Graphiteng

Generated by Doxygen 1.9.1

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:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

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

gr_space_unknown	no information is known.
gr_space_none	the space character never occurs in any rules.
gr_space_left_only	the space character only occurs as the first element in a rule.
gr_space_right_only	the space character only occurs as the last element in a rule.
gr_space_either_only	the space character only occurs as the only element in a rule.
gr_space_both	the space character may occur as the first or last element of a rule.
gr_space_cross	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 contextuals somewhere

4.2.3.6 space_contextuals

```
enum gr_faceinfo::gr_space_contextuals space_contextuals
```

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

appFontHandle	is the unique information passed to gr_make_font_with_advance()
glyphid	is the glyph to retireve 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

appFaceHandle	is the unique information passed to gr_make_face()
name	is a 32bit tag to the table name.
len	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

appFaceHandle	is the unique information passed to gr_make_face()
pointer	to table memory returned by get_table.

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

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

pFace	The font face to get feature values from
langname	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 langauge 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

pFace	Font face to get information on.
featId	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

pFace	face to test within
usv	Unicode Scalar Value of character to test
script	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

pfeatureref	gr_feature_ref to the feature
feats	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

pfeatureref	gr_feature_ref of the feature
langId	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
utf	Encoding form for the string
length	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

pfeatureref	gr_feature_ref to the feature
val	value to set the feature to
pDest	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

pfeatureref	gr_feature_ref of the feature of interest
settingno	Index up to the return value of gr_fref_n_values() 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

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

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

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

appFaceHandle	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.
getTable	The function graphite calls to access font table data
segCacheMaxSize	How large the segment cache is.
faceOptions	Bitfield of values from enum gr_face_options

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

appFaceHandle	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.
face_ops	Pointer to face specific callback structure for table management. Must stay alive for the duration of the call only.
segCacheMaxSize	Unused.
faceOptions	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

filename	Full path and filename to font file
faceOptions	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

filename	Full path and filename to font file
segCacheMaxSize	Specifies how big to make the cache in segments.
faceOptions	Bitfield from enum gr_face_options 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

ppm	Resolution of the font in pixels per em
face	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

ppm	size of font in pixels per em
appFontHandle	font specific information that must stay alive as long as the font does
getAdvance	callback function reference that returns horizontal advance in pixels for a glyph.
face	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

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

str	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

tag	contains the tag to convert
str	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

face	the gr_face whose segments you want to log to the given file
log_path	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

face	the gr_face 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

logfile	FILE reference to output logging to
mask	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_slatBidiLevel = 56 , gr_slatColFlags , gr_slatColLimitblx , gr_slatColLimitbly ,
  gr_slatColLimittrx , gr_slatColLimittry , 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_bidirtl { 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 mush 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

gr_justCompleteLine	Indicates that this segment is a complete line.
gr_justStartInline	Indicates that the start of the slot list is not at the start of a line.
gr_justEndInline	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

p	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

p	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

p	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

p	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

p	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

enc	Specifies the type of data in the string: utf8, utf16, utf32
buffer_begin	The start of the string
buffer_end	Measure up to the first nul or when end is reached, whichever is earliest. This parameter may be NULL.
pError	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

font	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.
face	The face containing all the non-size dependent information.
script	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.
pFeats	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.
enc	Specifies what encoding form the string is in (utf8, utf16, utf32)
pStart	Start of the string
nChars	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.
dir	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 gr_bidirtl 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

p	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

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

p	Slot to give results for
face	gr_face of the glyphs. May be NULL if unhinted advances used
font	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 interpretted 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 interpretted 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 glyphid which is rendered at the position given by the slot. This glyphid 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 br 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 previos 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