAttXOff, 35  
     gr\_slatAttY, 35  
     gr\_slatAttYOff, 35  
     gr\_slatBidiLevel, 35  
     gr\_slatBreak, 35  
     gr\_slatColExclGlyph, 35  
     gr\_slatColExclOffx, 35  
     gr\_slatColExclOffy, 36  
     gr\_slatColFlags, 35  
     gr\_slatColLimitblx, 35  
     gr\_slatColLimitbly, 35  
     gr\_slatColLimittrx, 35  
     gr\_slatColLimittry, 35

- gr\_slatColMargin, [35](#)
- gr\_slatColMarginWt, [35](#)
- gr\_slatColShiftx, [35](#)
- gr\_slatColShifty, [35](#)
- gr\_slatCompRef, [35](#)
- gr\_slatDir, [35](#)
- gr\_slatInsert, [35](#)
- gr\_slatJShrink, [35](#)
- gr\_slatJStep, [35](#)
- gr\_slatJStretch, [35](#)
- gr\_slatJWeight, [35](#)
- gr\_slatJWidth, [35](#)
- gr\_slatMax, [36](#)
- gr\_slatMeasureEol, [35](#)
- gr\_slatMeasureSol, [35](#)
- gr\_slatNoEffect, [36](#)
- gr\_slatPosX, [35](#)
- gr\_slatPosY, [35](#)
- gr\_slatSegSplit, [35](#)
- gr\_slatSeqAboveWt, [36](#)
- gr\_slatSeqAboveXoff, [36](#)
- gr\_slatSeqBelowWt, [36](#)
- gr\_slatSeqBelowXlim, [36](#)
- gr\_slatSeqClass, [36](#)
- gr\_slatSeqOrder, [36](#)
- gr\_slatSeqProxClass, [36](#)
- gr\_slatSeqValignHt, [36](#)
- gr\_slatSeqValignWt, [36](#)
- gr\_slatShiftX, [35](#)
- gr\_slatShiftY, [35](#)
- gr\_slatUserDefn, [35](#)
- gr\_slatUserDefnV1, [35](#)
- gr\_slatWithGpt, [35](#)
- gr\_slot, [34](#)
- gr\_slot\_advance\_X, [42](#)
- gr\_slot\_advance\_Y, [42](#)
- gr\_slot\_after, [43](#)
- gr\_slot\_attached\_to, [43](#)
- gr\_slot\_attr, [43](#)
- gr\_slot\_before, [43](#)
- gr\_slot\_can\_insert\_before, [43](#)
- gr\_slot\_first\_attachment, [44](#)
- gr\_slot\_gid, [44](#)
- gr\_slot\_index, [44](#)
- gr\_slot\_linebreak\_before, [44](#)
- gr\_slot\_next\_in\_segment, [44](#)
- gr\_slot\_next\_sibling\_attachment, [45](#)
- gr\_slot\_origin\_X, [45](#)
- gr\_slot\_origin\_Y, [45](#)
- gr\_slot\_original, [45](#)
- gr\_slot\_prev\_in\_segment, [45](#)
- size
  - gr\_face\_ops, [8](#)
  - gr\_font\_ops, [12](#)
- space\_contextuals
  - gr\_faceinfo, [11](#)
- Types.h, [46](#)
  - GR2\_API, [46](#)
  - GR2\_DEPRECATED\_API, [46](#)
  - gr\_byte, [47](#)
  - gr\_encform, [47](#)
  - gr\_int16, [47](#)
  - gr\_int32, [47](#)
  - gr\_int8, [47](#)
  - gr\_uint16, [47](#)
  - gr\_uint32, [47](#)
  - gr\_uint8, [47](#)
  - gr\_utf16, [48](#)
  - gr\_utf32, [48](#)
  - gr\_utf8, [48](#)
- upem
  - gr\_faceinfo, [11](#)