UF_DRF_begin_line_fn_t (view source)

Defined in: uf_drf_types.h

Overview

Callbacks for UF_DRF_render_text. Default set is used for other functions.

Each return either UF_DRF_RENDER_OK or UF_DRF_RENDER_CANNOT_RENDER_CHAR (UF_DRF_draw_char_fn_t only) or UF_DRF_RENDER_CANNOT_RENDER_SYMBOL (UF_DRF_draw_user_symbol_fn_t only).

If a cannot render or not drawn status is returned for a user symbol or an individual character, the resultant object will be stroked.

Lines are output as polyline segments bracketed by calls to the begin and end line functions.

```
UF_DRF_render_text_status_t UF_DRF_begin_line_fn_t
(
    void * client
)
```

void *	**client**	Input	client data

UF_DRF_draw_arc_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_draw_arc_fn_t
(
    double center [ 3 ] ,
    double radius,
    double start_param,
    double end_param,
    void * client
)
```

double	**center [3]**	Input	center point in coord system of annotation
double	**radius**	Input	radius of arc
double	**start_param**	Input	start angle in radians
double	**end_param**	Input	end angle in radians
void *	**client**	Input	client data

UF_DRF_draw_char_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_draw_char_fn_t
(
    double origin [ 3 ] ,
    unsigned char character,
    void * client
)
```

double	**origin [3]**	Input	text_origin coord system of annotation
unsigned char	**character**	Input	character (possibly multi-byte)to be drawn
void *	**client**	Input	client data

UF_DRF_draw_standard_font_string_fn_t (view source)

Defined in: uf_drf_types.h

Overview

Function called to render a text string using a standard font for UF_DRF_render_table_t when the use_standard_font_string_fn UF_DRF_render_table_t flag is set.
This function should be used with the text mode set to UF_TEXT_ALL_UTF8.
Data returned by this function may be outside the user's locale mode.

```
UF_DRF_render_text_status_t UF_DRF_draw_standard_font_string_fn_t
(
    const char* textString,
    const double origin [ 3 ] ,
    int fontId,
    double textAngle,
    double fontSize,
    double gapSize,
    double aspectRatio,
    logical suppressKerning,
    logical vertical,
    logical bold,
    logical italic,
    void * client
)
```

const char*	**textString**	Input	Text string to display. If the text mode is UF_TEXT_LOCALE then the '#' character will be returned for each character that is outside the user's locale mode. Note that text control sequence symbols displayed using standard fonts may return characters that are defined outside the user's locale.
const double	**origin [3]**	Input	Origin of text baseline

int	**fontId**	Input	Standard font number
double	**textAngle**	Input	Text angle
double	**fontSize**	Input	Character height to use for standard font. This is the actual size used when using the FreeType Font library. Note that this differs from the NX character height.
double	**gapSize**	Input	Size of spacing between characters
double	**aspectRatio**	Input	Aspect ratio to use for display
logical	**suppressKerning**	Input	True to not use kerning when rendering
logical	**vertical**	Input	True to draw in vertical direction
logical	**bold**	Input	True if using bold style
logical	**italic**	Input	True if using italic style
void *	**client**	Input	Client data

UF_DRF_draw_to_position_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_draw_to_position_fn_t
(
    double in_point [ 3 ] ,
    double last_out_point [ 3 ] ,
    logical last_status,
    void * client,
    double out_point [ 3 ] ,
    logical * out_status
)
```

double	**in_point [3]**	Input	3d point in coord system of annotation
double	**last_out_point [3]**	Input	last output point from set_to_position or draw_to_position
logical	**last_status**	Input	last status from set_to_position or draw_to_position
void *	**client**	Input	client data
double	**out_point [3]**	Output	internally used
logical *	**out_status**	Output	used by draw_to_position

UF_DRF_draw_user_symbol_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_draw_user_symbol_fn_t
(
    char * symbol_name,
    double origin [ 3 ] ,
    void * symbol_params,
    void * client
)
```

char *	**symbol_name**	Input	name of user defined symbol to render
double	**origin [3]**	Input	
void *	**symbol_params**	Input	
void *	**client**	Input	client data

UF_DRF_end_line_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_end_line_fn_t
(
    void * client
)
```

void *	**client**	Input	client data

UF_DRF_fill_region_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_fill_region_fn_t
(
    int num_points,
    double * points,
    double fill_dist,
    double last_pt [ 3 ] ,
    logical * was_drawn,
    void * client
```

)

int	**num_points**	Input	number of points in region
double *	**points**	Input	array of 3d points
double	**fill_dist**	Input	fill distance
double	**last_pt [3]**	Output	last on screen point
logical *	**was_drawn**	Output	return TRUE if anything drawn
void *	**client**	Input	client data

UF_DRF_pop_orientation_fn_t (view source)

Defined in: uf_drf_types.h

UF_DRF_render_text_status_t UF_DRF_pop_orientation_fn_t
(
void * client
)

void *	**client**	Input	client data

UF_DRF_push_orientation_fn_t (view source)

Defined in: uf_drf_types.h

UF_DRF_render_text_status_t UF_DRF_push_orientation_fn_t
(
double matrix [9] ,
void * client
)

double	**matrix [9]**	Input	orientation matrix
void *	**client**	Input	client data

UF_DRF_set_cfw_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_set_cfw_fn_t
(
    UF_DRF_cfw_p_t cfw,
    void * client
)
```

UF_DRF_cfw_p_t	**cfw**	Input	new color, line font and line width
void *	**client**	Input	client data

UF_DRF_set_to_position_fn_t (view source)

Defined in: uf_drf_types.h

```
UF_DRF_render_text_status_t UF_DRF_set_to_position_fn_t
(
    double in_point [ 3 ] ,
    void * client,
    double out_point [ 3 ] ,
    logical * out_status
)
```

double	**in_point [3]**	Input	3d point in coord system of annotation
void *	**client**	Input	client data
double	**out_point [3]**	Output	internally used
logical *	**out_status**	Output	used by draw_to_position
