UF_DISP_activate_grid (view source)

Defined in: uf_disp.h

Overview

This causes the grid to be display immediately instead of waiting for a system determined regeneration (at more opportune time). However, trying to activate a grid in the wrong context will not work. For example, a drawing grid cannot be displayed in the Modeling application context.

product_context - either UF_DISP_SKETCH_GRID for sketcher application,
or UF_DISP_DRAWING_GRID for drawing,
or UF_DISP_SHED_GRID for True Shading display,
or UF_DISP_MODEL_GRID for others (or default).

Environment

Internal and External

See Also

UF_DISP_ask_grid_parameters
UF_DISP_set_grid_parameters
UF_DISP_deactivate_grid
UF_DISP_ask_current_grid_context

History

Originally released in V19.0

Required License(s)

gateway

```
void UF_DISP_activate_grid
(
    UF_DISP_grid_context_t product_context
)
```

UF_DISP_grid_context_t	**product_context**	Input	either UF_DISP_SKETCH_GRID or UF_DISP_DRAWING_GRID or UF_DISP_MODEL_GRID or UF_DISP_SHED_GRID

UF_DISP_add_item_to_display (view source)

Defined in: uf_disp_ugopenint.h

Overview

Adds an object to the display. You can use this routine to update the display of a new or modified object, if UF_DISP_set_display has been used to turn off the display update.

Environment

Internal

Required License(s)

gateway

```
int UF_DISP_add_item_to_display
(
    tag_t object_id
)
```

tag_t	**object_id**	Input	Object identifier of the object to be added to display

---

UF_DISP_ask_closest_color (view source)

Defined in: uf_disp.h

Overview

Determines the color in the Color Table Object (CTO) "closest" to the given color values, based on the specified Color Comparison Method. The index of the closest color is returned.
UF_DISP_CCM_EUCLIDEAN_DISTANCE - this method returns the color which is the least Euclidean distance away from the given color in the RGB color cube.
Note that the background color is not considered as a candidate.
There must be a part loaded when this function is called.

Environment

Internal

See Also

Please see the example

Required License(s)

gateway

```
int UF_DISP_ask_closest_color
(
    int clr_model,
    double clr_values [ 3 ] ,
    int clr_cmp_mtd,
    int * clr_num
)
```

int	**clr_model**	Input	The color model of the values in clr_values; the following constants are defined in uf_disp.h: UF_DISP_rgb_model UF_DISP_hsv_model UF_DISP_hls_model
double	**clr_values [3]**	Input	Three doubles representing the color, where the meaning and range of each value depends on the color model specified: rgb: clr_values[0]: 0.0 <= red <= 1.0 clr_values[1]: 0.0 <=green <= 1.0 clr_values[2]: 0.0 <= blue <= 1.0 hsv: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[1]: 0.0 <= saturation <= 1.0 clr_values[2]: 0.0 <= value <= 1.0 hls: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[2]: 0.0 <= light <= 1.0 clr_values[1]: 0.0 <= saturation <= 1.0
int	**clr_cmp_mtd**	Input	The Color Comparison Method currently only UF_DISP_CCM_EUCLIDEAN_DISTANCE is defined.
int *	**clr_num**	Output	The number of the closest color. Range: 1 .. (# color records in CTO) - 1 Note: the background color is not considered as a candidate.

UF_DISP_ask_closest_color_in_displayed_part (view source)

Defined in: uf_disp.h

Overview

Returns the index of the color in the color table of the currently displayed part that is most similar to the color indicated by the given color name symbol.

If the specified part cannot be found, then the closest color in the default color definition file (CDF) specified by the customer defaults file is returned. If no such file exists, then the closest color in the system color table is returned.

Return

Zero is returned upon successful execution.
Any other return code indicates an error.

Environment

Internal and External

See Also

UF_DISP_color_name_t
UF_DISP_ask_closest_color
UF_DISP_ask_closest_color_in_part

History

Originally released in NX3.

Required License(s)

gateway

```
int UF_DISP_ask_closest_color_in_displayed_part
(
    UF_DISP_color_name_t color_name,
    int * color_index
)
```

UF_DISP_color_name_t	**color_name**	Input	symbol for color name
int *	**color_index**	Output	color index of most similar color

UF_DISP_ask_closest_color_in_part (view source)

Defined in: uf_disp.h

Overview

Returns the index of the color in the color table of the specified part that is most similar to the color indicated by the given color name symbol. Parameter object_in_part identifies the part; the parameter can be set to the part tag or the tag of an object in the part.

If the specified part cannot be found, then the closest color in the default color definition file (CDF) specified by the customer defaults file is returned. If no such file exists, then the closest color in the system color table is returned.

Return
Zero is returned upon successful execution.
Any other return code indicates an error.

Environment
Internal and External

See Also
UF_DISP_color_name_t
UF_DISP_ask_closest_color
UF_DISP_ask_closest_color_in_displayed_part

History
Originally released in NX3.

Required License(s)
gateway

```
int UF_DISP_ask_closest_color_in_part
(
    tag_t object_in_part,
    UF_DISP_color_name_t color_name,
    int * color_index
)
```

tag_t	**object_in_part**	Input	object in part of interest
UF_DISP_color_name_t	**color_name**	Input	symbol for color name
int *	**color_index**	Output	color index of most similar color

---

UF_DISP_ask_color (view source)

Defined in: uf_disp.h

Overview
Obtains the name and values of the specified color from the Color Table object (CTO) of the Display part. The color values are given using the specified color model.
Note that the maximum value permitted for clr_num is the number of records presently in the CTO, minus 1. The number of CTO records may be obtained by calling UF_DISP_ask_color_count. There must be a part loaded when this function is called.

Environment
Internal and External

See Also
See example

Required License(s)
gateway

```
int UF_DISP_ask_color
(
    int clr_num,
    int clr_model,
    char * * clr_name,
    double clr_values [ 3 ]
)
```

int	**clr_num**	Input	The number of the color whose name and value are to be returned; Range: 0 .. (# color records in CTO) - 1 Note: 0 is for the background color
int	**clr_model**	Input	The color model of the values in clr_values: UF_DISP_rgb_model UF_DISP_hsv_model UF_DISP_hls_model
char * *	**clr_name**	Output to UF_*free*	A pointer to the name of the color. This must be freed by the caller using UF_free
double	**clr_values [3]**	Output	Three doubles representing the color, where the meaning and range of each value depends on the color model specified: rgb: clr_values[0]: 0.0 <= red <= 1.0 clr_values[1]: 0.0 <=green <= 1.0 clr_values[2]: 0.0 <= blue <= 1.0 hsv: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[1]: 0.0 <= saturation <= 1.0 clr_values[2]: 0.0 <= value <= 1.0 hls: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[2]: 0.0 <= light <= 1.0 clr_values[1]: 0.0 <= saturation <= 1.0

UF_DISP_ask_color_count (view source)

Defined in: uf_disp.h

Overview

Returns the actual number of records currently in the Color Table object of the Display part, including the record for the background color. There must be a part loaded when this function is called.

Environment

Internal and External

See Also

See example

Required License(s)

gateway

```
int UF_DISP_ask_color_count
(
    int * count
)
```

int *	**count**	Output	number of records in the color table object (includes background color)

UF_DISP_ask_current_grid_context (view source)

Defined in: uf_disp.h

Overview

This function returns the current application for task environment

return - either UF_DISP_SKETCH_GRID for sketcher application,
or UF_DISP_DRAWING_GRID for drawing,
or UF_DISP_SHED_GRID for True Shading display,
or UF_DISP_MODEL_GRID for others (or default).
or UF_DISP_NULL_GRID if no grid is setup yet

Environment
Internal and External

See Also
UF_DISP_ask_grid_parameters
UF_DISP_set_grid_parameters
UF_DISP_activate_grid
UF_DISP_deactivate_grid

History
Originally released in V19.0

Required License(s)
gateway

```
UF_DISP_grid_context_t UF_DISP_ask_current_grid_context
(
    void
)
```

UF_DISP_ask_currently_selected_material (view source)

Defined in: uf_disp.h

Overview
This function returns ERROR_OK or zero if either
a material in the Materials in Part Palette is selected or
a material in the Materials Library is selected depending which was selected lastly.
If material selected is from the Materials in Part palette, then both material_tag and
material_full_archive_name will be returned. If the material selected is from
the Materials library, then only material_full_archive_name will be returned.
The function also returns the material source of UF_DISP_material_source_t type.
This can be used to determine which of UF_DISP_ask_work_part_material_lwa_user_area_data
or UF_DISP_ask_library_material_lwa_user_area_data to call to find the
LWA (LightWork Archive) user area data.
If no material is found selected, non-zero is returned.
This function applies to materials of current renderer type.

Environment
Internal only

History
This function is created for NX604.

Required License(s)
gateway

```
int UF_DISP_ask_currently_selected_material
(
    UF_DISP_material_source_t * material_source,
    tag_t * material_tag,
    char material_full_archive_name [ MAX_FSPEC_BUFSIZE ]
)
```

UF_DISP_material_source_t *	**material_source**	Output	UF_DISP_lw_material_in_Materials_Library or UF_DISP_lw_material_in_Materials_in_Part_Palette

tag_t *	**material_tag**	Output	The tag of material that is currently selected in the Materials in Part Palette
char	**material_full_archive_name [MAX_FSPEC_BUFSIZE]**	Output	The full_arachive_name of the material currently selected in the Materials in part palette or the Materials Library

---

UF_DISP_ask_display (view source)

Defined in: uf_disp.h

Overview
Queries the display status. You can use UF_DISP_ask_display to find out the current state of the graphics display.

Environment
Internal

History
Released in V16.0

Required License(s)
gateway

```
int UF_DISP_ask_display
(
    int * display_code
)
```

int *	**display_code**	Output	display code: UF_DISP_SUPPRESS_DISPLAY- set display off UF_DISP_UNSUPPRESS_DISPLAY- set display on

---

UF_DISP_ask_display_context (view source)

Defined in: uf_disp.h

Overview
Inquires the context pointer and returns information about display, selection, fit, and attention point information in the output structure. This function can be called prior to displaying geometry, after geometry is displayed, or both. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

NOTE: All values in the inquiry structure are set to NULL or FALSE by this function except for the following:
If this routine is being called from the attention point callback the is_attn_pt_valid field is set to TRUE.
If this routine is being called from the fit callback only the view_tag is set.
If this routine is being called from the display callback, the view_tag, is_draw_open_disp, is_view_mode_valid, and view_mode fields are set.

Environment
Internal and External

See Also

UF_DISP_inquire_p_t
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

History
Modified in V17 to return the view tag

Required License(s)
gateway

int UF_DISP_ask_display_context
(
void * context,
UF_DISP_inquire_p_t inquiry
)

void *	**context**	Input	The private context pointer obtained from the callback.
UF_DISP_inquire_p_t	**inquiry**	Output	The inquiry structure.

UF_DISP_ask_drawing_display (view source)

Defined in: uf_disp.h

Overview
Returns the data that affects drawing monochrome display.
This routine MUST be called to populate the data structure, before
UF_DISP_set_drawing_display is called.

Please reference ufd_disp.c to review a sample execution of this function.

Return
UF_err_program_not_initialized

Environment
Internal and External

See Also
UF_DISP_drawing_display_data_t

History
None

Required License(s)
gateway

int UF_DISP_ask_drawing_display
(
UF_DISP_drawing_display_data_p_t drawing_display
)

UF_DISP_drawing_display_data_p_t	**drawing_display**	Output	The drawing monochrome display structure.

UF_DISP_ask_geometry_of_material (view source)

Defined in: uf_disp.h

Overview

This function returns a list of geometry object tags that are associated to the given material tag.

The object_tags are limited to these types: UF_solid_type, UF_solid_face_subtype and UF_faceted_model_type.

This function supports both types of materials.

Environment

Internal

See Also

See example

History

This function is created for NX7.5 QRM.

Required License(s)

gateway

```
int UF_DISP_ask_geometry_of_material
(
    const tag_t material_tag,
    int * object_count,
    tag_p_t * object_tags
)
```

const tag_t	**material_tag**	Input	The tag of material that is assigned to the returned objects.
int *	**object_count**	Output	Number of objects found
tag_p_t *	**object_tags**	Output to UF_*free*	A pointer to a list of geometry object tags associated to the given material.

UF_DISP_ask_grid_parameters (view source)

Defined in: uf_disp.h

Overview

Queries the information about the grid in the designated application context. The 'product_context' variable must be set.

product_context - either UF_DISP_SKETCH_GRID for sketcher application,
or UF_DISP_DRAWING_GRID for drawing,
or UF_DISP_SHED_GRID for True Shading display,
or UF_DISP_MODEL_GRID for others (or default).

Environment

Internal and External

See Also

UF_DISP_set_grid_parameters
UF_DISP_activate_grid
UF_DISP_deactivate_grid
UF_DISP_ask_current_grid_context

History

Originally released in V19.0

Required License(s)
gateway

void UF_DISP_ask_grid_parameters
(
UF_DISP_grid_context_t product_context,
UF_DISP_grid_p_t output_grid
)

UF_DISP_grid_context_t	**product_context**	Input	either UF_DISP_SKETCH_GRID or UF_DISP_DRAWING_GRID or UF_DISP_MODEL_GRID or UF_DISP_SHED_GRID
UF_DISP_grid_p_t	**output_grid**	Output	a grid structure

UF_DISP_ask_library_material_lwa_user_area_data (view source)

Defined in: uf_disp.h

Overview
This function returns the LWA user area data based on the material_full_archive_name and the specification_attribute_string_key.
If the specified data is found in the LWA user area data, ERROR_OK or 0 is returned.
This function only supports Author based materials and libraries.

Environment
Internal only

History
This function is created for NX604.

Required License(s)
gateway

int UF_DISP_ask_library_material_lwa_user_area_data
(
char * material_full_archive_name,
const char * attribute_string_key,
const char * * attribute_data
)

char *	**material_full_archive_name**	Input	the full archive_name of the material to get lwa user area data
const char *	**attribute_string_key**	Input	key to identify which data in the lwa user data area to get
const char * *	**attribute_data**	Output	return the user area data specified by the attribute_string_key

---

UF_DISP_ask_material (view source)

Defined in: uf_disp.h

Overview

This function returns the material tag and name of the material assigned to the object defined by object_tag.
The object_tag is limited to these types: UF_solid_type, UF_solid_face_subtype and UF_faceted_model_type.
This function supports both types of materials, but will only work on the materials of the current renderer type.
For example, if you are using the Author RTS renderer(vs. Iray+ RTS renderer), then this API will apply to Author type materials.

Environment
Internal and External

See Also
See example

History
This function is created for NX3 QRM.

Required License(s)
gateway

```
int UF_DISP_ask_material
(
    const tag_t object_tag,
    tag_p_t material_tag,
    char material_name [ UF_SF_MAX_MAT_NAME_BUFSIZE ]
)
```

const tag_t	**object_tag**	Input	The tag of the object.
tag_p_t	**material_tag**	Output	The tag of material that is assigned to the specified object.
char	**material_name [UF_SF_MAX_MAT_NAME_BUFSIZE]**	Output	The name of the material

---

UF_DISP_ask_model_bounds (view source)

Defined in: uf_disp.h

Overview
Returns the model bounds of the work part. Each part should contain exactly one "model bounds object" which ideally contains a 3D box in absolute space inside of which is all of the geometry of the work part. NOTE: In general, the model bounds returned by this function are NOT accurate, as NX does not usually update the model bounds as objects are created, modified, and deleted. This function is intended for use by certain NX applications such as translators which know that their application has earlier computed model bounds which are still known to be valid.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DISP_ask_model_bounds
(
    const tag_t model_bounds_obj,
    double model_bounds [ 6 ]
)
```

const tag_t	**model_bounds_obj**	Input	The tag of the Model Bounds object of the work part. Use UF_DISP_ask_model_bounds_tag to find this tag.
double	**model_bounds [6]**	Output	The model bounds of the work part. The six values are (Minimum_X, Maximum_X, Minimum_Y, Maximum_Y, Minimum_Z, Maximum_Z)

UF_DISP_ask_model_bounds_tag (view source)

Defined in: uf_disp.h

Overview
Returns the model bounds tag of the work part. Each part should contain exactly one "model bounds object" which ideally contains a 3D box in absolute space inside of which is all of the geometry of the work part.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DISP_ask_model_bounds_tag
(
    tag_t * model_bounds_object
)
```

tag_t *	**model_bounds_object**	Output	The tag of the Model Bounds object of the work part. If the work part has no model bounds object (which is an abnormal condition), NULL_TAG is returned.

UF_DISP_ask_name_display_status (view source)

Defined in: uf_disp.h

Overview
Returns the current name display setting.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DISP_ask_name_display_status
(
    int * current_status
)
```

int *	**current_status**	Output	Name Display Status: UF_DISP_NAME_DISPLAY_OFF

UF_DISP_NAME_DISPLAY_ON

UF_DISP_ask_name_view_status (view source)

Defined in: uf_disp.h

Overview

Returns the current name display view setting. The constant UF_DISP_NAMES_IN_VIEW_OF_DEFN is the view which was the work view when the object was defined.

Environment

Internal and External

Required License(s)

gateway

```
int UF_DISP_ask_name_view_status
(
    int * current_status
)
```

int *	**current_status**	Output	View Status 0 = UF_DISP_NAMES_IN_WORK_VIEW 1 = UF_DISP_NAMES_IN_VIEW_OF_DEFN

UF_DISP_ask_srfanl_params (view source)

Defined in: uf_disp.h

Overview

Reads the face analysis display parameters into the supplied UF_DISP_srfanl_data_t structure. This routine uses a pointer to the structure UF_DISP_srfanl_data_s.

Environment

Internal

Required License(s)

gateway

```
int UF_DISP_ask_srfanl_params
(
    UF_DISP_srfanl_data_t * params
)
```

UF_DISP_srfanl_data_t *	**params**	Output	Face analysis display parameters

---

UF_DISP_ask_system_parameters (view source)

Defined in: uf_disp.h

Overview
Reads system display parameters. This routine uses a pointer to the structure UF_DISP_system_params_s.

To access parameters related to shaded face edges, please use UF_VIEW_ask_shaded_edge_options instead.

Environment
Internal and External

See Also
UF_DISP_system_params_p_t

History
V12.0 light_source_angles field removed from the structure
V13.0 Added show_shaded_face_edges and hidden_shaded_face_edges to the structure
V19.0 Added fields use_face_edges_color
face_edges_color
hidden_geometry_color
random_color_displayed
random_color_object_type

Required License(s)
gateway

int UF_DISP_ask_system_parameters
(
UF_DISP_system_params_p_t system_parameters
)

UF_DISP_system_params_p_t	**system_parameters**	Output	System display parameters

---

UF_DISP_ask_texture_space_info (view source)

Defined in: uf_disp.h

Overview
This function returns the texture space information required for the specified material.
For certain material texture (e.g., logos), texture space information is required to describe how to place the texture on a body so the texture image or pattern will appear correctly on the model (orientation, position and scale). If texture space information is not found for the material (e.g., for non-textured materials, such as metals), ts_info_defined will be set to 0 and no texture space information will be returned.
This function only supports Author based materials.

Environment
Internal and External

See Also
See example

History
This function is created for NX3 QRM.

Required License(s)
gateway

```
int UF_DISP_ask_texture_space_info
(
    tag_t material_tag,
    UF_DISP_texture_space_info_t * ts_info_ptr,
    int * ts_info_defined
)
```

tag_t	**material_tag**	Input	tag of material to request texture space information for.
UF_DISP_texture_space_info_t *	**ts_info_ptr**	Output	texture space information
int *	**ts_info_defined**	Output	value of 1 is returned if texture space information is returned

---

UF_DISP_ask_work_part_material_lwa_user_area_data (view source)

Defined in: uf_disp.h

Overview

This function returns the LWA user area data based on the material tag and the specification_attribute_string_key.
If the specified data is found from the LWA user area data, ERROR_OK or 0 is returned.
This function only supports Author based materials.

Environment

Internal only

History

This function is created for NX604.

Required License(s)

gateway

```
int UF_DISP_ask_work_part_material_lwa_user_area_data
(
    tag_t material_tag,
    const char * attribute_string_key,
    const char * * attribute_data
)
```

tag_t	**material_tag**	Input	the tag of the material to get the lwa user area data
const char *	**attribute_string_key**	Input	key to identify which data in the lwa user data area to get NOTE max key length is 31
const char * *	**attribute_data**	Output	return the user area data specified by the attribute_string_key

---

UF_DISP_ask_work_plane_emphasis (view source)

Defined in: uf_disp.h

Overview

Queries the current work plane emphasis setting. When work plane emphasis is on, objects that do not lie on the work plane appear de-emphasized, in which the object color is blended with the De-emphasis Blend Color according to the De-emphasis Blend Percentage. A de-emphasized object is unselectable

unless the work plane emphasis selection filter is overridden. Work plane emphasis is a session dependent value; it is not saved with any part.

Environment
Internal and External

See Also
UF_DISP_set_work_plane_emphasis
UF_DISP_ask_work_plane_sel
UF_DISP_set_work_plane_sel
See example

History
Original release was in V13.0.

Required License(s)
gateway

int UF_DISP_ask_work_plane_emphasis
(
int * emphasis
)

int *	**emphasis**	Output	Emphasis setting. Must be one of: UF_DISP_WORK_PLANE_EMPHASIS_ON, or UF_DISP_WORK_PLANE_EMPHASIS_OFF

UF_DISP_ask_work_plane_sel (view source)

Defined in: uf_disp.h

Overview
Queries the value of the selection override for work plane emphasis.
When work plane emphasis is enabled and the selection override is equal to UF_DISP_SELECT_WORK_DIMMED, no objects are filtered out of selection because they are off the work plane.
When work plane emphasis is enabled and the selection override is equal to UF_DISP_NO_SELECT_WORK_DIMMED, no objects dimmed for work plane emphasis are selectable.
When work plane emphasis is disabled, the selection override setting has no affect although it can be changed. The selection override setting takes effect once work plane emphasis is enabled.
The work plane emphasis selection override is a session dependent value; it is not saved with any part.

Environment
Internal and External

See Also
UF_DISP_set_work_plane_emphasis
UF_DISP_ask_work_plane_emphasis
UF_DISP_set_work_plane_sel

History
Original release was in V13.0.

Required License(s)
gateway

int UF_DISP_ask_work_plane_sel
(
int * override

)

int *	**override**	Output	Emphasis selection override setting. Must be either: UF_DISP_SELECT_WORK_DIMMED, or UF_DISP_NO_SELECT_WORK_DIMMED

UF_DISP_assign_material (view source)

Defined in: uf_disp.h

Overview

This function assigns the material to object
NOTE object types accepted include: UF_solid_type, UF_solid_face_subtype and UF_faceted_model_type.
This function supports assigning of both types of materials (Iray+ and Author).

Environment

Internal and external

See Also

See example

History

This function is created for NX3 QRM.

Required License(s)

gateway

int UF_DISP_assign_material
(
const tag_t material_tag,
const tag_t object_tag
)

const tag_t	**material_tag**	Input	The tag of the material to assign to object
const tag_t	**object_tag**	Input	The tag of object to assign the material to

---

UF_DISP_batch_shade (view source)

Defined in: uf_disp.h

Overview

Creates a shaded image using the capabilities of NX Photo and saves that image as a TIFF, GIF, or JPEG file. You specify the image type by including the appropriate file extension in the save_filename parameter. Use any one of the following file extensions:
.tif - TIFF
.gif - GIF
.jpg - JPEG
For example, specifying "test.jpg" for the filename parameter creates a JPEG file.
The batch shade function shades the active view using the light sources, backgrounds, foregrounds, materials and textures that were setup using interactive NX. If the current layout is a drawing then this function will return without creating an output file.

If using a shade method of UF_DISP_high_quality, UF_DISP_preview,

UF_DISP_photo_real or UF_DISP_raytrace then a Studio Render license is required. If this license is not available then the method will default back to UF_DISP_phong.

Please note this API generates output shaded image file using Lightworks Author renderer and supports Lightworks Author materials and textures. IrayPlus materials won't be processed. Please set NX_RTS_IRAY=0 before executing NX Open utilities using this API.

Environment
Internal and External

See Also
See example

History
This function was modified in V14.0 to also produce GIF and JPEG files

Required License(s)
gateway

```
int UF_DISP_batch_shade
(
    char* filename,
    int x_size,
    int y_size,
    UF_DISP_shade_method_t method
)
```

char*	**filename**	Input	Name of output TIFF, GIF, or JPEG file
int	**x_size**	Input	X size of output in pixels
int	**y_size**	Input	Y size of output in pixels
UF_DISP_shade_method_t	**method**	Input	Type of shade to produce. Either: UF_DISP_flat, UF_DISP_gouraud, UF_DISP_phong, UF_DISP_high_quality, UF_DISP_preview, UF_DISP_photo_real, or UF_DISP_raytrace

---

UF_DISP_batch_shade_options (view source)

Defined in: uf_disp.h

Overview
Creates a shaded image using the capabilities of NX Photo and saves that image as a TIFF, GIF, or JPEG file. You specify the image type by including the appropriate file extension in the save_filename parameter. Use any one of the following file extensions:
.tif - TIFF
.gif - GIF
.jpg - JPEG
For example, specifying "test.jpg" for the filename parameter creates a JPEG file.
The batch shade function shades the active view using the light sources, backgrounds, foregrounds, materials and textures that were setup using interactive NX. If the current layout is a drawing then this function will return without creating an output file.
This function allows setting of some options for the shading process.

If using a shade method of UF_DISP_high_quality, UF_DISP_preview, UF_DISP_photo_real or UF_DISP_raytrace then a Studio Render license

is required. If this license is not available then the method will default back to UF_DISP_phong.

Please note this API generates output shaded image file using Lightworks Author renderer and supports Lightworks Author materials and textures. IrayPlus materials won't be processed. Please set NX_RTS_IRAY=0 before executing NX Open utilities using this API.

Environment
Internal and External

History
This function was first released in NX3.0.3

Required License(s)
gateway

```
int UF_DISP_batch_shade_options
(
    char* filename,
    int x_size,
    int y_size,
    UF_DISP_shade_method_t method,
    UF_DISP_shade_options_p_t options
)
```

char*	**filename**	Input	Name of output TIFF, GIF, or JPEG file
int	**x_size**	Input	X size of output in pixels
int	**y_size**	Input	Y size of output in pixels
UF_DISP_shade_method_t	**method**	Input	Type of shade to produce. Either: UF_DISP_flat, UF_DISP_gouraud, UF_DISP_phong, UF_DISP_high_quality, UF_DISP_preview, UF_DISP_photo_real, or UF_DISP_raytrace
UF_DISP_shade_options_p_t	**options**	Input	Options to control the shading process.

UF_DISP_compute_model_bounds (view source)

Defined in: uf_disp.h

Overview
Computes the model bounds for the work part. The model bounds defines a box in Absolute coordinates which contains all of the potentially displayable geometry of the part. The sides of the box are parallel to the axes of the absolute coordinate system. Objects which are not currently displayable, because they are blanked or on an invisible layer, are still included within the model bounds.
If there are no displayable objects in the part, bounds_computed will be FALSE, and model_bounds will contain default bounds, which are the size of the current graphics in X and Y, with Z the same as the smaller of X and Y.
NOTE: This function is of limited value for most NX Open API application developers. It is intended to be used by certain NX applications such as translators.

Environment
Internal and External

Required License(s)

gateway

int UF_DISP_compute_model_bounds
(
logical * bounds_computed,
double model_bounds [6]
)

logical *	**bounds_computed**	Output	TRUE if there are some displayable objects in the work part, so that normal model bounds were computed. FALSE if the work part has no displayable objects
double	**model_bounds [6]**	Output	The model bounds for the work part. This is a box in absolute space, whose sides are parallel to the axes of the absolute coordinate system. The six values of the array are the Minimum_X, Maximum_X, Minimum_Y, Maximum_Y, Minimum_Z and Maximum_Z of the box. If bounds_computed is FALSE, default bounds are returned. The default bounds are roughly the size of the current graphics window.

UF_DISP_conehead (view source)

Defined in: uf_disp_ugopenint.h

Overview
Displays a temporary conehead vector with either the base of the staff, the arrow tip or the base of the arrow head at the reference point in the specified view.

Return
void

Environment
Internal

See Also
See example

Required License(s)
gateway

void UF_DISP_conehead
(
int display_flag,
double coord [3] ,
double vector [3] ,
int anchor_flag
)

int	**display_flag**	Input	Display views to draw the conehead vector in. UF_DISP_ALL_VIEWS_BUT_DRAWING UF_DISP_VIEW_OF_LAST_CURSOR UF_DISP_ALL_ACTIVE_VIEWS UF_DISP_WORK_VIEW_ONLY >0 = View sequence number
double	**coord [3]**	Input	3D absolute coordinates of anchor.

double	**vector [3]**	Input	3D vector which gives conehead direction
int	**anchor_flag**	Input	Denotes the anchor point 0 = Anchor point at base of staff 1 = Anchor point at tip of arrowhead 2 = Anchor point at base of arrowhead

---

UF_DISP_copy_LWA_archive_material_to_work_part (view source)

Defined in: uf_disp.h

Overview
This function copies a material specified by the material_name input argument from the currently opened Lightworks LWA archive material library to the work part.
NOTE: this API will not work on CgFX materials.

Returns
0 = Success
-1 = Failed to find an opened LWA archive materials library
-2 = Failed to import named material.
Not -1,-2, 0 = Error code
Standard UF Error Codes:
UF_err_program_not_initialized
UF_err_bad_parameter_number_1: problem found with material_name argument
UF_err_part_not_loaded: need to have an opened work part.

This function only supports Author based materials and libraries(lwa).

Environment
Internal and external

See Also
See example

History
This function is created for NX8.5.3

Required License(s)
gateway

```
int UF_DISP_copy_LWA_archive_material_to_work_part
(
    const char* material_name,
    tag_p_t material_tag
)
```

const char*	**material_name**	Input	The name of material in the currently opened archive material library to copy to the work part.
tag_p_t	**material_tag**	Output	The tag of the copied material.

UF_DISP_copy_material (view source)

Defined in: uf_disp.h

Overview
This function creates a new material based on an existing material attributes defined by the given material tag.

This function supports copying of both type of materials(Iray+ and Author).

Environment
Internal and External

See Also
See example

History
This function is created for NX3 QRM.

Required License(s)
gateway

```
int UF_DISP_copy_material
(
    const tag_t material_tag,
    tag_t * new_material_tag,
    char new_material_name [ MAX_FSPEC_BUFSIZE ]
)
```

const tag_t	**material_tag**	Input	The tag of material that the new material is copied from.
tag_t *	**new_material_tag**	Output	new material tag
char	**new_material_name [MAX_FSPEC_BUFSIZE]**	Output	The name of the newly copied material

---

UF_DISP_create_animation (view source)

Defined in: uf_disp.h

Overview
Allows the batch creation of an MPEG movie based on an existing animation saved in the opened part. An Animation can be created using the Create Animation Dialog. See the NX Gateway Online help for details on creating an animation name and frames.

Note that only animations of "Define Key Frames" type are supported. The Default animation is "Define Trajectory Curves" type. To create a "Define Key Frames" type animation, the user needs to select "Define Key Frames" prior to selecting Add/Copy when creating a new animation.

Returns
0 = Success
-1 = Unable to find data for "animation_name"
-2 = General error creating MPEG data
not -1,-2, 0 = Error code
Standard UF Error Codes

Environment
Internal and External

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_DISP_create_animation
(
```

```
    char * filename,
    char * animation_name,
    int first_frame,
    int last_frame
)
```

char *	**filename**	Input	Name of output file for MPEG data
char *	**animation_name**	Input	Name of a Key Frame camera pan in the part
int	**first_frame**	Input	Initial frame to generate (usually 0)
int	**last_frame**	Input	Final frame to generate

UF_DISP_create_framed_image (view source)

Defined in: uf_disp_ugopenint.h

Overview

Creates an image file with the file type and rectangular frame you specify using the image currently displayed in the graphics window.
Export a PNG/JPEG/TIFF/GIF/XWD/BMP full window or rectangular area image
Full window image: set width and height to zero
Rectangular area image: set width, height greater than zero and set upper left point window coordinate

Return

Return code:
= 0 No error
= not 0 Error code, including:
UF_DISP_err_failed_to_create_image_file
Standard UF error codes

Environment

Internal

History

Original release was in V15.0.

Required License(s)

gateway

```
int UF_DISP_create_framed_image
(
    char* filename,
    UF_DISP_image_format_t format,
    UF_DISP_background_color_t color,
    int upper_left_corner [ 2 ] ,
    int width,
    int height
)
```

char*	**filename**	Input	Name of output image file
UF_DISP_image_format_t	**format**	Input	Image type to produce: UF_DISP_PNG UF_DISP_JPEG UF_DISP_TIFF UF_DISP_COMPRESSED_TIFF UF_DISP_GIF UF_DISP_XWD (Only on UNIX workstations) UF_DISP_BMP (Only on Windows workstations)

UF_DISP_background_color_t	**color**	Input	Back ground color: UF_DISP_ORIGINAL - original background color UF_DISP_WHITE - white background color.
int	**upper_left_corner [2]**	Input	The Image's upper left corner in the window. Set both x and y to 0 (zero) for a full screen image. upper_left_corner[0] = x position upper_left_corner[1] = y position
int	**width**	Input	Image width in pixels. Set to 0 (zero) for full screen image.
int	**height**	Input	Image height in pixels. Set to 0 (zero) for full screen image.

UF_DISP_create_image (view source)

Defined in: uf_disp_ugopenint.h

Overview
Export a PNG/JPEG/TIFF/GIF/XWD/BMP full window area image
Creates an image file with the file type you specify using the image currently displayed in the graphics window.

Return
Return code:
= 0 No error
= not 0 Error code, including:
UF_DISP_err_failed_to_create_image_file
Standard UF error codes

Environment
Internal

See Also
See example

History
Original release was in V13.0.

Required License(s)
gateway

```
int UF_DISP_create_image
(
    char* filename,
    UF_DISP_image_format_t format,
    UF_DISP_background_color_t color
)
```

char*	**filename**	Input	Name of output image file
UF_DISP_image_format_t	**format**	Input	Image type to produce: UF_DISP_PNG UF_DISP_JPEG UF_DISP_TIFF UF_DISP_COMPRESSED_TIFF UF_DISP_GIF UF_DISP_XWD (Only on UNIX workstations) UF_DISP_BMP (Only on Windows workstations)

UF_DISP_background_color_t	**color**	Input	Back ground color: UF_DISP_ORIGINAL UF_DISP_WHITE

UF_DISP_deactivate_grid (view source)

Defined in: uf_disp.h

Overview

This is mainly called when existing from a 'task' environment (eg. Sketcher) This causes the grid to be removed from display. In general, if switching from one application to another this call is not necessary because the DSS subsystem manages all this. However, in a task environment, existing the environment doesn't always cause display regeneration. Therefore, it is a good 'house keeping' practice to call this when another grid is expected to be restored when existing from the current task environment.

product_context - either UF_DISP_SKETCH_GRID for sketcher application,
or UF_DISP_DRAWING_GRID for drawing,
or UF_DISP_SHED_GRID for True Shading display,
or UF_DISP_MODEL_GRID for others (or default).
restore_prev_context_grid - TRUE to assure a proper display regeneration
FALSE, if you don't care

Environment

Internal and External

See Also

UF_DISP_ask_grid_parameters
UF_DISP_set_grid_parameters
UF_DISP_activate_grid
UF_DISP_ask_current_grid_context

History

Originally released in V19.0

Required License(s)

gateway

```
void UF_DISP_deactivate_grid
(
    UF_DISP_grid_context_t product_context,
    logical restore_prev_context_grid
)
```

UF_DISP_grid_context_t	**product_context**	Input	either UF_DISP_SKETCH_GRID or UF_DISP_DRAWING_GRID or UF_DISP_MODEL_GRID or UF_DISP_SHED_GRID
logical	**restore_prev_context_grid**	Input	either TRUE or FALSE

UF_DISP_delete_material (view source)

Defined in: uf_disp.h

Overview
This function deletes the material specified by the material_tag.
This function supports deletion of both types of materials (Iray+ and Author).

Environment
Internal and external

See Also
See example

History
This function is created for NX3 QRM.

Required License(s)
gateway

```
int UF_DISP_delete_material
(
    const tag_t material_tag
)
```

const tag_t	**material_tag**	Input	The tag of material to be deleted

---

UF_DISP_display_arc (view source)

Defined in: uf_disp.h

Overview
Displays a User Defined Object arc. The arc is drawn from the start angle to the end angle in a counter clockwise manner. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

Environment
Internal and External

See Also
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)
gateway

```
int UF_DISP_display_arc
(
    double matrix [ 9 ] ,
    double start_angle,
    double end_angle,
    double arc_center [ 3 ] ,
    double radius,
    void * context
)
```

double	**matrix [9]**	Input	A matrix for the CSYS in which the arc exists. Use UF_MTX3_initialize to create a matrix from X and Y vectors
double	**start_angle**	Input	Start angle expressed in radians

double	**end_angle**	Input	End angle expressed in radians. The start angle must be less than the end angle and the difference between the two must be less than 2 PI
double	**arc_center [3]**	Input	Coordinates in arc coordinates system CSYS
double	**radius**	Input	Radius from the arc_center
void *	**context**	Input / Output	A private context pointer obtained from the callback

---

UF_DISP_display_circle (view source)

Defined in: uf_disp.h

Overview

Displays a User Defined Object circle which can be solid filled. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

Environment

Internal and External

See Also

UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)

gateway

```
int UF_DISP_display_circle
(
    double matrix [ 9 ] ,
    double circle_center [ 3 ] ,
    double radius,
    logical filled,
    void * context
)
```

double	**matrix [9]**	Input	A matrix for the CSYS in which the circle exists. Use UF_MTX3_initialize to create a matrix from X and Y vectors.
double	**circle_center [3]**	Input	Circle's center coordinates (x, y, and z) from the CSYS in which the circle exists (see the matrix[9] argument).
double	**radius**	Input	Radius from the circle_center
logical	**filled**	Input	TRUE = Solid fill the circle's interior FALSE = Circle's interior not filled
void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_display_facets (view source)

Defined in: uf_disp.h

Overview

Displays User Defined Object facets. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

Environment

Internal and External

See Also

UF_DISP_facet_p_t
UF_DISP_facet_type_t
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)

gateway

```
int UF_DISP_display_facets
(
    UF_DISP_facet_p_t facets,
    int num_vertices,
    int num_facets,
    UF_DISP_facet_type_t type_of_facet,
    void * context
)
```

UF_DISP_facet_p_t	**facets**	Input	An array of facets.
int	**num_vertices**	Input	The number of vertices in one facet.
int	**num_facets**	Input	The number of facets in the facet array.
UF_DISP_facet_type_t	**type_of_facet**	Input	The format of the facet in the facet array.
void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_display_ogp_arc (view source)

Defined in: uf_disp_ugopenint.h

Overview

This function defines an arc to be rendered in the specified view. The "ogp" in the name indicates that the arc will be drawn as an Overlay Graphics primitive (please see the Overview to this section); as such, it neither creates, nor is directly associated with, an NX object.
The arc is drawn in a positive angular direction ("counterclockwise") from the start angle to the end angle.
The orientation matrix must be ortho-normal. (If necessary, use UF_MTX3_initialize to create the matrix from X and Y vectors.)
This function returns an error if the absolute value of the difference between the start and end angle is greater than two pi (plus the system tolerance). An error is also returned if the start angle is greater than the end angle.
UF_DISP_display_ogp_ functions are intended to be used

exclusively from motion callback functions. Please see the discussion and example provided with UF_UI_specify_screen_position.

Environment
Internal and External

See Also
See example

Required License(s)
gateway

```
int UF_DISP_display_ogp_arc
(
    tag_t view_tag,
    double orientation [ 9 ] ,
    double start_angle,
    double end_angle,
    double center [ 3 ] ,
    double radius
)
```

tag_t	**view_tag**	Input	Tag of the view in which the arc is to be rendered
double	**orientation [9]**	Input	The rotation matrix of the arc (relative to the Work Part Absolute Coordinate System). The matrix must be ortho-normal. If necessary, use UF_MTX3_initialize to create the matrix from X and Y vectors.
double	**start_angle**	Input	Start angle in radians; the zero degree vector is the positive x-axis of the space described by the orientation matrix.
double	**end_angle**	Input	End angle in radians; the zero degree vector is the positive x-axis of the space described by the orientation matrix.
double	**center [3]**	Input	The arc center in the coordinate system of the arc (as defined by "orientation" above).
double	**radius**	Input	The radius of the arc (Work Part Absolute units)

UF_DISP_display_ogp_circle (view source)

Defined in: uf_disp_ugopenint.h

Overview
This function defines a circle to be rendered in the specified view. The "ogp" in the name indicates that the arc will be drawn as an Overlay Graphics primitive (please see the Overview to this section); as such, it neither creates, nor is directly associated with, an NX object. The orientation matrix must be ortho-normal. (If necessary, use UF_MTX3_initialize to create the matrix from X and Y vectors.) UF_DISP_display_ogp_ functions are intended to be used exclusively from motion callback functions. Please see the discussion and example provided with UF_UI_specify_screen_position.

Environment
Internal and External

See Also

See example

Required License(s)
gateway

int UF_DISP_display_ogp_circle
(
tag_t view_tag,
double orientation [9] ,
double center [3] ,
double radius
)

tag_t	**view_tag**	Input	Tag of the view in which the circle is to be rendered
double	**orientation [9]**	Input	The rotation matrix of the circle (relative to the Work Part Absolute Coordinate System). The matrix must be ortho-normal. If necessary, use UF_MTX3_initialize to create the matrix from X and Y vectors.
double	**center [3]**	Input	The circle center in the coordinate system of the arc (as defined by "orientation" above).
double	**radius**	Input	The radius of the circle (Work Part Absolute units)

UF_DISP_display_ogp_line (view source)

Defined in: uf_disp_ugopenint.h

Overview
This function defines a simple line to be rendered in the specified view. The "ogp" in the name indicates that the line is drawn as an Overlay Graphics primitive (please see the Overview to this section); as such, it neither creates, nor is directly associated with, an NX object. UF_DISP_display_ogp_ functions are intended to be used exclusively from motion callback functions. Please see the discussion and example provided with UF_UI_specify_screen_position.

Environment
Internal Only

See Also
See example

Required License(s)
gateway

int UF_DISP_display_ogp_line
(
tag_t view_tag,
double pos1 [3] ,
double pos2 [3]
)

tag_t	**view_tag**	Input	Tag of the view in which the line is to be rendered

double	**pos1 [3]**	Input	1st endpoint (Work Part Abs Coords)
double	**pos2 [3]**	Input	2nd endpoint (Work Part Abs Coords)

UF_DISP_display_ogp_polyline (view source)

Defined in: uf_disp_ugopenint.h

Overview

This function defines a polyline to be rendered in the specified view. The "ogp" in the name indicates that the polyline will be drawn as an Overlay Graphics primitive (please see the Overview to this section); as such, it neither creates, nor is directly associated with, an NX object. UF_DISP_display_ogp_ functions are intended to be used exclusively from motion callback functions. Please see the discussion and example provided with UF_UI_specify_screen_position.

Environment

Internal and External

See Also

See example

Required License(s)

gateway

```
int UF_DISP_display_ogp_polyline
(
    tag_t view_tag,
    double pos_array [ ] [ 3 ] ,
    int pos_count
)
```

tag_t	**view_tag**	Input	Tag of the view in which the polyline is to be rendered
double	**pos_array [ ] [3]**	Input	Array of positions [pos_count][3] each position is an (x,y,z) triplet in Work Part Absolute Coordinates.
int	**pos_count**	Input	count of positions in pos_array

UF_DISP_display_points (view source)

Defined in: uf_disp.h

Overview

Displays User Defined Object points. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

Environment

Internal and External

See Also

UF_DISP_poly_marker_t
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb

UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)
gateway

int UF_DISP_display_points
(
double * points,
int num_points,
UF_DISP_poly_marker_t marker_type,
void * context
)

double *	**points**	Input	num_points An array of point coordinates.
int	**num_points**	Input	The number of points in the points array.
UF_DISP_poly_marker_t	**marker_type**	Input	The type of marker to be displayed.
void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_display_polygon (view source)

Defined in: uf_disp.h

Overview
Displays a closed User Defined Object polygon and may be solid filled. The polygon must be planar and convex when using the filled option. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.
NOTE: A polygon is convex if a line joining any two interior points of the polygon lies completely inside the polygon

Environment
Internal and External

See Also
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)
gateway

int UF_DISP_display_polygon
(
double * points,
int num_points,
logical filled,
void * context
)

double *	**points**	Input	num_points An array of coordinates resulting in a closed polygon.

int	**num_points**	Input	The number of points in the points array.
logical	**filled**	Input	TRUE = Solid fill polygon's interior. FALSE = Do not fill polygon's interior
void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_display_polyline (view source)

Defined in: uf_disp.h

Overview

Displays a User Defined Object polyline which is a connected set of line segments. This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.

Environment

Internal and External

See Also

UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)

gateway

```
int UF_DISP_display_polyline
(
    double * poly_points,
    int num_points,
    void * context
)
```

double *	**poly_points**	Input	An array of absolute coordinates which represent a connected set of line segments.
int	**num_points**	Input	The number of points in the poly_points array.
void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_display_rpo_dimensions (view source)

Defined in: uf_disp_ugopenint.h

Overview

Displays all or some rpo dimensions which belong to the specified feature. The dimensions are specified indirectly by an array of expression tags. If the count of the array is -1 then the array is ignored and all rpo dimensions of the feature are displayed. If the count is 0 then nothing is displayed. The display is temporary and can be removed by a display refresh operation. The background color can also be used to erase the dimensions but it may leave gaps on the

display. There must be a part loaded when this function is called. Any specified expression which does not tie directly to a dimension of the specified feature is considered invalid.

Environment
Internal

Required License(s)
gateway

```
int UF_DISP_display_rpo_dimensions
(
    tag_t feature_tag,
    int exp_count,
    tag_t* exp_tags,
    int view_option,
    int color_option,
    int color
)
```

tag_t	**feature_tag**	Input	Tag of the feature to which the dimensions belong
int	**exp_count**	Input	>=0 = Number of expression tags in exp_tags array -1 = all rpo dimensions of the feature
tag_t*	**exp_tags**	Input	exp_count Array of expression tags of which the dimensions need to be displayed.(is ignored if count <=0)
int	**view_option**	Input	Option for the view, the constants are in uf_disp.h: UF_DISP_ALL_VIEWS_BUT_DRAWING UF_DISP_VIEW_OF_LAST_CURSOR UF_DISP_ALL_ACTIVE_VIEWS UF_DISP_WORK_VIEW_ONLY
int	**color_option**	Input	Option for the color, the constants are in uf_disp.h: UF_DISP_USE_SYSTEM_COLOR UF_DISP_USE_BACKGROUND_COLOR UF_DISP_USE_ORIGINAL_COLOR UF_DISP_USE_SPECIFIED_COLOR
int	**color**	Input	Color used to display the dimensions if color_option= UF_DISP_USE_SPECIFIED_COLOR. Use the color constants listed in uf_obj.h.

UF_DISP_display_sket_dimensions (view source)

Defined in: uf_disp_ugopenint.h

Overview
Displays all or some sketch dimensions which belong to a specified sketch. The dimensions are specified indirectly by an array of expression tags. If the count of the array is -1 then the array is ignored and all dimensions of the sketch are displayed. If the count is 0 then nothing is displayed. The display is temporary and can be removed by a display refresh operation. The background color can also be used to erase the dimensions but it may leave gaps on the

display. There must be a part loaded when this function is called. Any specified expression which does not tie directly to a dimension of the specified feature is considered invalid.

Environment
Internal

Required License(s)
gateway

int UF_DISP_display_sket_dimensions
(
tag_t sketch_tag,
int exp_count,
tag_t* exp_tags,
int view_option,
int color_option,
int color
)

tag_t	**sketch_tag**	Input	Tag of the sketch to which the dimensions belong
int	**exp_count**	Input	>= 0 = Number of expression tags in exp_tags array -1 = all dimensions of the sketch
tag_t*	**exp_tags**	Input	exp_count Array of expression tags of which the dimensions need to be displayed. (is ignored if count <= 0)
int	**view_option**	Input	Option for the view, the constants are in uf_disp.h: UF_DISP_ALL_VIEWS_BUT_DRAWING UF_DISP_VIEW_OF_LAST_CURSOR UF_DISP_ALL_ACTIVE_VIEWS UF_DISP_WORK_VIEW_ONLY
int	**color_option**	Input	Option for the color, the constants are in uf_disp.h: UF_DISP_USE_SYSTEM_COLOR UF_DISP_USE_BACKGROUND_COLOR UF_DISP_USE_ORIGINAL_COLOR UF_DISP_USE_SPECIFIED_COLOR
int	**color**	Input	Color used to display the dimensions if color_option= UF_DISP_USE_SPECIFIED_COLOR. Use the color constants listed in uf_obj.h.

UF_DISP_display_temporary_arc (view source)

Defined in: uf_disp.h

Overview
Displays a temporary arc. A display refresh erases temporary geometry. The start angle must be less than the end angle and the absolute value of the difference must be less than two pi.

Environment
Internal

See Also
UF_OBJ_disp_props_t
UF_DISP_view_type_t

Required License(s)
gateway

```
int UF_DISP_display_temporary_arc
(
    tag_t view_tag,
    UF_DISP_view_type_t which_views,
    double matrix [ 9 ] ,
    double start_angle,
    double end_angle,
    double arc_center [ 3 ] ,
    double radius,
    UF_OBJ_disp_props_t * attrib
)
```

tag_t	**view_tag**	Input	The tag of the view in which to display the temporary arc.
UF_DISP_view_type_t	**which_views**	Input	The view mode to use.
double	**matrix [9]**	Input	A matrix for the CSYS in which the arc exists. Use UF_MTX3_initialize to create a matrix from X and Y vectors.
double	**start_angle**	Input	Start angle in radians
double	**end_angle**	Input	End angle in radians.
double	**arc_center [3]**	Input	Coordinates are with respect to the CSYS of the matrix in the work part.
double	**radius**	Input	Radius from the arc center.
UF_OBJ_disp_props_t *	**attrib**	Input	The color, font, and width elements of the structure should be set to: attrib->color = color index attrib->font = 1 for solid 2-7, user defined 0 use system default attrib->line_width = line width values in uf_obj.h -1 use the default width

---

UF_DISP_display_temporary_line (view source)

Defined in: uf_disp.h

Overview
Displays a temporary line. A display refresh erases temporary geometry.

Environment
Internal

See Also
UF_OBJ_disp_props_t
UF_DISP_view_type_t

Required License(s)
gateway

int UF_DISP_display_temporary_line
(
tag_t view_tag,
UF_DISP_view_type_t which_views,
double start_line [3] ,
double end_line [3] ,
UF_OBJ_disp_props_t * attrib
)

tag_t	**view_tag**	Input	The tag of the view in which the temporary line is to display.
UF_DISP_view_type_t	**which_views**	Input	The view mode to use
double	**start_line [3]**	Input	The absolute coordinates of the start point of the line.
double	**end_line [3]**	Input	The absolute coordinates of the work part of the end point of the line.
UF_OBJ_disp_props_t *	**attrib**	Input	The color, font, and width elements of the structure should be set to: attrib->color = color index attrib->font = 1 for solid 2-7, user defined 0 use system default attrib->line_width = line width values in uf_obj.h -1 use the default width

---

UF_DISP_display_temporary_point (view source)

Defined in: uf_disp.h

Overview
Displays a temporary marker. A display refresh erases temporary geometry.

Environment
Internal

See Also
color constants
UF_OBJ_disp_props_t
UF_DISP_poly_marker_t
UF_DISP_view_type_t

Required License(s)
gateway

int UF_DISP_display_temporary_point
(
tag_t view_tag,
UF_DISP_view_type_t which_views,
double markerpos [3] ,
UF_OBJ_disp_props_t * color,
UF_DISP_poly_marker_t marker_type
)

tag_t	**view_tag**	Input	The tag of the view in which to display the temporary maker.
UF_DISP_view_type_t	**which_views**	Input	The view mode to use.
double	**markerpos [3]**	Input	The marker position in absolute coordinates of the work part
UF_OBJ_disp_props_t *	**color**	Input	The color element of the structure. All other elements of the structure are ignored. If color->color = 0, then the system default color is used.
UF_DISP_poly_marker_t	**marker_type**	Input	The marker type

UF_DISP_display_temporary_text (view source)

Defined in: uf_disp.h

Overview

Displays temporary text. A display refresh erases temporary text. The text is entered in an array and each newline is specified by entering the '\n' character in the text. The text must be null terminated. The text is displayed parallel to the screen. The size of the text is in the given size (or the system default size) on the screen, and remains at the same size regardless of the view scale.

Environment

Internal

See Also

UF_OBJ_disp_props_t
color constants
UF_DISP_view_type_t

UF_DISP_text_ref_t

History

V18.0 Add comments.

Required License(s)

gateway

```
int UF_DISP_display_temporary_text
(
    tag_t view_tag,
    UF_DISP_view_type_t which_views,
    char text [ ] ,
    double text_coord [ 3 ] ,
    UF_DISP_text_ref_t ref_point,
    UF_OBJ_disp_props_t * color,
    double char_size,
    int hardware
)
```

tag_t	**view_tag**	Input	The tag of the view in which to display the temporary text.
UF_DISP_view_type_t	**which_views**	Input	The view mode to use.
char	**text [ ]**	Input	Lines of text

double	**text_coord [3]**	Input	Position of text box reference point in absolute coordinates of the displayed part.
UF_DISP_text_ref_t	**ref_point**	Input	Reference point of text box
UF_OBJ_disp_props_t *	**color**	Input	The color element of the structure. All other elements of the structure are ignored. If color->color = 0, then the system default color is used.
double	**char_size**	Input	The character size in part units (Metric or inches). Used only if software text is used. If <= 0, then uses system default.
int	**hardware**	Input	hardware/software flag: 0 = Use hardware font 1 = Use software font

---

UF_DISP_display_text (view source)

Defined in: uf_disp.h

Overview

Displays User Defined Object text which rotates with the view. The text is entered in an array and each newline is specified by entering the '\n' character in the text. The text must be null terminated. For example, "Hello \n World!". This routine may only be called from a UDO registered callback routine. The context argument passed into this routine will be passed from NX to the UDO callback routine.
The text is drawn using the character size found in the Annotations Preference dialog, under Lettering, General. This size may be changed by UF_DRF_set_preferences, setting mpr[4] to the desired size.

Environment

Internal and External

See Also

UF_DISP_text_ref_t
UF_UDOBJ_register_display_cb
UF_UDOBJ_register_fit_cb
UF_UDOBJ_register_attn_pt_cb
UF_UDOBJ_register_select_cb

Required License(s)

gateway

```
int UF_DISP_display_text
(
    char text [ ] ,
    double text_coord [ 3 ] ,
    UF_DISP_text_ref_t ref_point,
    void * context
)
```

char	**text [ ]**	Input	Lines of text
double	**text_coord [3]**	Input	Position of text box reference point in absolute coordinates.
UF_DISP_text_ref_t	**ref_point**	Input	Reference point of text box

void *	**context**	Input / Output	A private context pointer obtained from the callback.

UF_DISP_export_windows_metafile (view source)

Defined in: uf_disp.h

Overview

Creates a Windows Enhanced Metafile of the image in the NX graphics window. This function is available only on Windows NT. Calling it on any other platform will result in an error being returned. There are options to export the Enhanced Metafile to the clipboard or to a device context. This is similar to exporting a plot file or a CGM file of the current display. Any views which are in Shaded, Partially Shaded, or in Face Analysis mode are treated as if they were in Wireframe mode.
The Enhanced Metafile will be created using part colors and a single (standard thin) line width value.
The size of the image in the generated Enhanced Metafile is the same as that of the current graphics window. Therefore, you may wish to resize the graphics window to the desired size prior to invoking this function. When in External mode, the size of the graphics window is the size the last time the part was saved.
The output Enhanced Metafile does not include the Work Coordinate System, the grid, nor any temporary display.
You can only use the UF_DISP_WMF_TO_CLIPBOARD option when in Internal mode. If you use this option in External mode, this function returns an error code.

Environment

Internal and External

History

Original release was in V14.0.

Required License(s)

gateway

```
int UF_DISP_export_windows_metafile
(
    UF_DISP_wmf_output_t output_type,
    char * file_spec
)
```

UF_DISP_wmf_output_t	**output_type**	Input	Specifies where the output Enhanced Metafile should be written: UF_DISP_WMF_TO_CLIPBOARD writes directly to the Windows clipboard. This is not available in external NX Open programs. UF_DISP_WMF_TO_FILE uses the following file_spec to output the metafile data to.
char *	**file_spec**	Input	The file specification of the file to create. This parameter is used only when the output_type is UF_DISP_WMF_TO_FILE. This file specification must be at most MAX_FSPEC_SIZE characters.

UF_DISP_get_conehead_attrb (view source)

Defined in: uf_disp_ugopenint.h

Overview
Gets the current attribute settings with which the conehead vector displays by a call to UF_DISP_conehead. You can then obtain the values of the fields of the UF_DISP_conehead_attrb_s structure (see the example below).

Return
void

Environment
Internal

See Also
See example

Required License(s)
gateway

```
void UF_DISP_get_conehead_attrb
(
    UF_DISP_conehead_attrb_s * attributes
)
```

UF_DISP_conehead_attrb_s *	**attributes**	Output	Pointer to the attributes structure.

UF_DISP_j3d_free_geometry (view source)

Defined in: uf_disp.h

Overview
Cleans up the data returned from UF_DISP_j3d_geometry

Environment
Internal and External

See Also
None

History
None

Required License(s)
gateway

```
int UF_DISP_j3d_free_geometry
(
    int num_entities,
    UF_DISP_j3d_entity_p_t entity_list
)
```

int	**num_entities**	Input
UF_DISP_j3d_entity_p_t	**entity_list**	Input

UF_DISP_j3d_geometry (view source)

Defined in: uf_disp.h

Overview

Creates polygon and vector data for the geometry of a part file. The output polygon data is triangle strips for polygon data and vectors for curves.

wireframe - a boolean value where if the value is 1, then solids and sheets will be returned as wireframe vector data for the edges of the body. If the value is 0, then solids and sheets will be returned as triangle strip polygon data.
num_entities - returns the number of elements in the entity_list
entitiy_list - An array of display data structures

Environment

Internal and External

See Also

None

History

None

Required License(s)

gateway

```
int UF_DISP_j3d_geometry
(
    int wireframe,
    int * num_entities,
    UF_DISP_j3d_entity_p_t * entity_list
)
```

int	**wireframe**	Input
int *	**num_entities**	Output
UF_DISP_j3d_entity_p_t *	**entity_list**	Output to UF_*free*

UF_DISP_labeled_conehead (view source)

Defined in: uf_disp_ugopenint.h

Overview

Displays a temporary, labeled conehead vector with either the base of the staff, the arrow tip or the base of the arrowhead at the reference point in the specified view(s). A label will be placed at the cone's tip.

Environment

Internal

Return

void

Required License(s)

gateway

```
void UF_DISP_labeled_conehead
(
    int display_flag,
    double coord [ 3 ] ,
    double vector [ 3 ] ,
    int anchor_flag,
    char * label
)
```

int	**display_flag**	Input	Display views to draw the conehead vector in. UF_DISP_ALL_VIEWS_BUT_DRAWING UF_DISP_VIEW_OF_LAST_CURSOR UF_DISP_ALL_ACTIVE_VIEWS UF_DISP_WORK_VIEW_ONLY >0 = View sequence number
double	**coord [3]**	Input	3D absolute coordinates of anchor or reference point in absolute space.
double	**vector [3]**	Input	3D vector which gives conehead direction in absolute space.
int	**anchor_flag**	Input	Denotes the anchor point's location: 0 = Anchor point at base of staff 1 = Anchor point at tip of arrowhead 2 = Anchor point at base of arrowhead
char *	**label**	Input	Labels the conehead at its tip.

UF_DISP_load_color_table (view source)

Defined in: uf_disp_ugopenint.h

Overview

Downloads the color table to the graphics display, so that all geometry displayed in the graphics window will now use the current contents of the Color Table object (of the Display part); this is equivalent to selecting Apply from the color palette dialog in interactive NX. There must be a part loaded before calling this function.

Environment

Internal

See Also

Please see the example

Required License(s)

gateway

```
int UF_DISP_load_color_table
(
    void
)
```

UF_DISP_make_display_up_to_date (view source)

Defined in: uf_disp.h

Overview
Ensures that previously invoked display operations are complete.

Function UF_DISP_make_display_up_to_date should be called only by NX Open API programs that directly invoke Windows, MFC, or Motif functions to display dialogs. This function is unneeded when NX Open dialogs are used.

When NX Open functions perform display operations, the display operations can be buffered. NX Open dialog logic ensures that the buffered operations are completed at appropriate times. If an NX Open API program bypasses NX Open dialog logic, the program might need to explicitly invoke UF_DISP_make_display_up_to_date to complete buffered operations. The function should be called immediately before invoking Windows, MFC, or Motif functions to display dialogs.

Return
Zero is returned upon successful execution.
Any other return code indicates an error.

Environment
Internal

History
Originally released in V17.0.2

Required License(s)
gateway

```
int UF_DISP_make_display_up_to_date
(
    void
)
```

UF_DISP_open_LWA_archive_materials_library (view source)

Defined in: uf_disp.h

Overview
This function opens the Lightworks LWA archive material library specified by the input argument lwa_material_library_name

Returns
0 = Success
-1 = Failed to initialize Lightworks session_manager
-2 = Failed to open the specified lwa material archive library file
not -1,-2, 0 = Error code
Standard UF Error Codes
UF_err_program_not_initialized
UF_err_bad_parameter_number_1: problem found with lwa_archive_library_name , file does not exist.
UF_err_part_not_loaded: need to have an opened work part.
This function only supports Author based materials and libraries(lwa).

Environment
Internal and external

See Also
See example

History
This function is created for NX8.5.3

Required License(s)
gateway

int UF_DISP_open_LWA_archive_materials_library
(
const char* lwa_archive_library_name
)

const char*	**lwa_archive_library_name**	Input	The name including full path of the LWA archive material library to be opened.

UF_DISP_print_window_ug_image (view source)

Defined in: uf_disp_ugopenint.h

Overview
Prints the contents of the graphics window to the default printer.
This function is available only on Windows workstations and returns an error if called on a non-Windows workstation.

Environment
Internal

History
Originally released in V16.0.
Second parameter redefined in V18.0.

Required License(s)
gateway

int UF_DISP_print_window_ug_image
(
int format,
int color_usage
)

int	**format**	Input	Specifies the format of the printed image. 0: A vector-format image is printed. 1: A raster-format (bitmap) image is printed. If only wireframe views are displayed, then a vector-format image is printed even if the format parameter is set to 1.
int	**color_usage**	Input	Specifies how color is used in the printed image. 0: If a vector image is printed, line color is determined by the setting of the Black Lines Only option in the File->Print dialog. If a raster image is printed, the image background is the same as the background of the graphics window. 1: If a vector image is printed, line color is determined by the setting of the Black Lines Only option in the File->Print dialog. If a raster image is printed, the image background is white. 2: If a vector image is printed, lines are printed

	in color (or shades of gray on a monochrome printer). If a raster image is printed, the image background is the same as the background of the graphics window. 3: If a vector image is printed, all lines are printed in black. If a raster image is printed, the image background is white.

UF_DISP_refresh (view source)

Defined in: uf_disp.h

Overview
Performs an entire display refresh.

Return
void

Environment
Internal

Required License(s)
gateway

```
void UF_DISP_refresh
(
    void
)
```

UF_DISP_regenerate_display (view source)

Defined in: uf_disp.h

Overview
Regenerates the display.

Environment
Internal

Required License(s)
gateway

```
int UF_DISP_regenerate_display
(
    void
)
```

UF_DISP_regenerate_view (view source)

Defined in: uf_disp.h

Overview

Regenerates the display of a single view . The view must be currently displayed. If you specify an inactive view (i.e. a view on a layout or a drawing which is not displayed), an error occurs. Use UF_VIEW_ask_tag_of_view_name to find the view_tag.

Environment
Internal

See Also
UF_VIEW_ask_tag_of_view_name

Required License(s)
gateway

```
int UF_DISP_regenerate_view
(
    tag_t view_tag
)
```

tag_t	**view_tag**	Input	The tag of the view to regenerate

UF_DISP_remove_material_assignment (view source)

Defined in: uf_disp.h

Overview
This function applies "None" material to a geometry identified by the input object tag argument.
This removes all the material texture links to the object.
It does not remove the materials or textures from the part.

Returns
0 = Success
Standard UF Error Codes:
UF_err_program_not_initialized
UF_err_bad_parameter_number_1: problem found with object_tag
UF_err_part_not_loaded

This function aplies to materials of current renderer type.
This means if user is in Iray+ renderer mode, then only materials of Iray+ type will be removed.

Environment
Internal and external

See Also
See example

History
This function is created for NX8.5.3

Required License(s)
gateway

```
int UF_DISP_remove_material_assignment
(
    tag_t object_tag
)
```

tag_t	**object_tag**	Input	The tag of the geometry object to remove the material from.

UF_DISP_reset_conehead_attrb (view source)

Defined in: uf_disp_ugopenint.h

Overview
Resets the conehead attributes to the default values.

Return
void

Environment
Internal

Required License(s)
gateway

```
void UF_DISP_reset_conehead_attrb
(
    void
)
```

UF_DISP_set_color (view source)

Defined in: uf_disp.h

Overview
Modifies the name and values of the specified color in the Color Table object (CTO) of the Display part. The color values are given using the specified color model.
Note that the maximum value permitted for clr_num is the number of records presently in the CTO, minus 1. The number of CTO records may be obtained by calling UF_DISP_ask_color_count. There must be a part loaded when this function is called.

Environment
Internal and External

See Also
See example

Required License(s)
gateway

```
int UF_DISP_set_color
(
    int clr_num,
    int clr_model,
    char * clr_name,
    double clr_values [ 3 ]
)
```

int	**clr_num**	Input	The number of the color whose name and value are to be returned; Range: 0 .. (# color records in CTO) - 1 Note: 0 is for the background color
int	**clr_model**	Input	The color model of the values in clr_values: UF_DISP_rgb_model UF_DISP_hsv_model UF_DISP_hls_model

char *	**clr_name**	Input	The name of the color; the character array must be null terminated. The length must not exceed UF_DISP_MAX_NAME_SIZE, as defined in uf_disp.h ignored if clr_num == 0
double	**clr_values [3]**	Input	Three doubles representing the color, where the meaning and range of each value depends on the color model specified: rgb: clr_values[0]: 0.0 <= red <= 1.0 clr_values[1]: 0.0 <=green <= 1.0 clr_values[2]: 0.0 <= blue <= 1.0 hsv: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[1]: 0.0 <= saturation <= 1.0 clr_values[2]: 0.0 <= value <= 1.0 hls: clr_values[0]: 0.0 <= hue <= 360.0 clr_values[2]: 0.0 <= light <= 1.0 clr_values[1]: 0.0 <= saturation <= 1.0

UF_DISP_set_conehead_attrb (view source)

Defined in: uf_disp_ugopenint.h

Overview
Allows changes to the current attribute settings with which the conehead vector displays by a call to UF_DISP_conehead. Note that it is best to get the current attribute settings and then adjust the ones you need, as shown in the example.

Return
void

Environment
Internal

See Also
See example

Required License(s)
gateway

```
void UF_DISP_set_conehead_attrb
(
    UF_DISP_conehead_attrb_s * attributes
)
```

UF_DISP_conehead_attrb_s *	**attributes**	Input	Pointer to attributes structure.

UF_DISP_set_display (view source)

Defined in: uf_disp.h

Overview
Sets the display status. You can use UF_DISP_set_display to turn off the graphics display before executing a long series of graphics intensive operations such as manipulations of views, layouts, or drawings. You must call UF_DISP_regenerate_display to update the display after turning it back on.
Note that there are limitations to this function in conjunction with layout and drawing changes. If you open a layout or drawing, the display will change to show the outlines of the views in the layout

or drawing, even though the suppress state is enabled; the contents of the views will not be shown however.

Environment
Internal

Required License(s)
gateway

int UF_DISP_set_display
(
int display_code
)

int	**display_code**	Input	display code: UF_DISP_SUPPRESS_DISPLAY- set display off UF_DISP_UNSUPPRESS_DISPLAY- set display on

---

UF_DISP_set_drawing_display (view source)

Defined in: uf_disp.h

Overview
Update the data that affects drawing monochrome display.
The routine UF_DISP_ask_drawing_display MUST be called to populate the data structure, before this routine is called.

Please reference ufd_disp.c to review a sample execution of this function.

Return
UF_err_program_not_initialized, UF_err_bad_parameter_number_1

Environment
Internal and External

See Also
UF_DISP_drawing_display_data_t

History
None

Required License(s)
gateway

int UF_DISP_set_drawing_display
(
UF_DISP_drawing_display_data_p_t drawing_display
)

UF_DISP_drawing_display_data_p_t	**drawing_display**	Input	The drawing monochrome display structure.

UF_DISP_set_grid_parameters (view source)

Defined in: uf_disp.h

Overview

Sets up the information for the grid in the designated application context. Any field not set will have a default value provided.

product_context - either UF_DISP_SKETCH_GRID for sketcher application,
or UF_DISP_DRAWING_GRID for drawing,
or UF_DISP_SHED_GRID for True Shading display,
or UF_DISP_MODEL_GRID for others (or default).

Environment
Internal and External

See Also
UF_DISP_ask_grid_parameters
UF_DISP_activate_grid
UF_DISP_deactivate_grid
UF_DISP_ask_current_grid_context

History
Originally released in V19.0

Required License(s)
gateway

void UF_DISP_set_grid_parameters
(
UF_DISP_grid_context_t product_context,
UF_DISP_grid_p_t input_grid
)

UF_DISP_grid_context_t	**product_context**	Input	either UF_DISP_SKETCH_GRID or UF_DISP_DRAWING_GRID or UF_DISP_MODEL_GRID or UF_DISP_SHED_GRID
UF_DISP_grid_p_t	**input_grid**	Input	a polulated grid structure

---

UF_DISP_set_highlight (view source)

Defined in: uf_disp.h

Overview
Turns object/feature highlight on or off.

Environment
Internal

Required License(s)
gateway

int UF_DISP_set_highlight
(
tag_t object_id,
int action_switch
)

tag_t	**object_id**	Input	Object/Feature identifier
int	**action_switch**	Input	Action Switch 0 = Turn Highlight Off 1 = Turn Highlight On

UF_DISP_set_highlights (view source)

Defined in: uf_disp.h

Overview
Turns object/feature highlight on or off.

Environment
Internal

History
This function was first released in NX 7.5.5

Required License(s)
gateway

```
int UF_DISP_set_highlights
(
    int object_count,
    tag_p_t object_list,
    int action_switch
)
```

int	**object_count**	Input	The number of objects in the object_list
tag_p_t	**object_list**	Input	object list to set highlights
int	**action_switch**	Input	Action Switch 0 = Turn Highlight Off 1 = Turn Highlight On

---

UF_DISP_set_model_bounds (view source)

Defined in: uf_disp.h

Overview
Stores the given model bounds into the one and only model bounds object for the work part. The given tag must be that of the model bounds object.
NOTE: This function is of limited value for most NX Open API application developers. It is intended to be used by certain NX applications such as translators.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DISP_set_model_bounds
(
    const tag_t model_bounds_object,
    const double model_bounds [ 6 ]
)
```

const tag_t	**model_bounds_object**	Input	The tag of the Model Bounds object of the work part. You will have received

			this from a prior call to UF_DISP_ask_model_bounds_tag
const double	**model_bounds [6]**	Input	The model bounds of the current work part. The six values are (Minimum_X, Maximum_X, Minimum_Y, Maximum_Y, Minimum_Z, Maximum_Z). You probably have determined these bounds in a prior call to UF_DISP_compute_model_bounds.

UF_DISP_set_name_display_status (view source)

Defined in: uf_disp.h

Overview
Sets the name display status for objects. Names are created with UF_OBJ_set_name, and the location where they are displayed is set with UF_OBJ_set_name_origin.

Environment
Internal and External

See Also
UF_OBJ_set_name
UF_OBJ_set_name_origin

Required License(s)
gateway

```
int UF_DISP_set_name_display_status
(
    const int new_status
)
```

const int	**new_status**	Input	Name display status: UF_DISP_NAME_DISPLAY_OFF UF_DISP_NAME_DISPLAY_ON

UF_DISP_set_name_view_status (view source)

Defined in: uf_disp.h

Overview
Sets the name display view status. The constant UF_DISP_NAMES_IN_VIEW_OF_DEFN is the view which is the work view when the object is defined.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DISP_set_name_view_status
(
    const int new_status
)
```

const int	**new_status**	Input	Name display view status: UF_DISP_NAMES_IN_WORK_VIEW UF_DISP_NAMES_IN_VIEW_OF_DEFN UF_DISP_NAMES_IN_ALL_VIEWS

UF_DISP_set_srfanl_params (view source)

Defined in: uf_disp.h

Overview
Sets the face analysis display parameters. This routine uses a pointer to the structure UF_DISP_srfanl_data_s.

Environment
Internal

See Also
UF_DISP_srfanl_data_t

Required License(s)
gateway

```
int UF_DISP_set_srfanl_params
(
    UF_DISP_srfanl_data_t * params
)
```

UF_DISP_srfanl_data_t *	**params**	Input	New system display parameters

---

UF_DISP_set_system_parameters (view source)

Defined in: uf_disp.h

Overview
Sets new system display parameters. This routine uses a pointer to the structure.
UF_DISP_system_params_p_t

To access parameters related to shaded face edges, please use
UF_VIEW_set_shaded_edge_options
instead.

Environment
Internal and External

History
V12.0 light_source_angles field removed from the structure
V13.0 Added show_shaded_face_edges and hidden_shaded_face_edges to the structure
V19.0 Added fields use_face_edges_color
face_edges_color
hidden_geometry_color
random_color_displayed
random_color_object_type

Required License(s)
gateway

```
int UF_DISP_set_system_parameters
(
    UF_DISP_system_params_p_t system_parameters
)
```

UF_DISP_system_params_p_t	**system_parameters**	Input	New system display parameters

---

UF_DISP_set_texture_space_info (view source)

Defined in: uf_disp.h

Overview
This function allows user to set new texture space information of the material.
This function only supports Author based materials.

Environment
Internal and External

See Also
See example

History
This function is created for NX3 QRM.

Required License(s)
gateway

```
int UF_DISP_set_texture_space_info
(
    const tag_t material_tag,
    UF_DISP_texture_space_info_t * ts_info_ptr
)
```

const tag_t	**material_tag**	Input	tag of material
UF_DISP_texture_space_info_t *	**ts_info_ptr**	Input	new texture space information

---

UF_DISP_set_work_plane_emphasis (view source)

Defined in: uf_disp.h

Overview
Sets work plane emphasis on or off. When work plane emphasis is on, objects that do not lie on the work plane appear de-emphasized, in which the object color is blended with the De-emphasis Blend Color according to the De-emphasis Blend Percentage. A de-emphasized object is unselectable unless the work plane emphasis selection filter is overridden. Work plane emphasis is a session dependent value; it is not saved with any part.

Environment
Internal and External

See Also
UF_DISP_ask_work_plane_emphasis
UF_DISP_ask_work_plane_sel
UF_DISP_set_work_plane_sel

History
Original release was in V13.0.

Required License(s)
gateway

int UF_DISP_set_work_plane_emphasis
(
int emphasis
)

int	**emphasis**	Input	Emphasis setting. Must be either: UF_DISP_WORK_PLANE_EMPHASIS_ON, or UF_DISP_WORK_PLANE_EMPHASIS_OFF

UF_DISP_set_work_plane_sel (view source)

Defined in: uf_disp.h

Overview
Sets (or unsets) the work plane emphasis selection override.
When work plane emphasis is enabled and the selection override is set to UF_DISP_SELECT_WORK_DIMMED, no objects are filtered out of selection because they are off the work plane.
When work plane emphasis is enabled and the selection override is set to UF_DISP_NO_SELECT_WORK_DIMMED, no objects dimmed for work plane emphasis are selectable.
When work plane emphasis is disabled, the selection override setting has no affect although it can be changed. The selection override setting takes effect once work plane emphasis is enabled.
The work plane emphasis selection override is a session dependent value; it is not saved with any part.

Environment
Internal and External

See Also
UF_DISP_ask_work_plane_emphasis
UF_DISP_set_work_plane_emphasis
UF_DISP_ask_work_plane_sel

History
Original release was in V13.0.

Required License(s)
gateway

int UF_DISP_set_work_plane_sel
(
int override
)

int	**override**	Input	Emphasis selection override setting. Must be either: UF_DISP_SELECT_WORK_DIMMED, or UF_DISP_NO_SELECT_WORK_DIMMED

UF_DISP_update_material_display_of_geometry (view source)

Defined in: uf_disp.h

Overview

This function updates the display of the given geometry objects in Studio rendering style.

The display of UF_solid_type, UF_solid_face_subtype and UF_faceted_model_type are updated. All other object types are ignored.

This function supports materials update of current renderer type.

Environment

Internal

See Also

See example

History

This function is created for NX7.5.2 QRM.

Required License(s)

gateway

```
int UF_DISP_update_material_display_of_geometry
(
    const int object_count,
    const tag_p_t object_list
)
```

const int	**object_count**	Input	The number of objects in the object_list
const tag_p_t	**object_list**	Input	The tags of objects with material applied to