UF_FLTR_ask_box_of_assy (view source)

Defined in: uf_fltr.h

Overview

Returns the bounding box information for an assembly and all of the components in it, or for a particular component. The edges of the box are parallel to the axis of the absolute coordinate system.

Environment

Internal and External

History

This function was originally released in V13.0.4.

Required License(s)

gateway

```
int UF_FLTR_ask_box_of_assy
(
    tag_t assy,
    double centroid [ 3 ] ,
    double corner [ 3 ] ,
    double orientation [ 9 ]
)
```

tag_t	**assy**	Input	Tag of the assembly, component, or subassembly to box.
double	**centroid [3]**	Output	Center of box
double	**corner [3]**	Output	Corner vector (diagonal corner coordinates of the box can be calculated by adding and subtracting this vector from the centroid location)
double	**orientation [9]**	Output	Orientation of box with respect to absolute coordinate system.

UF_FLTR_ask_box_zone (view source)

Defined in: uf_fltr.h

Overview

Outputs the parameters of the box zone specified by a tag.

The corner value represents an offset position which can be subtracted from the centroid position to give the minimum corner, or added to the centroid position to give the maximum corner. The corner value is always positive in value.

Say the bounding box has a minimum corner of 0,0,0 and maximum corner of 100,100,100. The centroid is at ((0,0,0)+(100,100,100))/2 = (50,50,50) whilst the corner calculation is ((100,100,100)-(0,0,0))/2 = (50,50,50).

Environment

Internal and External

See Also
See the example for UF_FLTR_create_box_zone

Required License(s)
gateway

```
int UF_FLTR_ask_box_zone
(
    tag_t box_zone,
    char* name,
    double centroid [ 3 ] ,
    double corner [ 3 ] ,
    double matrix [ 9 ]
)
```

tag_t	**box_zone**	Input	Tag of zone being queried
char*	**name**	Output	Name of zone being queried (30 chars max)
double	**centroid [3]**	Output	Centroid of zone (in Absolute Coordinates.)
double	**corner [3]**	Output	Corner vector
double	**matrix [9]**	Output	Orientation of zone.

UF_FLTR_ask_filter (view source)

Defined in: uf_fltr.h

Overview
Returns name and condition of a given filter.

Environment
Internal and External

Required License(s)
gateway

```
int UF_FLTR_ask_filter
(
    tag_t filter_tag,
    char name [ UF_OBJ_NAME_BUFSIZE ] ,
    char condition [ 132 ]
)
```

tag_t	**filter_tag**	Input	Filter which is being queried
char	**name [UF_OBJ_NAME_BUFSIZE]**	Output	Name of filter (30 chars max).
char	**condition [132]**	Output	Condition of filter (132 chars max).

UF_FLTR_ask_plane_zone (view source)

Defined in: uf_fltr.h

Overview
Outputs the parameters of the plane zone referred to by a tag.

Environment
Internal and External

See Also
See the example for UF_FLTR_create_box_zone

Required License(s)
gateway

```
int UF_FLTR_ask_plane_zone
(
    tag_t plane_zone,
    char name [ UF_OBJ_NAME_BUFSIZE ] ,
    double origin [ 3 ] ,
    double matrix [ 9 ]
)
```

tag_t	**plane_zone**	Input	tag of zone being queried.
char	**name [UF_OBJ_NAME_BUFSIZE]**	Output	The output should be declared as char buffer[UF_OBJ_NAME_BUFSIZE] Name of zone being queried
double	**origin [3]**	Output	Origin of zone (in Absolute Coordinates.)
double	**matrix [9]**	Output	Orientation of plane.

UF_FLTR_auto_create_box_zones (view source)

Defined in: uf_fltr.h

Overview
Automatically generates box zones along all 3 axes of the WCS in a part. If use_part_volume is TRUE, the zones are created to divide up the entire volume occupied by the part.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_FLTR_auto_create_box_zones
(
    tag_t part_tag,
    char * prefix_text,
    int num_in_dir [ 3 ] ,
    logical use_part_volume,
    double user_spec_vol [ 3 ] ,
    double user_spec_origin [ 3 ] ,
    tag_t * * zone_list,
    int * num_zones_created
)
```

tag_t	**part_tag**	Input	part in which zones are to be created.
char *	**prefix_text**	Input	Name to be used as prefix of auto-generated zones. (e.g. "ZONE" makes ZONE1, ZONE2 etc.)
int	**num_in_dir [3]**	Input	Number of zones to be created in each direction of the WCS. num_in_dir[0] = number of zones in X direction num_in_dir[1] = number of zones in Y direction num_in_dir[2] = number of zones in Z direction
logical	**use_part_volume**	Input	Flag which indicates region to be zoned. if TRUE, part extents are used. If FALSE, user_spec_vol and user_spec_origin are used (See below).
double	**user_spec_vol [3]**	Input	User specified volume to be zoned. If use_part_volume is TRUE, user_spec_vol is ignored. user_spec_vol[0] extent of volume in X direction user_spec_vol[1] extent of volume in Y direction user_spec_vol[2] extent of volume in Z direction
double	**user_spec_origin [3]**	Input	User specified origin of volume to be zoned. If use_part_volume is TRUE, user_spec_origin is ignored. user_spec_origin[0] X coordinate of origin user_spec_origin[1] Y coordinate of origin user_spec_origin[2] Z coordinate of origin.
tag_t * *	**zone_list**	Output to UF_*free*	List of resulting zone tags. Must be freed by the caller using UF_free.
int *	**num_zones_created**	Output	Number of zones in zone_list.

UF_FLTR_auto_create_plane_zones (view source)

Defined in: uf_fltr.h

Overview

Automatically generates plane zones along the Z axis of the WCS in a part. If use_part_disp is TRUE, the zones are created to cover the entire Z displacement of the part.

Environment

Internal and External

See Also
Refer to the example

Required License(s)
adv_assemblies

```
int UF_FLTR_auto_create_plane_zones
(
    tag_t part_tag,
    char * prefix_text,
    int num_in_dir,
    logical use_part_disp,
    double user_spec_z_disp,
    double user_spec_origin [ 3 ] ,
    tag_t * * zone_list,
    int * num_zones_created
)
```

tag_t	**part_tag**	Input	part in which zones are to be created.
char *	**prefix_text**	Input	Name to be used as prefix of auto-generated zones. (e.g. "ZONE" makes ZONE1, ZONE2 etc.)
int	**num_in_dir**	Input	Number of zones to be created in the Z direction of the WCS.
logical	**use_part_disp**	Input	Flag which indicates region to be zoned. if TRUE, part extents along the Z axis of WCS are used. If FALSE, user_spec_z_disp and user_spec_origin are used (See below).
double	**user_spec_z_disp**	Input	User specified Z displacement to be zoned. If use_part_disp is TRUE, this argument is ignored.
double	**user_spec_origin [3]**	Input	User specified origin of volume to be zoned. If use_part_disp is TRUE, user_spec_origin is ignored. user_spec_origin[0] X coordinate of origin user_spec_origin[1] Y coordinate of origin user_spec_origin[2] Z coordinate of origin.
tag_t * *	**zone_list**	Output to UF_*free*	List of resulting zone tags. Must be freed by the caller using UF_free.
int *	**num_zones_created**	Output	Number of zones in zone_list.

UF_FLTR_create_box_zone (view source)

Defined in: uf_fltr.h

Overview
Creates a box zone and returns its tag. The centroid is the center point of the box zone (in Absolute coordinates). The corner is a vector describing the most positive corner of the box relative to the centroid. The orientation is the orientation of the zone to be created.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
adv_assemblies

```
int UF_FLTR_create_box_zone
(
    tag_t part_tag,
    char * name,
    double centroid [ 3 ] ,
    double corner [ 3 ] ,
    double orientation [ 9 ] ,
    tag_t * zone_tag
)
```

tag_t	**part_tag**	Input	Part in which zone is to be created.
char *	**name**	Input	Name of zone to be created.
double	**centroid [3]**	Input	Centroid of zone (in Absolute Coordinates.)
double	**corner [3]**	Input	Corner vector (points to most positive box corner)
double	**orientation [9]**	Input	Orientation of zone with respect to absolute.
tag_t *	**zone_tag**	Input / Output	Tag of zone created.

UF_FLTR_create_filter (view source)

Defined in: uf_fltr.h

Overview
Creates a filter object given the name and condition.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_FLTR_create_filter
(
    tag_t part_tag,
    char * name,
    char * condition,
    tag_t * zone_tag
)
```

tag_t	**part_tag**	Input	part in which filter is to be created.

char *	**name**	Input	Name of filter to be created.
char *	**condition**	Input	Condition of filter to be created. For example, "Within(0,ZONE_PLANE_P1)"
tag_t *	**zone_tag**	Input / Output	Tag of filter created.

UF_FLTR_create_plane_zone (view source)

Defined in: uf_fltr.h

Overview
Creates a plane zone given the origin and orientation. Origin and
Creates a plane zone given the origin and orientation. Origin and

Environment
Internal and External

See Also
See the example for UF_FLTR_create_box_zone

Required License(s)
adv_assemblies

```
int UF_FLTR_create_plane_zone
(
    tag_t part_tag,
    char * name,
    double origin [ 3 ] ,
    double orientation [ 9 ] ,
    tag_t * zone_tag
)
```

tag_t	**part_tag**	Input	Part in which zone is to be created.
char *	**name**	Input	Name of zone to be created.
double	**origin [3]**	Input	Origin of zone (in Absolute Coordinates.)
double	**orientation [9]**	Input	Orientation of plane zone.
tag_t *	**zone_tag**	Input / Output	Tag of zone created.

UF_FLTR_edit_box_zone (view source)

Defined in: uf_fltr.h

Overview
Modifies the box zone referred to by a tag to have the values passed
in by name, centroid, corner and matrix.

Environment
Internal and External

See Also
See the example for UF_FLTR_create_box_zone

Required License(s)
adv_assemblies

```
int UF_FLTR_edit_box_zone
(
    tag_t zone,
    char * name,
    double centroid [ 3 ] ,
    double corner [ 3 ] ,
    double matrix [ 9 ]
)
```

tag_t	**zone**	Input	Tag of zone to be edited.
char *	**name**	Input	New name of zone.
double	**centroid [3]**	Input	New centroid of zone (in Absolute Coordinates.)
double	**corner [3]**	Input	New corner vector
double	**matrix [9]**	Input	New orientation of zone.

UF_FLTR_edit_filter (view source)

Defined in: uf_fltr.h

Overview
Gives filter new name and condition.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_FLTR_edit_filter
(
    tag_t filter_tag,
    char * name,
    char * condition
)
```

tag_t	**filter_tag**	Input	Filter which is being edited
char *	**name**	Input	New name of filter (30 chars max).

char *	**condition**	Input	New condition of filter (132 chars max).

UF_FLTR_edit_plane_zone (view source)

Defined in: uf_fltr.h

Overview
Modifies the plane zone referred to by a tag to have the values passed in by name, origin and matrix.

Environment
Internal and External

See Also
See the example for UF_FLTR_create_box_zone

Required License(s)
adv_assemblies

```
int UF_FLTR_edit_plane_zone
(
    tag_t zone,
    char * name,
    double origin [ 3 ] ,
    double matrix [ 9 ]
)
```

tag_t	**zone**	Input	tag of zone to be modified.
char *	**name**	Input	New name of zone.
double	**origin [3]**	Input	New origin of zone (in Absolute Coordinates.)
double	**matrix [9]**	Input	New orientation of plane.

UF_FLTR_evaluate_filter (view source)

Defined in: uf_fltr.h

Overview
Evaluate whether object matches the given filter.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_FLTR_evaluate_filter
(
    tag_t input_object,
    tag_t filter_tag,
    logical * result
)
```

tag_t	**input_object**	Input	Object to test against filter. Currently only part occurrences are supported.
tag_t	**filter_tag**	Input	Tag of filter in question.
logical *	**result**	Output	Result of filter evaluation. TRUE if object matches the given filter. Otherwise FALSE.

UF_FLTR_is_obj_above_plane_zone (view source)

Defined in: uf_fltr.h

Overview

Tests whether a part occurrence is above a planar zone. Note that this function can only be used with a plane zone.

Environment

Internal and External

Required License(s)

gateway

```
int UF_FLTR_is_obj_above_plane_zone
(
    tag_t zone,
    tag_t object,
    logical * result
)
```

tag_t	**zone**	Input	Tag of zone. Must be a plane zone.
tag_t	**object**	Input	tag of object. Currently only part occurrences are supported.
logical *	**result**	Output	Result of comparison: TRUE = if any portion of object is above the plane specified by zone. FALSE = if no portion of object is above the plane i specified by zone.

UF_FLTR_is_obj_inside_box_zone (view source)

Defined in: uf_fltr.h

Overview
Tests whether an object is further than some distance of a zone.

Environment
Internal and External

Required License(s)
gateway

```
int UF_FLTR_is_obj_inside_box_zone
(
    double distance,
    tag_t zone,
    tag_t object,
    logical * result
)
```

double	**distance**	Input	Distance between zone and part occurrence (in part units).
tag_t	**zone**	Input	Tag of zone. (Must be box zone)
tag_t	**object**	Input	Tag of object. Currently only part occurrences are supported.
logical *	**result**	Output	Result of comparison. result = TRUE if no portion of object is further than distance from zone. result = FALSE if any portion of object is further than distance from zone.

UF_FLTR_is_obj_intsct_zone (view source)

Defined in: uf_fltr.h

Overview
Tests whether an object is within some distance of a zone. Currently the object can only be a part occurrence. The zone can be either a box or a plane zone.

Environment
Internal and External

Required License(s)
gateway

```
int UF_FLTR_is_obj_intsct_zone
(
    double distance,
    tag_t zone,
    tag_t object,
    logical * result
)
```

double	**distance**	Input	Distance between zone and part occurrence (in part units).

tag_t	**zone**	Input	Tag of zone.
tag_t	**object**	Input	tag of object. Currently only part occurrences are supported.
logical *	**result**	Output	Result of comparison. result = TRUE if some portion of object is within distance of zone. result = FALSE if no portion of object is within distance of zone.

UF_FLTR_object_has_box (view source)

Defined in: uf_fltr.h

Overview
Returns if the given NX object has a bounding box recorded for it.

Environment
Internal and External

History
This function was originally released in V18.0.5.2

Required License(s)
adv_assemblies

```
int UF_FLTR_object_has_box
(
    tag_t object,
    logical * has_box
)
```

tag_t	**object**	Input	Tag of the object to question.
logical *	**has_box**	Output	True if the object has a box, false otherwise

---

UF_FLTR_update_structure (view source)

Defined in: uf_fltr.h

Overview
Updates the assembly structure of the given part.
The interactively equivalent is found on the assembly node in the Assembly Navitagor (ANT) as Update Structure.
It uses file time stamps to see which parts have been modified since the assembly was last saved, these parts are then loaded to get the assembly structure, and component attributes, up to date, and are then unloaded.

Environment

Internal and External

History

This function was originally released in V15.0.

Required License(s)

adv_assemblies

int UF_FLTR_update_structure
(
tag_t part
)

tag_t	**part**	Input	part to update assembly structure of
