uc6476 (view source)

Defined in: uf_draw.h

Overview

(use UF_DRAW_set_display_state)

set drawing display state in the drafting application

Return code:
1 = OK
if not 1,
error_code = UF_DRAWING_STATE_OK
error_code = UF_DRAWING_INVALID_STATE_FLAG
error_code = UF_DRAWING_NOOP_WORK_MEM_VIEW
error_code = UF_DRAWING_STATE_NOT_SET
error_code = UF_DRAWING_NO_VIEW_AVAILABLE

Environment

Internal and External

Required License(s)

drafting

```
int uc6476
(
    int view_type
)
```

int	**view_type**	Input	type of view 1 = modeling view 2 = drawing view

---

uc6477 (view source)

Defined in: uf_draw.h

Overview

(use UF_DRAW_ask_display_state)

ask drawing display state in the drafting application

Return code:
1 = modeling view
2 = drawing view
if not 1 or 2, error_code

Environment

Internal and External

Required License(s)

drafting

```
int uc6477
(
    void
)
```

---

uc6478 (view source)

Defined in: uf_draw.h

Overview

(use UF_DRAW_create_drawing)

create drawing

Return code:
0 = OK
if not 0,
error_code = UF_DRAWING_DESIGN_IN_CONTEXT
error_code = UF_DRAWING_WORK_IN_MEMBER_VIEW
error_code = UF_DRAWING_INVALID_DRAWING_NAME
error_code = UF_DRAWING_DRAWING_ALREADY_EXISTS
error_code = UF_DRAWING_VIEW_ALREADY_EXISTS
error_code = UF_DRAWING_INVALID_UNITS
error_code = UF_DRAWING_INVALID_SIZE_CODE
error_code = UF_DRAWING_INVALID_DRAWING_SIZE

Environment

Internal and External

Required License(s)

drafting

```
int uc6478
(
    const char * drawing_name,
    const int drawing_units,
    const int size_code,
    const double * custom_size
)
```

const char *	**drawing_name**	Input	drawing name " " = current drawing (UF_OBJ_NAME_NCHARS characters max)
const int	**drawing_units**	Input	drawing units 1 = inches 2 = millimeters
const int	**size_code**	Input	drawing size code 0 = custom size 1 = A/A0 2 = B/A1 3 = C/A2 4 = D/A3 5 = E/A4 6 = F/-- 7 = H/-- 8 = J/--

const double *	**custom_size**	Input	custom drawing size [0] height [1] width

uc6479 (view source)

Defined in: uf_draw.h

Overview

(use UF_DRAW_ask_drawing_info)

ask drawing size

Return code:
0 = OK
if not 0 = error code
21 = DRAWING DOES NOT EXIST
22 = INVALID DRAWING NAME
29 = NO CURRENT DRAWING

Environment

Internal and External

Required License(s)

gateway

```
int uc6479
(
    const char * drawing_name,
    int * drawing_units,
    int * size_code,
    double * custom_size
)
```

const char *	**drawing_name**	Input	drawing name " " = current drawing (UF_OBJ_NAME_NCHARS characters max)
int *	**drawing_units**	Output	drawing units 1 = inches 2 = millimeters
int *	**size_code**	Output	drawing size code 0 = custom size 1 = A/A0 2 = B/A1 3 = C/A2 4 = D/A3 5 = E/A4
double *	**custom_size**	Output	custom drawing size [0] height [1] width

uc6480 (view source)

Defined in: uf_draw.h

Overview

(use UF_DRAW_set_drawing_info)

set drawing size

Return code:
0 = OK
if not 0 = error code
21 = DRAWING DOES NOT EXIST
22 = INVALID DRAWING NAME
29 = NO CURRENT DRAWING
31 = INVALID UNITS
32 = INVALID SIZE CODE
33 = INVALID DRAWING SIZE
36 = CANNOT CHANGE DRAWING SIZE WHILE WORK IN MEMBER VIEW

Environment

Internal and External

Required License(s)

gateway

```
int uc6480
(
    const char * drawing_name,
    const int drawing_units,
    const int size_code,
    const double * custom_size
)
```

const char *	**drawing_name**	Input	drawing name " " = current drawing (UF_OBJ_NAME_NCHARS characters max)
const int	**drawing_units**	Input	drawing units 1 = inches 2 = millimeters
const int	**size_code**	Input	drawing size code 0 = custom size 1 = A/A0 2 = B/A1 3 = C/A2 4 = D/A3 5 = E/A4 6 = F/-- 7 = H/-- 8 = J/--
const double *	**custom_size**	Input	custom drawing size [0] height [1] width

uc6481 (view source)

Defined in: uf_draw.h

Overview

Use UF_DRAW_import_view
uc6481 add view to drawing

Required License(s)

gateway

```
int uc6481
(
    const char * dwg_name,
    const char * view_name,
    const double * reference_pt,
    const int view_status
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input
const double *	**reference_pt**	Input
const int	**view_status**	Input

---

uc6482 (view source)

Defined in: uf_draw.h

Overview

Use UF_VIEW_delete
uc6482 remove view from drawing

Required License(s)

gateway

```
int uc6482
(
    const char * dwg_name,
    const char * view_name
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input

---

uc6483 (view source)

Defined in: uf_draw.h

Overview

uc6483 read view reference point on drawing
Use UF_DRAW_ask_drawing_ref_pt
This is actually the drawing reference point

Required License(s)

gateway

```
int uc6483
(
    const char * dwg_name,
    const char * view_name,
    double * reference_pt
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input
double *	**reference_pt**	Output

---

uc6484 (view source)

Defined in: uf_draw.h

Overview

uc6484 set view reference point on drawing
Use UF_DRAW_set_drawing_ref_pt

Required License(s)

gateway

```
int uc6484
(
    const char * dwg_name,
    const char * view_name,
    const double * reference_pt
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input
const double *	**reference_pt**	Input

---

uc6485 (view source)

Defined in: uf_draw.h

Overview

uc6485 read view borders on current drawing
Use UF_DRAW_ask_view_borders

Required License(s)

gateway

```
int uc6485
(
    const char * cp1,
    double * rr2
)
```

const char *	**cp1**	Input
double *	**rr2**	

---

uc6486 (view source)

Defined in: uf_draw.h

Overview

uc6486 set view borders on current drawing

Required License(s)

gateway

```
int uc6486
(
    const char * cp1,
    double * rp2
)
```

const char *	**cp1**	Input
double *	**rp2**	

---

uc6488 (view source)

Defined in: uf_draw.h

Overview

uc6488 read view status in drawing
Use UF_DRAW_ask_view_status

Required License(s)

gateway

```
int uc6488
(
    const char * dwg_name,
    const char * view_name,
    int * view_status
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input
int *	**view_status**	Output

---

uc6489 (view source)

Defined in: uf_draw.h

Overview

uc6489 set view status in drawing
Use UF_DRAW_set_view_status

Required License(s)

gateway

```
int uc6489
(
    const char * dwg_name,
    const char * view_name,
    const int view_status
)
```

const char *	**dwg_name**	Input
const char *	**view_name**	Input
const int	**view_status**	Input

---

uc6492 (view source)

Defined in: uf_draw.h

Overview

Use UF_DRAW_ask_current_drawing and UF_OBJ_ask_name.
uc6492 read current drawing name

Required License(s)

gateway

int uc6492
(
    char cr1 [UF_OBJ_NAME_BUFSIZE]
)

char	**cr1 [UF_OBJ_NAME_BUFSIZE]**	Output	This parameter must be a character buffer large enough to hold the returned drawing name (UF_OBJ_NAME_LEN + 1).

---

uc6494 (view source)

Defined in: uf_draw.h

Overview

uc6494 retrieve drawing
Use UF_DRAW_open_drawing

Required License(s)

gateway

int uc6494
(
    const char * dwg_name
)

const char *	**dwg_name**	Input

---

uc6495 (view source)

Defined in: uf_draw.h

Overview

uc6495 delete drawing
Use UF_DRAW_delete_drawing

Required License(s)

gateway

int uc6495
(
    const char * dwg_name
)

const char *	**dwg_name**	Input

---

uc6496 (view source)

Defined in: uf_draw.h

Overview

uc6496 rename drawing
Use UF_DRAW_rename_drawing

Required License(s)

gateway

```
int uc6496
(
    const char * old_dwg_name,
    const char * new_dwg_name
)
```

const char *	**old_dwg_name**	Input
const char *	**new_dwg_name**	Input

---

uc6497 (view source)

Defined in: uf_draw.h

Overview

uc6497 cycle drawings in a part

Return code of 0 = OK
21 = Drawing does not exist
22 = Invalid Drawing Name
all other numbers - error

Environment

Internal and External

Required License(s)

gateway

```
int uc6497
(
    char ca1 [ UF_OBJ_NAME_BUFSIZE ]
)
```

char	**ca1 [UF_OBJ_NAME_BUFSIZE]**	Input / Output	Drawing name Pass in an empty string to start the cycle. By calling this function in a loop and passing

			the last drawing name - the next drawing will be returned. An empty string is returned when all drawings have been cycled.

uc6498 (view source)

Defined in: uf_draw.h

Overview
uc6498 read number of views in a drawing

Required License(s)
gateway

```
int uc6498
(
    const char * cp1,
    int * ir2
)
```

const char *	**cp1**	Input
int *	**ir2**	

uc6499 (view source)

Defined in: uf_draw.h

Overview
uc6499 cycle views in drawing

Required License(s)
gateway

```
int uc6499
(
    const char * cp1,
    char ca2 [ UF_OBJ_NAME_BUFSIZE ]
)
```

const char *	**cp1**	
char	**ca2 [UF_OBJ_NAME_BUFSIZE]**	Input / Output

UF_DRAW_add_auxiliary_view (view source)

Defined in: uf_draw.h

Overview
Adds an auxiliary view to the current drawing.

Environment
Internal and External

See Also
See the example

History
This function was originally released in V15.0

Required License(s)
drafting

```
int UF_DRAW_add_auxiliary_view
(
    const tag_t drawing_tag,
    const tag_t parent_view_tag,
    const tag_t hinge_line_tag,
    double dwg_reference_point [ 2 ] ,
    tag_t * aux_view_tag
)
```

const tag_t	**drawing_tag**	Input	The drawing tag (must be current drawing).
const tag_t	**parent_view_tag**	Input	tag of parent view.
const tag_t	**hinge_line_tag**	Input	tag of smart hinge line
double	**dwg_reference_point [2]**	Input	Drawing reference point (drawing coordinates)
tag_t *	**aux_view_tag**	Output	tag of auxiliary view

UF_DRAW_add_circ_detail_view (view source)

Defined in: uf_draw.h

Overview
Adds an associative circular detail view to the current drawing.

Environment
Internal and External

See Also
See example

History
Original release was in V15.0.

Required License(s)

drafting

```
int UF_DRAW_add_circ_detail_view
(
    const tag_t drawing_tag,
    const tag_t parent_view_tag,
    const tag_t center_pt_tag,
    const tag_t circle_pt_tag,
    const double view_scale,
    double dwg_reference_point [ 2 ] ,
    tag_t * detail_view_tag
)
```

const tag_t	**drawing_tag**	Input	The drawing tag (must be current drawing).
const tag_t	**parent_view_tag**	Input	tag of parent view.
const tag_t	**center_pt_tag**	Input	tag of smart center point for the detail circle.
const tag_t	**circle_pt_tag**	Input	tag of smart point on the detail circle.
const double	**view_scale**	Input	The desired view scale
double	**dwg_reference_point [2]**	Input	Drawing reference point (drawing coordinates)
tag_t *	**detail_view_tag**	Output	tag of detail view

UF_DRAW_add_detail_view (view source)

Defined in: uf_draw.h

Overview

This function adds a detailed view to the current drawing.
There is currently a restriction requiring the input drawing
tag to be the tag of the current drawing. We intend to relax this
restriction in the future. As a result, we are requiring the input tag to
ensure that future code changes will not be required by NX Open
API developers.

Environment

Internal and External

See Also

See the example

History

Original release was in V13.0.

Required License(s)

drafting

```
int UF_DRAW_add_detail_view
(
    const tag_t drawing_tag,
    const tag_t parent_view_tag,
    double xy1 [ 2 ] ,
    double xy2 [ 2 ] ,
    const double view_scale,
    double dwg_reference_point [ 2 ] ,
    tag_t * detail_view_tag
)
```

const tag_t	**drawing_tag**	Input	Drawing Tag, must be the current drawing.
const tag_t	**parent_view_tag**	Input	tag of parent view.
double	**xy1 [2]**	Input	Left lower corner of area to create detailed view, in drawing coordinates (x1,y1).
double	**xy2 [2]**	Input	Right upper corner of area to create detailed view, in drawing coordinates (x2, y2). This corner should be the diagonal corner to corner1.
const double	**view_scale**	Input	desired view scale of detailed view.
double	**dwg_reference_point [2]**	Input	Drawing Reference point (in drawing coordinates x,y)
tag_t *	**detail_view_tag**	Output	tag of new detail view.

UF_DRAW_add_orthographic_view (view source)

Defined in: uf_draw.h

Overview
Adds an orthographic view to the current drawing.

Environment
Internal and External

See Also
UF_DRAW_proj_dir_t
See the example

History
This function was originally released in V15.0

Required License(s)
drafting

```
int UF_DRAW_add_orthographic_view
(
    const tag_t drawing_tag,
    const tag_t parent_view_tag,
    const UF_DRAW_proj_dir_t projection_direction,
    double dwg_reference_point [ 2 ] ,
    tag_t * ortho_view_tag
```

)

const tag_t	**drawing_tag**	Input	The drawing tag (must be current drawing).
const tag_t	**parent_view_tag**	Input	tag of parent view.
const UF_DRAW_proj_dir_t	**projection_direction**	Input	direction of projection if none specified, the projection direction is determined by the location of the drawing reference point relative to the center of the parent view
double	**dwg_reference_point [2]**	Input	Drawing reference point (drawing coordinates)
tag_t *	**ortho_view_tag**	Output	tag of orthographic view

UF_DRAW_add_sxline_sxseg (view source)

Defined in: uf_draw.h

Overview

Adds a segment to a section line, then updates all of its associated section views located on the current drawing. Associated section views not on the current drawing are marked out of date. To perform this edit on the section line without an update, use the suppress view update feature provided in UF_DRF_set_suppress_view_update.

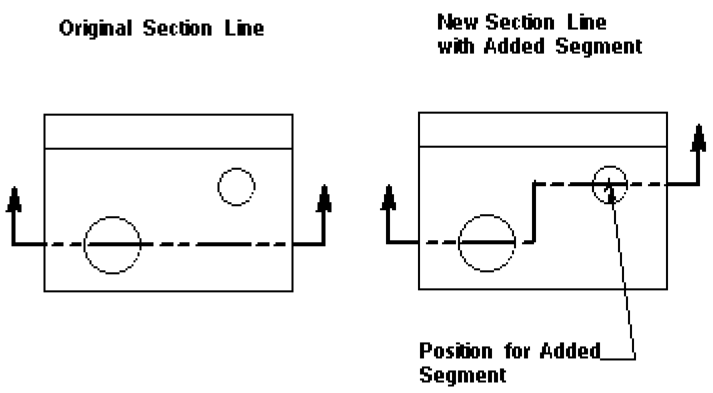
Original Section Line
New Section Line with Added Segment
Position for Added Segment

Figure. Adding a new segment to a section line.

Environment

Internal and External

See Also

UF_DRF_set_suppress_view_update

Required License(s)

drafting

int UF_DRAW_add_sxline_sxseg
(

```
    tag_t sxline_tag,
    UF_DRAW_sxseg_type_t sxseg_type,
    UF_DRAW_sxline_leg_t sxline_leg,
    UF_DRF_object_p_t object,
    tag_t * sxseg_tag
)
```

tag_t	**sxline_tag**	Input	Tag of section line
UF_DRAW_sxseg_type_t	**sxseg_type**	Input	Segment_type: UF_DRAW_sxseg_cut = cut segment (only type currently supported for add)
UF_DRAW_sxline_leg_t	**sxline_leg**	Input	If a section line is a two-legged revolved type , UF_DRAW_sxline_leg1= add segment to leg 1 UF_DRAW_sxline_leg2= add segment to leg 2
UF_DRF_object_p_t	**object**	Input	Object to associate new segment to
tag_t *	**sxseg_tag**	Output	Tag of added section segment

UF_DRAW_add_sxseg (view source)

Defined in: uf_draw.h

Overview
Adds a segment to a section line, then updates all of its associated section views located on the current drawing. Associated section views not on the current drawing are marked out of date. To perform this edit on the section line without an update, use the suppress view update feature provided in UF_DRF_set_suppress_view_update.

Environment
Internal and External

See Also
UF_DRAW_sxline_sxsegs_t
UF_DRF_set_suppress_view_update
See the example

History
Original release was in V14.0.

Required License(s)
drafting

```
int UF_DRAW_add_sxseg
(
    tag_t sxline_tag,
    UF_DRAW_sxline_sxsegs_p_t sxseg_data,
    tag_t * sxseg_tag
)
```

tag_t	**sxline_tag**	Input	Tag of section line

UF_DRAW_sxline_sxsegs_p_t	**sxseg_data**	Input	Segment data contains: segment type: UF_DRAW_sxseg_cut, linear segment angle, and object to associate to
tag_t *	**sxseg_tag**	Output	Tag of newly created section line segment.

UF_DRAW_ask_auto_update (view source)

Defined in: uf_draw.h

Overview
Queries the current value of the Automatic Update preference setting.

Environment
Internal and External

See Also
UF_DRAW_set_auto_update
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_auto_update
(
    tag_t view_tag,
    logical * auto_update
)
```

tag_t	**view_tag**	Input	Tag of section view
logical *	**auto_update**	Output	TRUE = Automatically update the view FALSE = Do not automatically update the view

---

UF_DRAW_ask_body_sils_in_view (view source)

Defined in: uf_draw.h

Overview
Outputs an array of silhouettes of the input body that reside in the input drawing member view.

Environment
Internal and External

See Also
See the example

History
Original Release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_ask_body_sils_in_view
(
    tag_t body_tag,
    tag_t view_tag,
    int * num_silhouettes,
    tag_p_t * silhouette_tags
)
```

tag_t	**body_tag**	Input	Tag of a body
tag_t	**view_tag**	Input	Tag of a drawing member view
int *	**num_silhouettes**	Output	Number of silhouettes
tag_p_t *	**silhouette_tags**	Output to UF_*free*	If num_silhouettes > 0, this is an array of silhouette tags. Use UF_free to deallocate this memory.

UF_DRAW_ask_border_color (view source)

Defined in: uf_draw.h

Overview
UF_DRAW_ask_border_color

DESCRIPTION -
Output the color of view borders

PARAMETERS -
border_color <O> Color of view borders
Returns -
0 = OK
if not 0 = error code

Required License(s)
gateway

```
int UF_DRAW_ask_border_color
(
    int * border_color
)
```

int *	**border_color**	Output

UF_DRAW_ask_border_display (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_ask_border_display

DESCRIPTION -
Ask the view border display status

PARAMETERS -
border_display <O> True if borders are displayed
Returns -
0 = OK
if not 0 = error code

Required License(s)

gateway

```
int UF_DRAW_ask_border_display
(
    logical * border_display
)
```

logical *	**border_display**	Output

UF_DRAW_ask_bound_by_objects (view source)

Defined in: uf_draw.h

Overview

Retrieves the view's bounded objects.

Environment

Internal and External

See Also

UF_DRAW_define_bound_by_objects
See the example

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_DRAW_ask_bound_by_objects
(
    const tag_t view_tag,
    int* num_objects,
    tag_t* * bounded_objects
)
```

const tag_t	**view_tag**	Input	Tag of view whose model reference point is to be obtained.
int*	**num_objects**	Output	Pointer to int that represents the number of object tags returned.
tag_t* *	**bounded_objects**	Output to UF_*free*	Pointer to tag array of objects used to calculate the bounding box for the view boundary. Use UF_free() to free this array when done.

UF_DRAW_ask_boundary_curves (view source)

Defined in: uf_draw.h

Overview

Retrieves the view boundary curves, the smart defining points of those curves, and the tolerance. These items are used to create an arbitrary clipping bound for a view.

Environment

Internal and External

See Also

UF_DRAW_define_view_boundary
See the example

Required License(s)

gateway

```
int UF_DRAW_ask_boundary_curves
(
    tag_t view_tag,
    double * tolerance,
    int * num_curves,
    UF_DRAW_view_boundary_t * * boundary_curves
)
```

tag_t	**view_tag**	Input	Tag of view whose view boundary curves are to be obtained.
double *	**tolerance**	Output	The chain tolerance used to create the arbitrary clipping bounds for this view.
int *	**num_curves**	Output	Count of structures. = 0 if view does not have an arbitrary boundary defined.
UF_DRAW_view_boundary_t * *	**boundary_curves**	Output to UF_*free*	Pointer to view boundary structure. Use UF_DRAW_free_boundary to free the pointer. If a boundary curve has been deleted, its curve_tag will be NULL.

UF_DRAW_ask_boundary_type (view source)

Defined in: uf_draw.h

Overview
Retrieves the view's boundary type.

Environment
Internal and External

See Also
See the example

History
Original Release was in V13.0.

Required License(s)
gateway

```
int UF_DRAW_ask_boundary_type
(
    const tag_t view_tag,
    UF_DRAW_boundary_type_t * boundary_type
)
```

const tag_t	**view_tag**	Input	Tag of view whose model reference point is to be obtained.
UF_DRAW_boundary_type_t *	**boundary_type**	Output	Boundary type of member view: UF_DRAW_BREAK_DETAIL_TYPE, UF_DRAW_MANUAL_RECTANGLE_TYPE, UF_DRAW_AUTOMATIC_RECTANGLE_TYPE, UF_DRAW_BOUND_BY_OBJECTS_TYPE

UF_DRAW_ask_break_region_data (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the break region data for a given break region.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_broken_view.c to review an example of using
this function.

Environment

Internal and External

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_break_region_data
(
    tag_t region,
    UF_DRAW_break_region_data_p_t break_region_data
)
```

tag_t	**region**	Input	Tag of the break region.
UF_DRAW_break_region_data_p_t	**break_region_data**	Output	Break region data.

UF_DRAW_ask_break_regions (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the array of break regions for a given view.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_broken_view.c to review an example of using
this function.

Environment
Internal and External

History
New for V17.0

Required License(s)
gateway

```
int UF_DRAW_ask_break_regions
(
    tag_t view_tag,
    int * num_regions,
    tag_p_t * break_regions
)
```

tag_t	**view_tag**	Input	View to be queried
int *	**num_regions**	Output	Number of break regions

tag_p_t *	**break_regions**	Output to UF_*free*	Tags of the break regions; may be NULL

UF_DRAW_ask_breakout_data (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the breakout data from a given breakout section.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message string associated with the error code.

Please reference ufd_drw_cre_breakout.c to review an example of using this function.

Environment
Internal and External

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_breakout_data
(
    tag_t breakline,
    tag_p_t view_tag,
    UF_DRAW_breakout_data_p_t breakout_data
)
```

tag_t	**breakline**	Input	Tag of the breakout section.
tag_p_t	**view_tag**	Output	View which contains the given breakout section.
UF_DRAW_breakout_data_p_t	**breakout_data**	Output	Breakout section data.

UF_DRAW_ask_comp_section_in_view (view source)

Defined in: uf_draw.h

Overview
This routine asks whether or not the component of a given section view is defined to be sectioned or non-sectioned.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_comp_section_in_view.c to review an example
of using this function.

Environment
Internal and External

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_comp_section_in_view
(
    const tag_t component,
    const tag_t sx_view,
    UF_DRAW_comp_section_in_view_t * sx_property
)
```

const tag_t	**component**	Input	Tag of the component in the section view.
const tag_t	**sx_view**	Input	Tag of the section view.
UF_DRAW_comp_section_in_view_t *	**sx_property**	Output	Sectioning property UF_DRAW_NON_SECTIONED UF_DRAW_SECTIONED UF_DRAW_NOT_VIEW_SPECIFIED

UF_DRAW_ask_current_drawing (view source)

Defined in: uf_draw.h

Overview
n
Returns the tag of the current drawing.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DRAW_ask_current_drawing
(
    tag_t * drawing_tag
)
```

tag_t *	**drawing_tag**	Output	Tag of current drawing

UF_DRAW_ask_curve_group_members (view source)

Defined in: uf_draw.h

Overview
This routine gets members and count of members of a drafting curve group.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Environment
Internal and External

History
New for V19.0

Required License(s)
gateway

```
int UF_DRAW_ask_curve_group_members
(
    tag_t curve_group,
    tag_t* * curves,
    int * curve_count
)
```

tag_t	**curve_group**	Input	Tag of drafting curve group
tag_t* *	**curves**	Output to UF_*free*	curve_count Tag array of all curves in the drafting curve group and must be freed by UF_free()
int *	**curve_count**	Output	Number of curves in the drafting curve group

---

UF_DRAW_ask_curve_of_sxedge (view source)

Defined in: uf_draw.h

Overview
Given the tag of a section edge, outputs the tag of its curve. Note that section edges that result from a section line's bend segment do not have an associated curve tag. If a section edge from a bend segment is input to this function, curve_tag is returned as a pointer to a null tag.

Return

Return code:
0 = No error
UF_DRAW_no_sxedge_curve = warning
not 0 = Error code

Environment
Internal and External

See Also
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_curve_of_sxedge
(
    tag_t sxseg_tag,
    tag_t* curve_tag
)
```

tag_t	**sxseg_tag**	Input	section edge tag
tag_t*	**curve_tag**	Output	section edge curve tag

UF_DRAW_ask_display_state (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the drawing display state in the drafting application. The view_type parameter does not indicate what type of view is being displayed, it indicates what will be displayed when you are in the Drafting application.
To determine if a drawing is currently displayed in any application, use UF_DRAW_ask_current_drawing.
To determine the type of the work view, use UF_VIEW_ask_work_view then UF_VIEW_ask_type.

Environment
Internal and External

See Also
See the example program ufd_drw_ask_display_state.c

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_display_state
(
    int * view_type
)
```

int *	**view_type**	Output	Flag Setting: 1 = Modeling View 2 = Drawing View

UF_DRAW_ask_displayed_objects (view source)

Defined in: uf_draw.h

Overview
This routine outputs the diusplayed objects which were defined by the user.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Environment
Internal and External

History
New for V18.0

Required License(s)
gateway

```
int UF_DRAW_ask_displayed_objects
(
    const tag_t view,
    int * num_objects,
    tag_p_t * objects
)
```

const tag_t	**view**	Input	Tag of the view
int *	**num_objects**	Output	Number of objects
tag_p_t *	**objects**	Output to UF_*free*	num_objects An array of objects and must be freed by UF_free()

UF_DRAW_ask_dmv_rotation_plane (view source)

Defined in: uf_draw.h

Overview
This routine outputs the plane tag which is associated with the view
orientation.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Environment

Internal and External

History

New for V18.0

Required License(s)

gateway

```
int UF_DRAW_ask_dmv_rotation_plane
(
    const tag_t view,
    tag_p_t plane
)
```

const tag_t	**view**	Input	Tag of the view
tag_p_t	**plane**	Output	The associated plane

UF_DRAW_ask_drafting_curve_parents (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_ask_drafting_curve_parents

DESCRIPTION -
This routine returns the parents and count of parents for input drafting curve.

Returns -
0 = OK
if not 0 = error code

History

Initially released in NX 9.0.2-MP1

Required License(s)

gateway

```
int UF_DRAW_ask_drafting_curve_parents
(
    tag_t input_curve_tag,
    int * parents_count,
    tag_t * * parents
)
```

tag_t	**input_curve_tag**	Input	Tag of the input drafting curve whose parents are to queried

int *	**parents_count**	Output	Number of parents
tag_t * *	**parents**	Output to UF_*free*	Array of parents for input drafting curve and must be freed by UF_free()

UF_DRAW_ask_drafting_curve_type (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_ask_drafting_curve_type

DESCRIPTION -
This routine returns the type of input drafting curve.

Returns -
0 = OK
if not 0 = error code

History

Initially released in NX 9.0.2-MP1

Required License(s)

gateway

```
int UF_DRAW_ask_drafting_curve_type
(
    tag_t input_curve_tag,
    UF_DRAW_drafting_curve_type_t* curve_type
)
```

tag_t	**input_curve_tag**	Input	Tag of the input drafting curve whose parents are to queried
UF_DRAW_drafting_curve_type_t*	**curve_type**	Output	drafting curve type

---

UF_DRAW_ask_drawing_info (view source)

Defined in: uf_draw.h

Overview

This function retrieves the information about a drawing, including the size, scale, units, and projection angle.

Environment

Internal and External

See Also

UF_DRAW_set_drawing_info
See the example

History
Original Release was in V13.0.

Required License(s)
gateway

```
int UF_DRAW_ask_drawing_info
(
    tag_t drawing_tag,
    UF_DRAW_info_t * drawing_info
)
```

tag_t	**drawing_tag**	Input	Drawing Tag - if a NULL_TAG is passed in for the drawing_tag, the current drawing will be used.
UF_DRAW_info_t *	**drawing_info**	Output	Pointer to Drawing Info Structure

---

UF_DRAW_ask_drawing_of_view (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the tag of the drawing which contains the given member view or drawing sheet view.

Environment
Internal and External

See Also
See the example program ufd_drf_edit_dim_assoc.c

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_drawing_of_view
(
    const tag_t member_view,
    tag_t * drawing
)
```

const tag_t	**member_view**	Input	Tag of view whose drawing is unknown. may also be a drawing sheet view
tag_t *	**drawing**	Output	Tag of the drawing containing member_view. NULL_TAG if not found.

UF_DRAW_ask_drawing_ref_pt (view source)

Defined in: uf_draw.h

Overview

This routine reads the drawing reference point for a view on the drawing.
This is the point which controls where the view is on the drawing sheet.

Environment

Internal and External

See Also

See the example program ufd_drw_cre_simple_sxvw.c

History

New for V16.0

Required License(s)

gateway

```
int UF_DRAW_ask_drawing_ref_pt
(
    const tag_t view_tag,
    double reference_pt [ 2 ]
)
```

const tag_t	**view_tag**	Input	Tag of the view. Must be a member view. If NULL_TAG, the work view is used. It will cause an error if the work view is not a member view.
double	**reference_pt [2]**	Output	Reference point (Drawing Coordinates). [0] - X-coordinate [1] - Y-coordinate

---

UF_DRAW_ask_drawings (view source)

Defined in: uf_draw.h

Overview

This routine returns an array of all of the drawings in
the current work part.

Please reference ufd_drw_object_out_of_date.c to review an example of
using this function.

See Also

See the example program ufd_drw_object_out_of_date.c

Environment

Internal and External

History

New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_drawings
(
    int * num_drawings,
    tag_p_t * drawing_tags
)
```

int *	**num_drawings**	Output	Number of drawings.
tag_p_t *	**drawing_tags**	Output to UF_*free*	Array of drawing tags. Use UF_free to free this memory.

UF_DRAW_ask_face_of_sil (view source)

Defined in: uf_draw.h

Overview
Outputs the associated face of a given silhouette that resides in a drawing member view. The silhouette must be a line, circle, conic or spline type object.

Environment
Internal and External

See Also
See the example

History
Original Release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_ask_face_of_sil
(
    tag_t silhouette_tag,
    tag_t * face_tag
)
```

tag_t	**silhouette_tag**	Input	Tag of silhouette in a drawing member view Must be a line, circle, conic, or spline type.
tag_t *	**face_tag**	Output	Tag of face that the silhouette is associated to

UF_DRAW_ask_face_sils_in_view (view source)

Defined in: uf_draw.h

Overview
Outputs an array of silhouettes of the input face that reside in the input drawing member view.

Environment
Internal and External

See Also
Please see the example

History
Original Release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_ask_face_sils_in_view
(
    tag_t face_tag,
    tag_t view_tag,
    int * num_silhouettes,
    tag_p_t * sil_tags
)
```

tag_t	**face_tag**	Input	Tag of a face
tag_t	**view_tag**	Input	Tag of a drawing member view
int *	**num_silhouettes**	Output	Number of silhouettes
tag_p_t *	**sil_tags**	Output to UF_*free*	If num_silhouettes > 0, this is an array of silhouette tags. Use UF_free to deallocate memory when done.

UF_DRAW_ask_folded_sxline (view source)

Defined in: uf_draw.h

Overview
Retrieves folded section line information, given the tag of the section line.
NOTE: If a section line has an invalid section line status it does not independently cause the return code to be set to a non-zero value.

Environment
Internal and External

See Also
UF_DRAW_sxline_status_t

History

Original release was in NX903.

Required License(s)

gateway

```
int UF_DRAW_ask_folded_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of unfolded section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Arrow direction vector (unitized) relative to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
int *	**num_sxviews**	Output	Number of section views associated to section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section views. Use UF_free to free memory.
int *	**num_sxsegs**	Output	Number of section line segments
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line's segment tags. Use UF_free to free memory.
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

UF_DRAW_ask_group_of_curve (view source)

Defined in: uf_draw.h

Overview

Given the tag of a curve, outputs the tag of the associated drawing member view curve group, the group's type, and subtype.

Environment

Internal and External

See Also

See the example

Required License(s)

gateway

int UF_DRAW_ask_group_of_curve
(
    tag_t curve_tag,
    tag_t * group_tag,
    int * group_type,
    int * group_subtype
)

tag_t	**curve_tag**	Input	Tag of curve
tag_t *	**group_tag**	Output	Tag of drawing member view curve group. The group_tag is a NULL_TAG if curve is not a drafting edge.
int *	**group_type**	Output	Type of drawing member view curve group, either: UF_solid_silhouette_type, UF_solid_section_type or UF_curve_group_type
int *	**group_subtype**	Output	Subtype of drawing member view curve group: If group is UF_solid_silhouette_type, either UF_solid_silhouette_sl_subtype, UF_solid_silhouette_uvhatch_subtype, or UF_vicurve_subtype Else if group is UF_curve_group_type, either UF_dropped_edge_group_subtype or UF_simplified_group_subtype Else = 0

UF_DRAW_ask_half_sxline (view source)

Defined in: uf_draw.h

Overview

Retrieves half section line information, given the tag of the section line. If a section line has an invalid section line status it does not independently cause the return code to be set to a non-zero value.

Environment

Internal and External

See Also

UF_DRAW_sxline_status_t
See UF_DRAW_create_half_sxview
for more information about half section lines and views.

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_half_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of half section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Arrow direction vector (unitized) relative
			to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
int *	**num_sxviews**	Output	Number of section views associated to the input section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of Section line's section views Use UF_free to free memory
int *	**num_sxsegs**	Output	Number of section line segments
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line segment tags Use UF_free to free memory
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

UF_DRAW_ask_num_drawings (view source)

Defined in: uf_draw.h

Overview

This routine returns the number of drawings in the current work part.

Environment
Internal and External

See Also
See the example program ufd_drw_ask_num_drawings.c

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_num_drawings
(
    int * num_drawings
)
```

int *	**num_drawings**	Output	Number of drawings.

UF_DRAW_ask_num_views (view source)

Defined in: uf_draw.h

Overview
This routine returns the number of views in the given drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_ask_num_views.c

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_num_views
(
    const tag_t drawing_tag,
    int * num_views
)
```

const tag_t	**drawing_tag**	Input	Tag of the drawing. If NULL_TAG, use current drawing.
int *	**num_views**	Output	Number of views in the drawing.

UF_DRAW_ask_pictorial_sxline (view source)

Defined in: uf_draw.h

Overview

This routine retrieves information from a pictorial section line, given the tag of the section line. A pictorial section line cannot be revolved or unfolded.

Return

0 = OK
if not 0 = Failure error code
Use UF_get_fail_message to obtain the message string associated with the error code.

Please reference ufd_drw_cre_dmv_sxvw.c to review an example of using this function.

See Also

UF_DRAW_sxline_status_t

Environment

Internal and External

History

New for V16.0

Required License(s)

gateway

```
int UF_DRAW_ask_pictorial_sxline
(
    tag_t sxline_tag,
    UF_DRAW_sxline_type_t * sxline_type,
    double cut_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * parent_view_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    logical * pictorial_sxview,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of the section line.
UF_DRAW_sxline_type_t *	**sxline_type**	Output	Type of section line.

double	**cut_dir [3]**	Output	Original user-specified cut direction:
			cut_dir[0] = x value
			cut_dir[1] = y value
			cut_dir[2] = z value

double	**arrow_dir [3]**	Output	Original user-specified arrow direction:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**parent_view_tag**	Output	Parent view tag.
int *	**num_sxviews**	Output	Number of section views associated to the input section line.
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section views. Use UF_free to free this.
int *	**num_sxsegs**	Output	Number of section line segments.
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line segment tags. Use UF_free to free this.
logical *	**pictorial_sxview**	Output	Whether or not the section view is pictorial.
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

UF_DRAW_ask_render_set_objects (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the objects (solids or component sets) the given render set reference.

Environment
Internal and External

See Also
See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
gateway

```
int UF_DRAW_ask_render_set_objects
(
    tag_t render_set,
    int * number_objects,
    tag_p_t * objects
)
```

tag_t	**render_set**	Input	Tag of render set.
int *	**number_objects**	Output	Number of objects referenced by the render set.
tag_p_t *	**objects**	Output to UF_*free*	Array of solids or component sets to be referenced by the render set. Use UF_free() to

free the memory pointed to by this variable.

UF_DRAW_ask_render_set_parms (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the display parameters for a given render set.

Environment
Internal and External

See Also
See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
gateway

```
int UF_DRAW_ask_render_set_parms
(
    tag_t render_set,
    UF_DRAW_render_prefs_t * render_parms
)
```

tag_t	**render_set**	Input	Tag of render set. If this is a NULL_TAG, the function retrieves the default parameters.
UF_DRAW_render_prefs_t *	**render_parms**	Output	Pointer to render set preferences structure, with settings for visible and hidden line color, font, and widths, edge hiding edge and hidden line options.

UF_DRAW_ask_render_sets (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the render sets in the current part.

Environment
Internal and External

See Also
See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
gateway

int UF_DRAW_ask_render_sets
(
int * number_render_sets,
tag_p_t * render_sets
)

int *	**number_render_sets**	Output	Number of render sets in the part.
tag_p_t *	**render_sets**	Output to UF_*free*	Array of render sets in the current part. Use UF_free to free the memory.

UF_DRAW_ask_render_sets_of_view (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the render sets references by the given drawing member view. These render sets are listed in the order they are rendered in.

Environment
Internal and External

See Also
See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
gateway

int UF_DRAW_ask_render_sets_of_view
(
tag_t view,
int * number_render_sets,
tag_p_t * render_sets
)

tag_t	**view**	Input	Tag of drawing member view.
int *	**number_render_sets**	Output	Number of render sets
tag_p_t *	**render_sets**	Output to UF_*free*	Array of render sets referenced by the given view. They are listed in the rendering order. This array must be freed by calling UF_free.

UF_DRAW_ask_revolved_sxline (view source)

Defined in: uf_draw.h

Overview

Retrieves revolved section line information, given the tag of the section line. Note that if a section line has a invalid section line status it does not independently cause the return code to be set to a non-zero value.

Environment

Internal and External

See Also

UF_DRAW_sxline_status_t
See UF_DRAW_create_revolved_sxview
for more information about revolved section lines and views.
See the example

Required License(s)

gateway

```
int UF_DRAW_ask_revolved_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    UF_DRF_object_t * rotpt_object,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    int * num_leg1_sxsegs,
    UF_DRAW_sxline_leg_t * cut_plane_leg,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of revolved section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Arrow direction vector (unitized) relative to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
UF_DRF_object_t *	**rotpt_object**	Output	Object associated to rotation point

int *	**num_sxviews**	Output	Number of section views associated to section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section view tags Use UF_free to free memory
int *	**num_sxsegs**	Output	Number of section line segments.
int *	**num_leg1_sxsegs**	Output	Number of section segments input to define the first leg. Cannot be greater than num_sxsegs. Any remaining segments define leg2.
UF_DRAW_sxline_leg_t *	**cut_plane_leg**	Output	Leg used to define cut plane = UF_DRAW_sxline_leg1 = UF_DRAW_sxline_leg2
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line segment tags Use UF_free to free memory
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

---

UF_DRAW_ask_simple_sxline (view source)

Defined in: uf_draw.h

Overview
Retrieves simple section line information, given the tag of the section line. If a section line has an invalid section line status, this does not independently cause the return code to be set to a non-zero value.

Environment
Internal and External

See Also
UF_DRAW_sxline_status_t
See UF_DRAW_create_simple_sxview
for more information about simple section lines and views.
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_simple_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1]= y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1]= y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
int *	**num_sxviews**	Output	Number of section views associated to section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section views. Use UF_free to free memory.
int *	**num_sxsegs**	Output	Number of section line segments
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line segment tags. Use UF_free to free memory
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

---

UF_DRAW_ask_simplified_curve (view source)

Defined in: uf_draw.h

Overview

Given a drawing curve (a silhouette or a section edge) or an edge that is a conic or a spline, and the tag to a drawing member view where this curve or edge resides, this function returns simplified curves (if they exist), and the simplification creation information. If a NULL_TAG is input as the view_tag and the master_curve_tag is a section edge or silhouette, the view is inferred, and if the master_curve_tag is an edge, the first view found where the edge is simplified is used.

Environment

Internal and External

See Also

See the example

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_DRAW_ask_simplified_curve
(
    tag_t master_curve_tag,
    tag_t * view_tag,
    logical * flat_arc_to_line,
    double * tolerance,
    int * num_segments,
    tag_p_t * segments
)
```

tag_t	**master_curve_tag**	Input	Tag of the master curve or edge that was simplified.
tag_t *	**view_tag**	Input / Output	Tag of the drawing member view of the master_curve.
logical *	**flat_arc_to_line**	Output	If TRUE, a post processing of the arc segments output from the simplification was performed to convert flat arc segments with chordal tolerance less than half the view display tolerance to line segments. Also, adjacent line segments and arc segments may have been joined, if the result was within half of the view display tolerance.
double *	**tolerance**	Output	Tolerance that was used to produce the simplification.
int *	**num_segments**	Output	Number of simplified curves
tag_p_t *	**segments**	Output to UF_*free*	Array of simplified curves. Use UF_free to free memory.

UF_DRAW_ask_solid_of_section (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_ask_solid_of_section

DESCRIPTION -
Given the tag of a section solid, output the tag of its solid.

PARAMETERS -
sxsolid_tag <I> Tag of section
solid_tag <O> Tag of solid
Returns -
0 = OK
if not 0 = error code

Required License(s)

gateway

```
int UF_DRAW_ask_solid_of_section
(
    tag_t sxsolid_tag,
    tag_t * solid_tag
)
```

tag_t	**sxsolid_tag**	Input	Tag of section
tag_t *	**solid_tag**	Output	Tag of solid

UF_DRAW_ask_stepped_sxline (view source)

Defined in: uf_draw.h

Overview

Retrieves stepped section line information, given the tag of the section line. NOTE: If a section line has a invalid section line status it does not independently cause the return code to be set to a non-zero value.

Environment

Internal and External

See Also

UF_DRAW_sxline_status_t
See UF_DRAW_create_stepped_sxview
for more information about stepped section lines and views.
See the example

Required License(s)

gateway

```
int UF_DRAW_ask_stepped_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of stepped section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Arrow direction vector (unitized) relative to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
int *	**num_sxviews**	Output	Number of section views associated to section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section views. Use UF_free to free memory.
int *	**num_sxsegs**	Output	Number of section line segments
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line's segment tags. Use UF_free to free memory.
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

---

UF_DRAW_ask_suppress_view_updat (view source)

Defined in: uf_draw.h

Overview

Queries the current value of the Suppress View Update preference.
This preference is saved with a part. Retrieves value from the root part.
If no parts are loaded, an error will occur.
If the preference is TRUE, then functions which perform implicit
drawing update, will not update the drawing member views.

Note: Starting in nx2, this preference was saved with the part.
Some parts may have been inadvertently saved in nx2 with suppress
view updated turned off.
To allow users to override the suppress view update setting, an
environment variable "UGII_SUPPRESS_VIEW_UPDATE" can be set with
values "0" or "1". If set to 1, this will prevent an automatic
update of the out-of-date views.

Environment

Internal and External

See Also

UF_DRAW_set_suppress_view_updat
UF_DRAW_update_one_view
UF_DRAW_is_object_out_of_date
See the example

Required License(s)

gateway

```
int UF_DRAW_ask_suppress_view_updat
(
    logical * suppress_view_update
)
```

logical *	**suppress_view_update**	Output	Suppress View Update preference setting: TRUE = suppress all system initiated view updates FALSE = allow all system initiated view updates

UF_DRAW_ask_sxedges_of_sxsolid (view source)

Defined in: uf_draw.h

Overview
Given the tag of a solid section, outputs the number of associated section edges and an array of tags of these section edges.

Environment
Internal and External

See Also
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_sxedges_of_sxsolid
(
    tag_t sxsolid_tag,
    int* num_sxedges,
    tag_t* * sxedge_tags
)
```

tag_t	**sxsolid_tag**	Input	solid section tag
int*	**num_sxedges**	Output	number of section edges of solid section
tag_t* *	**sxedge_tags**	Output to UF_*free*	Array of section edge tags. Use UF_free to free memory.

UF_DRAW_ask_sxline_default_prfs (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the default section line display preferences, including section line visibility and arrow parameters.

Environment
Internal and External

See Also

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_sxline_default_prfs
(
    UF_DRAW_arrow_parms_t * arrow_parms,
    UF_DRAW_sxline_display_t * sxline_display
)
```

UF_DRAW_arrow_parms_t *	**arrow_parms**	Output	Section line arrow parameters
UF_DRAW_sxline_display_t *	**sxline_display**	Output	Section line display: UF_DRAW_display_sxline = display section line. UF_DRAW_no_display_sxline = do not display section line.

UF_DRAW_ask_sxline_display (view source)

Defined in: uf_draw.h

Overview

Retrieves the input section line's display preferences, including the
the section line's visibility and its arrow parameters.

NOTE: Do not invoke this routine for section lines of type UF_DRAW_breakline.

Environment

Internal and External

See Also

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_sxline_display
(
    tag_t sxline_tag,
    UF_DRAW_arrow_parms_t * arrow_parms,
    UF_DRAW_sxline_display_t * sxline_display
)
```

tag_t	**sxline_tag**	Input	Tag of section line
UF_DRAW_arrow_parms_t *	**arrow_parms**	Output	Section line arrow parameters
UF_DRAW_sxline_display_t *	**sxline_display**	Output	Section line display: UF_DRAW_display_sxline = display section line. UF_DRAW_no_display_sxline = do not display section line.

UF_DRAW_ask_sxline_of_sxview (view source)

Defined in: uf_draw.h

Overview
Retrieves the section line tag that is associated to the given section view.

Environment
Internal and External

See Also
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_sxline_of_sxview
(
    tag_t sxview_tag,
    tag_t * sxline_tag
)
```

tag_t	**sxview_tag**	Input	Tag of section view
tag_t *	**sxline_tag**	Output	Tag of associated section line

UF_DRAW_ask_sxline_sxseg (view source)

Defined in: uf_draw.h

Overview
Retrieves section line segment information, given the tag of the section line segment.

Environment
Internal and External

See Also
UF_DRAW_sxline_sxsegs_t
See the example

Required License(s)
gateway

```
int UF_DRAW_ask_sxline_sxseg
(
    tag_t sxseg_tag,
    UF_DRAW_sxseg_info_t * sxseg_info,
    tag_t * curve_tag,
    UF_DRF_object_p_t * object
)
```

tag_t	**sxseg_tag**	Input	Tag of section line segment to query
UF_DRAW_sxseg_info_t *	**sxseg_info**	Output	Segment information
tag_t *	**curve_tag**	Output	Tag of segment curve
UF_DRF_object_p_t *	**object**	Output to UF_*free*	Object associated to segment. Use UF_free to free the memory.

UF_DRAW_ask_sxline_type (view source)

Defined in: uf_draw.h

Overview

Retrieves the section line type, given the tag of the section line.

Environment

Internal and External

See Also

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_sxline_type
(
    tag_t sxline_tag,
    UF_DRAW_sxline_type_t * sxline_type
)
```

tag_t	**sxline_tag**	Input	Tag of section line to query its type

UF_DRAW_sxline_type_t *	**sxline_type**	Output	UF_DRAW_simple_sxline= simple section line
			UF_DRAW_stepped_sxline= stepped section line
			UF_DRAW_half_sxline = half section line
			UF_DRAW_revolved_sxline = revolved section line
			UF_DRAW_unfolded_sxline = unfolded section line

UF_DRAW_ask_sxsolids_of_sxview (view source)

Defined in: uf_draw.h

Overview

Returns the count and array of solid sections created by the sectioning of an input leg. Depending on whether the section view is a revolved or non-revolved type the following descriptions will apply:
For Non-Revolved Section Views
Given the tag of a section view output an array of solid section tags that are sectioned in the input section view.
For Revolved Section Views
Given the tag of a section view and the leg number, output an array of solid section tags that are sectioned in the input section view by the leg of the section view's section line.

Environment

Internal and External

See Also

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_sxsolids_of_sxview
(
    tag_t sxview_tag,
    UF_DRAW_sxline_leg_t leg_num,
    int* num_sxsolids,
    tag_t* * sxsolid_tags
)
```

tag_t	**sxview_tag**	Input	Tag of section view
UF_DRAW_sxline_leg_t	**leg_num**	Input	Leg number of the cut, UF_DRAW_sxline_leg1 or UF_DRAW_sxline_leg2. Required only if sxview_tag was created from a two legged revolved section line.
int*	**num_sxsolids**	Output	Number of solid sections created by the sectioning of the input leg.
tag_t* *	**sxsolid_tags**	Output to UF_*free*	Array of solid sections created by the sectioning of the input leg. Use UF_free to free memory.

UF_DRAW_ask_sxview_display (view source)

Defined in: uf_draw.h

Overview

Returns the structure of a single specified section view display setting.

Environment

Internal and External

See Also

UF_DRAW_set_sxview_display
See the example

Required License(s)

gateway

```
int UF_DRAW_ask_sxview_display
(
    tag_t view_tag,
    UF_DRAW_sxview_prfs_t * sxview_parms
)
```

tag_t	**view_tag**	Input	Tag of section view
UF_DRAW_sxview_prfs_t *	**sxview_parms**	Output	Data structure contains the section view preference parameters.

UF_DRAW_ask_unfolded_sxline (view source)

Defined in: uf_draw.h

Overview

Retrieves unfolded section line information, given the tag of the section line.
NOTE: If a section line has an invalid section line status it does not independently cause the return code to be set to a non-zero value.

Environment

Internal and External

See Also

UF_DRAW_sxline_status_t
See UF_DRAW_create_unfolded_sxview
for more information about unfolded section lines and views.
See the example

History

Original release was in V14.0.

Required License(s)

gateway

```
int UF_DRAW_ask_unfolded_sxline
(
    tag_t sxline_tag,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t * pview_tag,
    int * num_sxviews,
    tag_p_t * sxview_tags,
    int * num_sxsegs,
    tag_p_t * sxseg_tags,
    UF_DRAW_sxline_status_t * sxline_status
)
```

tag_t	**sxline_tag**	Input	Tag of unfolded section line to query

double	**step_dir [3]**	Output	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Output	Arrow direction vector (unitized) relative to the drawing:
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t *	**pview_tag**	Output	Parent view tag
int *	**num_sxviews**	Output	Number of section views associated to section line
tag_p_t *	**sxview_tags**	Output to UF_*free*	Array of section line's section views. Use UF_free to free memory.
int *	**num_sxsegs**	Output	Number of section line segments
tag_p_t *	**sxseg_tags**	Output to UF_*free*	Array of section line's segment tags. Use UF_free to free memory.
UF_DRAW_sxline_status_t *	**sxline_status**	Output	Section line status

---

UF_DRAW_ask_view_anchor (view source)

Defined in: uf_draw.h

Overview
Retrieves the view's anchor point.

Environment
Internal and External

See Also
See the example

History
Original Release was in V13.0.

Required License(s)
gateway

```
int UF_DRAW_ask_view_anchor
(
    const tag_t view_tag,
    tag_t * anchor_point
)
```

const tag_t	**view_tag**	Input	Tag of view whose model reference point is to be obtained.
tag_t *	**anchor_point**	Output	Pointer to tag of point that represents the view reference point.

UF_DRAW_ask_view_angle (view source)

Defined in: uf_draw.h

Overview
Returns the view angle in degrees.

Environment
Internal & External

See Also
See UF_DRAW_set_view_angle
See example

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_DRAW_ask_view_angle
(
    tag_t view_tag,
    double * angle_value
)
```

tag_t	**view_tag**	Input	The view tag
double *	**angle_value**	Output	The angle of the view in degrees

---

UF_DRAW_ask_view_borders (view source)

Defined in: uf_draw.h

Overview
This routine reads the view borders on the current drawing.

Environment
Internal and External

See Also
See the example program ufd_draw_ask_view_borders.c

History

New for V16.0

Required License(s)

gateway

int UF_DRAW_ask_view_borders
(
    const tag_t view_tag,
    double view_borders [4]
)

const tag_t	**view_tag**	Input	Tag of the view. If NULL_TAG, use Work view.
double	**view_borders [4]**	Output	View Borders (Drawing Coordinates). [0] - X min [1] - Y min [2] - X max [3] - Y max

---

UF_DRAW_ask_view_display (view source)

Defined in: uf_draw.h

Overview

Returns the structure of a single specified view display setting for a specific drawing member view. The display structures are as follows:

Environment

Internal and External

See Also

See the example
UF_DRAW_view_prfs_t

Required License(s)

gateway

int UF_DRAW_ask_view_display
(
    tag_t view_tag,
    UF_DRAW_view_prfs_t * view_parms
)

tag_t	**view_tag**	Input	Tag of drawing member view
UF_DRAW_view_prfs_t *	**view_parms**	Output	Data structure contains the hidden line removal, edge hiding edge, silhouette, uv hatch, smooth edge display, smooth edge color, font,, width and gap, virtual intersection color, font, width, gap, and status, tolerance, hidden line color, font, and width parameters, and the extracted edges setting.

UF_DRAW_ask_view_label (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the tag of the view label associated to a view.
If no view label is associated, view_label_tag is set to NULL_TAG.

Return
0 = successful
= UF_DRAW_tag_not_view

Environment
Internal and External

History
Created in v17.0

Required License(s)
gateway

```
int UF_DRAW_ask_view_label
(
    tag_t view_tag,
    tag_t * view_label_tag
)
```

tag_t	**view_tag**	Input	Drawing member view tag
tag_t *	**view_label_tag**	Output	View label tag

UF_DRAW_ask_view_label_parms (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the parameters from a view label object. If the view label has been customized and the parameters cannot be determined, UF_DRAW_invalid_parameter will be returned and view_label_parms will not be changed.

Note: If the input parameter view_label_tag is NULL_TAG, the view_label_parm_type element in the view_label_parms structure must be set to a valid type in order to retrieve global view label parameters for that type of view label.

Return
0 = successful
= UF_DRAW_invalid_parameter

Environment

Internal and External

History
Created in v17.0

Required License(s)
gateway

```
int UF_DRAW_ask_view_label_parms
(
    tag_t view_label_tag,
    UF_DRAW_view_label_parms_p_t view_label_parms
)
```

tag_t	**view_label_tag**	Input	View label tag OR NULL_TAG to ask global preferences
UF_DRAW_view_label_parms_p_t	**view_label_parms**	Input / Output	Structure that will be filled with the view label parameters

---

UF_DRAW_ask_view_notes (view source)

Defined in: uf_draw.h

Overview
Checks whether a drawing member view has associated notes. If true, this function returns the tags of the notes as output. Otherwise, the number of the associated notes is zero (0).

Environment
Internal and External

See Also
UF_DRAW_ask_view_of_note
UF_DRAW_attach_note_to_view
UF_DRAW_detach_note_from_view
See the example

History
Original release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_ask_view_notes
(
    tag_t view_tag,
    int * num_notes,
    tag_p_t * note_tags
)
```

tag_t	**view_tag**	Input	Drawing member view tag

int *	**num_notes**	Output	Number of associated notes
tag_p_t *	**note_tags**	Output to UF_*free*	If num_notes > 0, the array of note tags must be freed. Use UF_free to free note_tags.

UF_DRAW_ask_view_of_drawing (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the tag of the drawing view for the given drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_ask_num_views.c

History
New for V18.0

Required License(s)
gateway

```
int UF_DRAW_ask_view_of_drawing
(
    const tag_t drawing,
    tag_t * view
)
```

const tag_t	**drawing**	Input	Tag of drawing whose view is unknown.
tag_t *	**view**	Output	Tag of the view for the given drawing. NULL_TAG if not found.

UF_DRAW_ask_view_of_note (view source)

Defined in: uf_draw.h

Overview
Detemines if a note is associated to a drawing member view. If so, this function returns the view tag as an output. This routine is not meant to determine the view associated to a view label.
Use UF_DRAW_ask_view_of_view_label() to determine the view associated to a view label.

Environment
Internal and External

See Also

UF_DRAW_ask_view_notes
UF_DRAW_attach_note_to_view
UF_DRAW_detach_note_from_view
UF_DRAW_ask_view_of_view_label
See the example

History
Original release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_ask_view_of_note
(
    tag_t note_tag,
    tag_t * view_tag
)
```

tag_t	**note_tag**	Input	Note tag
tag_t *	**view_tag**	Output	The drawing member view the note is associated to. NULL_TAG if the note is not associated to a view.

UF_DRAW_ask_view_of_view_label (view source)

Defined in: uf_draw.h

Overview
This routine retrieves the tag of the view associated to a view label.
If no view is associated, view_tag is set to NULL_TAG.

Return
0 = successful
= UF_DRAW_tag_not_view, UF_DRAW_invalid_parameter

Environment
Internal and External

History
Created in v19.0

Required License(s)
gateway

```
int UF_DRAW_ask_view_of_view_label
(
    tag_t view_label_tag,
    tag_t * view_tag
)
```

tag_t	**view_label_tag**	Input	view label tag
tag_t *	**view_tag**	Output	Drawing member view label tag

UF_DRAW_ask_view_parm_scale (view source)

Defined in: uf_draw.h

Overview

If the view scale is associative, this function outputs an expression tag and view scale value. If the view scale is not associative, this function still outputs the view scale value, but the expression is equal to a NULL_TAG and returns an error.

Return

Return code:
0 = No error
UF_DRAW_hinge_not_linear = warning
not 0 = Error code

Environment

Internal and External

See Also

See the example

Required License(s)

gateway

```
int UF_DRAW_ask_view_parm_scale
(
    tag_t view_tag,
    tag_t * exp_tag,
    double * scale_value
)
```

tag_t	**view_tag**	Input	View from which the scale value and associated expression is obtained.
tag_t *	**exp_tag**	Output	Expression associated to the view scale, NULL_TAG if the view scale is not associative
double *	**scale_value**	Output	The scale value of the view.

UF_DRAW_ask_view_scale (view source)

Defined in: uf_draw.h

Overview

Returns the view scale. Also returns the expression tag if the view scale is parametric. If the view scale is not parametric, it returns a NULL_TAG.

Environment

Internal & External

See Also
See UF_DRAW_set_view_scale
See example

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_DRAW_ask_view_scale
(
    tag_t view_tag,
    tag_t * exp_tag,
    double * scale_value
)
```

tag_t	**view_tag**	Input	The view tag
tag_t *	**exp_tag**	Output	The expression tag
double *	**scale_value**	Output	The scale of the view

UF_DRAW_ask_view_status (view source)

Defined in: uf_draw.h

Overview
This routine reads the view status in the drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_set_view_status.c

History
New for V16.0

Required License(s)
gateway

```
int UF_DRAW_ask_view_status
(
    const tag_t view_tag,
    UF_DRAW_view_status_t * view_status
)
```

const tag_t	**view_tag**	Input	Tag of the view. If NULL_TAG, use Work view.

UF_DRAW_view_status_t *	**view_status**	Output	View Status UF_DRAW_ACTIVE_VIEW UF_DRAW_REFERENCE_VIEW

UF_DRAW_ask_view_thd_app_pitch (view source)

Defined in: uf_draw.h

Overview

Accepts a view tag as an input parameter and returns the minimum apparent pitch. The minimum apparent pitch is defined as the minimum value used to visually represent the pitch when rendering symbolic thread features on drawing member views. It can be used to preserve the visual clarity of threads whose pitch is small relative to the view scale.

For example, if a view with a scale of 2.0 contained threads with an actual pitch of 0.0125 inches, the actual thread pitch would render at 0.0125 2 = 0.025 inches which may be unreadable when plotted at full scale. If the view's minimum apparent pitch had been set to 0.0625 inches, then that value (in drawing units) would be used to render the thread pitch.

Environment

Internal and External

See Also

UF_DRAW_ask_view_thd_meth
UF_DRAW_set_view_thd_meth
UF_DRAW_set_view_thd_app_pitch

Required License(s)

gateway

```
int UF_DRAW_ask_view_thd_app_pitch
(
    tag_t view,
    double * app_pitch
)
```

tag_t	**view**	Input	Tag of drawing member view
double *	**app_pitch**	Output	The minimum pitch for rendered threads in the given view

---

UF_DRAW_ask_view_thd_meth (view source)

Defined in: uf_draw.h

Overview

Accepts a view tag as an input parameter and returns the view rendering method. Each rendering method corresponds with an ANSI or ISO thread standard. Valid rendering methods are:

UF_DRAW_THD_METH_NONE
UF_DRAW_THD_METH_ANSI_SIMPLIFIED
UF_DRAW_THD_METH_ANSI_SCHEMATIC
UF_DRAW_THD_METH_ANSI_DETAILED
UF_DRAW_THD_METH_ISO_SIMPLIFIED
UF_DRAW_THD_METH_ISO_DETAILED
UF_DRAW_THD_METH_ESKD_SIMPLIFIED
The default rendering method for pre-V12 member views is
UF_DRAW_THD_METH_NONE.

Environment

Internal and External

See Also

See UF_DRAW_ask_view_thd_app_pitch
UF_DRAW_set_view_thd_meth
UF_DRAW_set_view_thd_app_pitch

Required License(s)

gateway

```
int UF_DRAW_ask_view_thd_meth
(
    tag_t view,
    int * method
)
```

tag_t	**view**	Input	tag of the drawing member view
int *	**method**	Output	the view's current rendering method

UF_DRAW_ask_views (view source)

Defined in: uf_draw.h

Overview

This routine reads the number of member views in the given drawing and returns an array of the member view tags.

Environment

Internal and External

See Also

See the example program ufd_draw_ask_view_angle.c

History

New for V16.0

Required License(s)

gateway

```
int UF_DRAW_ask_views
(
    const tag_t drawing_tag,
    int * num_views,
    tag_p_t * view_tag
)
```

const tag_t	**drawing_tag**	Input	Tag of the drawing. If NULL_TAG, use current drawing.
int *	**num_views**	Output	Number of views in the drawing.
tag_p_t *	**view_tag**	Output to UF_*free*	Array of the view tags in the drawing. Use UF_free to free this memory.

UF_DRAW_ask_xhatch_of_sxsolid (view source)

Defined in: uf_draw.h

Overview

Given the tag of a solid section, output the tag of its associated crosshatch entity. Note that if a section view has its crosshatch preference turned off, a tag to a crosshatch entity that has no line information will be output.

Environment

Internal and External

See Also

See the example of

Required License(s)

gateway

```
int UF_DRAW_ask_xhatch_of_sxsolid
(
    tag_t sxsolid_tag,
    tag_t* xhatch_tag
)
```

tag_t	**sxsolid_tag**	Input	section solid tag
tag_t*	**xhatch_tag**	Output	crosshatch tag

UF_DRAW_attach_note_to_view (view source)

Defined in: uf_draw.h

Overview

Associates an existing note to an existing drawing member view. If the note is already associated to another drawing member view, it need not be disassociated from the view prior to calling this function.

Environment

Internal and External

See Also

UF_DRAW_ask_view_of_note
UF_DRAW_ask_view_notes
UF_DRAW_detach_note_from_view
See the example

History

Original release was in V14.0.

Required License(s)

drafting

```
int UF_DRAW_attach_note_to_view
(
    tag_t note_tag,
    tag_t view_tag
)
```

tag_t	**note_tag**	Input	Note tag to be associated to the view
tag_t	**view_tag**	Input	The drawing member view the note will be associated to.

UF_DRAW_copy_view (view source)

Defined in: uf_draw.h

Overview

This routine copies a view and its associated annotation, leaving the new view positioned on top of the original view. The new view can be moved to another drawing with UF_DRAW_move_view_to_drawing, or to another position on the same drawing with UF_DRAW_move_view.

Environment

Internal and External

See Also

UF_DRAW_move_view_to_drawing
UF_DRAW_move_view
See the example

History

Original Release was in V13.0.

Required License(s)

drafting

```
int UF_DRAW_copy_view
(
    tag_t view_tag,
    tag_t * new_view
)
```

tag_t	**view_tag**	Input	Tag of drawing member view to copy. It must be on the current drawing.
tag_t *	**new_view**	Output	Pointer to new view (copy of the original view).

UF_DRAW_create_break_region (view source)

Defined in: uf_draw.h

Overview
This routine creates a break region for a given view.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_broken_view.c to review an example of using
this function.

Environment
Internal and External

History
New for V17.0

Required License(s)
drafting

```
int UF_DRAW_create_break_region
(
    tag_t view_tag,
    tag_t anchor_point,
    int num_curves,
    UF_DRAW_break_region_boundary_p_t curves,
    tag_p_t break_region
)
```

tag_t	**view_tag**	Input	View to which the break region is to be added.
tag_t	**anchor_point**	Input	tag of anchor_point for break region
int	**num_curves**	Input	number of boundary curves
UF_DRAW_break_region_boundary_p_t	**curves**	Input	Break region boundary data.
tag_p_t	**break_region**	Output	Tag of the break region created.

UF_DRAW_create_breakout (view source)

Defined in: uf_draw.h

Overview
This routine creates a breakout section within a given view.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_cre_breakout.c to review an example of using
this function.

Environment
Internal and External

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_create_breakout
(
    tag_t view_tag,
    UF_DRAW_breakout_data_p_t breakout_data,
    tag_p_t breakline
)
```

tag_t	**view_tag**	Input	View to which the breakout is to be added.
UF_DRAW_breakout_data_p_t	**breakout_data**	Input	Breakout section data.
tag_p_t	**breakline**	Output	Tag of the breakout section created.

UF_DRAW_create_drawing (view source)

Defined in: uf_draw.h

Overview
This routine creates a new drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_create_drawing.c

History
New for V16.0

Required License(s)

gateway

```
int UF_DRAW_create_drawing
(
    const char * drawing_name,
    const UF_DRAW_info_p_t drawing_info,
    tag_p_t drawing_tag
)
```

const char *	**drawing_name**	Input	Name of the drawing.
const UF_DRAW_info_p_t	**drawing_info**	Input	The desired drawing info.
tag_p_t	**drawing_tag**	Output	Tag of the new drawing.

UF_DRAW_create_half_sxview (view source)

Defined in: uf_draw.h

Overview

Creates a half section line and view.

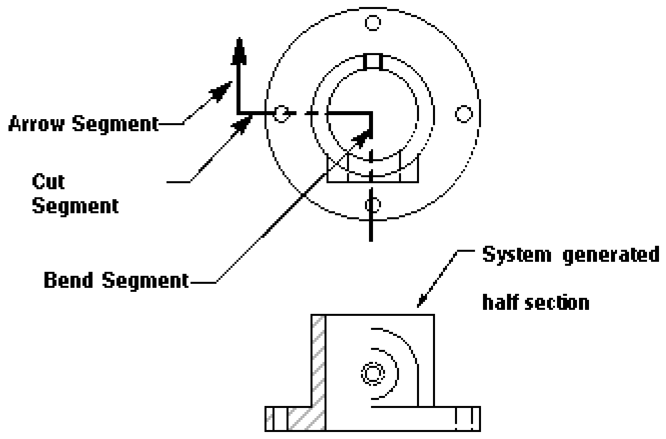

Figure. A half section line and section view

Environment

Internal and External

See Also

See the discussion on the half_sxsegs structure in the Types and

See Also

UF_DRAW_half_sxsegs_t
See the example

Required License(s)

drafting

```
int UF_DRAW_create_half_sxview
(
    tag_t dwg_tag,
    double sxview_scale,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t pview_tag,
    int num_sxsegs,
    UF_DRAW_half_sxsegs_p_t half_sxseg_objects,
    double view_placement_pt [ 2 ] ,
    tag_t* sxview_tag
)
```

tag_t	**dwg_tag**	Input	Drawing tag
double	**sxview_scale**	Input	Section view scale

double	**step_dir [3]**	Input	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the drawing.
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t	**pview_tag**	Input	Parent view tag
int	**num_sxsegs**	Input	Number of section segments. A cut segment and bend segment must be provided (num_segs=2). If the arrow segment location is optionally provided, num_segs=3.
UF_DRAW_half_sxsegs_p_t	**half_sxseg_objects**	Input	Section line segment object structure
double	**view_placement_pt [2]**	Input	View placement point on drawing in absolute drawing coordinates (x,y))
tag_t*	**sxview_tag**	Output	Tag of newly created section view

---

UF_DRAW_create_render_set (view source)

Defined in: uf_draw.h

Overview

This routine creates a render set with the given display preferences

Environment

Internal and External

See Also

See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
drafting

```
int UF_DRAW_create_render_set
(
    char * render_set_name,
    UF_DRAW_render_prefs_t * render_parms,
    tag_t * render_set
)
```

char *	**render_set_name**	Input	Desired name of render set.
UF_DRAW_render_prefs_t *	**render_parms**	Input	Pointer to render set preferences structure, with desired settings for visible and hidden line color, font, and widths, edge hiding edge and hidden line options.
tag_t *	**render_set**	Output	Tag of newly created render set, if successful.

UF_DRAW_create_revolved_sxview (view source)

Defined in: uf_draw.h

Overview
Creates a revolved section view. The system generates a bend when two cut segments are input with no intervening bend definition. The system also ignores the second of two consecutive input bend segments. The system generates arrow segments if they are not input.

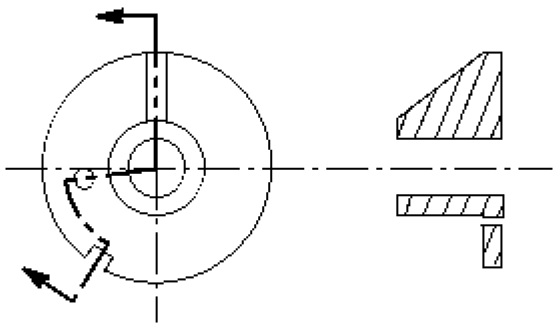

Figure. A revolved section line and section view.

Environment
Internal and External

See Also

UF_DRAW_sxline_sxsegs_t
See the example

Required License(s)

drafting

```
int UF_DRAW_create_revolved_sxview
(
    tag_t dwg_tag,
    double sxview_scale,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t pview_tag,
    UF_DRF_object_p_t rotpt_object,
    int num_sxsegs,
    int num_leg1_sxsegs,
    UF_DRAW_sxline_sxsegs_p_t rev_sxsegs,
    double view_placement_pt [ 2 ] ,
    tag_t * sxview_tag
)
```

tag_t	**dwg_tag**	Input	Drawing tag
double	**sxview_scale**	Input	Section view scale

double	**step_dir [3]**	Input	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1]= y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the drawing.
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t	**pview_tag**	Input	Parent view tag
UF_DRF_object_p_t	**rotpt_object**	Input	Object defining rotation point associativity.
int	**num_sxsegs**	Input	Number of section segments used to initially define the section line. A cut segment must be defined (num_segs must be > 0). This number defines the size of the rev_sxseg_objects array. A section line cannot have more than 99 segments and cannot have more than 49 cut segments.
int	**num_leg1_sxsegs**	Input	Number of section segments input to define the first leg. Cannot be greater than num_segs. Any remaining segments define leg2.
UF_DRAW_sxline_sxsegs_p_t	**rev_sxsegs**	Input	For each section segment defined, a segment type is given that determines whether the segment is a cut, bend,

			or arrow. Bend and arrow segments are optional. Also, for each section segment defined, an object is given to define the section segment associativity.
double	**view_placement_pt [2]**	Input	View placement point on drawing in absolute drawing coordinates (x,y))
tag_t *	**sxview_tag**	Output	Tag of newly created section view

UF_DRAW_create_simple_sxview (view source)

Defined in: uf_draw.h

Overview

Creates a simple section line and view. Only the cut segment of the section line is input. The arrow segments of the section line are automatically generated.

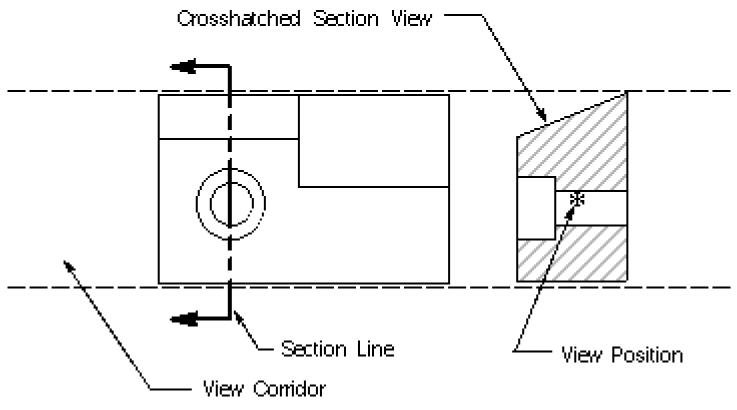

Figure. A simple section line and section view

Environment

Internal and External

See Also

See the example

Required License(s)

drafting

```
int UF_DRAW_create_simple_sxview
(
    tag_t dwg_tag,
    double sxview_scale,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t pview_tag,
    UF_DRF_object_p_t cut_object,
    double view_placement_pt [ 2 ] ,
    tag_t * sxview_tag
)
```

tag_t	**dwg_tag**	Input	Drawing tag
double	**sxview_scale**	Input	Section view scale

double	**step_dir [3]**	Input	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the drawing.
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t	**pview_tag**	Input	Parent view tag
UF_DRF_object_p_t	**cut_object**	Input	Object associated to cut segment
double	**view_placement_pt [2]**	Input	View placement point on drawing in absolute drawing coordinates (x,y)
tag_t *	**sxview_tag**	Output	Tag of newly created section view

---

UF_DRAW_create_simplified_curve (view source)

Defined in: uf_draw.h

Overview

Given a drawing curve (a silhouette or a section edge) or an edge that is a conic or a spline, and the tag to a drawing member view where this curve or edge is displayed, this function creates new arcs or lines that approximate that conic or spline curve.
The approximation uses the drawing member view's tolerance view display preference in its calculations.
Note that the simplification can be deleted by deleting any one of the curves of the simplification.

Environment

Internal and External

See Also

See UF_DRAW_set_view_display
to change the tolerance value. This tolerance value is defined in
UF_DRAW_view_prfs_t
See the example

History

Original release was in V13.0.

Required License(s)

drafting

int UF_DRAW_create_simplified_curve
(

```
        tag_t master_curve_tag,
        tag_t view_tag,
        logical flat_arc_to_line,
        int * num_segments,
        tag_p_t * segments
    )
```

tag_t	**master_curve_tag**	Input	Tag of the master curve or edge to simplify
tag_t	**view_tag**	Input	Tag of the drawing member view of the master_curve. If the master_curve is a section edge or silhouette, view_tag will be ignored as simple curves will be created in the view of the master curve.
logical	**flat_arc_to_line**	Input	If TRUE, a post processing of the arc segments output from the simplification will be performed to convert flat arc segments with chordal tolerance less than half the view display tolerance to line segments. Also, adjacent line segments and arc segments will be joined if the result is within half the view display tolerance.
int *	**num_segments**	Output	Number of simplified curves
tag_p_t *	**segments**	Output to UF_*free*	Array of simplified curves. Use UF_free to free memory.

UF_DRAW_create_stepped_sxview (view source)

Defined in: uf_draw.h

Overview

Creates a stepped section view. The system generates a bend when two cut segments are input with no intervening bend definition. The system also ignores the second of two consecutive input bend segments. The system generates arrow segments if they are not input.

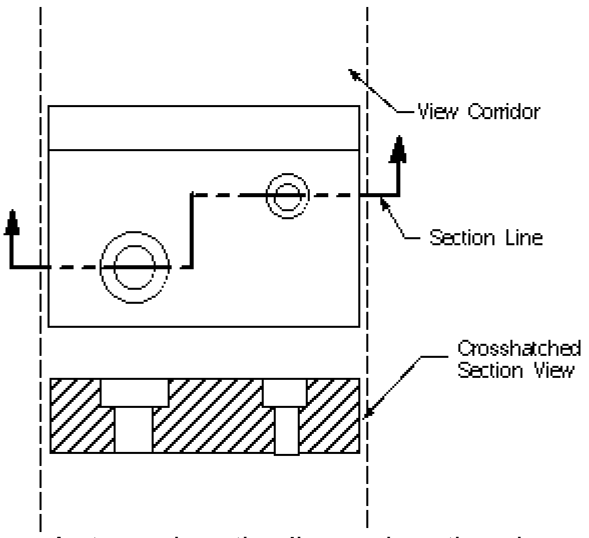

Figure. A stepped section line and section view

Environment
Internal and External

See Also
UF_DRAW_sxline_sxsegs_t

Required License(s)
drafting

int UF_DRAW_create_stepped_sxview
(
tag_t dwg_tag,
double sxview_scale,
double step_dir [3] ,
double arrow_dir [3] ,
tag_t pview_tag,
int num_sxsegs,
UF_DRAW_sxline_sxsegs_p_t stepped_sxsegs,
double view_placement_pt [2] ,
tag_t * sxview_tag
)

tag_t	**dwg_tag**	Input	Drawing tag
double	**sxview_scale**	Input	Section view scale

double	**step_dir [3]**	Input	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the drawing.
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t	**pview_tag**	Input	Parent view tag
int	**num_sxsegs**	Input	Number of section segments used to initially define the section line. A cut segment must be provided (num_sxsegs>0). This number determines the length of the stepped_sxsegs array.
UF_DRAW_sxline_sxsegs_p_t	**stepped_sxsegs**	Input	For each section segment defined, a segment type is given that determines whether the segment is a cut, bend or arrow. Bend and arrow segments are optional. Also, for each section seg ment defined, an object is given to de fine the section segment associativity. A section line cannot have more than 99 segments and cannot have more than 49 cut segments.
double	**view_placement_pt [2]**	Input	View placement point on drawing in absolute drawing coordinates (x,y))

tag_t *	**sxview_tag**	Output	Tag of newly created section view

UF_DRAW_create_sxview_from_dmv (view source)

Defined in: uf_draw.h

Overview

This routine allows the user to create a section line and a
section view from any view, excluding revolved section view
and unfolded section view. The display of the section line
is pictorial.

Return

0 = OK
if not 0 = Failure error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_cre_dmv_sxvw.c to review an example of
using this function.

Environment

Internal and External

History

New for V16.0

Required License(s)

drafting

```
int UF_DRAW_create_sxview_from_dmv
(
    tag_t drawing_tag,
    tag_t parent_view_tag,
    UF_DRAW_sxline_type_t section_type,
    double section_view_scale,
    double cut_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    int num_sxsegs,
    UF_DRAW_sxline_sxsegs_p_t segment_data,
    double view_placement_pt [ 2 ] ,
    logical expect_pictorial_sxview,
    tag_t * section_view_tag
)
```

tag_t	**drawing_tag**	Input	Drawing tag.
tag_t	**parent_view_tag**	Input	Parent view tag.
UF_DRAW_sxline_type_t	**section_type**	Input	Type of the section.
double	**section_view_scale**	Input	Section view scale.
double	**cut_dir [3]**	Input	Cut direction vector (unitized) relative to the model space

			cut_dir[0] = x value
			cut_dir[1] = y value
			cut_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the model space
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

int	**num_sxsegs**	Input	Number of section line segments
UF_DRAW_sxline_sxsegs_p_t	**segment_data**	Input	Data of these section segments
double	**view_placement_pt [2]**	Input	Location of the section view relative to the drawing.
logical	**expect_pictorial_sxview**	Input	Whether or not we expect a pictorial sxview.
tag_t *	**section_view_tag**	Output	Tag of the newly created section view.

UF_DRAW_create_unfolded_sxview (view source)

Defined in: uf_draw.h

Overview
Creates an unfolded section view.
NOTE: The system generates arrow segments if they are not input.

Environment
Internal and External

See Also
UF_DRAW_sxline_sxsegs_t
See the example

History
Original release was in V14.0.

Required License(s)
drafting

```
int UF_DRAW_create_unfolded_sxview
(
    tag_t dwg_tag,
    double sxview_scale,
    double step_dir [ 3 ] ,
    double arrow_dir [ 3 ] ,
    tag_t pview_tag,
    int num_sxsegs,
    UF_DRAW_sxline_sxsegs_p_t unfolded_sxsegs,
    double view_placement_pt [ 2 ] ,
    tag_t * sxview_tag
)
```

tag_t	**dwg_tag**	Input	Tag of drawing
double	**sxview_scale**	Input	Scale of section view

double	**step_dir [3]**	Input	Step direction vector (unitized) relative to the drawing:
			step_dir[0] = x value
			step_dir[1] = y value
			step_dir[2] = z value

double	**arrow_dir [3]**	Input	Arrow direction vector (unitized) relative to the drawing.
			arrow_dir[0] = x value
			arrow_dir[1] = y value
			arrow_dir[2] = z value

tag_t	**pview_tag**	Input	Parent view tag
int	**num_sxsegs**	Input	Number of section segments used to initially define the section line. A cut segment must be provided (num_sxsegs>0). This number determines the length of the stepped_sxsegs array.
UF_DRAW_sxline_sxsegs_p_t	**unfolded_sxsegs**	Input	For each section segment defined, a segment type is given that determines whether the segment is a cut, bend or arrow. Bend and arrow segments are op- tional. Also, for each section segment defined, an object is given to define the section segment associativity. A section line cannot have more than 99 segments and cannot have more than 49 cut segments.
double	**view_placement_pt [2]**	Input	View placement point on drawing in absolute drawing coordinates (x,y))
tag_t *	**sxview_tag**	Output	Tag of newly created section view

---

UF_DRAW_create_view_label (view source)

Defined in: uf_draw.h

Overview

This routine adds a view label to the specified view. If a view already has a view label, the function edits the existing view label.

Return

= 0 : successful
= UF_DRAW_tag_not_view

= UF_DRAW_invalid_parameter

Environment
Internal and External

History
Created in v17.0

Required License(s)
drafting

```
int UF_DRAW_create_view_label
(
    tag_t view_tag,
    UF_DRAW_view_label_parms_p_t view_label_parms,
    tag_t * view_label_tag
)
```

tag_t	**view_tag**	Input	Drawing member view tag
UF_DRAW_view_label_parms_p_t	**view_label_parms**	Input	Structure that is filled with the view view label parameters
tag_t *	**view_label_tag**	Output	View label tag

UF_DRAW_define_bound_by_objects (view source)

Defined in: uf_draw.h

Overview
Sets the view boundary type to Bound By Objects and defines an array of objects that must be contained with the bounds of the view.

Environment
Internal and External

See Also
UF_DRAW_ask_bound_by_objects
See the example

History
Original release was in V13.0

Required License(s)
drafting

```
int UF_DRAW_define_bound_by_objects
(
    const tag_t view_tag,
    const int num_objects,
    tag_t* bounded_objects
)
```

const tag_t	**view_tag**	Input	Tag of the view whose location is to be defined.
const int	**num_objects**	Input	Number of tags in the bounded_objects array.
tag_t*	**bounded_objects**	Input	num_objects Array of tags of objects to be used to calculate the bounding box for the view boundary.

UF_DRAW_define_view_auto_rect (view source)

Defined in: uf_draw.h

Overview
Defines the view boundary for the input view as an automatic view boundary. The view cannot be a detail view. The view must be active and on the current drawing. No view may be expanded prior to calling this function.

Environment
Internal and External

See Also
See the example

Required License(s)
drafting

```
int UF_DRAW_define_view_auto_rect
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	Tag of view.

UF_DRAW_define_view_boundary (view source)

Defined in: uf_draw.h

Overview
Creates a view boundary of a closed, connected, non-self-intersecting loop of curves. Valid curve types are UF_line_type, UF_circle_type, and UF_spline_type. All curves must be visible in the view. If the curve type is UF_circle_type, the circle must be in the plane of the view.

The Figure below shows an example of the the construction of an arbitrary view bound defined by selecting several curves. In the figure, the numbers associated with each curve are used to illustrate the curve selection order.

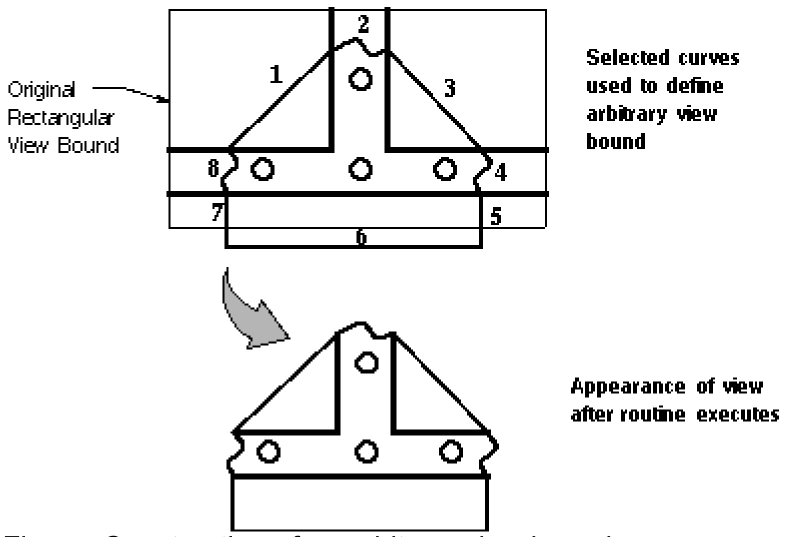
Original Rectangular View Bound
1
2
3
4
5
6
7
8
Selected curves used to define arbitrary view bound
Appearance of view after routine executes

Figure. Construction of an arbitrary view bound

Environment
Internal and External

See Also
UF_DRAW_ask_boundary_curves
See the example

Required License(s)
drafting

int UF_DRAW_define_view_boundary
(
tag_t view_tag,
int curve_count,
UF_DRAW_define_boundary_p_t* boundary_curves
)

tag_t	**view_tag**	Input	Tag of the view whose boundary is to be defined.
int	**curve_count**	Input	Count of tags of curves in curve_list
UF_DRAW_define_boundary_p_t*	**boundary_curves**	Input	curve_count Pointer to array of structures that are used to define the view boundary.

UF_DRAW_define_view_boundary1 (view source)

Defined in: uf_draw.h

Overview
In order to provide appropriate .NET binding for UF_DRAW_define_view_boundary, UF_DRAW_define_view_boundary1 is introduced.

Note: C/C++ users can continue to use UF_DRAW_define_view_boundary.

For docuementation, refer to documentation of UF_DRAW_define_view_boundary.

History
Originally released in NX8.5.1

Required License(s)
drafting

```
int UF_DRAW_define_view_boundary1
(
    tag_t view_tag,
    int curve_count,
    UF_DRAW_define_boundary_p_t* boundary_curves
)
```

tag_t	**view_tag**	Input	Tag of the view whose boundary is to be defined.
int	**curve_count**	Input	Count of tags of curves in curve_list
UF_DRAW_define_boundary_p_t*	**boundary_curves**	Input	curve_count Pointer to array of structures that are used to define the view boundary.

UF_DRAW_define_view_manual_rect (view source)

Defined in: uf_draw.h

Overview
Sets view borders for the given view on the current drawing. The view must be active on the current drawing. No view may be expanded before calling this function.

Environment
Internal and External

See Also
See the example

Required License(s)
drafting

```
int UF_DRAW_define_view_manual_rect
(
    tag_t view_tag,
    double view_borders [ 4 ]
)
```

tag_t	**view_tag**	Input	Tag of view.
double	**view_borders [4]**	Input	View Borders (Drawing Coordinates), can be any two diagonal corners (X1,Y1,X2,Y2) of the view box.

UF_DRAW_delete_drawing (view source)

Defined in: uf_draw.h

Overview
This routine deletes the drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_delete_drawing.c

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_delete_drawing
(
    const tag_t drawing_tag
)
```

const tag_t	**drawing_tag**	Input	Tag of drawing to delete.

UF_DRAW_delete_sxline_sxseg (view source)

Defined in: uf_draw.h

Overview
Deletes a segment of a section line, then updates all of the section
line's associated section views located on the current drawing.
Associated section views not on the current drawing are marked out
of date. To perform this edit on the section line without an update, use
the suppress view update feature provided in UF_DRF_set_suppress_view_update.

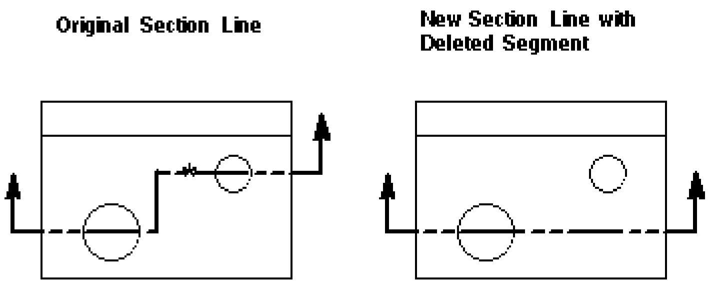

Figure. The deletion of a section line cut segment.

Environment
Internal and External

See Also

UF_DRF_set_suppress_view_update

Required License(s)

drafting

```
int UF_DRAW_delete_sxline_sxseg
(
    tag_t sxseg_tag
)
```

tag_t	**sxseg_tag**	Input	Tag of section line segment to delete

---

UF_DRAW_delete_view_label (view source)

Defined in: uf_draw.h

Overview

This routine deletes a view label from the specified view. The routine silently ignores views that do not already include a view label.

Return

= 0 : successful
= UF_DRAW_tag_not_view

Environment

Internal and External

History

Created in v17.0

Required License(s)

drafting

```
int UF_DRAW_delete_view_label
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	Drawing member view tag

UF_DRAW_detach_note_from_view (view source)

Defined in: uf_draw.h

Overview

Disassociates an existing note from an existing drawing member view.
NOTE: A section view label cannot be disassociated from the section view. A section line cannot be disassociated from its parent view.

Environment
Internal and External

See Also
UF_DRAW_ask_view_of_note
UF_DRAW_ask_view_notes
UF_DRAW_attach_note_to_view
See the example

History
Original release was in V14.0.

Required License(s)
drafting

```
int UF_DRAW_detach_note_from_view
(
    tag_t note_tag
)
```

tag_t	**note_tag**	Input	Note tag to be associated to the view

---

UF_DRAW_edit_boundary_point (view source)

Defined in: uf_draw.h

Overview
Replaces a smart defining point for an associative view boundary
curve with a new smart defining point. This function may return errors
used in UF_DRAW_define_view_boundary, because when defining_pt is
replaced with new_pt, the break line/detail boundary is redefined,
based on the new point location.

Environment
Internal and External

See Also
UF_DRAW_set_boundary_assoc
See the figure
See the example

History
Original release was in V14.0.

Required License(s)
drafting

```
int UF_DRAW_edit_boundary_point
(
    tag_t defining_point,
    tag_t new_point,
    tag_t view_tag
)
```

tag_t	**defining_point**	Input	Tag of the defining point which is to be replaced
tag_t	**new_point**	Input	Tag of the new point which will replace the defining point.
tag_t	**view_tag**	Input	Tag of the view whose defining point is being replaced. The view must have a break line/detail type boundary, and the boundary must be associated to model geometry (via UF_DRAW_set_boundary_assoc).

UF_DRAW_edit_sxline_display (view source)

Defined in: uf_draw.h

Overview

Edits the input section line's display preferences, including the the section line's visibility and its arrow parameters, and updates the display of the section line. The input preferences are also saved as the new global section line display preferences.

Environment

Internal and External

See Also

UF_DRAW_arror_parms_t

Required License(s)

drafting

```
int UF_DRAW_edit_sxline_display
(
    tag_t sxline_tag,
    UF_DRAW_arrow_parms_p_t arrow_parms,
    UF_DRAW_sxline_display_t sxline_display
)
```

tag_t	**sxline_tag**	Input	Tag of section line to edit
UF_DRAW_arrow_parms_p_t	**arrow_parms**	Input	Section line arrow parameters
UF_DRAW_sxline_display_t	**sxline_display**	Input	Section line display: UF_DRAW_display_sxline=display section line UF_DRAW_no_display_sxline=do not display section line

---

UF_DRAW_erase_sxview_objects (view source)

Defined in: uf_draw.h

Overview

This routine displays the input faces and/or bodies, and erases all the others in the section view.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message string associated with the error code.

Environment

Internal and External

History

New for V18.0

Required License(s)

drafting

```
int UF_DRAW_erase_sxview_objects
(
    const tag_t view,
    const int num_objects,
    const tag_p_t objects
)
```

const tag_t	**view**	Input	Tag of the view
const int	**num_objects**	Input	Number of objects
const tag_p_t	**objects**	Input	num_objects An array of objects

---

UF_DRAW_free_boundary (view source)

Defined in: uf_draw.h

Overview

Frees the memory in the boundary_curves structure.

Environment

Internal and External

See Also

UF_DRAW_ask_bound_by_objects
See the example

History

Original release was in V14.0.

Required License(s)

drafting

```
int UF_DRAW_free_boundary
(
        int curve_count,
        UF_DRAW_view_boundary_p_t boundary_curves
    )
```

int	**curve_count**	Input	Count of curves in curve list
UF_DRAW_view_boundary_p_t	**boundary_curves**	Input	Pointer to curve list, i.e. array of structures containing the boundary curves and their defining points.

UF_DRAW_get_view_model_view_part (view source)

Defined in: uf_draw.h

Overview
This function gets the part name of the model view that is imported into the drawing member view

Environment
Internal and External

History
Original release in NX5.0.3

Required License(s)
drafting

```
    int UF_DRAW_get_view_model_view_part
    (
        tag_t view,
        char model_view_partname [ MAX_FSPEC_BUFSIZE ]
    )
```

tag_t	**view**	Input	drawing member view
char	**model_view_partname [MAX_FSPEC_BUFSIZE]**	Output	(filespec with path) part name of view

---

UF_DRAW_import_view (view source)

Defined in: uf_draw.h

Overview
This function imports a view onto the current drawing. Use the view info structure to set the view status, anchor point, view scale, use reference point, clean model view, transfer annotation, and inherit boundary parameters.
NOTE: There is currently a restriction requiring the input drawing tag to be the tag of the current drawing. We intend to relax this restriction in the future. As a result, we are requiring the input tag to ensure that future code changes will not be required by NX or by

NX Open API developers.

Environment
Internal and External

History
Original release in V13.0.

Required License(s)
ddrafting

```
int UF_DRAW_import_view
(
    const tag_t drawing_tag,
    const tag_t view_tag,
    double dwg_reference_point [ 2 ] ,
    UF_DRAW_view_info_t * view_info,
    tag_t * draw_view_tag
)
```

const tag_t	**drawing_tag**	Input	Drawing Tag, must be the current drawing.
const tag_t	**view_tag**	Input	Tag of model view to import.
double	**dwg_reference_point [2]**	Input	Drawing Reference Point (in drawing coordinates, x,y).
UF_DRAW_view_info_t *	**view_info**	Input	View Info (see uf_draw_types.h).
tag_t *	**draw_view_tag**	Output	View Tag of imported view on drawing

---

UF_DRAW_initialize_view_info (view source)

Defined in: uf_draw.h

Overview
This function initializes the view info structure that
is used as an input to UF_DRAW_import_view.

The default settings that are assinged to this structure are:
view_info->view_status = UF_DRAW_ACTIVE_VIEW
view_info->anchor_point = NULL_TAG
view_info->view_scale = 1.0
view_info->use_ref_pt = FALSE
view_info->transfer_annotation = TRUE
view_info->inherit_boundary = FALSE
view_info->model_name[0] = '\0'
view_info->arrangement_name[0] = '\0'

Environment
Internal and External

History

Original release in V16.0.

Required License(s)
gateway

```
void UF_DRAW_initialize_view_info
(
    UF_DRAW_view_info_t * view_info
)
```

UF_DRAW_view_info_t *	**view_info**	Output	info structure to be initialized

---

UF_DRAW_is_drafting_component (view source)

Defined in: uf_draw.h

Overview
This function determines if the component is a Drafting component,
that is, if the component is a result of Add View from Part

Environment
Internal and External

History
Original release in NX5.0.3

Required License(s)
drafting

```
int UF_DRAW_is_drafting_component
(
    tag_t component,
    logical * is_drafting_component
)
```

tag_t	**component**	Input	component being tested
logical *	**is_drafting_component**	Output	true - component is a Drafting component

---

UF_DRAW_is_object_out_of_date (view source)

Defined in: uf_draw.h

Overview
Queries the up-to-date status of an object. Currently, only two types
of objects are valid: views and drawings.

Environment
Internal and External

See Also
UF_DRAW_ask_suppress_view_updat
UF_DRAW_set_suppress_view_updat
UF_DRAW_update_one_view
See the example

Required License(s)
gateway

```
int UF_DRAW_is_object_out_of_date
(
    tag_t object,
    logical * out_of_date
)
```

tag_t	**object**	Input	tag of view or drawing
logical *	**out_of_date**	Output	object out-of-date status

UF_DRAW_is_sxview (view source)

Defined in: uf_draw.h

Overview
Given the tag of a view, outputs whether the view is a section view.

Environment
Internal and External

See Also
See the example

History
Original release was in V14.0.

Required License(s)
gateway

```
int UF_DRAW_is_sxview
(
    tag_t view_tag,
    logical * is_a_sxview
)
```

tag_t	**view_tag**	Input	Tag of view
logical *	**is_a_sxview**	Output	If TRUE, the view is a section view, else it is not.

UF_DRAW_is_thread_curve (view source)

Defined in: uf_draw.h

Overview

Determines whether the specified curve is a thread curve. This can be used in conjunction with UF_DRAW_ask_group_of_curve to determine if a thread curve is a thread silhouette or thread section edge.

Environment

Internal and External

Required License(s)

gateway

```
int UF_DRAW_is_thread_curve
(
    tag_t curve_tag,
    logical * is_thread_curve
)
```

tag_t	**curve_tag**	Input	Tag of curve
logical *	**is_thread_curve**	Output	TRUE = curve is a thread curve (thread silhouette or section edge) FALSE = curve is not a thread curve

---

UF_DRAW_move_sxline_rotpt (view source)

Defined in: uf_draw.h

Overview

Moves a section line rotation point to a location defined by the input new object, then updates all of the section line's associated section views located on the current drawing. Associated section views not on the current drawing are marked out of date. To perform edits on the section line without an update, use the suppress view update feature provided in UF_DRF_set_suppress_view_update.

Environment

Internal and External

See Also

UF_DRAW_create_revolved_sxview
UF_DRF_set_suppress_view_update
for more information on revolved section lines and section views.
See the example

Required License(s)

drafting

```
int UF_DRAW_move_sxline_rotpt
(
    tag_t sxline_tag,
    UF_DRF_object_p_t new_object
```

)

tag_t	**sxline_tag**	Input	Tag of revolved section line
UF_DRF_object_p_t	**new_object**	Input	New object to associate rotation point

UF_DRAW_move_sxline_sxseg (view source)

Defined in: uf_draw.h

Overview

Moves a section line segment to a location defined by the input new_object, then updates all of the section line's associated section views located on the current drawing. Associated section views not on the current drawing are marked out of date. To perform edits on the section line without an update, use the suppress view update feature provided in UF_DRF_set_suppress_view_update.

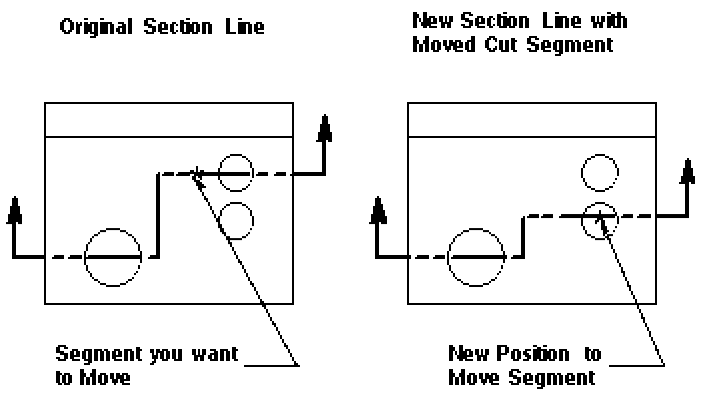
Original Section Line
New Section Line with Moved Cut Segment
Segment you want to Move
New Position to Move Segment

Figure. Moving section line segments

Environment

Internal and External

See Also

UF_DRF_set_suppress_view_update
See the example

Required License(s)

drafting

int UF_DRAW_move_sxline_sxseg
(
  tag_t sxseg_tag,
  UF_DRF_object_p_t new_object
)

tag_t	**sxseg_tag**	Input	Tag of section line segment to move.
UF_DRF_object_p_t	**new_object**	Input	New object to associate segment.

UF_DRAW_move_sxseg (view source)

Defined in: uf_draw.h

Overview

Moves a section line segment to a location defined by the input new object of sxseg_data, then updates all of the section line's associated section views located on the current drawing. Associated section views not on the current drawing are marked out of date. To perform this edit on the section line without an update, use the suppress view update feature provided in UF_DRF_set_suppress_view_update.

Environment

Internal and External

See Also

UF_DRF_set_suppress_view_update
UF_DRAW_sxline_sxsegs_t
See the example

History

Original release was in V14.0.

Required License(s)

drafting

```
int UF_DRAW_move_sxseg
(
    tag_t sxseg_tag,
    UF_DRAW_sxline_sxsegs_p_t sxseg_data
)
```

tag_t	**sxseg_tag**	Input	Tag of section line segment to move.
UF_DRAW_sxline_sxsegs_p_t	**sxseg_data**	Input	Segment data contains: segment type: UF_DRAW_sxseg_cut, linear segment angle, and object to associate to. The sxseg_object is used to move all segments. The sxseg_angle is used to move an unfolded sxseg.

UF_DRAW_move_view (view source)

Defined in: uf_draw.h

Overview

This routine moves a specified view to the given position on the current drawing.

Environment

Internal and External

See Also
UF_DRAW_copy_view
UF_DRAW_move_view_to_drawing
See the example

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRAW_move_view
(
    const tag_t view_tag,
    const double drawing_reference_point [ 2 ]
)
```

const tag_t	**view_tag**	Input	Drawing member view to move. It must be on the current drawing.
const double	**drawing_reference_point [2]**	Input	Desired drawing reference point, in drawing coordinates.

UF_DRAW_move_view_to_drawing (view source)

Defined in: uf_draw.h

Overview
This routine moves a drawing member view to the specified drawing. Annotation spanning the view and other views on the drawing are deleted. If the view does not fit on a smaller destination drawing, an error is returned.

Environment
Internal and External

See Also
UF_DRAW_copy_view
UF_DRAW_move_view
See the example

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRAW_move_view_to_drawing
(
    tag_t view_tag,
    const tag_t drawing_tag
)
```

tag_t	**view_tag**	Input	Tag of drawing member view to move. It must be on the current drawing.
const tag_t	**drawing_tag**	Input	Destination drawing tag. It must not be the current drawing.

UF_DRAW_open_drawing (view source)

Defined in: uf_draw.h

Overview
This routine opens a drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_create_drawing.c

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_open_drawing
(
    const tag_t drawing_tag
)
```

const tag_t	**drawing_tag**	Input	Tag of drawing to open.

UF_DRAW_redefine_sxline_hinge (view source)

Defined in: uf_draw.h

Overview
Redefines the section line's hinge line given the tag of a section line
and a drafting line object. The section view and any details of the
section view are reoriented about their center to reflect the newly
defined hinge line.

To perform this edit on the section line without an update, use the
suppress view updated feature provided in UF_DRF_set_suppress_view_update.

NOTE: The hinge_line.line_assoc_type must be either
UF_DRF_dwg_line, UF_DRF_existing_line or UF_DRF_two_points.
If hinge_line.line_assoc_type is UF_DRF_dwg_line, the hinge line has
no associativity and is defined by a direction vector defined from
hinge_line.object1.assoc_dwg_pos to hinge_line.point_object2.assoc_dwg_pos.

Return

Return code:
0 = No error
UF_DRAW_hinge_not_linear = warning
not 0 = Error code

Environment

Internal and External

See Also

UF_DRF_set_suppress_view_update
See the example

Required License(s)

drafting

```
int UF_DRAW_redefine_sxline_hinge
(
    tag_t sxline_tag,
    UF_DRF_line_object_p_t hinge_line,
    logical arrow_same_dir
)
```

tag_t	**sxline_tag**	Input	section line tag
UF_DRF_line_object_p_t	**hinge_line**	Input	object to associate hinge line to
logical	**arrow_same_dir**	Input	If true, the new arrow direction will point away from the new hinge line in the same relative direction as the old arrow pointed away from the old hinge line. Else, the direction is flipped.

UF_DRAW_remove_break_region (view source)

Defined in: uf_draw.h

Overview

This routine removes a break region from a given view.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_broken_view.c to review an example of using
this function.

Environment

Internal and External

History

New for V17.0

Required License(s)

drafting

```
int UF_DRAW_remove_break_region
(
    tag_t break_region,
    logical delete_curves
)
```

tag_t	**break_region**	Input	Tag of the break region to be removed
logical	**delete_curves**	Input	Whether or not to delete the visible break region boundary curves.

UF_DRAW_remove_breakout (view source)

Defined in: uf_draw.h

Overview

This routine removes a breakout section from a given view.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_cre_breakout.c to review an example of using
this function.

Environment

Internal and External

History

New for V16.0

Required License(s)

drafting

```
int UF_DRAW_remove_breakout
(
    tag_t breakline,
    logical delete_curves
)
```

tag_t	**breakline**	Input	Tag of a curve in the breakout.
logical	**delete_curves**	Input	Whether or not to delete the breakout curves.

UF_DRAW_remove_dmv_rotation_plane (view source)

Defined in: uf_draw.h

Overview
This routine removes the associativity between the plane tag and the view, and restores the view orientation.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message string associated with the error code.

Environment
Internal and External

History
New for V18.0

Required License(s)
drafting

```
int UF_DRAW_remove_dmv_rotation_plane
(
    const tag_t view
)
```

const tag_t	**view**	Input	Tag of the view

UF_DRAW_rename_drawing (view source)

Defined in: uf_draw.h

Overview
This routine renames a drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_rename_drawing.c

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_rename_drawing
(
    const tag_t drawing_tag,
    const char * new_drawing_name
)
```

const tag_t	**drawing_tag**	Input	Tag of the drawing. If NULL_TAG, use current drawing
const char *	**new_drawing_name**	Input	New name for the drawing.

UF_DRAW_retrieve_drawing_cgm (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_retrieve_drawing_cgm

DESCRIPTION -
Function will retrieve drawing cgm data stored in NX QAF.
The drawing cgm data will be stored in TMP_DIR, the file name will be
TMP_DIR/file_name-sheet_name.cgm

Returns -
0 = OK
if not 0 = error code

Required License(s)

drafting

```
int UF_DRAW_retrieve_drawing_cgm
(
    char * file_name,
    char * * * out_file_names,
    int * num_sheets
)
```

char *	**file_name**	Input	file name used to query cgm data
char * * *	**out_file_names**	Output to UF_*free*	an array of output name files
int *	**num_sheets**	Output	number of output file names

UF_DRAW_set_auto_update (view source)

Defined in: uf_draw.h

Overview

This function sets the current value of the Automatic Update
preference.

Environment

Internal and External

See Also

UF_DRAW_ask_auto_update
See the example

Required License(s)

drafting

```
int UF_DRAW_set_auto_update
(
    tag_t view_tag,
    logical * auto_update
)
```

tag_t	**view_tag**	Input	Tag of view object
logical *	**auto_update**	Output	TRUE = Automatically update the view FALSE = Do not automatically update the view

---

UF_DRAW_set_border_color (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_set_border_color

DESCRIPTION -
Set the color of view borders

PARAMETERS -
border_color <I> Color of view borders
Returns -
0 = OK
if not 0 = error code

Required License(s)

drafting

```
int UF_DRAW_set_border_color
(
    int border_color
)
```

int	**border_color**	Input

UF_DRAW_set_border_display (view source)

Defined in: uf_draw.h

Overview

UF_DRAW_set_border_display

DESCRIPTION -

Set the view border display status

PARAMETERS -
border_display <I> True, if borders are to be displayed
Returns -
0 = OK
if not 0 = error code

Required License(s)

drafting

int UF_DRAW_set_border_display
(
    logical border_display
)

logical	**border_display**	Input

UF_DRAW_set_boundary_assoc (view source)

Defined in: uf_draw.h

Overview

This routine makes the boundary associative with the model by making each defining point in the boundary a smart point which is at a fixed offset from the anchor point. The view's anchor point must be defined, and it must be a smart point which is associated with the model. This function uses the structure boundary_curves to return the tags of each boundary curve's smart defining points. For this function to successfully complete, none of the curves in the boundary can already be associative with the model. If the boundary contains a spline, that spline must be defined via defining points and cannot have tangency or curvature constraints.

Environment

Internal and External

See Also

UF_DRAW_define_view_boundary
UF_DRAW_edit_boundary_point

Return

Return code:
UF_DRAW_NO_ERRORS - The view boundary was successfully made to be associative.
UF_DRAW_tag_is_null - The view tag or the model_reference_pt tag is null.
UF_DRAW_invalid_parameter - A parameter is invalid, such as curve_count is less than or equal
to zero.
UF_DRAW_anchor_point_is_not_smart_point - The anchor point is not a smart point
associated with the
model.
UF_DRAW_sketch_object - At least one of the curves in the boundary belongs to a sketch.

UF_DRAW_curve_is_associative - At least one of the curves in the boundary is already associative
and has smart defining points.
UF_DRAW_invalid_spline - There is a spline in the boundary which violates the following rule. The spline
must be defined via defining points and
cannot have tangency or curvature
constraints.

See Also

See the example

History

Original release was in V14.0.

Required License(s)

drafting

```
int UF_DRAW_set_boundary_assoc
(
    tag_t view,
    int * curve_count,
    UF_DRAW_view_boundary_p_t * boundary_curves
)
```

tag_t	**view**	Input	Tag of the view whose boundary is to be made associative. The view must have a break line/detail boundary type. The view must have an anchor point defined.
int *	**curve_count**	Output	Pointer to count of curves in curve list
UF_DRAW_view_boundary_p_t *	**boundary_curves**	Output to UF_*free*	Pointer to curve list, i.e. array of structures containing the boundary curves and the points used to define them. Use UF_DRAW_free_boundary to free the pointer.

UF_DRAW_set_break_region_data (view source)

Defined in: uf_draw.h

Overview

This routine modifies the break region data of an existing break region.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_broken_view.c to review an example of using this function.

Environment

Internal and External

History

New for V17.0

Required License(s)

drafting

```
int UF_DRAW_set_break_region_data
(
    tag_t break_region,
    UF_DRAW_break_region_data_p_t break_region_data
)
```

tag_t	**break_region**	Input	Tag of the break region.
UF_DRAW_break_region_data_p_t	**break_region_data**	Input	Break region data.

UF_DRAW_set_breakout_data (view source)

Defined in: uf_draw.h

Overview

This routine modifies the breakout data of an existing breakout section by deleting the existing breakout section and creating a new one.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message string associated with the error code.

Please reference ufd_drw_cre_breakout.c to review an example of using this function.

Environment

Internal and External

History

New for V16.0

Required License(s)

drafting

```
int UF_DRAW_set_breakout_data
(
    tag_t breakline,
    UF_DRAW_breakout_data_p_t breakout_data,
    tag_p_t new_breakline
)
```

tag_t	**breakline**	Input	Tag of the existing breakout section.
UF_DRAW_breakout_data_p_t	**breakout_data**	Input	Breakout section data.
tag_p_t	**new_breakline**	Output	Tag of the new (modified) breakout section.

UF_DRAW_set_comp_section_in_view (view source)

Defined in: uf_draw.h

Overview

This routine sets the component of a given section view to be sectioned
or non-sectioned.

Return

0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Please reference ufd_drw_comp_section_in_view.c to review an example
of using this function.

Environment

Internal and External

History

New for V16.0

Required License(s)

drafting

```
int UF_DRAW_set_comp_section_in_view
(
    const tag_t component,
    const tag_t sx_view,
    const UF_DRAW_comp_section_in_view_t sx_property
)
```

const tag_t	**component**	Input	Tag of the component in the section view.
const tag_t	**sx_view**	Input	Tag of the section view.
const UF_DRAW_comp_section_in_view_t	**sx_property**	Input	Sectioning property UF_DRAW_NON_SECTIONED UF_DRAW_SECTIONED UF_DRAW_NOT_VIEW_SPECIFIED

UF_DRAW_set_display_state (view source)

Defined in: uf_draw.h

Overview
This routine sets the drawing display state.

Environment
Internal and External

See Also
See the example program ufd_draw_aux_view.c

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_set_display_state
(
    const int view_type
)
```

const int	**view_type**	Input	Setting Flag 1 = Modeling View 2 = Drawing View

---

UF_DRAW_set_dmv_rotation_plane (view source)

Defined in: uf_draw.h

Overview
This routine sets the plane to the drawing member view. The view will be
rotated to the plane after view update.
Either the plane or the x_vector, or both, needs to be defined.
If a tag of an xform is used for the plane, then an x_vector is not needed to fully define the
orientation.
If a tag of a plane is used for the plane, then an x_vector is needed to fully define the orientation.

Return
0 = OK
if not 0 = Failing error code
Use UF_get_fail_message to obtain the message
string associated with the error code.

Environment
Internal and External

History
New for V18.0

Required License(s)
drafting

```
int UF_DRAW_set_dmv_rotation_plane
(
    const tag_t view,
    const tag_t plane,
    const tag_t x_vector
)
```

const tag_t	**view**	Input	Tag of the view
const tag_t	**plane**	Input	Tag of the plane
const tag_t	**x_vector**	Input	Tag of the x direction

UF_DRAW_set_drawing_info (view source)

Defined in: uf_draw.h

Overview

Sets the information about the current drawing, including the size, scale, units, and projection angle. The projection angle cannot be changed if a drawing has one or more auxiliary or orthogonal views. NOTE: There is currently a restriction requiring the input drawing tag to be the tag of the current drawing. We intend to relax this restriction in the future. As a result, we are requiring the input tag to ensure that future code changes will not be required by NX or by
NX Open API developers.

Environment

Internal and External

See Also

UF_DRAW_ask_drawing_info

History

Original release was in V13.0.

Required License(s)

drafting

```
int UF_DRAW_set_drawing_info
(
    const tag_t drawing_tag,
    UF_DRAW_info_p_t drawing_info
)
```

const tag_t	**drawing_tag**	Input	Drawing Tag, must be the current drawing.
UF_DRAW_info_p_t	**drawing_info**	Input	Pointer to Drawing Info Structure

---

UF_DRAW_set_drawing_ref_pt (view source)

Defined in: uf_draw.h

Overview

This routine sets the drawing reference point for a view on the drawing.
This is the point which controls where the view is on the drawing sheet.

Environment

Internal and External

See Also

See the example program ufd_drw_set_drawing_ref_pt.c

History

New for V16.0

Required License(s)

drafting

```
int UF_DRAW_set_drawing_ref_pt
(
    const tag_t drawing_tag,
    const tag_t view_tag,
    const double reference_pt [ 2 ]
)
```

const tag_t	**drawing_tag**	Input	Tag of the drawing. If NULL_TAG, use current drawing.
const tag_t	**view_tag**	Input	Tag of the view. If NULL_TAG, use work view.
const double	**reference_pt [2]**	Input	Reference point (Drawing Coordinates). [0] - X-coordinate [1] - Y-coordinate

---

UF_DRAW_set_render_set_objects (view source)

Defined in: uf_draw.h

Overview

This routine defines the objects (solids or component sets) the given
render set will reference. The objects will replace the objects the render set
referenced prior to calling this function.

Environment

Internal and External

See Also

See the example program ufd_draw_render_set.c

History

Created in v16.0

Required License(s)

drafting

```
int UF_DRAW_set_render_set_objects
(
    tag_t render_set,
    int number_objects,
    tag_t * objects
)
```

tag_t	**render_set**	Input	Tag of render set.
int	**number_objects**	Input	Number of objects to be included in the render set. Set to zero if you want the render set to contain no objects.
tag_t *	**objects**	Input	number_objects Array of solids or component sets to be referenced by the render set. These objects will replace the objects the render set referenced prior to calling this function. Set to NULL if you want the render set to contain no objects.

UF_DRAW_set_render_set_parms (view source)

Defined in: uf_draw.h

Overview
This routine sets the display parameters for a given render set.

Environment
Internal and External

See Also
See the example program ufd_draw_render_set.c

History
Created in v16.0

Required License(s)
drafting

```
int UF_DRAW_set_render_set_parms
(
    tag_t render_set,
    UF_DRAW_render_prefs_t * render_parms
)
```

tag_t	**render_set**	Input	Tag of render set.
UF_DRAW_render_prefs_t *	**render_parms**	Input	Pointer to render set preferences structure, with desired settings for visible and hidden line color, font, and widths, edge hiding edge and hidden line options.

UF_DRAW_set_render_sets_for_view (view source)

Defined in: uf_draw.h

Overview

This routine defines the render sets to be rendered in the given drawing member view. List the render sets in the desired rendering order.

Environment

Internal and External

See Also

See the example program ufd_draw_render_set.c

History

Created in v16.0

Required License(s)

drafting

```
int UF_DRAW_set_render_sets_for_view
(
    tag_t view,
    int number_render_sets,
    tag_t * render_sets
)
```

tag_t	**view**	Input	Tag of drawing member view.
int	**number_render_sets**	Input	Number of render sets
tag_t *	**render_sets**	Input	Array of render sets to be rendered in the given view. List them in the desired rendering order.

UF_DRAW_set_suppress_view_updat (view source)

Defined in: uf_draw.h

Overview

Changes the value of the Suppress View Update preference. This preference is saved to the root part.
If no parts are loaded, an error will occur.
If the preference is TRUE, then functions which perform implicit drawing update, will not update the drawing member views.
Please note that manual views do not update automatically.
For a view to automatically update, set UF_DRAW_set_auto_update to TRUE, and set suppress_view_update to FALSE.

Note: Starting in nx2, this preference was saved with the part.
Some parts may have been inadvertently saved with suppress view updated

turned off. To allow users to override the suppress view update setting,
an environment variable "UGII_SUPPRESS_VIEW_UPDATE" can be set with
values "0" or "1". If set to 1, this will prevent an automatic
update of the out-of-date views.

Environment

Internal and External

See Also

UF_DRF_ask_suppress_view_update
UF_DRF_update_views
UF_DRF_is_object_out_of_date
See the example

Required License(s)

drafting

```
int UF_DRAW_set_suppress_view_updat
(
    logical suppress_view_update
)
```

logical	**suppress_view_update**	Input	Suppress View Update preference setting: TRUE = suppress all system initiated view updates. FALSE = allow all system initiated view updates.

UF_DRAW_set_sxline_default_prfs (view source)

Defined in: uf_draw.h

Overview

Sets the section line default display preferences, including visibility
and arrow parameters.

Environment

Internal and External

See Also

UF_DRAW_arror_parms_t
See the example

Required License(s)

drafting

```
int UF_DRAW_set_sxline_default_prfs
(
    UF_DRAW_arrow_parms_p_t arrow_parms,
    UF_DRAW_sxline_display_t sxline_display
)
```

UF_DRAW_arrow_parms_p_t	**arrow_parms**	Input	Section line arrow parameters
UF_DRAW_sxline_display_t	**sxline_display**	Input	Section line display: UF_DRAW_display_sxline = display section line.

			UF_DRAW_no_display_sxline = do not display section line.

UF_DRAW_set_sxview_display (view source)

Defined in: uf_draw.h

Overview
Sets the value of a specified section view display preference.

Environment
Internal and External

See Also
UF_DRAW_ask_sxview_display
See the example

Required License(s)
drafting

```
int UF_DRAW_set_sxview_display
(
    tag_t view_tag,
    UF_DRAW_sxview_prfs_t * sxview_parms
)
```

tag_t	**view_tag**	Input	Tag of section view object
UF_DRAW_sxview_prfs_t *	**sxview_parms**	Input	Data structure contains the section view preference parameters.

UF_DRAW_set_view_anchor (view source)

Defined in: uf_draw.h

Overview
Defines an associative view reference point that is to coincide with the view anchor point. The anchor point must be on the model, or in the view (not on the drawing sheet). An anchor point cannot be a point on a drawing member view's boundary curves. The anchor point must be a smart point. If you wish to use a "dumb point" to set the view reference point, use uc6484.

Please reference ufd_drw_set_view_anchor.c to review an example of using this function.

Environment
Internal and External

See Also
UF_DRAW_ask_view_anchor
See the example

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRAW_set_view_anchor
(
    const tag_t view_tag,
    const tag_t anchor_point
)
```

const tag_t	**view_tag**	Input	Tag of the view whose location is to be defined.
const tag_t	**anchor_point**	Input	Tag of a smart point which defines the location of the model that will coincide with the drawing reference point The point must be a smart point (see uf_so.h).

UF_DRAW_set_view_angle (view source)

Defined in: uf_draw.h

Overview
Sets the view angle to a specific value.

Environment
Internal & External

See Also
See UF_DRAW_ask_view_angle
See example

History
This function was originally released in V15.0.

Required License(s)
drafting

```
int UF_DRAW_set_view_angle
(
    tag_t view_tag,
    double angle
)
```

tag_t	**view_tag**	Input	The view tag
double	**angle**	Input	The new angle in degrees.

UF_DRAW_set_view_display (view source)

Defined in: uf_draw.h

Overview

Sets the value of a specified view display preference for a specific drawing member view.

Note: Call UF_DRAW_update_one_view() to force this view to update after calling UF_DRAW_set_view_display()

Environment

Internal and External

See Also

UF_DRAW_ask_view_display
See the example

Required License(s)

drafting

```
int UF_DRAW_set_view_display
(
    tag_t view_tag,
    UF_DRAW_view_prfs_t* view_parms
)
```

tag_t	**view_tag**	Input	Tag of drawing member view
UF_DRAW_view_prfs_t*	**view_parms**	Input	Data structure contains the hidden line removal, edge hiding edge, silhouette, uv hatch, smooth edge display, smooth edge color, font, width and gap, virtual intersection color, font, width, gap, and status, tolerance, and hidden line color, font, and width parameters.

UF_DRAW_set_view_label_parms (view source)

Defined in: uf_draw.h

Overview

This routine sets the parameters of a view label and updates the view label.

Return

0 = successful
= UF_DRAW_invalid_parameter

Environment

Internal and External

History

Created in v17.0

Required License(s)

drafting

```
int UF_DRAW_set_view_label_parms
(
    tag_t view_label_tag,
    UF_DRAW_view_label_parms_p_t view_label_parms
)
```

tag_t	**view_label_tag**	Input	View label tag OR NULL_TAG to set global preferences
UF_DRAW_view_label_parms_p_t	**view_label_parms**	Input	Structure that is filled with the view view label parameters

---

UF_DRAW_set_view_parm_scale (view source)

Defined in: uf_draw.h

Overview

Associates the expression to the view scale of drawing member views or model views. The expression does not have to be in the same part file as the view. An Expression from a different part file can be associated to the view scale as long as the part containing the expression is loaded.

Return

Return code:
0 = No error
UF_DRAW_hinge_not_linear = warning
not 0 = Error code

Environment

Internal and External

See Also

See the example

Required License(s)

drafting

```
int UF_DRAW_set_view_parm_scale
(
    tag_t view,
    tag_t exp_tag
)
```

tag_t	**view**	Input	The scale of this view is made associative to the expression.
tag_t	**exp_tag**	Input	The expression which is to be associated to the view scale.

UF_DRAW_set_view_scale (view source)

Defined in: uf_draw.h

Overview
Sets the view scale to a specific value.

Environment
Internal and External

See Also
UF_DRAW_ask_view_scale
See the example

History
This function was originally released in V15.0.

Required License(s)
drafting

```
int UF_DRAW_set_view_scale
(
    tag_t view_tag,
    double scale
)
```

tag_t	**view_tag**	Input	The view tag
double	**scale**	Input	The new scale (must be > 0.0)

UF_DRAW_set_view_status (view source)

Defined in: uf_draw.h

Overview
This routine sets the view status in the drawing.

Environment
Internal and External

See Also
See the example program ufd_drw_set_view_status.c

History
New for V16.0

Required License(s)
drafting

```
int UF_DRAW_set_view_status
(
    const tag_t view_tag,
    const UF_DRAW_view_status_t view_status
)
```

const tag_t	**view_tag**	Input	Tag of the view. If NULL_TAG, use work view.
const UF_DRAW_view_status_t	**view_status**	Input	View Status. UF_DRAW_ACTIVE_VIEW UF_DRAW_REFERENCE_VIEW

UF_DRAW_set_view_thd_app_pitch (view source)

Defined in: uf_draw.h

Overview

Sets the minimum apparent pitch for all the threads in a drawing member view. When set, any previously existing threads with an actual pitch less than the minimum value are rendered using this value instead.

Environment

Internal and External

See Also

UF_DRAW_ask_view_thd_app_pitch
UF_DRAW_ask_view_thd_meth
UF_DRAW_set_view_thd_meth

Required License(s)

drafting

```
int UF_DRAW_set_view_thd_app_pitch
(
    tag_t view,
    double app_picth
)
```

tag_t	**view**	Input	tag of drawing member view
double	**app_picth**	Input	This is a double value representing the new minimum apparent pitch for the view. This means that although a thread may have a smaller pitch, the smallest pitch which will be drawn is specified by app _pitch for visual clarity. For example, if a thread had a pitch of 0.001 inches but the minimum apparent pitch was 0.1, the thread would be drawn showing a pitch of 0.1, although the actual modelling data remains unchanged with a true pitch of 0.001 inches. the minimum apparent pitch for the view

---

UF_DRAW_set_view_thd_meth (view source)

Defined in: uf_draw.h

Overview

Accepts a view tag and a thread rendering method as input parameters. Each rendering method corresponds with an ANSI or ISO thread standard. Valid rendering methods are:
UF_DRAW_THD_METH_NONE
UF_DRAW_THD_METH_ANSI_SIMPLIFIED
UF_DRAW_THD_METH_ANSI_SCHEMATIC
UF_DRAW_THD_METH_ANSI_DETAILED
UF_DRAW_THD_METH_ISO_SIMPLIFIED
UF_DRAW_THD_METH_ISO_DETAILED
UF_DRAW_THD_METH_ESKD_SIMPLIFIED
Changing the method affects the display of all symbolic thread features rendered in the specified view.

Environment
Internal and External

See Also
UF_DRAW_ask_view_thd_app_pitch
UF_DRAW_ask_view_thd_meth
UF_DRAW_set_view_thd_app_pitch

Required License(s)
drafting

```
int UF_DRAW_set_view_thd_meth
(
    tag_t view,
    int method
)
```

tag_t	**view**	Input	tag of the drawing member view
int	**method**	Input	the thread rendering method you wish to use for this view.

UF_DRAW_upd_out_of_date_views (view source)

Defined in: uf_draw.h

Overview
This function updates those views that are out of date.

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
drafting

```
int UF_DRAW_upd_out_of_date_views
(
    tag_t drawing_tag
)
```

tag_t	**drawing_tag**	Input	If drawing_tag is the tag of a drawing, it updates all the out of date views on the drawing. If drawing_tag is a NULL_TAG, it updates all the views in the part.

UF_DRAW_update_one_view (view source)

Defined in: uf_draw.h

Overview

Updates one drawing member view on a drawing. The view update process includes updating the view bounds, resectioning section views, updating silhouettes, and updating hidden line display when applicable.
NOTE: When running this function interactively you must be operating in an application that creates, retrieves, or requires the presence of a drawing (Gateway, Drafting, Valisys, and Genconnect).

Environment

Internal and External

See Also

UF_DRAW_ask_suppress_view_updat
UF_DRAW_set_suppress_view_updat
UF_DRAW_is_object_out_of_date
See the example

Required License(s)

drafting

```
int UF_DRAW_update_one_view
(
    tag_t drawing_tag,
    tag_t view_tag
)
```

tag_t	**drawing_tag**	Input	Tag of drawing
tag_t	**view_tag**	Input	Tag of view to update