uc5530 (view source)

Defined in: uf_drf.h

Overview

uc5530 create a horizontal, vertical, parallel, or

Required License(s)

gateway

```
void uc5530
(
    int ip1,
    tag_t np2,
    int ip3,
    int ip4,
    tag_t np5,
    int ip6,
    int ip7,
    const char * cp8,
    int ip9,
    const UF_DRF_one_apptext_line_t cp10 [ ] ,
    double * rp11,
    tag_t * nr12
)
```

int	**ip1**	
tag_t	**np2**	
int	**ip3**	
int	**ip4**	
tag_t	**np5**	
int	**ip6**	
int	**ip7**	
const char *	**cp8**	
int	**ip9**	Input
const UF_DRF_one_apptext_line_t	**cp10 [ ]**	
double *	**rp11**	Input
tag_t *	**nr12**	

---

uc5531 (view source)

Defined in: uf_drf.h

Overview

uc5531 create a perpendicular dimension

Required License(s)

gateway

```
void uc5531
(
    tag_t np1,
    tag_t np2,
    int ip3,
    int ip4,
    const char * cp5,
    int ip6,
    const UF_DRF_one_apptext_line_t cp7 [ ] ,
    double * rp8,
    tag_t * nr9
)
```

tag_t	**np1**	
tag_t	**np2**	
int	**ip3**	
int	**ip4**	
const char *	**cp5**	
int	**ip6**	Input
const UF_DRF_one_apptext_line_t	**cp7 [ ]**	
double *	**rp8**	Input
tag_t *	**nr9**	

---

uc5532 (view source)

Defined in: uf_drf.h

Overview

uc5532 create angular dimension

Required License(s)

gateway

```
void uc5532
(
    int ip1,
    tag_t np2,
    int ip3,
    tag_t np4,
    int ip5,
    const char * cp6,
    int ip7,
    const UF_DRF_one_apptext_line_t cp8 [ ] ,
    double * rp9,
    tag_t * nr10
)
```

int	**ip1**	
tag_t	**np2**	
int	**ip3**	
tag_t	**np4**	
int	**ip5**	
const char *	**cp6**	
int	**ip7**	Input
const UF_DRF_one_apptext_line_t	**cp8 [ ]**	
double *	**rp9**	Input
tag_t *	**nr10**	

---

uc5533 (view source)

Defined in: uf_drf.h

Overview

uc5533 create an arc length, radius, diameter, or hole

Required License(s)

gateway

```
void uc5533
(
    int ip1,
    tag_t np2,
    const char * cp3,
    int ip4,
    const UF_DRF_one_apptext_line_t cp5 [ ] ,
    double * rp6,
    tag_t * nr7
)
```

int	**ip1**	
tag_t	**np2**	
const char *	**cp3**	
int	**ip4**	Input
const UF_DRF_one_apptext_line_t	**cp5 [ ]**	
double *	**rp6**	Input
tag_t *	**nr7**	

---

uc5534 (view source)

Defined in: uf_drf.h

Overview
uc5534 create a concentric circle dimension

Required License(s)
gateway

```
void uc5534
(
    tag_t np1,
    tag_t np2,
    const char * cp3,
    int ip4,
    const UF_DRF_one_apptext_line_t cp5 [ ] ,
    double * rp6,
    tag_t * nr7
)
```

tag_t	**np1**	
tag_t	**np2**	
const char *	**cp3**	
int	**ip4**	Input
const UF_DRF_one_apptext_line_t	**cp5 [ ]**	
double *	**rp6**	Input
tag_t *	**nr7**	

---

uc5540 (view source)

Defined in: uf_drf.h

Overview
Create a drafting aid note. The line length of the note string cannot exceed 132 characters (in C this is 133 characters which includes the null character terminator `\0'). You can limit the number of lines that display from a character string array by specifying the number of lines (ip1). For example, if you make the following declarations:

int ip1=6;
char cp2[10][133];
then only the first 6 lines of the 10 line note will display.

Return
void

Environment
Internal and External

Required License(s)
gateway

```
void uc5540
(
    int ip1,
    const UF_DRF_one_apptext_line_t cp2 [ ] ,
    double * rp3,
    tag_t * nr4
)
```

int	**ip1**	Input	Number Of Lines Of Text
const UF_DRF_one_apptext_line_t	**cp2 [ ]**	Input	Text Array Of ip1 Strings.
double *	**rp3**	Input	3D Object Origin (In WCS Coordinates)
tag_t *	**nr4**	Output	Object Identifier Of Created Drafting Aid

---

uc5541 (view source)

Defined in: uf_drf.h

Overview

uc5541 create a label

Required License(s)

gateway

```
void uc5541
(
    int ip1,
    const UF_DRF_one_apptext_line_t cp2 [ ] ,
    double * rp3,
    int ip4,
    tag_t np5,
    double * rp6,
    tag_t * nr7
)
```

int	**ip1**	Input
const UF_DRF_one_apptext_line_t	**cp2 [ ]**	
double *	**rp3**	
int	**ip4**	
tag_t	**np5**	
double *	**rp6**	Input
tag_t *	**nr7**	

---

uc5542 (view source)

Defined in: uf_drf.h

Overview
uc5542 create an id symbol

Required License(s)
drafting

```
int uc5542
(
    int ip1,
    const char * cp2,
    const char * cp3,
    double * rp4,
    int ip5,
    int ip6,
    tag_t np7,
    double * rp8,
    tag_t * nr9
)
```

int	**ip1**	Input
const char *	**cp2**	
const char *	**cp3**	Input
double *	**rp4**	
int	**ip5**	
int	**ip6**	
tag_t	**np7**	
double *	**rp8**	Input
tag_t *	**nr9**	

---

uc5543 (view source)

Defined in: uf_drf.h

Overview
uc5543 create a form and positional

Required License(s)
gateway

```
void uc5543
(
    int ip1,
    const UF_DRF_one_apptext_line_t cp2 [ ] ,
    double * rp3,
    int ip4,
    int ip5,
```

```
    tag_t np6,
    double * rp7,
    int ip8,
    tag_t * nr9
)
```

int	**ip1**	Input
const UF_DRF_one_apptext_line_t	**cp2 [ ]**	
double *	**rp3**	
int	**ip4**	
int	**ip5**	
tag_t	**np6**	
double *	**rp7**	
int	**ip8**	Input
tag_t *	**nr9**	

---

uc5550 (view source)

Defined in: uf_drf.h

Overview

uc5550 read the drafting object creation - replaced by UF_DRF_ask_object_preferences

Required License(s)

gateway

```
void uc5550
(
    tag_t np1,
    int * ir2,
    double * rr3,
    char cr4 [ 27 ] ,
    char cr5 [ 27 ]
)
```

tag_t	**np1**	
int *	**ir2**	
double *	**rr3**	Output
char	**cr4 [27]**	
char	**cr5 [27]**	Output

---

uc5551 (view source)

Defined in: uf_drf.h

Overview

uc5551 regenerate a drafting object with - replaced by UF_DRF_set_object_preferences

Required License(s)

drafting

```
int uc5551
(
    tag_t np1,
    int * ip2,
    double * rp3,
    const char * cp4,
    const char * cp5
)
```

tag_t	**np1**	
int *	**ip2**	
double *	**rp3**	Input
const char *	**cp4**	
const char *	**cp5**	Input

---

uc5563 (view source)

Defined in: uf_drf.h

Overview

Add Text to an Assorted Parts Annotation (dimensions by parts)

Routine UF_DRF_create_assortpart_dim or UF_DRF_create_assortpart_aid must be called first to initialize Open API for dimension/drafting aid creation by parts. This routine, along with uf5561, uf5562, and uf5564, can then be called to add lines, arcs, text, and arrows to the dimension/drafting aid. Routine uf5565 must be called to close the definition of the object.

Environment

Internal and External

Required License(s)

gateway

```
void uc5563
(
    int num_lines,
    const UF_DRF_one_apptext_line_t text [ ] ,
    const double text_origin [ 2 ]
)
```

int	**num_lines**	Input	Number Of lines of text
const UF_DRF_one_apptext_line_t	**text [ ]**	Input	Array of text strings
const double	**text_origin [2]**	Input	2d origin of text string in coordinates of annotation

---

uc5566 (view source)

Defined in: uf_drf.h

Overview

Edit Drafting Object Text. Replaces existing text in an existing dimension/drafting aid. Can not be used to add text.
If the dimension text (dimensions) or main text (drafting aids) is changed, the object is changed to indicate manual text.

Return

Return code:
1 = Success
2 = Failure
3 = Zero Object Identifier passed in
4 = Text string in a line is too long
5 = Too many text string words

Environment

Internal and External

Required License(s)

drafting

```
int uc5566
(
    tag_t np1,
    int ip2,
    int ip3,
    const UF_DRF_one_apptext_line_t cp4 [ ]
)
```

tag_t	**np1**	Input	Drafting Object Identifier
int	**ip2**	Input	Text Type To Edit 1 = Dimension/Main Text 2 = Dual Dimension Text 3 = Tolerance Text 4 = Dual Tolerance Text 5 = Diameter/Radius Symbol Text 6 = Text Appended During Creation 7 = Text Appended During Editing 8 = Second Appended Text
int	**ip3**	Input	Number Of Lines Of Text
const UF_DRF_one_apptext_line_t	**cp4 [ ]**	Input	Array Of ip3 Text Strings.

---

uf5505 (view source)

Defined in: uf_drf.h

Overview

Create Kanji Note

The 23.ugf file must be accessible to the program calling this subroutine. The maximum number of characters per row is 132. The maximum number of rows is 50. Subject to the above, the maximum number of characters in the note is 650.

Environment

Internal and External

Required License(s)

gateway

```
void uf5505
(
    int * ia1,
    int * ip2,
    double * rp3,
    double * rp4,
    double * ra5,
    int * ip6,
    double * rp7,
    double * rp8,
    double * rp9,
    tag_t * nr10
)
```

int *	**ia1**	Input	Kanji Codes (1) = Count Of Characters In First Row (2..N) = Codes For First Row (N+1) = Count Of Characters In Second Row (N+2..M)= Codes For Second Row (Z) = 0 (End Of Data)
int *	**ip2**	Input	Density: 0 = Use Kanji Default 1 = Normal 2 = Heavy 3 = Thin
double *	**rp3**	Input	Character Size 0 = Use Kanji Default
double *	**rp4**	Input	Text Angle (In Radians)
double *	**ra5**	Input	Origin (X,Y in Work Coordinates)
int *	**ip6**	Input	Orientation: 0 = Use Kanji Default 1 = Left To Right 2 = Top To Bottom
double *	**rp7**	Input	Character Height
double *	**rp8**	Input	Character Width
double *	**rp9**	Input	Character Spacing
tag_t *	**nr10**	Output	Kanji Note Object Identifier

uf5506 (view source)

Defined in: uf_drf.h

Overview

Read Kanji Note

The 23.ugf file must be accessible to the program calling that subroutine. The maximum number of characters per row is 132. The maximum number of rows is 50. Subject to the above, the maximum number of characters in the note is 650.
The origin that returns is in units relative to the creation matrix of the Kanji entity. To convert these coordinates to WCS space, you can use the following code:
UF_CSYS_ask_matrix_of_object(entity_tag[inx1], &matrix_id);
UF_CSYS_ask_matrix_values(matrix_id, matrix);
UF_MTX3_transpose(matrix, matrix_transpose);
UF_MTX3_vec_multiply(origin, matrix_transpose, model_origin)

Environment

Internal and External

Required License(s)

gateway

```
void uf5506
(
    tag_t * np1,
    int * iar2,
    int * ir3,
    double * rr4,
    double * rr5,
    double * rar6,
    int * ir7,
    double * rr8,
    double * rr9,
    double * rr10
)
```

tag_t *	**np1**	Input	Kanji Note Object Identifier
int *	**iar2**	Output	Kanji Codes (1) = Count Of Characters In First Row (2..N) = Codes For First Row (N+1) = Count Of Characters In Second Row (N+2..M)= Codes For Second Row (Z) = 0 (End Of Data)
int *	**ir3**	Output	Density: 1 = Normal 2 = Heavy 3 = Thin
double *	**rr4**	Output	Character Size
double *	**rr5**	Output	Text Angle (In Radians)
double *	**rar6**	Output	Origin (X,Y,Z). See description above.
int *	**ir7**	Output	Orientation: 1 = Left To Right

			2 = Top To Bottom
double *	**rr8**	Output	Character Height
double *	**rr9**	Output	Character Width
double *	**rr10**	Output	Character Spacing

---

uf5522 (view source)

Defined in: uf_drf.h

Overview
Set the drafting parameters to the same values as an object.

Environment
Internal and External

Required License(s)
gateway

```
void uf5522
(
    tag_t * np1,
    int * ir2
)
```

tag_t *	**np1**	Input	The object that we want the drafting parameters to be set from.
int *	**ir2**	Output	The error code from the operation. 0 is success, 1 is failure

---

uf5523 (view source)

Defined in: uf_drf.h

Overview
Read the current drafting display parameters.

Environment
Internal and External

Required License(s)
gateway

```
void uf5523
(
    int * ir1,
    double * rr2
)
```

int *	**ir1**	Output	A 10 element array that will be filled with the drafting display parameters. The elements of the arrary are defined as: (0) MARGIN DISPLAY

			1 = DISPLAY MARGINS 2 = DO NOT DISPLAY MARGINS (1) FAST FONT 1 = FAST FONT ON 2 = FAST FONT OFF (2) TEXT BOX MODE 1 = TEXT BOX MODE ON 2 = TEXT BOX MODE OFF (3) INDICATOR SITE 1 = DISPLAY INDICATOR SITE 2 = DO NOT DISPLAY INDICATOR SITE (4) TEXT SCALING FOR PART MERGE AND TRANSFORMATIONS 1 = SCALE TEXT 2 = DO NOT SCALE TEXT (5) FILLED ARROWHEAD DISPLAY 1 = DISPLAY FILLED ARROWHEAD 2 = DO NOT DISPLAY FILLED ARROWHEAD (6) - (9) RESERVED FOR FUTURE USE
double *	**rr2**	Output	A 5 element array that will be filled with the drafting display parameters. Currently only the first parameter rr2[0] is used. rr2[0] holds the character slant (in degrees).

---

uf5524 (view source)

Defined in: uf_drf.h

Overview
Set the current drafting display parameters.

Environment
Internal and External

Required License(s)
gateway

```
void uf5524
(
    int * ip1,
    double * rp2
)
```

int *	**ip1**	Input	A 10 element array that contains the new drafting display parameters.
double *	**rp2**	Output	A 5 element array that contains the new drafting display parameters.

---

uf5547 (view source)

Defined in: uf_drf.h

Overview
uf5547 create a area fill

Required License(s)

gateway

```
void uf5547
(
    int * ip1,
    int * ip2,
    tag_t * np3,
    tag_t * nr4,
    int * ir5
)
```

int *	**ip1**
int *	**ip2**
tag_t *	**np3**
tag_t *	**nr4**
int *	**ir5**

---

uf554a (view source)

Defined in: uf_drf.h

Overview

Validates that selected object (entity) for creation of centerline is either an arc or a point. If the entity is an arc or a point, a drafting point is output.

Environment

Internal and External

Required License(s)

gateway

```
void uf554a
(
    tag_t * np1,
    tag_t * nr2
)
```

tag_t *	**np1**	Input	EID of selected entity
tag_t *	**nr2**	Output	EID of drafting point NULL_TAG = Invalid entity selected (Not an arc or point) if not NULL_TAG = drafting point

---

uf554b (view source)

Defined in: uf_drf.h

Overview
Validates that selected object (entity) is either an arc or a point. If the entity is an arc or a point, the coordinates of the point or arc center point are output.

Environment
Internal and External

Required License(s)
gateway

```
void uf554b
(
    tag_t * np1,
    double * rr2,
    int * ir3
)
```

tag_t *	**np1**	Input	EID of point or arc
double *	**rr2**	Output	Coordinates of point or arc center point
int *	**ir3**	Output	Return Code: 0 = Successful 1 = Invalid entity object selected (Not a point or an arc)

---

uf554c (view source)

Defined in: uf_drf.h

Overview
Ufunc program for creation of centerlines

Environment
Internal and External

Required License(s)
gateway

```
void uf554c
(
    const int * ip1,
    const int * ip2,
    const tag_t * np3,
    const tag_t * np4,
    const double * rp5,
    tag_t * nr6,
    int * ir7
)
```

const int *	**ip1**	Input	TYPE OF CENTERLINE TO CREATE 1 = LINEAR 2 = FULL CIRCULAR --- CENTER POINT METHOD 3 = PARTIAL CIRCULAR --- CENTER POINT METHOD 4 = FULL BOLT CIRCLE --- CENTER POINT METHOD 5 = PARTIAL BOLT CIRCLE --- CENTER POINT METHOD 6 = FULL CIRCULAR --- THRU 3 POINTS METHOD 7 = PARTIAL CIRCULAR --- THRU 3 POINTS METHOD

			8 = FULL BOLT CIRCLE --- THRU 3 POINTS METHOD 9 = PARTIAL BOLT CIRCLE --- THRU 3 POINTS METHOD 10 = OFFSET CYLINDRICAL --- KEYIN OFFSET DISTANCE METHOD 11 = OFFSET CYLINDRICAL --- CALCULATE OFFSET DISTANCE METHOD 12 = SYMMETRICAL 13 = OFFSET CENTER POINT --- XC-AXIS, DISTANCE FROM ARC NORMAL METHOD 14 = OFFSET CENTER POINT --- XC-AXIS, DISTANCE FROM ARC CENTER METHOD 15 = OFFSET CENTER POINT --- XC-AXIS, DISTANCE CALCULATED METHOD 16 = OFFSET CENTER POINT --- YC-AXIS, DISTANCE FROM ARC NORMAL METHOD 17 = OFFSET CENTER POINT --- YC-AXIS, DISTANCE FROM ARC CENTER METHOD 18 = OFFSET CENTER POINT --- YC-AXIS, DISTANCE CALCULATED METHOD
const int *	**ip2**	Input	NUMBER OF ENTITIES IN ARRAY
const tag_t *	**np3**	Input	ARRAY OF ENTITY IDs THAT ARE TO BE ASSOCIATED TO THE CENTERLINE
const tag_t *	**np4**	Input	ENTITY ID IF "IP1" IS 1 ..... NOT USED IF "IP1" IS 2 - 5 ..... ENTITY SELECTED FOR CENTERLINE CENTER POINT IF "IP1" IS 6 - 9 ..... NOT USED IF "IP1" IS 10, 11 ..... ENTITY SELECTED FOR CALCULATING THE OFFSET DISTANCE IF "IP1" IS 12 - 14, 16, 17 ... NOT USED IF "IP1" IS 15, 18 ..... ENTITY SELECTED FOR CALCULATING DISTANCE TO BE USED IN PLACEMENT OF THE OFFSET CENTER POINT
const double *	**rp5**	Input	DISTANCE VALUE IF "IP1" IS 1 - 9 ..... NOT USED IF "IP1" IS 10 ..... OFFSET DISTANCE IF "IP1" IS 11, 12, 15, 18 .... NOT USED IF "IP1" IS 13, 16 ..... DISTANCE FROM ARC NORMAL IF "IP1" IS 14, 17 ..... DISTANCE FROM ARC CENTER
tag_t *	**nr6**	Output	ENTITY ID OF CENTERLINE
int *	**ir7**	Output	RETURN CODE: = 0 SUCCESSFUL COMPLETION = 1 NO ENTITIES SUPPLIED = 2 MORE THAN 100 ENTITIES SUPPLIED = 3 CENTERLINE TYPE REQUIRES 3 OR MORE ENTITIES = 4 CENTERLINE TYPE REQUIRES 2 ENTITIES = 5 CENTERLINE TYPE REQUIRES 1 ENTITY = 6 INVALID ENTITY TYPE FOR CENTERLINE = 7 POINTS ARE COINCIDENT = 8 POINT IS NOT ON CENTERLINE = 9 INVALID ENTITY TYPE FOR CENTER POINT = 10 POINT IS COINCIDENT WITH CENTER = 11 POINTS ARE COLLINEAR = 12 INVALID ENTITY TYPE FOR OFFSET POINT = 13 INVALID CENTERLINE TYPE = 14 OFFSET CENTER POINT CENTERLINE REQUIRES AN ARC = 15 MODEL ENTITIES WERE SUPPLIED ON A DRAWING (NOT SUPPORTED)

uf5552 (view source)

Defined in: uf_drf.h

Overview
Read drafting object origin.

Environment
Internal and External

Required License(s)
gateway

```
void uf5552
(
    tag_t * np1,
    double * rr2,
    int * ir3
)
```

tag_t *	**np1**	Input	Drafting object identifier.
double *	**rr2**	Output	3D object origin in model space coordinates.
int *	**ir3**	Output	Return code. 0 = success, 1 = failure.

---

uf5553 (view source)

Defined in: uf_drf.h

Overview
Edits the drafting object origin.

Environment
Internal and External

Required License(s)
gateway

```
void uf5553
(
    tag_t * np1,
    double * rp2,
    int * ir3
)
```

tag_t *	**np1**	Input	Drafting Object Identifier
double *	**rp2**	Input	3-D Object Origin (In model space coordinates)
int *	**ir3**	Output	Return code: 1 = Success 2 = Failure

---

uf5554 (view source)

Defined in: uf_drf.h

Overview
Returns the number of objects that are associated to a drafting object.

Environment
Internal and External

Required License(s)
gateway

```
void uf5554
(
    tag_t * np1,
    int * ir2
)
```

tag_t *	**np1**	Input	Object Identifier Of Drafting Object
int *	**ir2**	Output	Number Of Associated Objects

---

uf5555 (view source)

Defined in: uf_drf.h

Overview
Returns the n-th associated object of a drafting object.

Environment
Internal and External

Required License(s)
gateway

```
void uf5555
(
    tag_t * np1,
    int * ip2,
    tag_t * nr3
)
```

tag_t *	**np1**	Input	Object Identifier Of Drafting Object
int *	**ip2**	Input	Index Number Of The Associated Object
tag_t *	**nr3**	Output	Object Identifier Of N-th Associated Object

---

uf5561 (view source)

Defined in: uf_drf.h

Overview

Add Lines to an Assorted Parts Annotation (dimensions by parts)

Routine UF_DRF_create_assortpart_dim or UF_DRF_create_assortpart_aid must be called first to initialize Open API for dimension/drafting aid creation by parts. This routine, along with uf5562, uf5563, and uf5564 can then be called to add lines, arcs, text, and arrows to the dimension/drafting aid. Routine uf5565 must be called to close the definition of the object.

Environment

Internal and External

Required License(s)

gateway

```
void uf5561
(
    int * num_lines,
    double * line_coords
)
```

int *	**num_lines**	Input	Number Of Lines. This must not exceed MAX_LINES.
double *	**line_coords**	Input	Array Of Line Points [0]Line 1, End 1, X Pos [1]Line 1, End 1, Y Pos [2]Line 1, End 2, X Pos [3]Line 1, End 2, Y Pos [4]Line 2, End 1, X Pos . . . Line IP1, End 2, Y Pos

---

uf5562 (view source)

Defined in: uf_drf.h

Overview

Add Arcs to an Assorted Parts Annotation (dimensions by parts)

Routine UF_DRF_create_assortpart_dim or UF_DRF_create_assortpart_aid must be called first to initialize Open API for dimension/drafting aid creation by parts. This routine, along with uf5561, uf5563, and uf5564 can then be called to add lines, arcs, text, and arrows to the dimension/drafting aid. Routine uf5565 must be called to close the definition of the object.

Environment

Internal and External

Required License(s)

gateway

```
void uf5562
(
```

```
    int * num_arcs,
    double * arc_data
)
```

int *	**num_arcs**	Input	Number Of Arc segments (not to exceed MAX_ARC_SEGMENTS)
double *	**arc_data**	Input	Arc data [0,1] 2D Arc center In Coordinates Of annotation [2] Arc Radius [3,4] Start and End Angles (In Radians) . . . [2num_arcs+1,2num_arcs+2] segment angles

---

uf5564 (view source)

Defined in: uf_drf.h

Overview

Add an Arrow to an Assorted Parts Annotation (dimensions by parts)

Routine UF_DRF_create_assortpart_dim or UF_DRF_create_assortpart_aid must be called first to initialize Open API for dimension/drafting aid creation by parts. This routine, along with uf5561, uf5562, and uf5563, can then be called to add lines, arcs, text, and arrows to the dimension/ drafting aid. Routine uf5565 must be called to close the definition of the object.

Environment

Internal and External

Required License(s)

gateway

```
void uf5564
(
    int * arrowhead_subtype,
    double * arrow_origin,
    double * arrow_angle
)
```

int *	**arrowhead_subtype**	Input	Arrowhead Of Subtype 1 = Closed 2 = Open 3 = Arch Cross 4 = Dot 5 = Origin Symbol 6 = None
double *	**arrow_origin**	Input	2d arrow origin in coordinates of annotation
double *	**arrow_angle**	Input	Arrow Angle (Radians)

---

uf5565 (view source)

Defined in: uf_drf.h

Overview

Close an Assorted Parts Annotation (dimensions by parts)

This routine closes the definition by parts and creates the dimension/drafting aid. This routine is called after routine UF_DRF_create_assortpart_dim or UF_DRF_create_assortpart_aid has been called to initialize the definition and routines uf5561 through uf5564 have been called to add lines, arcs, text, and arrows.

Environment

Internal and External

Required License(s)

gateway

```
void uf5565
(
    tag_t * annotation_tag
)
```

tag_t *	**annotation_tag**	Output	Annotation tag

---

uf5575 (view source)

Defined in: uf_drf.h

Overview

Obtain Coordinates of Dimension Area Box

Returns the lower left and upper right coordinates of a dimension area box.

Environment

Internal and External

Required License(s)

gateway

```
void uf5575
(
    tag_t * np1,
    double * rar2,
    double * rar3,
    int * ir4
)
```

tag_t *	**np1**	Input	Drafting Object Identifier
double *	**rar2**	Output	Lower Left Coordinate
double *	**rar3**	Output	Upper Right Coordinate
int *	**ir4**	Output	Return code: 0 = Success 1 = Error - NP1 Not A Drafting Object

---

UF_DRF_add_assortpart_to_ann (view source)

Defined in: uf_drf.h

Overview
Adds Assorted Parts to an Annotation.

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_add_assortpart_to_ann
(
    tag_t annotation_tag,
    int number_of_objects,
    tag_t * list_of_objects
)
```

tag_t	**annotation_tag**	Input	annotation tag to add assorted parts
int	**number_of_objects**	Input	number of objects to add
tag_t *	**list_of_objects**	Input	number_of_objects list of object tags to add

---

UF_DRF_add_compound_weld_symbol (view source)

Defined in: uf_drf.h

Overview
Add compound weld symbol

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_NO_ERRORS - No error
UF_err_program_not_initialized - Open C API has not been initialized
UF_DRF_null_object_structure

Environment
Internal and External

See Also
UF_DRF_ weld_symbols_t
UF_DRF_object_t

History
Originally released in v18.0

Required License(s)

gateway

int UF_DRF_add_compound_weld_symbol
(
tag_t weld_symbol,
UF_DRF_weld_sym_info_t * top_info,
UF_DRF_weld_sym_info_t * bottom_info
)

tag_t	**weld_symbol**	Input	tag of the weld symbol for which compound weld info has to be added
UF_DRF_weld_sym_info_t *	**top_info**	Input	compound weld symbol info for top
UF_DRF_weld_sym_info_t *	**bottom_info**	Input	compound weld symbol info for bottom

---

UF_DRF_add_controlling_exp (view source)

Defined in: uf_drf.h

Overview
Link drafting object to the controlling expression

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
drafting

int UF_DRF_add_controlling_exp
(
tag_t object,
tag_t exp_id
)

tag_t	**object**	Input	Drafting object
tag_t	**exp_id**	Input	Controlling expression

UF_DRF_add_symbol_to_object (view source)

Defined in: uf_drf.h

Overview
Adds a symbol to a drafting object.

Environment
Internal and External

See Also

Refer to the example

Required License(s)
drafting

int UF_DRF_add_symbol_to_object
(
UF_DRF_symbol_create_data_t * symbol_data,
tag_t object_tag
)

UF_DRF_symbol_create_data_t *	**symbol_data**	Input	symbol data (see uf_drf_types.h)
tag_t	**object_tag**	Input	drafting object tag

UF_DRF_add_to_dimension (view source)

Defined in: uf_drf.h

Overview
Append text, form-position tolerance, or ID symbol to a dimension.

Environment
Internal and External

See Also
UF_DRF_ask_ann_data

Required License(s)
drafting

int UF_DRF_add_to_dimension
(
unsigned int * entity_id,
int * segment_num,
int ann_data [10] ,
int * text_type,
int * text_position,
int * relative_just,
int * line_space,
int * number_lines,
const UF_DRF_one_apptext_line_t text_array [ ]
)

unsigned int *	**entity_id**	Input	Object identifier of the dimension object to be added to.
int *	**segment_num**	Input	Segment number to add to. UF_DRF_ask_ann_data can be called to find out the number of segments in a dimension.
int	**ann_data [10]**	Input	The annotation data for this dimension, returned from UF_DRF_ask_ann_data.
int *	**text_type**	Input	The type of appended text: 18 = Pure text

			19 = circle ID SYMBOL text 20 = divided circle ID SYMBOL 21 = square ID SYMBOL 22 = divided square ID SYMBOL 23 = hexagon ID SYMBOL 24 = divided hexagon ID SYMBOL 25 = triangle pointed up ID SYMBOL 26 = triangle pointed down ID SYMBOL 27 = datum target ID SYMBOL 28 = rounded box ID SYMBOL 29 = Form and positional tolerance text 44 = Multi-type text
int *	**text_position**	Input	Position of the appended text: 1 = below 2 = after 3 = above 4 = before
int *	**relative_just**	Input	Relative text justification: 1 = left 2 = center 3 = right
int *	**line_space**	Input	Flag for line spacing: 0 = No line spacing, this is the default. 1 = Use a line spacing adjustment when adding text blocks above or below.
int *	**number_lines**	Input	Number of lines of text to append.
const UF_DRF_one_apptext_line_t	**text_array [ ]**	Input	number_lines Array of text strings to be appended to the dimension.

UF_DRF_are_draft_objects_const (view source)

Defined in: uf_drf.h

Overview
This function will determine if the current drafting objects in the system match a set of given drafting objects previously recorded with the UF_DRF_record_draft_objects function.

Returns
UF_DRF_NO_ERRORS
UF_DRF_OBJECTS_ARE_NOT_CONSTANT

Environment
Internal & External

See Also
UF_DRF_record_draft_objects

History
Created in NX4.0

Required License(s)
drafting

int UF_DRF_are_draft_objects_const
(

```
    void * objs,
    logical check_view_data
)
```

void *	**objs**	Input	Drafting object data from UF_DRF_record_draft_objects
logical	**check_view_data**	Input	Should view data be compared

---

UF_DRF_ask_ang_obj_suppress_zeros (view source)

Defined in: uf_drf.h

Overview
Determines if the suppress zeros option for an angular dimension is set.
Suppress zero option: display zero,
suppress leading zeros,
suppress any zeros
suppress trailing zeros.

Environment
Internal and External

History
This function originally released in V17.0

Required License(s)
gateway

```
int UF_DRF_ask_ang_obj_suppress_zeros
(
    tag_t object,
    UF_DRF_angular_suppress_zeros_t * option
)
```

tag_t	**object**	Input	Tag id of angular dimension object
UF_DRF_angular_suppress_zeros_t *	**option**	Output	Suppress zeros option

---

UF_DRF_ask_ang_obj_units_format (view source)

Defined in: uf_drf.h

Overview
Queries the nominal and tolerance units format for an angular dimension.

Environment
Internal and External

History
This function originally released in V17.0

Required License(s)
gateway

```
int UF_DRF_ask_ang_obj_units_format
(
    tag_t object,
    UF_DRF_angular_units_t * nominal_format,
    UF_DRF_angular_units_t * tolerance_format
)
```

tag_t	**object**	Input	angular dimension
UF_DRF_angular_units_t *	**nominal_format**	Output	angular dimension nominal format
UF_DRF_angular_units_t *	**tolerance_format**	Output	angular dimension tolerance format

---

UF_DRF_ask_ann_arc_seg_angles (view source)

Defined in: uf_drf.h

Overview

Ask start and end Angles of an Annotation Object Arc Segment.

Environment

Internal and External

Required License(s)

gateway

```
int UF_DRF_ask_ann_arc_seg_angles
(
    int * arc_segment,
    int * ann_data,
    double arc_angles [ 2 ]
)
```

int *	**arc_segment**	Input	arc segment number
int *	**ann_data**	Input / Output	annotation data
double	**arc_angles [2]**	Output	start and end angles of arc segment

---

UF_DRF_ask_ann_data (view source)

Defined in: uf_drf.h

Overview

Ask data of an Annotation Object. The text data for the annotation
can be read by passing the ann_data array to UF_DRF_ask_text_data

This method will not work for the annotations having new leaders i.e.
leaders created in or after NX7.5. For such case, below methods can be used:
1. NXOpen::Annotations::ComponentData::GetLineComponents
2. NXOpen::Annotations::ComponentData::GetArcComponents
3. NXOpen::Annotations::ComponentData::GetTextComponents
4. NXOpen::Annotations::ComponentData::GetArrowComponents

Environment

Internal and External

See Also

UF_DRF_ask_text_data

History

In V17.0 Appended Text was enhanced to allow any combination of the 4 appended text locations. 6 and 30 are no longer returned for the text type for 1st and 2nd appended text. Instead the following values are returned:
50 - above appended text
51 - below appended text
52 - before appended text
53 - after appended text

Required License(s)

gateway

```
int UF_DRF_ask_ann_data
(
    tag_t * annotation_tag,
    int search_mask [ 4 ] ,
    int * cycle_flag,
    int ann_data [ 10 ] ,
    int * ann_data_type,
    int * ann_data_form,
    int * num_segments,
    double ann_origin [ 2 ] ,
    double * radius_angle
)
```

tag_t *	**annotation_tag**	Input	tag of annotation object
int	**search_mask [4]**	Input	array of data types to search [0] != 0, line data [1] != 0, arc data [2] != 0, text data [3] != 0, arrow data
int *	**cycle_flag**	Input / Output	cycle flag must be set to zero on first call incremented to show the current number of the call 0 = no more data found
int	**ann_data [10]**	Input / Output	Array of annotation data. The user must declare this as int ann_data[10], and it will then be filled in by this routine.
int *	**ann_data_type**	Output	annotation data type 0 = no more data of requested types 1 = line 2 = arc 3 = text 4 = arrow
int *	**ann_data_form**	Output	annotation data type form if ann_data_type = 1 1 = RESERVED 2 = RESERVED 3 = EXTENSION LINE 4 = DIMENSION LINE 5 = STUB 6 = DUAL BRACKET 7 = BASIC TOLERANCE LINE 8 = ISO LINE 9 = ADDED LINE if ann_data_type = 2 not used

			if ann_data_type = 3 1 = DIM TEXT OR MAIN TEXT 2 = DUAL DIM TEXT 3 = TOLERANCE TEXT 4 = DUAL TOLERANCE TEXT 5 = RAD/DIA TEXT 6 = OBSOLETE 7 = TEXT APP AT EDITING ID SYM TEXT INPUT AT CREATION: 8 = CIRCLE 9 = DIVIDED CIRCLE 10 = SQUARE 11 = DIVIDED CIRCLE 12 = HEXAGON 13 = DIVIDED HEXAGON 14 = TRIANGLE, POINT UP 15 = TRIANGLE, POINT DOWN 16 = DATUM TARGET 17 = ROUNDED BOX 18 = F&P TOL TEXT INP AT CREATION ID SYM TEXT APPENDED AT EDITING: 19 = CIRCLE 20 = DIVIDED CIRCLE 21 = SQUARE 22 = DIVIDED CIRCLE 23 = HEXAGON 24 = DIVIDED HEXAGON 25 = TRIANGLE, POINT UP 26 = TRIANGLE, POINT DOWN 27 = DATUM TARGET 28 = ROUNDED BOX 29 = F&P TOL TEXT APP AT EDITING 30 = OBSOLETE SECOND ID SYM TEXT INPUT AT CREATION: 31 = CIRCLE 32 = DIVIDED CIRCLE 33 = SQUARE 34 = DIVIDED CIRCLE 35 = HEXAGON 36 = DIVIDED HEXAGON 37 = TRIANGLE, POINT UP 38 = TRIANGLE, POINT DOWN 39 = DATUM TARGET 40 = ROUNDED BOX 41 = SECOND F&P TOL TEXT INP AT CREATION 42 = MULTI-TYPE TEXT INP AT CREATION 43 = 2ND MULTI-TYPE TEXT APP AT CREATION 44 = MULTI-TYPE TEXT APP AT 50 = ABOVE APPENDED TEXT 51 = BELOW APPENDED TEXT 52 = BEFORE APPENDED TEXT 53 = AFTER APPENDED TEXT EDITING if ann_data_type = 4 1 = CLOSED ARROW 2 = OPEN ARROW 3 = ARCH CROSS 4 = DOT
int *	**num_segments**	Output	number of data segments if ann_data_type = 1, line segments = 2, arc segments = 3, lines of text = 4, not used
double	**ann_origin [2]**	Output	annotation data origin if ann_data_type = 1, not used = 2, arc origin

			= 3, text origin = 4, arrow origin
double *	**radius_angle**	Output	annotation radius or angle if ann_data_type = 1, not used = 2, arc radius = 3, not used = 4, arrow angle

---

UF_DRF_ask_ann_line_seg_ends (view source)

Defined in: uf_drf.h

Overview
Ask end points of an Annotation Object Line Segment.

Environment
Internal and External

Required License(s)
gateway

```
int UF_DRF_ask_ann_line_seg_ends
(
    int * line_segment,
    int ann_data [ 10 ] ,
    double line_endpoints [ 4 ]
)
```

int *	**line_segment**	Input	line segment number
int	**ann_data [10]**	Input / Output	annotation data
double	**line_endpoints [4]**	Output	2D coordinates of line segment

---

UF_DRF_ask_annotation_template (view source)

Defined in: uf_drf.h

Overview
Ask for the name of the template part which is to be used to be used in UF_DRF_inherit_feature_data.

Environment
Internal and External

See Also
UF_DRF_inherit_type_t
UF_DRF_ask_callout_of_annotation
UF_DRF_ask_controlling_member_of_callout
UF_DRF_ask_number_of_rows_in_callout
UF_DRF_ask_callout_row_members
UF_DRF_inherit_feature_data
UF_DRF_set_annotation_template

History

Originally released in v19.0

Required License(s)
gateway

```
int UF_DRF_ask_annotation_template
(
    char * * annotation_template_name
)
```

char * *	**annotation_template_name**	Output to UF_*free*	the name of the annotation template part which is used by UF_DRF_inherit_feature_data. The name is the same as the name appearing in the Feature Parameters dialog and does not contain the directory in which the part resides or the extension .atp.prt, for example, ansi or iso_din

---

UF_DRF_ask_annotation_text_box (view source)

Defined in: uf_drf.h

Overview
Description -
Ask the text box of an annotation object. The text box information returned is in the format of:
upper_left - this is the upper left location point of the box
length - this is the length of the box
height - this is the height of the box

Return
0 = No error
not 0 = Error code

Environment
Internal and External

Required License(s)
gateway

```
int UF_DRF_ask_annotation_text_box
(
    tag_t annotation,
    double upper_left [ 3 ] ,
    double * length,
    double * height
)
```

tag_t	**annotation**	Input	Tag of the annotation
double	**upper_left [3]**	Output	Upper left point
double *	**length**	Output	Length of box
double *	**height**	Output	Height of box

UF_DRF_ask_appended_text (view source)

Defined in: uf_drf.h

Overview
The following function will query all the appended text for a dimension.

Environment
Internal & External

See Also
UF_DRF_set_appended_text
UF_DRF_free_appended_text
UF_DRF_appended_text_t

History
Created in V17.0

Required License(s)
gateway

```
int UF_DRF_ask_appended_text
(
    tag_t dimension,
    int * num_text,
    UF_DRF_appended_text_p_t * appended_text
)
```

tag_t	**dimension**	Input	Dimension object to query
int *	**num_text**	Output	Number of appended text
UF_DRF_appended_text_p_t *	**appended_text**	Output to UF_*free*	Appended text. Please use UF_DRF_free_appended_text to free the data returned.

UF_DRF_ask_areafill_data (view source)

Defined in: uf_drf.h

Overview
Finds the area fill data for the specified area fill object. The data is contained in the area fill structure.

Environment
Internal & External

See Also
UF_DRF_set_areafill_material
UF_DRF_set_areafill_scale
UF_DRF_set_areafill_angle

Required License(s)
gateway

```
int UF_DRF_ask_areafill_data
(
    tag_t areafill_id,
    UF_DRF_areafill_t * areafill_data
)
```

tag_t	**areafill_id**	Input	Area fill object identifier
UF_DRF_areafill_t *	**areafill_data**	Output	Area fill data

UF_DRF_ask_arrow_data (view source)

Defined in: uf_drf.h

Overview
Gets drafting arrow block parameter data.

Environment
Internal and External

Return
0 = OK
1 = ERROR (a NULL entity found)
2 = ERROR (not a drafting aid nor a dimension entity)
3 = ERROR (no arrow data for this entity)
4 = ERROR (record instance is greater than total count)
5 = ERROR (included angle is 180 degree)
Other = Standard Error code

Required License(s)
gateway

```
int UF_DRF_ask_arrow_data
(
    int data_block [ 10 ] ,
    int * arrow_type,
    int * filled,
    double origin [ 2 ] ,
    double * arrow_angle,
    double * include_angle,
    double * arrow_height,
    double * arrow_length
)
```

int	**data_block [10]**	Input	Only the following array elements are used: [0] = drafting object identifier [4] = record instance - If a label had 4 leaders, it would then have 4 arrows thus possible record instance values of 1, 2, 3 and 4.
int *	**arrow_type**	Output	1 = Closed Arrow 2 = Open Arrow 3 = Arch Cross 4 = Dot 5 = Origin symbol 6 = None

int *	**filled**	Output	The arrow head is either filled or open: 0 = open 1 = filled
double	**origin [2]**	Output	The x and y coordinates of the arrowhead: [0] = x-coordinate [1] = y-coordinate
double *	**arrow_angle**	Output	The angle of the arrow line (degrees)
double *	**include_angle**	Output	The arrowhead included angle (degrees)
double *	**arrow_height**	Output	The height of the arrowhead. If the arrowhead type is 4 (dot), then this is the diameter of the dot.
double *	**arrow_length**	Output	The length of the arrowhead

UF_DRF_ask_assoc_exp (view source)

Defined in: uf_drf.h

Overview

Gets the expression, if it exists, associated to the specified dimension. If there is an expression associated with the specified dimension, this function returns its tag. If there is no expression associated with the specified dimension, a NULL_TAG is returned.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following:
UF_DRF_NO_ERRORS - No error
UF_DRF_null_object - The object tag is NULL.
UF_DRF_invalid_object - The object is invalid for this function.
UF_err_program_not_initialized - Open API has not been initialized.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

gateway

```
int UF_DRF_ask_assoc_exp
(
    tag_t object_tag,
    tag_t * exp_tag
)
```

tag_t	**object_tag**	Input	The tag of a dimension.
tag_t *	**exp_tag**	Output	The tag of the expression associated to the specified input object tag.

UF_DRF_ask_associative_origin (view source)

Defined in: uf_drf.h

Overview
The following function will query the associative origin information for the annotation.

Environment
Internal and External

See Also
UF_DRF_has_associative_origin
UF_DRF_set_associative_origin

History
Created in V17.0

Required License(s)
gateway

```
int UF_DRF_ask_associative_origin
(
    tag_t drafting_entity,
    UF_DRF_associative_origin_p_t * origin_data,
    double origin [ 3 ]
)
```

tag_t	**drafting_entity**	Input	Dimension or drafting object to query.
UF_DRF_associative_origin_p_t *	**origin_data**	Output to UF_*free*	Data used to define the associative origin. To free this allocated data, please use UF_free.
double	**origin [3]**	Output	Origin of the annotation in absolute coords.

UF_DRF_ask_associativity_data (view source)

Defined in: uf_drf.h

Overview
The following function queries all of the associativities for the annotation object.

Environment
Internal or External

History
Originally released in V19.0

Required License(s)
gateway

```
int UF_DRF_ask_associativity_data
(
```

```
    tag_t object,
    int * num_associativities,
    UF_DRF_object_assoc_data_p_t * associativity_data
)
```

tag_t	**object**	Input	Dimension or drafting object in which to query the associativity information. Valid objects include: all dimension types, labels, id symbols, centerlines (linear, symmetrical, cylindrical, bolt hole, etc.), intersection symbols, and GD&T symbols (with leaders).
int *	**num_associativities**	Output	The number of associativities on the object.
UF_DRF_object_assoc_data_p_t *	**associativity_data**	Output to UF_*free*	An array containing the information for each associativity for that object. Use UF_free to free this array of allocated memory.

UF_DRF_ask_boundaries (view source)

Defined in: uf_drf.h

Overview

Returns the number of boundaries, and the tags associated with the boundaries given the specified input tag.

Environment

Internal and External

See Also

UF_BOUND_ask_boundary_data
Refer to the example

Required License(s)

gateway

```
int UF_DRF_ask_boundaries
(
    tag_t draft_aid_tag,
    int * num_boundaries,
    tag_p_t * boundary_tags
)
```

tag_t	**draft_aid_tag**	Input	Tag of the area fill or crosshatch object
int *	**num_boundaries**	Output	The number of boundaries
tag_p_t *	**boundary_tags**	Output to UF_*free*	The array which contains the tags of the boundaries. This array must be freed by calling UF_free.

UF_DRF_ask_callout_of_annotation (view source)

Defined in: uf_drf.h

Overview
Get the callout that contains a specified annotation

Environment
Internal & External

History
Created in V18.0

Required License(s)
gateway

```
int UF_DRF_ask_callout_of_annotation
(
    tag_t annotation,
    tag_t * callout
)
```

tag_t	**annotation**	Input	Tag of annotation
tag_t *	**callout**	Output	Tag of callout or NULL_TAG if annotation is not part of callout

---

UF_DRF_ask_callout_row_members (view source)

Defined in: uf_drf.h

Overview
Get the feature annotation members of a callout row

Environment
Internal & External

History
Created in V18.0

Required License(s)
gateway

```
int UF_DRF_ask_callout_row_members
(
    tag_t callout,
    int row,
    int * num_members,
    tag_t * * members
)
```

tag_t	**callout**	Input	Tag of callout
int	**row**	Input	Row in callout (starting at zero)
int *	**num_members**	Output	Number of members in row

tag_t * *	**members**	Output to UF_*free*	Callout members

UF_DRF_ask_centerline_info (view source)

Defined in: uf_drf.h

Overview

Returns the centerline type, the number of centerline objects, and the data of the centerline given the specified centerline tag. For circular and bolt circle centerlines, the centerpoint method is always returned for centerline_type. For offset cylindrical centerlines, UF_DRF_offcyl_cline is always returned for centerline_type. Use UF_DRF_ask_associativity_data to query the centerline's associativity data including coordinates of the associated position(s).

Environment

Internal and External

See Also

Refer to the example

Required License(s)

gateway

```
int UF_DRF_ask_centerline_info
(
    tag_t centerline_tag,
    UF_DRF_valid_cline_form_t * centerline_type,
    double centerline_origin [ 3 ] ,
    UF_DRF_centerline_info_p_t * centerline_info
)
```

tag_t	**centerline_tag**	Input	Tag of the centerline
UF_DRF_valid_cline_form_t *	**centerline_type**	Output	Type of centerline (see uf_drf_types.h)
double	**centerline_origin [3]**	Output	Centerline origin
UF_DRF_centerline_info_p_t *	**centerline_info**	Output to UF_*free*	Centerline information. This must be freed by calling UF_DRF_free_centerline.

UF_DRF_ask_chamfer_dimension_data (view source)

Defined in: uf_drf.h

Overview

UF_DRF_ask_chamfer_dimension_data

Description -
ask chamfer dimension preferences

PARAMETERS -
cham_dim_tag, - <I> tag of cham dim

cham_dim_data - <O> cham dim prefs
return - <O> return code:
0 = OK
if not 0 = error code

Environment
Internal and External

History
As of NX4 this function will allocate the symbol_name in the UF_DRF_chamfer_dimension_data structure. It will be up to the user to free the symbol_name.

Required License(s)
gateway

```
int UF_DRF_ask_chamfer_dimension_data
(
    tag_t cham_dim_tag,
    UF_DRF_chamfer_dimension_data_p_t cham_dim_data
)
```

tag_t	**cham_dim_tag**	Input	tag of cham dim
UF_DRF_chamfer_dimension_data_p_t	**cham_dim_data**	Output to UF_*free*	chamfer dimension prefs - Must free the symbol_name with call to UF_free()

UF_DRF_ask_controlling_exp (view source)

Defined in: uf_drf.h

Overview
Get the controlling expression of the drafting object

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_controlling_exp
(
    tag_t object,
    tag_t * exp_id
)
```

tag_t	**object**	Input	drafting object
tag_t *	**exp_id**	Output	Controlling expression of the object

---

UF_DRF_ask_controlling_member_of_callout (view source)

Defined in: uf_drf.h

Overview
Get the controlling member of a callout. This member is used for positioning the entire callout and, if deleted, causes the entire callout to be deleted.

Environment
Internal & External

History
Created in V18.0

Required License(s)
gateway

```
int UF_DRF_ask_controlling_member_of_callout
(
    tag_t callout,
    tag_t * controlling_member
)
```

tag_t	**callout**	Input	Tag of callout
tag_t *	**controlling_member**	Output	Controlling member

UF_DRF_ask_custom_symbol_angle (view source)

Defined in: uf_drf.h

Overview
Ask the angle for a custom symbol instance.

Environment
Internal and External

History
Originally released in v19.0

Required License(s)
gateway

```
int UF_DRF_ask_custom_symbol_angle
(
    tag_t symbol_tag,
    double * angle
)
```

tag_t	**symbol_tag**	Input	tag of a custom symbol instance
double *	**angle**	Output	double value of the angle

UF_DRF_ask_custom_symbol_attach_locations (view source)

Defined in: uf_drf.h

Overview

This function will query the leader attachment locations for a custom symbol. The array indices are as follows:

locations[0] = left leader attachment location
locations[1] = right leader attachment location

If there is no leader on a given side, then the tag will be NULL_TAG.

Returns

UF_DRF_NO_ERRORS

Environment

Internal & External

History

Created in NX3.0.2

Required License(s)

gateway

```
int UF_DRF_ask_custom_symbol_attach_locations
(
    tag_t symbol,
    tag_t locations [ 2 ]
)
```

tag_t	**symbol**	Input	The tag of the custom symbol
tag_t	**locations [2]**	Output	An array of 2 tags corresponding to the left and right leader attachment locations

---

UF_DRF_ask_custom_symbol_leader (view source)

Defined in: uf_drf.h

Overview

The following function will query the leader of a custom symbol. If the symbol has no leader, the input variable will be set to NULL. The caller of the function is responsible for freeing the memory for the leader by using UF_DRF_free_leader_data.

NOTE: Since this function exposes components of the custom symbol that are otherwise hidden, this function should only be called by internal NX callers which need the Open API (e.g. translators)

Returns

UF_DRF_NO_ERRORS
UF_DRF_invalid_object
UF_DRF_NOT_DRAFTING_OBJECT
UF_DRF_INVALID_SYMBOL_TYPE
UF_err_bad_parameter_number_2

Environment

Internal & External

History
Created in NX3.0.2

Required License(s)
gateway

int UF_DRF_ask_custom_symbol_leader
(
tag_t symbol,
UF_DRF_leader_data_p_t * leader_data
)

tag_t	**symbol**	Input	The tag of the custom symbol
UF_DRF_leader_data_p_t *	**leader_data**	Output	The custom symbol's leader data

---

UF_DRF_ask_custom_symbol_name (view source)

Defined in: uf_drf.h

Overview
This function returns the name of custom symbol.

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX5.0.3

Required License(s)
gateway

int UF_DRF_ask_custom_symbol_name
(
tag_t custom_symbol,
char* * symbol_name
)

tag_t	**custom_symbol**	Input	Custom symbol
char* *	**symbol_name**	Output to UF_*free*	Custom symbol name Must be freed by calling UF_free

---

UF_DRF_ask_custom_symbol_scale (view source)

Defined in: uf_drf.h

Overview

Ask the symbol scale for a custom symbol instance.

Environment
Internal and External

History
Originally released in v19.0

Required License(s)
gateway

int UF_DRF_ask_custom_symbol_scale
(
tag_t symbol_tag,
double * scale
)

tag_t	**symbol_tag**	Input	tag of a custom symbol instance
double *	**scale**	Output	double value of the symbol scale

UF_DRF_ask_diameter_radius_preferences (view source)

Defined in: uf_drf.h

Overview
Returns preferences for the display of radial dimensions.

Environment
Internal & External

See Also
UF_DRF_set_diameter_radius_preferences
UF_DRF_units_diameter_radius_preferences_t

History
Originally released in V16.0

Required License(s)
gateway

int UF_DRF_ask_diameter_radius_preferences
(
UF_DRF_diameter_radius_preferences_t * diameter_radius_preferences
)

UF_DRF_diameter_radius_preferences_t *	**diameter_radius_preferences**	Input / Output	pointer to preferences structure to be populated with the diameter/radius preferences

---

UF_DRF_ask_dim_appended_text_space_factor (view source)

Defined in: uf_drf.h

Overview

Ask the factor for the space between the dimension and the appended text for the specified dimension.
This factor controls the spacing between the before appended text and the next piece of dimension text to its right,
and the spacing between the after appended text and the next piece of dimension text to its left.
This factor is applied to the appended text character size to determine the spacing.

Environment

Internal and External

History

NX 2.0 release.

Required License(s)

gateway

```
int UF_DRF_ask_dim_appended_text_space_factor
(
    tag_t dimension,
    double * space_factor
)
```

tag_t	**dimension**	Input	dimension tag
double *	**space_factor**	Output	factor for spacing between dimension and appended text

---

UF_DRF_ask_dim_dim_line_space_factor (view source)

Defined in: uf_drf.h

Overview

Ask the space factor for the space between the dimension and the dimension line for the specified dimension.
This factor controls the spacing between the bottom most piece of dimension text and the dimension line only
when the text orientation is UF_DRF_DIMENSION_TEXT_OVER_DIMENSION_LINE.
This factor is applied to the dimension text character size to determine the spacing.

Environment

Internal and External

History

NX 2.0 release.

Required License(s)

gateway

```
int UF_DRF_ask_dim_dim_line_space_factor
(
    tag_t dimension,
    double * space_factor
)
```

tag_t	**dimension**	Input	dimension tag
double *	**space_factor**	Output	factor for spacing between dimension and dimension line

---

UF_DRF_ask_dim_info (view source)

Defined in: uf_drf.h

Overview

Returns the dimension subtype, number of objects, data of objects, associated text, and dimension origin given the specified dimension tag. The origin returned will be associated with the WCS plane when the annotation was created. Use UF_DRF_ask_origin to get the origin in absolute space.
Use UF_DRF_ask_associativity_data to query the dimension's associativity data including coordinates of the associated position(s).

Environment

Internal and External

See Also

Refer to the example

Required License(s)

gateway

```
int UF_DRF_ask_dim_info
(
    tag_t dim_tag,
    int * dim_subtype,
    double dim_origin [ 3 ] ,
    UF_DRF_dim_info_p_t * dim_info
)
```

tag_t	**dim_tag**	Input	Tag of the dimension
int *	**dim_subtype**	Output	Dimension subtype
double	**dim_origin [3]**	Output	Dimension origin
UF_DRF_dim_info_p_t *	**dim_info**	Output to UF_*free*	Dimension information. This must be freed by calling UF_DRF_free_dimension.

UF_DRF_ask_dim_inspection_type (view source)

Defined in: uf_drf.h

Overview

Routine -
UF_DRF_ask_dim_inspection_type

Description -
Ask dimension inspection type preference.

Input - Dimension tag.

Output -
UF_DRF_inspection_type_t inspection_type

Return -
0 = No error
Error code if not zero.
UF_DRF_invalid_object - if the dimension is invalid
UF_err_program_not_initialized

Required License(s)
gateway

int UF_DRF_ask_dim_inspection_type
(
tag_t dim_tag,
UF_DRF_inspection_type_t * inspection_type
)

tag_t	**dim_tag**	Input	tag of dimension object
UF_DRF_inspection_type_t *	**inspection_type**	Output	Inspection type option

---

UF_DRF_ask_dim_reference_type (view source)

Defined in: uf_drf.h

Overview
Routine -
UF_DRF_ask_dim_reference_type

Description -
Ask dimension reference type preference.

Input - Dimension tag.

Output -
UF_DRF_text_above_leader_t option

Return -
0 = No error
Error code if not zero. (including if given object is not dimension)
UF_err_program_not_initialized

Required License(s)
gateway

int UF_DRF_ask_dim_reference_type
(
tag_t dim_tag,
UF_DRF_reference_symbol_type_t * ref_type
)

tag_t	**dim_tag**	Input	tag of dimension object
UF_DRF_reference_symbol_type_t *	**ref_type**	Output	reference type option

UF_DRF_ask_dim_tolerance_text_space_factor (view source)

Defined in: uf_drf.h

Overview
Ask the factor for the space between the dimension and the tolerance text for the specified dimension.
This factor controls the spacing between the dimension text and the after tolerance text.
This factor is applied to the tolerance text character size to determine the spacing.

Environment
Internal and External

History
NX 2.0 release.

Required License(s)
gateway

```
int UF_DRF_ask_dim_tolerance_text_space_factor
(
    tag_t dimension,
    double * space_factor
)
```

tag_t	**dimension**	Input	dimension tag
double *	**space_factor**	Output	factor for spacing between dimension and tolerance text

---

UF_DRF_ask_dimension_preferences (view source)

Defined in: uf_drf.h

Overview
Returns dimension preferences for arrow and line formatting, type of
placement, tolerance and precision, and text orientation.

Environment
Internal & External

See Also
UF_DRF_set_dimension_preferences
UF_DRF_dimension_preferences_t

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_dimension_preferences
(
    UF_DRF_dimension_preferences_t * dimension_preferences
)
```

UF_DRF_dimension_preferences_t *	**dimension_preferences**	Input / Output	pointer to preferences structure to be populated with the dimension preferences

---

UF_DRF_ask_dimension_preferences1 (view source)

Defined in: uf_drf.h

Overview

In order to provide appropriate .NET binding for UF_DRF_ask_dimension_preferences, UF_DRF_ask_dimension_preferences1 is introduced.

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

For docuementation, refer to documentation of UF_DRF_ask_dimension_preferences.

History

Originally released in NX7.5

Required License(s)

gateway

```
int UF_DRF_ask_dimension_preferences1
(
    UF_DRF_dimension_preferences1_t * * dimension_preferences
)
```

UF_DRF_dimension_preferences1_t * *	**dimension_preferences**	Output to UF_*free*	pointer to preferences structure to be populated with the dimension preferences Must be freed using UF_DRF_free_dimension_preferences1

UF_DRF_ask_dimension_set_offset (view source)

Defined in: uf_drf.h

Overview

Return the offset of the dimension in a dimension set. If a dimension set tag is given, the offset of the first dimension in the set is returned.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1 - if the dimension tag is invalid.

Environment

Internal and External

History

Originally released in v18.0

Required License(s)
gateway

int UF_DRF_ask_dimension_set_offset
(
tag_t dimension,
double * offset
)

tag_t	**dimension**	Input	Tag of the given dimension.
double *	**offset**	Output	Dimension set offset of the given dimension.

UF_DRF_ask_dimension_text (view source)

Defined in: uf_drf.h

Overview
This function returns the dimension text - both the main text and the dual text. The returned text may contain just the computed dimension value or manual text including the control characters.

Note that the text doesn't contain the tolerance text, appended text or text for flags like inspection/reference.

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX5.0.3

Required License(s)
drafting

int UF_DRF_ask_dimension_text
(
tag_t dimension,
int* num_main_text,
char* * * main_text,
int* num_dual_text,
char* * * dual_text
)

tag_t	**dimension**	Input	Dimension
int*	**num_main_text**	Output	Number of main dimension text lines
char* * *	**main_text**	Output to UF_*free*	Main dimension text lines
int*	**num_dual_text**	Output	Number of dual dimension text lines
char* * *	**dual_text**	Output to UF_*free*	Dual dimension text lines

UF_DRF_ask_dimensions_of_set (view source)

Defined in: uf_drf.h

Overview
Return the dimensions in a dimension set.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_null_object - if the dimension set is null
UF_DRF_invalid_object - if the dimension set is invalid
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
gateway

```
int UF_DRF_ask_dimensions_of_set
(
    tag_t dimension_set,
    tag_t * * sub_dimensions,
    int * num
)
```

tag_t	**dimension_set**	Input	Tag of the given dimension set.
tag_t * *	**sub_dimensions**	Output to UF_*free*	Point to the tag array of the dimensions in a given set.
int *	**num**	Output	number of dimensions in the array.

UF_DRF_ask_dogleg_info (view source)

Defined in: uf_drf.h

Overview
Returns the dogleg type, distance in WCS coordinates, and angle in degrees given the specified ordinate dimension tag.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_dogleg_info
(
    tag_t orddim_tag,
    UF_DRF_dogleg_info_t * dogleg_info
)
```

tag_t	**orddim_tag**	Input	Tag of ordinate origin
UF_DRF_dogleg_info_t *	**dogleg_info**	Output	Ordinate dimension dog leg information. The calling program must allocate a UF_DRF_dogleg_info_t structure and pass in the pointer to that structure.

UF_DRF_ask_draft_aid_text_info (view source)

Defined in: uf_drf.h

Overview
Returns character size, text angle, origin, length, height, and distance from the origin to the top of a given text block and text strings given the specified drafting aid tag.

Environment
Internal and External

See Also
Refer to the example

History
V17.0 Add full_string and full_num_char to UF_DRF_draft_aid_text_t to provide access to all of the full drafting aid text

Required License(s)
gateway

```
int UF_DRF_ask_draft_aid_text_info
(
    tag_t draft_aid_tag,
    int * num_text,
    UF_DRF_draft_aid_text_info_t * * text_info
)
```

tag_t	**draft_aid_tag**	Input	Tag of drafting aid object
int *	**num_text**	Output	Number of text strings
UF_DRF_draft_aid_text_info_t * *	**text_info**	Output to UF_*free*	Pointer to data structure which contains drafting aid text information (see uf_drf_types.h) of the drafting aid object. This must be freed by calling UF_DRF_free_text.

UF_DRF_ask_embedded_uds_font_info (view source)

Defined in: uf_drf.h

Overview
This function will return the embedded User Defined Symbol font name and stroke data. "stroke_info" can be freed by making a call to "UF_free".
NOTE: The pen_status from returned stroke_info can be either - draw or move.
UF_DRF_UD_SYMBOL_DRAW means start drawing a new stroke from this position.
UF_DRF_UD_SYMBOL_MOVE means draw the stroke till this position.

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

Environment
Internal and External

History
Created in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_embedded_uds_font_info
(
    tag_t symbol_font_tag,
    char symbol_name [ 9 ] ,
    int * num_of_strokes,
    UF_DRF_stroke_info_t * * stroke_info
)
```

tag_t	**symbol_font_tag**	Input	Tag of Symbol Font
char	**symbol_name [9]**	Output	Name of Symbol Font
int *	**num_of_strokes**	Output	Total number of stroke
UF_DRF_stroke_info_t * *	**stroke_info**	Output to UF_*free*	Stroke Data

---

UF_DRF_ask_folded_radius_info (view source)

Defined in: uf_drf.h

Overview
Returns the fold location in WCS coordinates and the fold angle in degrees given the specified folded radius tag.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_folded_radius_info
(
```

```
    tag_t frdim_tag,
    UF_DRF_foldedradius_dim_info_t * frdim_info
)
```

tag_t	**frdim_tag**	Input	Tag of folded radius dimension
UF_DRF_foldedradius_dim_info_t *	**frdim_info**	Output	Folded radius dimension information (see UF_DRF_foldedradius_dim_info_t)

UF_DRF_ask_gdt_symbol_info (view source)

Defined in: uf_drf.h

Overview
Given the GD&T Symbol tag, the function returns the associated text string, object origin, leader type and attachment type, and data of object to attach leader.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_gdt_symbol_info
(
    tag_t gdt_symbol_tag,
    double gdt_symbol_origin [ 3 ] ,
    UF_DRF_gdt_symbol_info_p_t * gdt_symbol_info
)
```

tag_t	**gdt_symbol_tag**	Input	Tag of GD&T Symbol
double	**gdt_symbol_origin [3]**	Output	GD&T Symbol origin
UF_DRF_gdt_symbol_info_p_t *	**gdt_symbol_info**	Output to UF_*free*	GD&T Symbol information. This must be freed by calling UF_DRF_free_gdtsymbol.

UF_DRF_ask_hatch_fill_preferences (view source)

Defined in: uf_drf.h

Overview
Returns the preferences for crosshatching and area fill

Environment
Internal & External

See Also
UF_DRF_set_hatch_fill_preferences

History
Originally released in V16.0

Required License(s)
gateway

int UF_DRF_ask_hatch_fill_preferences
(
UF_DRF_hatch_fill_preferences_t * hatch_fill_preferences
)

UF_DRF_hatch_fill_preferences_t *	**hatch_fill_preferences**	Input / Output	pointer to structure to be populated with the hatch/fill preferences

---

UF_DRF_ask_id_symbol_geometry (view source)

Defined in: uf_drf.h

Overview
Return the geometry for an id symbol.

Environment
Internal and External

History
Returns the geometry for an id symbol.

Required License(s)
gateway

int UF_DRF_ask_id_symbol_geometry
(
tag_t id_symbol,
int * num_lines,
double * * lines,
int * num_arcs,
UF_DRF_arc_info_p_t * arcs
)

tag_t	**id_symbol**	Input	The ID symbol to query
int *	**num_lines**	Output	The number of lines in the ID symbol
double * *	**lines**	Output to UF_*free*	Array of lines - 3d start, 3d end, the first line is lines[0] - lines[5] the second line is lines[6] - lines[11] ... the nth line (n counting from 1) is lines[6 (n - 1)] - lines[6 n - 1]. Line end points are in the coordinate system of the id symbol, as are other coordinates returned by UF_DRF routines. If the idysmbol is not on a drawing, the coordinates in absolute can be obtained by multiplying by the transpose of

			the matrix returned by the function UF_CSYS_ask_matrix_values for the matrix returned by UF_CSYS_ask_matrix_of_object. This array must be freed by calling UF_free.
int *	**num_arcs**	Output	the number of arcs in the ID symbol
UF_DRF_arc_info_p_t *	**arcs**	Output to UF_*free*	array of arcs (center is in coordinate system of the id symbol). This array must be freed by calling UF_free.

UF_DRF_ask_id_symbol_info (view source)

Defined in: uf_drf.h

Overview
Given the ID Symbol tag, this function returns the ID Symbol subtype, text and leader information, and the data of object to attach leader.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_id_symbol_info
(
    tag_t id_symbol_tag,
    UF_DRF_id_symbol_type_t * id_symbol_type,
    double id_symbol_origin [ 3 ] ,
    UF_DRF_id_symbol_info_p_t * id_symbol_info
)
```

tag_t	**id_symbol_tag**	Input	Tag of ID Symbol
UF_DRF_id_symbol_type_t *	**id_symbol_type**	Output	ID Symbol type
double	**id_symbol_origin [3]**	Output	ID Symbol origin
UF_DRF_id_symbol_info_p_t *	**id_symbol_info**	Output to UF_*free*	ID Symbol information. This must be freed by calling UF_DRF_free_idsymbol.

UF_DRF_ask_id_symbol_type (view source)

Defined in: uf_drf.h

Overview
Given the ID Symbol tag, this function returns the ID Symbol type.

Environment

Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_DRF_ask_id_symbol_type
(
tag_t id_symbol_tag,
UF_DRF_id_symbol_type_t * id_symbol_type
)

tag_t	**id_symbol_tag**	Input	Tag of ID Symbol
UF_DRF_id_symbol_type_t *	**id_symbol_type**	Output	ID Symbol type (see uf_drf_types.h)

UF_DRF_ask_image_data (view source)

Defined in: uf_drf.h

Overview
Get the image data

Returns
UF_DRF_NO_ERRORS if the image query was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

int UF_DRF_ask_image_data
(
tag_t image,
UF_DRF_image_data_t * data
)

tag_t	**image**	Input	Image to query

UF_DRF_image_data_t *	**data**	Output to UF_*free*	Image data (see uf_drf_types.h) The image data will need to be freed using UF_DRF_free_image_data

---

UF_DRF_ask_label_info (view source)

Defined in: uf_drf.h

Overview
Given the label tag, this function returns the number of lines of text, associated text string, object origin, leader type, and data of object to attach leader.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_label_info
(
    tag_t label_tag,
    double label_origin [ 3 ] ,
    UF_DRF_label_info_p_t * label_info
)
```

tag_t	**label_tag**	Input	Tag of label
double	**label_origin [3]**	Output	Label origin in the coordinate system of the drafting object.
UF_DRF_label_info_p_t *	**label_info**	Output to UF_*free*	Label information. This must be freed by calling UF_DRF_free_label.

UF_DRF_ask_lettering_preferences (view source)

Defined in: uf_drf.h

Overview
Returns the lettering preferences for dimension, appended, tolerance and general (notes, id symbols, etc,) text.

Environment
Internal & External

See Also
UF_DRF_set_lettering_preferences
UF_DRF_lettering_preferences_t

History

Originally released in V16.0
Horizontal text justification preference and GD&T frame height factor preference added in V17.0.
Dimension/Appended Text Spacing Factor,
Dimension /Tolerance Text Spacing Factor, and
Dimension/Dimension Line Spacing Factor added in NX 2.0.

Required License(s)
gateway

int UF_DRF_ask_lettering_preferences
(
UF_DRF_lettering_preferences_t * lettering_preferences
)

UF_DRF_lettering_preferences_t *	**lettering_preferences**	Input / Output	pointer to preferences structure to be populated with the lettering preferences

---

UF_DRF_ask_line_arrow_preferences (view source)

Defined in: uf_drf.h

Overview
Returns preferences that apply to leaders, arrows and extension lines for both dimensions and other annotations

Environment
Internal & External

See Also
UF_DRF_set_line_arrow_preferences
UF_DRF_line_arrow_preferences_t

History
Originally released in V16.0

Required License(s)
gateway

int UF_DRF_ask_line_arrow_preferences
(
UF_DRF_line_arrow_preferences_t * line_arrow_preferences
)

UF_DRF_line_arrow_preferences_t *	**line_arrow_preferences**	Input / Output	pointer to preferences structure to be populated with the line/arrow preferences

---

UF_DRF_ask_narrow_dimension_data (view source)

Defined in: uf_drf.h

Overview
Ask narrow dimension parameters.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form - if the dimension given is not linear dimension.
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
gateway

```
int UF_DRF_ask_narrow_dimension_data
(
    tag_t dimension_tag,
    UF_DRF_narrow_dimension_info_p_t narrow_data
)
```

tag_t	**dimension_tag**	Input	Object tag of a linear dimension
UF_DRF_narrow_dimension_info_p_t	**narrow_data**	Output	Data of narrow dimension preferences

---

UF_DRF_ask_number_blocks (view source)

Defined in: uf_drf.h

Overview
Given the drafting aid object tag, this function returns the number of blocks for this drafting aid.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_number_blocks
(
    tag_t annotation_tag,
    int num_block [ 5 ]
)
```

tag_t	**annotation_tag**	Input	Tag of the drafting aid object

int	**num_block [5]**	Output	Five integer array: UF_DRF_ASSOCIATIVITY_BLOCK num_block[UF_DRF_LINE_BLOCK] is number of line blocks num_block[UF_DRF_ARCS_BLOCK] is number of arc blocks num_block[UF_DRF_TEXT_BLOCK] is number of text blocks num_block[UF_DRF_ARROWS_BLOCK] is number of arrow blocks num_block[UF_DRF_ASSOCIATIVITY_BLOCK] is number of associativity blocks

---

UF_DRF_ask_number_rows_in_callout (view source)

Defined in: uf_drf.h

Overview
Get the number of rows in a callout

Environment
Internal & External

History
Created in V18.0

Required License(s)
gateway

```
int UF_DRF_ask_number_rows_in_callout
(
    tag_t callout,
    int * num_rows
)
```

tag_t	**callout**	Input	Tag of callout
int *	**num_rows**	Output	Number of rows in callout

UF_DRF_ask_obj_suppress_pre_zeros (view source)

Defined in: uf_drf.h

Overview
Determines if the suppress preceding zeros option for an angular dimension is set.

Environment
Internal and External

See Also
UF_DRF_ask_obj_suppress_zeros

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_DRF_ask_obj_suppress_pre_zeros
(
tag_t object,
logical * option
)

tag_t	**object**	Input	Tag id of angular dimension object
logical *	**option**	Output	Suppress preceding zeros mode

UF_DRF_ask_obj_text_above_ldr (view source)

Defined in: uf_drf.h

Overview
Returns the text above leader attribute for a label or dimension.

If a value of UF_DRF_NO_TEXT_ABOVE_LEADER is returned, the text is NOT displayed above the leader stub. The placement of the leader stub relative to the text is controlled by the vertical text justification.
This can be queried with UF_DRF_ask_object_preferences.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_DRF_ask_obj_text_above_ldr
(
tag_t object,
UF_DRF_text_above_leader_t * option
)

tag_t	**object**	Input	tag of label or dimension
UF_DRF_text_above_leader_t *	**option**	Output	text above leader attribute for the specified label or dimension

---

UF_DRF_ask_object_preferences (view source)

Defined in: uf_drf.h

Overview
Given the annotation tag, this function retrieves and checks drafting parameters from an existing annotation.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following:
UF_DRF_NO_ERRORS - No error
UF_DRF_NOT_DRAFTING_OBJECT
UF_DRF_crosshatch_file_not_found

Environment

Internal and External

See Also

UF_DRF_set_object_preferences
UF_DRF_ask_preferences
UF_DRF_set_preferences
UF_DRF_ask_ang_obj_units_format
Refer to the example
See drafting parameters

History

Original release was in V13.0. This function replaces uc/uf5550.
In V15.0, the maximum byte length for both radius_value and diameter_value was increased from 6 to 27 to accomodate internationalized text.
Updated in V16.0 to return UF_get_fail_message error codes and if the input object is not a crosshatch object, returns mpi[31] = -999 for the material index.
If UF_DRF_set_preferences is called with a value of -999 for the material index, the material preferences will not be changed.
mpr[51], area fill tolerance, is now obsolete; use mpr[13] instead.
In V17.0 flexible appended text locations were implemented for dimensions. If the appended text is located at one of these enhanced locations, -999 will be returned. Use the new UF_DRF_ask_appended_text to query the appended text.
In V17.0, a separate preference controls the angular nominal and tolerance units format. This routine will return only the nominal angular format when queried.
In NX2.0, occurrence support was added.

Required License(s)

gateway

```
int UF_DRF_ask_object_preferences
(
    tag_t drf_object_tag,
    int mpi [ 100 ] ,
    double mpr [ 70 ] ,
    char radius_val [ 27 ] ,
    char diameter_val [ 27 ]
)
```

tag_t	**drf_object_tag**	Input	Drafting object Identifier
int	**mpi [100]**	Output	MPI Array [100 elements] The size of this array is defined by NUM_INT_PARAMS
double	**mpr [70]**	Output	MPR Array [70 elements] The size of this array is defined by NUM_REAL_PARAMS
char	**radius_val [27]**	Output	Radius Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char radius_val[27];
char	**diameter_val [27]**	Output	Diameter Symbol String. This can be at most six characters, however due to internal

requirements, the buffer must be allocated as char diameter_val[27];

UF_DRF_ask_objects_controlled_by_exp (view source)

Defined in: uf_drf.h

Overview
Find all drafting objects which are controlled by a given expression

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_objects_controlled_by_exp
(
    tag_t exp_id,
    int * num_objs,
    tag_p_t * objects
)
```

tag_t	**exp_id**	Input	Object identifier of the expression.
int *	**num_objs**	Output	Number of objects controlled by the expression.
tag_p_t *	**objects**	Output to UF_*free*	Array of objects controlled by the expression. The caller is responsible for freeing this array by calling UF_free.

UF_DRF_ask_ordorigin_info (view source)

Defined in: uf_drf.h

Overview
Returns the data of the associated object, arrow and dimension line display status, origin symbol display status, and user supplied object name given the specified ordinate origin tag.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_ordorigin_info
(
```

```
    tag_t ordorigin_tag,
    UF_DRF_orddisp_info_t * origin_disp,
    int * num_assoc,
    UF_DRF_assoc_info_t * * assoc_objects
)
```

tag_t	**ordorigin_tag**	Input	Tag of ordinate origin
UF_DRF_orddisp_info_t *	**origin_disp**	Output	Ordinate dimension origin display in formation (see UF_DRF_orddisp_info_t)
int *	**num_assoc**	Output	Number of associated objects
UF_DRF_assoc_info_t * *	**assoc_objects**	Output to UF_*free*	Array of associated object information. This array must be freed by calling UF_free.

UF_DRF_ask_origin (view source)

Defined in: uf_drf.h

Overview
Return the origin of the annotation object.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1

Environment
Internal and External

History
Written in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_origin
(
    tag_t annotation,
    double origin [ 3 ]
)
```

tag_t	**annotation**	Input	Tag of the annotation.
double	**origin [3]**	Output	Origin for the annotation. If the annotation is on the drawing, the origin is with respect to drawing coordinates. Otherwise, the origin is with respect to model coordinates.

UF_DRF_ask_parent_of_inherited_pmi (view source)

Defined in: uf_drf.h

Overview
This function returns the parent of the given inherited PMI. The parent is the PMI Display Instance in modeling.

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX5.0

Required License(s)
drafting

```
int UF_DRF_ask_parent_of_inherited_pmi
(
    tag_t inherited_pmi,
    tag_p_t parent
)
```

tag_t	**inherited_pmi**	Input	The tag of the inherited PMI
tag_p_t	**parent**	Output	Parent of the inherited PMI

UF_DRF_ask_plot_drawing_images (view source)

Defined in: uf_drf.h

Overview
Ask whether or not the raster images on the drawing sheets will be plotted when drawing sheets are plotted.

Returns
UF_DRF_NO_ERRORS if the preference query was successful

Environment
Internal and External

See Also
UF_DRF_create_image_from_file
UF_DRF_create_image

History
This function was originally released in NX3.0.

Required License(s)
gateway

```
int UF_DRF_ask_plot_drawing_images
(
    logical * plot_images
)
```

logical *	**plot_images**	Output	TRUE if set to plot raster images on drawing sheets FALSE if set to not plot raster images on drawing sheets

---

UF_DRF_ask_preferences (view source)

Defined in: uf_drf.h

Overview

Sets arrays and character strings to the current settings of the drafting parameters.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following:
UF_DRF_NO_ERRORS - No error
UF_DRF_crosshatch_file_not_found
UF_DRF_INVALID_CROSSHATCH_FILE_FORMAT

Environment

Internal and External

See Also

UF_DRF_set_preferences
UF_DRF_ask_object_preferences
UF_DRF_set_object_preferences
Refer to the example
See drafting parameters

History

Original release was in V13.0. This function replaces uc5520.
Updated in V15 to increase the size of the character strings.
Updated in V16.0 to return UF_get_fail_message error codes.
In V16.0, there are separate text angle preferences for dimension text and drafting aid text. This function returns the dimension text angle. Use UF_DRF_ask_lettering_preferences to get the drafting aid text angle.
mpr[51], area fill tolerance, is now obsolete; use mpr[13] instead.
In V17.0, a separate preference control the angular nominal and tolerance units format. This routine will return the only the nominal angular format when queried.

Required License(s)

gateway

```
int UF_DRF_ask_preferences
(
    int mpi [ 100 ] ,
    double mpr [ 70 ] ,
    char radius_value [ 27 ] ,
    char diameter_value [ 27 ]
)
```

int	**mpi [100]**	Output	MPI Array [100 elements] The size of this array is defined by NUM_INT_PARAMS
double	**mpr [70]**	Output	MPR Array [70 elements] The size of this array is defined by NUM_REAL_PARAMS

char	**radius_value [27]**	Output	Radius Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char radius_value[27];
char	**diameter_value [27]**	Output	Diameter Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char diameter_value[27];

UF_DRF_ask_retain_color_font_width (view source)

Defined in: uf_drf.h

Overview
Gets the color, font and widths used for retained annotations.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized

Environment
Internal and External

Required License(s)
gateway

```
int UF_DRF_ask_retain_color_font_width
(
    int * color,
    int * font,
    int * width
)
```

int *	**color**	Output	Color used for retained annotations
int *	**font**	Output	Font used for retained annotations
int *	**width**	Output	Line width used for retained annotations

---

UF_DRF_ask_retained_state (view source)

Defined in: uf_drf.h

Overview
Gets the retained state for annotations whose dependencies have expired.
If UF_DRF_KEEP_RETAINED_ANNOTATIONS is returned, annotations are retained.
If UF_DRF_DELETE_RETAINED_ANNOTATIONS is returned, annotations are deleted.

Return
Return code:
0 = No error

not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized

Environment
Internal and External

Required License(s)
gateway

int UF_DRF_ask_retained_state
(
UF_DRF_retained_state_t * state
)

UF_DRF_retained_state_t *	**state**	Output	Behavior state for retained annotations. Either 'UF_DRF_KEEP_RETAINED_ANNOTATIONS' or 'UF_DRF_DELETE_RETAINED_ANNOTATIONS'

UF_DRF_ask_sbf_file (view source)

Defined in: uf_drf.h

Overview
Returns the current symbol font definition file name.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_DRF_ask_sbf_file
(
char sbf_name [UF_CFI_MAX_FILE_NAME_BUFSIZE]
)

char	**sbf_name [UF_CFI_MAX_FILE_NAME_BUFSIZE]**	Output	current symbol font definition name (blank if none)

UF_DRF_ask_set_of_dimension (view source)

Defined in: uf_drf.h

Overview
Return the dimension set which the given dimension belongs to.

Return
Return code:
0 = No error

not 0 = Error code
Possible return codes can include the following
UF_DRF_null_object - if the dimension is null
UF_DRF_invalid_object - if the dimension is invalid
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
gateway

int UF_DRF_ask_set_of_dimension
(
tag_t dimension,
tag_t * dimension_set
)

tag_t	**dimension**	Input	Tag of the given dimension.
tag_t *	**dimension_set**	Output	Tag of the dimension set which the given dimension belongs to.

UF_DRF_ask_suppress_pre_zeros (view source)

Defined in: uf_drf.h

Overview
Determines if the global preference for suppress preceding zeros is set.

Environment
Internal and External

See Also
UF_DRF_ask_units_format_preferences
Refer to the example

History
This function was originally released in V15.0.

Required License(s)
gateway

int UF_DRF_ask_suppress_pre_zeros
(
logical * option
)

logical *	**option**	Output	Suppress preceding zeros mode

UF_DRF_ask_suppress_view_update (view source)

Defined in: uf_drf.h

Overview
Query the current value of the Suppress View Update preference. This preference is an environment setting and is not specific to a part. If the preference is TRUE, then functions which perform implicit drawing updates, will not update the drawing member views.

Environment
Internal & External:

See Also
UF_DRF_set_suppress_view_update

Required License(s)
gateway

```
int UF_DRF_ask_suppress_view_update
(
    logical * suppress_view_update
)
```

logical *	**suppress_view_update**	Output	the current setting of view update suppression

---

UF_DRF_ask_symbol_data (view source)

Defined in: uf_drf.h

Overview
Returns standalone symbol object data.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_symbol_data
(
    tag_t symbol_tag,
    UF_DRF_symbol_data_t * symbol_data
)
```

tag_t	**symbol_tag**	Input	standalone symbol object tag
UF_DRF_symbol_data_t *	**symbol_data**	Output	symbol data (see uf_drf_types.h).

UF_DRF_ask_symbol_data_from_name (view source)

Defined in: uf_drf.h

Overview

Return the User Defined Symbol factor, length, height, left, and right connecting point, the count of strokes, and stroke data given the specified user defined symbol name

Environment

Internal and External

Required License(s)

gateway

```
int UF_DRF_ask_symbol_data_from_name
(
    const char * sbf_name,
    char * * * symbol_names,
    int * num_symbols,
    UF_DRF_ud_symbol_font_info_p_t * symbol_info
)
```

const char *	**sbf_name**	Input	Name of the sbf file
char * * *	**symbol_names**	Output to UF_*free*	List of symbol names in sbf
int *	**num_symbols**	Output	Number of symbols in the sbf file
UF_DRF_ud_symbol_font_info_p_t *	**symbol_info**	Output to UF_*free*	Symbol data (see uf_drf_types.h) use UF_free () to free memory

---

UF_DRF_ask_symbol_mirror_and_flip (view source)

Defined in: uf_drf.h

Overview

UF_DRF_ask_symbol_mirror_and_flip

Description -
This function ask standalone symbol create mask

PARAMETERS -
symbol_tag <I> Tag of symbol to inquire
mirrored <O> Symbol has mirrored
flip <O> Symbol has flip
return <O> return code:
0 = OK
if not 0 = error code

Required License(s)

gateway

```
int UF_DRF_ask_symbol_mirror_and_flip
(
    tag_t symbol_tag,
    logical * mirrored,
    logical * flip
)
```

tag_t	**symbol_tag**	Input	standalone symbol object tag
logical *	**mirrored**	Output	TRUE symbol is mirrored
logical *	**flip**	Output	TRUE symbol is flip

UF_DRF_ask_symbol_preferences (view source)

Defined in: uf_drf.h

Overview
Returns preferences that apply to ID, User Define, Centerline, Intersection, Target and GD&T symbols.

Environment
Internal & External

See Also
UF_DRF_set_symbol_preferences
UF_DRF_symbol_preferences_t

History
Originally released in V16.0

Required License(s)
gateway

int UF_DRF_ask_symbol_preferences
(
UF_DRF_symbol_preferences_t * symbol_preferences
)

UF_DRF_symbol_preferences_t *	**symbol_preferences**	Input / Output	pointer to preferences structure to be populated with the symbol preferences

---

UF_DRF_ask_symbols_used (view source)

Defined in: uf_drf.h

Overview
Returns symbols used in a drafting object.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_symbols_used
(
    tag_t object_tag,
    int * num_symbol_fonts,
    tag_t * symbol_font_tags
)
```

tag_t	**object_tag**	Input	drafting object tag
int *	**num_symbol_fonts**	Output	number of symbol fonts used
tag_t *	**symbol_font_tags**	Output	tags of the symbol font objects. The caller is responsible for passing in an array large enough to receive all of the symbol font tags.

UF_DRF_ask_text_above_leader (view source)

Defined in: uf_drf.h

Overview

Inquires the current global setting for the text above leader attribute.
This attribute controls displaying text above the leader stub when creating labels and radial type dimensions.

If the value is UF_DRF_NO_TEXT_ABOVE_LEADER, then the text is NOT displayed above the leader stub.
The placement of the leader stub is controlled by the vertical text justification preference which can be queried with UF_DRF_ask_preferences.

Environment

Internal and External

History

This function was originally released in V15.0.

Required License(s)

gateway

```
int UF_DRF_ask_text_above_leader
(
    UF_DRF_text_above_leader_t * option
)
```

UF_DRF_text_above_leader_t *	**option**	Output	text above leader option this option controls displaying text above the leader stub. this option applies to Labels and radial type dimensions (hole, radius, diameter, concentric circle and folded radius) when the Text Alignment is Horizontal or By Angle.

UF_DRF_ask_text_data (view source)

Defined in: uf_drf.h

Overview

Read Drafting Object Text Data:
This routine deciphers the ann_data returned from UF_DRF_ask_ann_data and returns just one text string. This routine can be called once for each text string (or segment) in the data block. The number of segments is returned by UF_DRF_ask_ann_data.

This user function replaces uc5574.

Environment
Internal and External

See Also
UF_DRF_ask_ann_data

Required License(s)
drafting

```
int UF_DRF_ask_text_data
(
    int ip1,
    int ann_data [ 10 ] ,
    char* * cr3,
    int * ir4,
    int * ir5
)
```

int	**ip1**	Input	Segment Number 1 < ip1 < = num_segments returned by UF_DRF_ask_ann_data
int	**ann_data [10]**	Input / Output	Array of ann_data returned by UF_DRF_ask_ann_data
char* *	**cr3**	Output to UF_*free*	Text String Must be freed by calling UF_free
int *	**ir4**	Output	Length of Line in 1/64th's Character Size, Expressed in Integers
int *	**ir5**	Output	Number of Characters in String

UF_DRF_ask_ud_symbol_font_info (view source)

Defined in: uf_drf.h

Overview
Returns the User Defined Symbol factor, length, height, left and right connecting point, the count of strokes, and stroke data given the specified user defined symbol tag.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_DRF_ask_ud_symbol_font_info
(
```

```
    tag_t ud_symbol_tag,
    int * num_symbols,
    UF_DRF_ud_symbol_font_info_p_t * font_info
)
```

tag_t	**ud_symbol_tag**	Input	Tag of User Defined Symbol
int *	**num_symbols**	Output	Number of User Defined Symbol
UF_DRF_ud_symbol_font_info_p_t *	**font_info**	Output to UF_*free*	Data structure which contains the User Defined Symbol font number, left and right connection points, factor, length, height and the count of strokes and stoke data. Use UF_DRF_free_font to deallocate memory when done.

UF_DRF_ask_uds_object_size (view source)

Defined in: uf_drf.h

Overview

The following function returns the user defined symbol length / height or scale / aspect ratio.

Environment

Internal or External

History

Originally released in V19.0

Required License(s)

gateway

```
int UF_DRF_ask_uds_object_size
(
    tag_t object,
    UF_DRF_uds_size_p_t uds_size
)
```

tag_t	**object**	Input	User defined symbols, or annotation with an embedded user defined symbol object (such as a note, label, or GD&T symbol).
UF_DRF_uds_size_p_t	**uds_size**	Output	User defined symbol scale / aspect ratio or length / height parameters.

UF_DRF_ask_units_format_preferences (view source)

Defined in: uf_drf.h

Overview

Returns preferences for the display of linear and angular dimensions values as well as dual dimension format.

Environment

Internal & External

See Also
UF_DRF_set_units_format_preferences
UF_DRF_units_format_preferences_t

History
Originally released in V16.0. In v17.0, the data structure has changed.
An element controlling the units of the tolerance of angular dimension and
an element controlling the zero suppression for angular dimension have been added.
The display leading zeros option has been removed.

Required License(s)
gateway

```
int UF_DRF_ask_units_format_preferences
(
    UF_DRF_units_format_preferences_t * units_format_preferences
)
```

UF_DRF_units_format_preferences_t *	**units_format_preferences**	Input / Output	pointer to preferences structure to be populated with the units/format preferences

---

UF_DRF_ask_vertical_note (view source)

Defined in: uf_drf.h

Overview
This function will return the orientation of a note. A note can be
either horizontal or vertical.

Environment
Internal and External

History
Created in V16.0

Required License(s)
gateway

```
int UF_DRF_ask_vertical_note
(
    tag_t note,
    logical * is_vertical
)
```

tag_t	**note**	Input	Tag of the note.
logical *	**is_vertical**	Output	Orientation of the note: True if the note is vertical False if the note is horizontal If the object is not a note, False is returned.

UF_DRF_ask_weld_symbol (view source)

Defined in: uf_drf.h

Overview
Given the weld symbol tag, this function will get the information regarding the weld symbol.

Returns
0 = No Error
Not 0 = Error
Possible Error Return Code:
UF_DRF_OBJECT_IS_NOT_A_WELD_SYMBOL - Object is not a weld symbol

Environment
Internal and External

See Also
UF_DRF_weld_symbols_t
UF_DRF_label_info_t
UF_DRF_create_weld_symbol

History
Originally released in v18.0

Required License(s)
gateway

```
int UF_DRF_ask_weld_symbol
(
    tag_t weld_symbol_tag,
    double label_origin [ 3 ] ,
    UF_DRF_label_info_p_t * label_info,
    UF_DRF_weld_symbols_p_t symbol_data
)
```

tag_t	**weld_symbol_tag**	Input	tag of the weld symbol for which information is required
double	**label_origin [3]**	Output	label origin
UF_DRF_label_info_p_t *	**label_info**	Output to UF_*free*	Label information. This must be freed by UF_DRF_free_label
UF_DRF_weld_symbols_p_t	**symbol_data**	Output	weld symbol data for the given weld symbol tag

---

UF_DRF_count_text_substring (view source)

Defined in: uf_drf.h

Overview
Find the number of substrings of the given text. A substring is defined as:
A continuous set of characters with similar characteristics
A single symbol e.g. <&1>

Environment

Internal and External

See Also
UF_DRF_get_text_substring
UF_DRF_ask_ann_data
UF_DRF_set_draft_common

Required License(s)
drafting

```
int UF_DRF_count_text_substring
(
    int * segment_number,
    int * ann_data,
    int * number_of_substring
)
```

int *	**segment_number**	Input	The requested text segment on the drafting object.
int *	**ann_data**	Input	The data defining the drafting object. See UF_DRF_ask_ann_data.
int *	**number_of_substring**	Output	The number of substrings in this segment of the drafting object.

UF_DRF_cre_text_block (view source)

Defined in: uf_drf.h

Overview
Create a text block on a drafting aid.

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_cre_text_block
(
    int * entity_id,
    int * text_type,
    double text_origin [ 2 ] ,
    int * number_lines,
    const UF_DRF_one_apptext_line_t text_array [ ]
)
```

int *	**entity_id**	Input	Object identifier of the drafting aid to add text to
int *	**text_type**	Input	The type of the text to add. 6 = Appended text 17 = Form and positional tolerance text
double	**text_origin [2]**	Input	X and Y coordinates of the origin of the text string.
int *	**number_lines**	Input	The number of lines to add to the drafting aid.

const UF_DRF_one_apptext_line_t	**text_array [ ]**	Input	number_lines An array of the text strings to add.

UF_DRF_create_3pt_cline_fbolt (view source)

Defined in: uf_drf.h

Overview

Creates and displays a full bolt circle centerline using the 3 point method (centerline type UF_DRF_3pt_cline_fbolt).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_3_or_more_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_collinear_points
UF_err_program_not_initialized

Environment

Internal and External

See Also

See the example
This example creates a full bolt circle centerline using the 3 point method.

Required License(s)

drafting

```
int UF_DRF_create_3pt_cline_fbolt
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_3pt_cline_fcir (view source)

Defined in: uf_drf.h

Overview
Creates and displays a full circular centerline using the 3 point method (centerline type UF_DRF_3pt_cline_fcir).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_3_or_more_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_collinear_points
UF_err_program_not_initialized

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_3pt_cline_fcir
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_3pt_cline_pbolt (view source)

Defined in: uf_drf.h

Overview
Creates and displays a partial bolt circle centerline using the 3 point method (centerline type UF_DRF_3pt_cline_pbolt).

Return
Return code:
0 = No error

not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_3_or_more_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_collinear_points
UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates a full bolt circle centerline using the 3 point method.

Required License(s)
drafting

```
int UF_DRF_create_3pt_cline_pbolt
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_3pt_cline_pcir (view source)

Defined in: uf_drf.h

Overview
Creates and displays a partial circular centerline using the 3 point method (centerline type UF_DRF_3pt_cline_pcir).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_3_or_more_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing

UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_collinear_points
UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates a partial circular centerline using the 3 point method.
UF_DRF_create_3pt_cline_fcir

Required License(s)
drafting

```
int UF_DRF_create_3pt_cline_pcir
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_angular_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays an angular dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_angular_dim
(
    int dimension_form,
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

int	**dimension_form**	Input	Angular dimension form 1 = minor angle 2 = major angle
UF_DRF_object_p_t	**object1**	Input	Data of first (line) object (see uf_drf_types.h) Valid object types: line, straight solid curve, planar face, cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second (line) object
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created dimension initialized to NULL_TAG because legacy programs processed returned NULL_TAG as an error.

UF_DRF_create_arclength_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays an arc length dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_arclength_dim
(
    UF_DRF_object_p_t object,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object**	Input	Data of arc object (see uf_drf_types.h) Valid object types: arc, circle solid curve
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created arc length dimension

---

UF_DRF_create_areafill (view source)

Defined in: uf_drf.h

Overview
Creates and displays an area fill or solid fill.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_areafill
(
    int num_bounds,
    int* num_obj_bnd,
    tag_t object_list [ ] ,
    tag_t view_tag,
    tag_t* areafill_tag
)
```

int	**num_bounds**	Input	Number of area fill or solid fill boundaries
int*	**num_obj_bnd**	Input	List of integer values containing the number of objects in each boundary. The sum of all values in this list should equal the number of boundary objects in object_list.
tag_t	**object_list [ ]**	Input	List of curve tags or point tags or both that comprise each of the boundaries; a boundary object tag may be included in the list.
tag_t	**view_tag**	Input	Member view tag
tag_t*	**areafill_tag**	Output	Object tag of created area fill or solid fill.

UF_DRF_create_assortpart_aid (view source)

Defined in: uf_drf.h

Overview
Creates and displays an assorted parts Drafting Aid.

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_create_assortpart_aid
(
    UF_DRF_assortpart_arc_p_t arc,
    UF_DRF_assortpart_arrow_p_t arrow,
    UF_DRF_assortpart_line_p_t line,
    UF_DRF_assortpart_text_p_t text,
    tag_t * assorted_parts_tag
)
```

UF_DRF_assortpart_arc_p_t	**arc**	Input	assorted part arc data
UF_DRF_assortpart_arrow_p_t	**arrow**	Input	assorted part arrow data
UF_DRF_assortpart_line_p_t	**line**	Input	assorted part line data
UF_DRF_assortpart_text_p_t	**text**	Input	assorted part text data
tag_t *	**assorted_parts_tag**	Output	assorted parts tag

UF_DRF_create_assortpart_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays an Assorted Parts Dimension.

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_create_assortpart_dim
(
    UF_DRF_assortpart_arc_p_t arc,
    UF_DRF_assortpart_arrow_p_t arrow,
    UF_DRF_assortpart_line_p_t line,
    UF_DRF_assortpart_text_p_t text,
    tag_t * assorted_parts_tag
)
```

UF_DRF_assortpart_arc_p_t	**arc**	Input	assorted part arc data
UF_DRF_assortpart_arrow_p_t	**arrow**	Input	assorted part arrow data
UF_DRF_assortpart_line_p_t	**line**	Input	assorted part line data
UF_DRF_assortpart_text_p_t	**text**	Input	assorted part text data
tag_t *	**assorted_parts_tag**	Output	assorted parts tag

UF_DRF_create_block_cline (view source)

Defined in: uf_drf.h

Overview
Creates and displays a block centerline.
(centerline type UF_DRF_block_cline)

Return-
Return Codes -
0 = No error
not 0 = Error code

Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v19.0

Required License(s)
drafting

int UF_DRF_create_block_cline
(
UF_DRF_object_p_t defining_obj_list,
UF_DRF_object_p_t limiting_obj_list,
tag_t * centerline_tag
)

UF_DRF_object_p_t	**defining_obj_list**	Input	objects that define the block centerline (see uf_drf_types.h). Valid object types: linear edges and linear curves. This should be an array of two.
UF_DRF_object_p_t	**limiting_obj_list**	Input	objects that specify the extents of the block centerline. If NULL is passed, the system will calculate the limits. If an array is passed, it should be of the size two. Valid object types: edges and curves
tag_t *	**centerline_tag**	Output	object tag of the block centerline

---

UF_DRF_create_chamfer_dim (view source)

Defined in: uf_drf.h

Overview
UF_DRF_create_chamfer_dim

Description -
create a chamfer dimension and display it

PARAMETERS -
object1 - <I> data of first object (see uf_drf_types.h)
valid object types: line, solid curve, planar face
object2 - <I> data of second object
dim_text - <I> manual dimension text
dim_3d_origin - <I> 3d dim origin in wcs coords
dim_tag - <O> tag of created cham dim
return - <O> return code:
0 = OK
if not 0 = error code

Environment
Internal and External

Required License(s)
drafting

int UF_DRF_create_chamfer_dim
(
UF_DRF_object_p_t object1,
UF_DRF_object_p_t object2,
UF_DRF_text_t * dim_text,
double dim_3d_origin [3] ,
tag_t * dim_tag
)

UF_DRF_object_p_t	**object1**	Input	data of first object valid object types: line, solid curve
UF_DRF_object_p_t	**object2**	Input	data of second object
UF_DRF_text_t *	**dim_text**	Input	manual dimension text
double	**dim_3d_origin [3]**	Input	3d dim origin in wcs coords
tag_t *	**dim_tag**	Output	tag of created cham dim

UF_DRF_create_concir_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a concentric circle dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_concir_dim
(
UF_DRF_object_p_t object1,
UF_DRF_object_p_t object2,
UF_DRF_text_t* drf_text,
double dimension_3d_origin [3] ,
tag_t* dimension_tag
)

UF_DRF_object_p_t	**object1**	Input	Data of first arc object (see uf_drf_types.h) Valid object types: arc, circle solid curve, cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second arc object (see uf_drf_types.h)
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)

double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created circular dimension initialized to NULL_TAG because legacy programs processed returned NULL_TAG as an error.

UF_DRF_create_cpt_cline_fbolt (view source)

Defined in: uf_drf.h

Overview

n
Creates and displays a full bolt circle centerline using the centerpoint method (centerline type UF_DRF_cpt_cline_fbolt).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_invalid_object_type_center_point
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_point_coincident_with_center
UF_err_program_not_initialized

Environment

Internal and External

See Also

Refer to the example
UF_DRF_create_cpt_cline_pcir

Required License(s)

drafting

```
int UF_DRF_create_cpt_cline_fbolt
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    UF_DRF_object_p_t center_point,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h)
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for centerline center point (see uf_drf_types.h)

tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_cpt_cline_fcir (view source)

Defined in: uf_drf.h

Overview

Creates and displays a full circular centerline using the center point method (centerline type UF_DRF_cpt_cline_fcir).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_invalid_object_type_center_point
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_point_coincident_with_center
UF_err_program_not_initialized

Environment

Internal and External

See Also

Refer to the example

Required License(s)

drafting

```
int UF_DRF_create_cpt_cline_fcir
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    UF_DRF_object_p_t center_point,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for centerline center point (see uf_drf_types.h)
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_cpt_cline_pbolt (view source)

Defined in: uf_drf.h

Overview

Creates and displays a partial bolt circle centerline using the center point method (centerline type UF_DRF_cpt_cline_pbolt).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_invalid_object_type_center_point
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_point_coincident_with_center
UF_err_program_not_initialized

Environment

Internal and External

See Also

Refer to the example
UF_DRF_create_cpt_cline_pcir

Required License(s)

drafting

```
int UF_DRF_create_cpt_cline_pbolt
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    UF_DRF_object_p_t center_point,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for centerline center point (see uf_drf_types.h)
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_cpt_cline_pcir (view source)

Defined in: uf_drf.h

Overview

n
Creates and displays a partial circular centerline using the center point method (centerline type UF_DRF_cpt_cline_pcir).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_invalid_object_type_center_point
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_DRF_point_coincident_with_center
UF_err_program_not_initialized

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_cpt_cline_pcir
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    UF_DRF_object_p_t center_point,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for centerline center point (see uf_drf_types.h)
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_crosshatch (view source)

Defined in: uf_drf.h

Overview
Creates and displays a crosshatch.

b1 b5 b9
--------- ---------- ------------------------
| | | | | |

| | | | | | |
b2 | | b4 b6 | | b8 b10 | | b13
| | | | | | |
| | | | | | |
--------- ---------- -----------!-------------
b3 b7 b11 b12
Loop1 Loop2 Loop3
Input -
num_bounds - Number of loops = 3
{ Loop1, Loop2, Loop3 }
num_obj_bnd - Array of int which contains number of curve in each boundary
{ 4, 4, 5 }
object_list - tags of curve which forms loop
{ b1, b2, b3, b4, b5, b6
b7, b8, b9, 10, b11,b12, b13 }

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_crosshatch
(
    int num_bounds,
    int* num_obj_bnd,
    tag_t object_list [ ] ,
    tag_t view_tag,
    tag_t* crosshatch_tag
)
```

int	**num_bounds**	Input	Number of crosshatch boundaries
int*	**num_obj_bnd**	Input	List of integer values containing the number of objects in each boundary. The sum of all values in this list should equal the number of boundary objects in object_list
tag_t	**object_list [ ]**	Input	List of curve tags or point tags or both that comprise each of the boundaries; a boundary object tag may be included in the list
tag_t	**view_tag**	Input	Member view tag
tag_t*	**crosshatch_tag**	Output	Object tag of created crosshatch.

UF_DRF_create_custom_symbol_instance (view source)

Defined in: uf_drf.h

Overview
This function will create and display a new custom symbol. This function uses existing objects of the following types to form the symbol: lines, arcs, splines, parabolas, hyperbolas, B-curves, crosshatch, area fill, and notes. All objects which are passed into this routine via the data structure are consumed by the function call (they are not available for use as individual objects after this function returns).

The symbol scale and angle may be edited, as well as its color, font, and width. However, all other edits (such as text-specific edits, or display edits of symbol pieces) must be performed interactively at this time.

Return
Return values include the following:
UF_DRF_invalid_custom_symbol_piece
UF_DRF_default_text_out_of_bounds
UF_DRF_objects_not_in_same_view
UF_DRF_NO_ERRORS

Environment
Internal and External

See Also
UF_DRF_is_sbf_symbol, UF_DRF_initialize_custom_symbol_data, UF_DRF_ask_custom_symbol_scale, UF_DRF_set_custom_symbol_scale, UF_DRF_ask_custom_symbol_angle, UF_DRF_set_custom_symbol_angle
Refer to the example

History
Created in V18.0.3

Required License(s)
drafting

int UF_DRF_create_custom_symbol_instance
(
const UF_DRF_custom_symbol_t * symbol_definition,
tag_t * new_symbol_tag
)

const UF_DRF_custom_symbol_t *	**symbol_definition**	Input	Structure containing data necessary for custom symbol definition
tag_t *	**new_symbol_tag**	Output	The tag of the new custom symbol instance

---

UF_DRF_create_cylindrical_dim (view source)

Defined in: uf_drf.h

Overview
n
Creates and displays a cylindrical dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_cylindrical_dim
(

```
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of first object (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, utility symbol (centerline), or cylindrical face(if one associative object is a cylindrical face, both associative objects must be the same face. If the face is perpendicular to WCS X-Y plane the associative type must be tangency type).
UF_DRF_object_p_t	**object2**	Input	Data of second object
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created cylindrical dimension

---

UF_DRF_create_diameter_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a diameter dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_diameter_dim
(
    UF_DRF_object_p_t object,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object**	Input	Data of arc object (see uf_drf_types.h) Valid object types: arc, circle solid curve, cylindrical face
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates

tag_t*	**dimension_tag**	Output	Object tag of created diameter dimension

---

UF_DRF_create_foldedradius_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a folded radius dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_foldedradius_dim
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    double fold_location [ 3 ] ,
    double fold_angle,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of first object (see uf_drf_types.h) Valid object types: arc, bolt circle, circular centerline, cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second object: offset center point Valid object types: point, line, arc, conic, B curve, solid curve, offset center point, cylindrical, symmetrical center line, target point, intersection point
double	**fold_location [3]**	Input	Fold location in wcs coordinates
double	**fold_angle**	Input	Fold angle in degrees
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created linear dimension initialized to NULL_TAG because legacy programs processed returned NULL_TAG as an error.

UF_DRF_create_gdt_symbol (view source)

Defined in: uf_drf.h

Overview

n
Creates and displays a GD&T symbol.

Environment

Internal and External

See Also

Refer to the example

History

V17 - Added the ability to attach datums to annotations or leader stubs.

Required License(s)

drafting

```
int UF_DRF_create_gdt_symbol
(
    int num_lines_text,
    char text_string [ ] [ MAX_LINE_BUFSIZE ] ,
    double origin_3d [ 3 ] ,
    UF_DRF_leader_type_t leader_type,
    UF_DRF_leader_attach_type_t leader_attach_type,
    UF_DRF_object_p_t object,
    double model_pos_3d [ 3 ] ,
    UF_DRF_frame_corner_t frame_corner,
    tag_t* gdt_symbol_tag
)
```

int	**num_lines_text**	Input	Number of lines of text
char	**text_string [ ] [MAX_LINE_BUFSIZE]**	Input	Associated text string
double	**origin_3d [3]**	Input	3d object origin in wcs coordinates
UF_DRF_leader_type_t	**leader_type**	Input	Leader type UF_DRF_leader_type_none = none UF_DRF_leader_type_line = leader line UF_DRF_leader_type_ext_line = extension line
UF_DRF_leader_attach_type_t	**leader_attach_type**	Input	Leader attachment type If leader_type = UF_DRF_leader_type_line, UF_DRF_leader_attach_object = attached to object UF_DRF_leader_attach_screen = screen position
UF_DRF_object_p_t	**object**	Input	Data of object to attach leader see uf_drf_types.h If leader_type = UF_DRF_leader_type_line and leader_attach_type = UF_DRF_leader_attach_object, Valid object types: point, line, arc, conic, cubic spline, B curve, and solid curve. If leader_type = UF_DRF_leader_type_ext_line Valid object types: line, straight solid curve. If the leader_attach_type is UF_DRF_leader_attach_screen, then object = NULL.

double	**model_pos_3d [3]**	Input	3d model space position If leader_attach_type = UF_DRF_leader_attach_object This position is used as an approximate point on the object to attach the leader If leader_attach_type = UF_DRF_leader_attach_screen This position is the endpoint of the leader
UF_DRF_frame_corner_t	**frame_corner**	Input	Frame corner If leader_type = UF_DRF_leader_type_ext_line UF_DRF_frame_upper_left = upper left UF_DRF_frame_upper_right = upper right UF_DRF_frame_lower_left = lower left UF_DRF_frame_lower_right = lower right
tag_t*	**gdt_symbol_tag**	Output	Object tag of created gdt symbol.

UF_DRF_create_gdt_symbol_with_multiple_leaders (view source)

Defined in: uf_drf.h

Overview

UF_DRF_create_gdt_symbol_with_multiple_leaders

Description -
Create a Geometric Dimensioning and Tolerancing symbol with multiple leader lines. With this routine, you can also create leaders with intermediate points.

Input -
num_lines_text The number of lines of text in the GD&T symbol.
text_string The lines of text. This should be declared before calling this routine. For example:
text_string[][MAX_LINE_BUFSIZE]
origin The origin in WCS coordinates.
leader The leader line information.
frame_corner The frame corner to use if the leader_type is UF_DRF_leader_type_ext_line.
UF_DRF_frame_upper_left = upper left
UF_DRF_frame_upper_right = upper right
UF_DRF_frame_lower_left = lower left
UF_DRF_frame_lower_right = lower right

Output -
gdt_symbol_tag The tag of GD&T Symbol

Return Codes -
UF_DRF_NO_ERRORS
otherwise
UF_DRF_no_text
UF_err_bad_parameter
UF_DRF_invalid_leader_type
UF_DRF_too_many_objects More than UF_DRF_LEADER_MAX_LEADERS leaders.
UF_DRF_INVALID_LEADER_LOCATION Leader side is neither left nor right.
UF_DRF_invalid_leader_mode Datum leader requires an object.
UF_err_program_not_initialized

Environment

Internal and External

History
Original release was in NX 1.0.3.2 and NX 2

Required License(s)
drafting

```
int UF_DRF_create_gdt_symbol_with_multiple_leaders
(
    const int num_lines_text,
    const char text_string [ ] [ MAX_LINE_BUFSIZE ] ,
    const double gdt_symbol_origin [ 3 ] ,
    const UF_DRF_gdt_leader_t * leader,
    const UF_DRF_frame_corner_t frame_corner,
    tag_t * const gdt_symbol_tag
)
```

const int	**num_lines_text**	Input	The number of lines of text in the GD&T.
const char	**text_string [ ] [MAX_LINE_BUFSIZE]**	Input	The lines of text.
const double	**gdt_symbol_origin [3]**	Input	The GD&T symbol's origin.
const UF_DRF_gdt_leader_t *	**leader**	Input	The GD&T symbol's leader line information.
const UF_DRF_frame_corner_t	**frame_corner**	Input	Valid when leader_type is UF_DRF_leader_type_ext_line
tag_t * const	**gdt_symbol_tag**	Output	Object tag of created GD&T symbol.

UF_DRF_create_hole_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a hole dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_hole_dim
(
    UF_DRF_object_p_t object,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object**	Input	Data of arc object (see uf_drf_types.h) Valid object types: arc, circle solid curve, cylindrical face
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created hole dimension

---

UF_DRF_create_horizontal_baseline_dimension (view source)

Defined in: uf_drf.h

Overview
Creates and displays a horizontal baseline dimension.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form
UF_DRF_null_object_structure - if any object in the object set has null object structure
UF_DRF_null_object - if any object in the object set is null
UF_DRF_invalid_number_of_objects - if num_of_objects is less than 3
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
drafting

```
int UF_DRF_create_horizontal_baseline_dimension
(
    UF_DRF_object_t * object_set,
    int num_of_objects,
    double dimension_3d_origin [ 3 ] ,
    tag_t * dimension_tag
)
```

UF_DRF_object_t *	**object_set**	Input	Array of associated objects to be dimensioned.
int	**num_of_objects**	Input	Number of associated objects in array.
double	**dimension_3d_origin [3]**	Input	3d dimension set origin in absolute coordinates.
tag_t *	**dimension_tag**	Output	Object tag of created dimension set.

UF_DRF_create_horizontal_chain_dimension (view source)

Defined in: uf_drf.h

Overview
Creates and displays a horizontal chain dimension.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form
UF_DRF_null_object_structure - if any object in the object set has null object structure
UF_DRF_null_object - if any object in the object set is null
UF_DRF_invalid_number_of_objects - if num_of_objects is less than 3
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
drafting

```
int UF_DRF_create_horizontal_chain_dimension
(
    UF_DRF_object_t * object_set,
    int num_of_objects,
    double dimension_3d_origin [ 3 ] ,
    tag_t * dimension_tag
)
```

UF_DRF_object_t *	**object_set**	Input	Array of associated objects to be dimensioned.
int	**num_of_objects**	Input	Number of associated objects in array.
double	**dimension_3d_origin [3]**	Input	3d dimension set origin in absolute coordinates.
tag_t *	**dimension_tag**	Output	Object tag of created dimension set.

UF_DRF_create_horizontal_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a horizontal dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_horizontal_dim
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of first object (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, utility symbol (centerline), planar face, or cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second object
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created horizontal dimension

---

UF_DRF_create_id_symbol (view source)

Defined in: uf_drf.h

Overview
Creates and displays an ID symbol.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_id_symbol
(
    UF_DRF_id_symbol_type_t id_symbol_type,
    const char* upper_text_string,
    const char* lower_text_string,
    double origin_3d [ 3 ] ,
    UF_DRF_leader_mode_t leader_mode,
    UF_DRF_leader_attach_type_t leader_attach_type,
    UF_DRF_object_p_t object,
    double model_pos_3d [ 3 ] ,
    tag_t* id_symbol_tag
)
```

UF_DRF_id_symbol_type_t	**id_symbol_type**	Input	id symbol type UF_DRF_sym_circle = circle UF_DRF_sym_divcir = divided circle UF_DRF_sym_square = square UF_DRF_sym_divsqr = divided square UF_DRF_sym_hexagon = hexagon UF_DRF_sym_divhex = divided hexagon UF_DRF_sym_triup = triangle,

			point up UF_DRF_sym_tridown = triangle, point down UF_DRF_sym_datum = datum target UF_DRF_sym_roundbox = rounded box
const char*	**upper_text_string**	Input	Upper text string (maximum MAX_ID_SYM_TEXT_LENGTH characters)
const char*	**lower_text_string**	Input	Lower text string (maximum MAX_ID_SYM_TEXT_LENGTH characters)
double	**origin_3d [3]**	Input	3d object origin in wcs coordinates
UF_DRF_leader_mode_t	**leader_mode**	Input	Leader mode UF_DRF_without_leader = without leader UF_DRF_with_leader = with leader
UF_DRF_leader_attach_type_t	**leader_attach_type**	Input	Leader attachment type UF_DRF_leader_attach_object = attached to object UF_DRF_leader_attach_screen = screen position
UF_DRF_object_p_t	**object**	Input	Data of object to attach leader (see uf_drf_types.h) valid object types: point, line, arc, conic, cubic spline, B curve, and solid curve. If the leader_attach_type is UF_DRF_leader_attach_screen, then object = NULL.
double	**model_pos_3d [3]**	Input	3d model space position If leader_attach_type = UF_DRF_leader_attach_object, This position is used as an approximate point on the object to attach the leader If leader_attach_type = UF_DRF_leader_attach_screen, This position is the endpoint of the leader
tag_t*	**id_symbol_tag**	Output	Object tag of created id symbol.

UF_DRF_create_image (view source)

Defined in: uf_drf.h

Overview
Create an image from an existing image in the part

Returns
UF_DRF_NO_ERRORS if the image was created

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock

UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
gateway

```
int UF_DRF_create_image
(
    char * image_name,
    tag_t drawing_sheet,
    double origin [ 3 ] ,
    tag_t * image
)
```

char *	**image_name**	Input	Name of image in part file
tag_t	**drawing_sheet**	Input	The drawing sheet on which to create the image or NULL_TAG to create the image on the current drawing sheet.
double	**origin [3]**	Input	Origin of new image in drawing sheet units
tag_t *	**image**	Output	Image tag if successful or NULL_TAG if creation failed

---

UF_DRF_create_image_from_file (view source)

Defined in: uf_drf.h

Overview
Create an image from a jpg, png or tif file

Returns
UF_DRF_NO_ERRORS if the image was created

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
gateway

```
int UF_DRF_create_image_from_file
(
    char * file_name,
    tag_t drawing_sheet,
    double origin [ 3 ] ,
    tag_t * image
)
```

char *	**file_name**	Input	Name of jpg, png or tif file to use to create image
tag_t	**drawing_sheet**	Input	The drawing sheet on which to create the image or NULL_TAG to create the image on the current drawing sheet.
double	**origin [3]**	Input	Origin of new image in drawing sheet units
tag_t *	**image**	Output	Image tag if successful or NULL_TAG if creation failed

---

UF_DRF_create_label (view source)

Defined in: uf_drf.h

Overview
Creates and displays a label.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_label
(
    int num_lines_text,
    char text_string [ ] [ MAX_LINE_BUFSIZE ] ,
    double origin_3d [ 3 ] ,
    UF_DRF_leader_attach_type_t leader_attach_type,
    UF_DRF_object_p_t object,
    double model_pos_3d [ 3 ] ,
    tag_t* label_tag
)
```

int	**num_lines_text**	Input	Number of lines of text
char	**text_string [ ] [MAX_LINE_BUFSIZE]**	Input	Associated text string
double	**origin_3d [3]**	Input	3d object origin in wcs coordinates

UF_DRF_leader_attach_type_t	**leader_attach_type**	Input	Leader attachment type UF_DRF_leader_attach_object = attached to object UF_DRF_leader_attach_screen = screen position
UF_DRF_object_p_t	**object**	Input	Data of object to attach leader (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, and solid curve. If the leader_attach_type is UF_DRF_leader_attach_screen, then object = NULL. If object->object_tag is NULL_TAG then UF_DRF_leader_attach_screen should be used as leader_attach_type so object should also be NULL for this case. NOTE: UF_DRF_object_t is a general purpose structure used by several functions. The input fields object_assoc_type and object_assoc_modifier are not supported by UF_DRF_create_label. For example, Attaching the leader to an arc center is not supported, nor is it supported for interactive creation of a label in NX.
double	**model_pos_3d [3]**	Input	3d model space position if leader_attach_type = UF_DRF_leader_attach_object This position is used as an approximate point on the object to attach the leader if leader_attach_type = UF_DRF_leader_attach_screen, This position is the endpoint of the leader
tag_t*	**label_tag**	Output	Object tag of created label.

UF_DRF_create_linear_cline (view source)

Defined in: uf_drf.h

Overview

Creates and displays a linear centerline (centerline type UF_DRF_linear_cline).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_coincident_points
UF_DRF_point_not_on_centerline
UF_err_program_not_initialized

Environment

Internal and External

See Also
Refer to the example

History
In NX4, the centerline angle is inherited from the auxiliary view hinge line when the session setting has that option enabled.

Required License(s)
drafting

```
int UF_DRF_create_linear_cline
(
    const int num_cline_objs,
    UF_DRF_object_t* cline_obj_list,
    tag_t* centerline_tag
)
```

const int	**num_cline_objs**	Input	Number of centerline objects in cline_obj_list Maximum is MAX_CENTERLINE_OBJECTS, currently 100
UF_DRF_object_t*	**cline_obj_list**	Input	Array of objects that are to be associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_non_assoc_hatch (view source)

Defined in: uf_drf.h

Overview
Create a non-associative crosshatch. Useful for translating data from other system or forms, e.g. IGES, that do not support associative crosshatching.

Return
Returns standard error codes, with success == UF_DRF_NO_ERRORS.

Environment
Internal and External

History
-

Required License(s)
drafting

```
int UF_DRF_create_non_assoc_hatch
(
    int num_lines,
    double * hatch_lines,
    tag_t matrix,
    tag_t view,
    int color,
    int line_width,
    tag_p_t new_hatch
)
```

int	**num_lines**	Input	number of hatch lines = sizeof(hatch_lines)/4
double *	**hatch_lines**	Input	num_lines crosshatch lines ([x1, y1, x2, y2]) quadruples As with all annotations the coordinates are in the coordinater system of the matrix
tag_t	**matrix**	Input	matrix for crosshatch if NULL_TAG then the matrix of the WCS is used
tag_t	**view**	Input	view of crosshatch if NULL_TAG then the crosshatch is not view dependent
int	**color**	Input	color
int	**line_width**	Input	line width
tag_p_t	**new_hatch**	Output	crosshatch created or NULL_TAG if error

---

UF_DRF_create_note (view source)

Defined in: uf_drf.h

Overview
Creates either a horizontal or vertical note and displays it.

Return
Return Codes
UF_DRF_no_text
UF_DRF_null_object_structure
UF_DRF_null_object
UF_DRF_invalid_object
UF_err_program_not_initialized

Environment
Internal and External

History
This function was originally released in V14.0.1

Required License(s)
drafting

```
int UF_DRF_create_note
(
    int num_lines_text,
    char* text_string [ ] ,
    double origin_3d [ 3 ] ,
    int orientation,
    tag_t* note_tag
)
```

int	**num_lines_text**	Input	Number of lines of text

char*	**text_string [ ]**	Input	associated text strings
double	**origin_3d [3]**	Input	3d object origin in WCS coordinates
int	**orientation**	Input	Orientation of the note: 0 - Horizontal 1 - Vertical
tag_t*	**note_tag**	Output	Object tag of created note. Initialized to NULL_TAG.

UF_DRF_create_offctrpt_cx (view source)

Defined in: uf_drf.h

Overview

n
Creates and displays an offset center point on the x-axis at a specified distance from the arc center ("centerline" type UF_DRF_offctrpt_cx).

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc
UF_err_program_not_initialized

Environment

Internal and External

See Also

See the example
This example creates an offset centerpoint on the x-axis a specified distance from the arc center.

Required License(s)

drafting

```
int UF_DRF_create_offctrpt_cx
(
    UF_DRF_object_p_t cline_object,
    const double distance,
    tag_t* offctrpt_tag
)
```

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
const double	**distance**	Input	Distance value from arc center
tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

UF_DRF_create_offctrpt_cy (view source)

Defined in: uf_drf.h

Overview
Creates and displays an offset center point on the y-axis at a specified distance from the arc center ("centerline" type UF_DRF_offctrpt_cy).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc
UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates an offset centerpoint on the y-axis at a specified distance from the arc center.

Required License(s)
drafting

```
int UF_DRF_create_offctrpt_cy
(
    UF_DRF_object_p_t cline_object,
    const double distance,
    tag_t* offctrpt_tag
)
```

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
const double	**distance**	Input	Distance value from arc center
tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

UF_DRF_create_offctrpt_fx (view source)

Defined in: uf_drf.h

Overview
Creates and displays an offset center point on the x-axis and calculates the offset distance ("centerline" type UF_DRF_offctrpt_fx).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc
UF_DRF_invalid_object_type_offset_point
UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates an offset centerpoint on the x-axis and calculates the offset distance.

Required License(s)
drafting

```
int UF_DRF_create_offctrpt_fx
(
    UF_DRF_object_p_t cline_object,
    UF_DRF_object_p_t center_point,
    tag_t* offctrpt_tag
)
```

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for calculating distance to be used in placing offset center point (see uf_drf_types.h)
tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

---

UF_DRF_create_offctrpt_fy (view source)

Defined in: uf_drf.h

Overview
Creates and displays an offset center point on the y-axis and calculates the offset distance ("centerline" type UF_DRF_offctrpt_fy).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc

UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates an offset centerpoint on the y-axis and calculates the offset distance.

Required License(s)
drafting

int UF_DRF_create_offctrpt_fy
(
UF_DRF_object_p_t cline_object,
UF_DRF_object_p_t center_point,
tag_t* offctrpt_tag
)

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for calculating distance to be used in placing offset center point (see uf_drf_types.h)
tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

---

UF_DRF_create_offctrpt_nx (view source)

Defined in: uf_drf.h

Overview
Creates and displays an offset center point on the x-axis at a specified distance from the arc normal ("centerline" type UF_DRF_offctrpt_nx).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc
UF_err_program_not_initialized

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_offctrpt_nx
(
UF_DRF_object_p_t cline_object,
const double distance,
tag_t* offctrpt_tag
)

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
const double	**distance**	Input	Distance value from arc normal
tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

---

UF_DRF_create_offctrpt_ny (view source)

Defined in: uf_drf.h

Overview
Creates and displays an offset center point on the y-axis at a specified distance from the arc normal ("centerline" type UF_DRF_offctrpt_ny).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_1_object
UF_DRF_invalid_centerline_form
UF_DRF_offset_center_point_requires_arc
UF_DRF_invalid_object_type_offset_point
UF_err_program_not_initialized

Environment
Internal and External

See Also
See the example
This example creates an offset centerpoint on the y-axis at a specified distance from the arc normal.

Required License(s)
drafting

int UF_DRF_create_offctrpt_ny
(
UF_DRF_object_p_t cline_object,
const double distance,
tag_t* offctrpt_tag
)

UF_DRF_object_p_t	**cline_object**	Input	Object associated to the centerline (see uf_drf_types.h) Valid object types: arc, curved solid curve
const double	**distance**	Input	Distance value from arc normal

tag_t*	**offctrpt_tag**	Output	Object tag of created offset center point

---

UF_DRF_create_offcyl_cline_obj (view source)

Defined in: uf_drf.h

Overview

Creates and displays an offset cylindrical centerline and calculates the offset distance from the object (centerline type UF_DRF_offcyl_cline_obj). When you create a cylindrical centerline with an offset, it does not appear any different than the one created without an offset distance. However, when you dimension to the cylindrical centerline, the dimension reflects this distance by adding the offset distance value.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_2_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_DRF_invalid_object_type_offset_point
UF_err_program_not_initialized

Environment

Internal and External

See Also

See the example
This example creates an offset cylindrical centerline and calculates the offset distance from an object.

Required License(s)

drafting

```
int UF_DRF_create_offcyl_cline_obj
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_object_p_t center_point,
    tag_t* centerline_tag
)
```

UF_DRF_object_p_t	**object1**	Input	First object associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**object2**	Input	Second object associated to the centerline
UF_DRF_object_p_t	**center_point**	Input	Data of object selected for calculating offset distance (see uf_drf_types.h)
tag_t*	**centerline_tag**	Output	Object tag of created centerline

UF_DRF_create_offcyl_cline_off (view source)

Defined in: uf_drf.h

Overview

Creates and displays an offset cylindrical centerline with a keyed-in offset distance (centerline type UF_DRF_offcyl_cline_off).
When you create a cylindrical centerline with an offset, it does not appear any different than the one created without an offset distance.
However, when you dimension to the cylindrical centerline, the dimension reflects this distance by adding the offset distance value.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_2_objects
UF_DRF_invalid_centerline_form
UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_err_program_not_initialized

Environment

Internal and External

See Also

Refer to the example

Required License(s)

drafting

```
int UF_DRF_create_offcyl_cline_off
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    const double distance,
    tag_t* centerline_tag
)
```

UF_DRF_object_p_t	**object1**	Input	First object associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**object2**	Input	Second object associated to the centerline
const double	**distance**	Input	Offset distance value
tag_t*	**centerline_tag**	Output	Object tag of created centerline

UF_DRF_create_orddimension (view source)

Defined in: uf_drf.h

Overview

Creates and displays an ordinate dimension.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

drafting

```
int UF_DRF_create_orddimension
(
    tag_t margin_origin_tag,
    int dimension_type,
    UF_DRF_object_p_t object,
    double dogleg_angle,
    double dogleg_distance,
    UF_DRF_text_t* drf_text,
    int text_origin_flag,
    double origin_3d [ 3 ] ,
    tag_t* dimension_tag
)
```

tag_t	**margin_origin_tag**	Input	Margin or origin tag
int	**dimension_type**	Input	Type of dimension (required when margin_origin_tag is origin, ignored otherwise) 1 = horizontal 2 = vertical
UF_DRF_object_p_t	**object**	Input	Data of object (see uf_drf_types.h) valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, offset center point, cylindrical, symmetrical centerline, intersection symbol, target point
double	**dogleg_angle**	Input	Dogleg angle in degrees (0.0 for no dogleg)
double	**dogleg_distance**	Input	Dogleg distance from dim point in wcs coordinates
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
int	**text_origin_flag**	Input	Text origin flag 1 = center of total text box 2 = center of dimension text box
double	**origin_3d [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created ordinate dimension.

UF_DRF_create_ordinate_dim (view source)

Defined in: uf_drf.h

Overview

Create a horizontal or vertical ordinate dimension

Environment

Internal and External

Required License(s)

drafting

```
int UF_DRF_create_ordinate_dim
(
    tag_t np1,
    int ip2,
    tag_t np3,
    int ip4,
    int ip5,
    double rp6,
    double rp7,
    char * cp8,
    int ip9,
    char cp10 [ ] [ 133 ] ,
    int ip11,
    double rp12 [ 3 ] ,
    tag_t * nr13
)
```

tag_t	**np1**	Input	Object identifier of the margin or ordinate origin.
int	**ip2**	Input	Type of dimension if object is an origin: 1 = Horizontal 2 = Vertical
tag_t	**np3**	Input	Object to be dimensioned.
int	**ip4**	Input	Type of associativity to the object 1 = Endpoint 2 = Arc center 3 = Tangency 4 = Utility symbol
int	**ip5**	Input	Associativity modifier. For endpoint associativity: 1 = First endpoint 2 = Second endpoint For a tangency associativity: 0-100 = parameter percent along the arc to identify the point used to pick the arc side to dimension. For a utility symbol: 1-100 centerline segment number
double	**rp6**	Input	Dogleg angle in degrees (0.0 for no dogleg).
double	**rp7**	Input	Dogleg distance from the dimension point in WCS units.
char *	**cp8**	Input	Dimension text string.
int	**ip9**	Input	Number of lines of appended text.
char	**cp10 [ ] [133]**	Input	Array of appended text strings.
int	**ip11**	Input	Text origin flag: 1 = Center of total text box. 2 = Center of dimension text box
double	**rp12 [3]**	Input	X, Y and Z of origin in WCS coordinates.

tag_t *	**nr13**	Output	Object identifier of created dimension

---

UF_DRF_create_ordinate_margin (view source)

Defined in: uf_drf.h

Overview
Create an ordinate margin

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_create_ordinate_margin
(
    int ip1,
    tag_t np2,
    tag_t np3,
    double rp4 [ 2 ] ,
    double rp5 [ 2 ] ,
    double rp6,
    tag_t * nr7
)
```

int	**ip1**	Input	Type of margin: 1 = Horizontal 2 = Vertical
tag_t	**np2**	Input	Object identifier of the ordinate origin.
tag_t	**np3**	Input	Object identifier of the line or a NULL_TAG.
double	**rp4 [2]**	Input	X, Y point on the margin, only used if no line object is passed in.
double	**rp5 [2]**	Input	X, Y direction of the margin, only used if no line object is passed in.
double	**rp6**	Input	Offset distance.
tag_t *	**nr7**	Output	Object identifier of the created margin.

UF_DRF_create_ordinate_origin (view source)

Defined in: uf_drf.h

Overview
Create an ordinate origin dimension

Environment
Internal and External

Required License(s)

drafting

```
int UF_DRF_create_ordinate_origin
(
    tag_t np1,
    int ip2,
    int ip3,
    int ip4,
    int ip5,
    int ip6,
    char * cp7,
    tag_t * nr8
)
```

tag_t	**np1**	Input	Tag of the associated object.
int	**ip2**	Input	Type of associativity to the associated object: 1 = endpoint 2 = arc center 3 = Utility symbol
int	**ip3**	Input	Associativity modifier. For an endpoint associativity: 1 = Associate to the first endpoint 2 = Associate to the second endpoint For an Utility symbol associativity: 1-100 Centerline segment number
int	**ip4**	Input	Positive quadrant identifier, determines the signs of the dimensions: 1 = Upper right 2 = Upper left 3 = Lower right 4 = Lower left 5 = all quadrants
int	**ip5**	Input	Arrow and dimension line display for the ordinate origin: 1 = Don't display 2 = Display
int	**ip6**	Input	Origin symbol display 1 = Display origin name 2 = No display
char *	**cp7**	Input	Entity name.
tag_t *	**nr8**	Output	Tag of the created dimension.

UF_DRF_create_ordmargin (view source)

Defined in: uf_drf.h

Overview

Creates and displays an ordinate margin.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

drafting

int UF_DRF_create_ordmargin
(
int margin_type,
tag_t ordinate_origin_tag,
UF_DRF_object_p_t object,
double margin_xy_point [3] ,
double margin_xy_direction [2] ,
double offset_distance,
tag_t* margin_tag
)

int	**margin_type**	Input	Margin type 1 = horizontal 2 = vertical
tag_t	**ordinate_origin_tag**	Input	Ordinate origin tag
UF_DRF_object_p_t	**object**	Input	Data of line object (see uf_drf_types.h)
double	**margin_xy_point [3]**	Input	x,y point on margin (required if the object_tag in the object structure is null, ignored otherwise)
double	**margin_xy_direction [2]**	Input	x,y direction of margin (required if the object_tag in the object structure is null, ignored otherwise)
double	**offset_distance**	Input	Offset distance
tag_t*	**margin_tag**	Output	Object tag of created ordinate margin.

UF_DRF_create_ordorigin (view source)

Defined in: uf_drf.h

Overview
Creates and displays an ordinate origin.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_ordorigin
(
UF_DRF_object_p_t object,
int positive_quad_id,
int arr_dim_line_display,
int origin_symbol_display,
char* user_object_name,
tag_t* origin_tag
)

UF_DRF_object_p_t	**object**	Input	Data of object (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, offset center point, cylindrical, symmetrical centerline, intersection symbol, drafting point
int	**positive_quad_id**	Input	Positive quadrant identifier determines signs of dimensions 1 = upper right 2 = upper left 3 = lower right 4 = lower left 5 = all quadrants
int	**arr_dim_line_display**	Input	Arrow and dimension line display 1 = no 2 = yes
int	**origin_symbol_display**	Input	Origin symbol display 1 = origin name 2 = no display
char*	**user_object_name**	Input	User supplied object name
tag_t*	**origin_tag**	Output	Object tag of created ordinate origin.

UF_DRF_create_parallel_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a parallel dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_parallel_dim
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of first object (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, utility symbol (centerline), or cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second object
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)

double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created parallel dimension

---

UF_DRF_create_perpendicular_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a perpendicular dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_perpendicular_dim
(
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of base (line) object (see uf_drf_types.h) Valid object types: line, linear, cylindrical, symmetrical centerline, straight solid curve, planar face, cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second object valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, planar face, cylindrical face
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created dimension.

UF_DRF_create_radius_dim (view source)

Defined in: uf_drf.h

Overview
n
Creates and displays a radius dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_radius_dim
(
    UF_DRF_object_p_t object,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object**	Input	Data of arc object (see uf_drf_types.h) Valid object types: arc, circle solid curve, cylindrical face
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created radius dimension

---

UF_DRF_create_sbf_file (view source)

Defined in: uf_drf.h

Overview
Creates a Symbol Font Definition File and sets it as current.

Environment
Internal and External

History
This function was originally released in V14.0
via the Release Letter.

Required License(s)
drafting

```
int UF_DRF_create_sbf_file
(
    const char sbf_name [ UF_CFI_MAX_FILE_NAME_BUFSIZE ]
)
```

const char	**sbf_name [UF_CFI_MAX_FILE_NAME_BUFSIZE]**	Input	name of file to be created (30 characters maximum)

UF_DRF_create_side_seam (view source)

Defined in: uf_drf.h

Overview
This function creates the side seam representation for the weld symbol

Environment
Internal and External

See Also
UF_DRF_weld_symbols_t
UF_DRF_create_weld_symbol

History
Originally released in v18.0

Required License(s)
gateway

```
int UF_DRF_create_side_seam
(
    tag_t weld_symbol_tag,
    tag_t view_tag,
    tag_t object,
    double point [ 3 ] ,
    UF_DRF_weld_symbols_p_t weld_symbol_data
)
```

tag_t	**weld_symbol_tag**	Input	tag of the weld symbol for which side seam has to be created
tag_t	**view_tag**	Input	tag of the view for which side seam has to be created
tag_t	**object**	Input	object used as guide curve for creating side seam. These can be any curve or a solid edge.
double	**point [3]**	Input	absolute co-ordinates of the point at which the side seam has to be created
UF_DRF_weld_symbols_p_t	**weld_symbol_data**	Input	pointer to weld symbol data which defines the shape and size of the side seam

---

UF_DRF_create_sym_cline (view source)

Defined in: uf_drf.h

Overview
Creates and displays a symmetrical centerline (centerline type UF_DRF_symmetrical_cline).

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_centerline_form
UF_DRF_no_objects
UF_DRF_too_many_objects
UF_DRF_form_requires_2_objects
UF_DRF_invalid_centerline_form

UF_DRF_invalid_object_type_centerline
UF_DRF_model_objects_on_drawing
UF_err_program_not_initialized

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_sym_cline
(
UF_DRF_object_p_t object1,
UF_DRF_object_p_t object2,
tag_t* centerline_tag
)

UF_DRF_object_p_t	**object1**	Input	First object associated to the centerline (see uf_drf_types.h) Valid object types: point, arc, solid curve
UF_DRF_object_p_t	**object2**	Input	Second object associated to the centerline
tag_t*	**centerline_tag**	Output	Object tag of created centerline

---

UF_DRF_create_symbol_font (view source)

Defined in: uf_drf.h

Overview
Creates a symbol font object and saves it in the current SBF file.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_create_symbol_font
(
const char symbol_name [UF_CFI_MAX_FILE_NAME_BUFSIZE] ,
double symbol_factor,
const double symbol_anchor [3] ,
const double symbol_orient [3] ,
int num_objects,
tag_t object [ ]
)

const char	**symbol_name [UF_CFI_MAX_FILE_NAME_BUFSIZE]**	Input	symbol name

double	**symbol_factor**	Input	symbol factor
const double	**symbol_anchor [3]**	Input	model space x,y,z of anchor point
const double	**symbol_orient [3]**	Input	model space x,y,z of orientation point
int	**num_objects**	Input	number of objects composing this symbol Maximum allowed is 512
tag_t	**object [ ]**	Input	num_objects list of object tags

UF_DRF_create_top_seam (view source)

Defined in: uf_drf.h

Overview
This function creates the top seam representation for the weld symbol

Environment
Internal and External

See Also
UF_DRF_weld_symbols_t
UF_DRF_create_weld_symbol

History
Originally released in v18.0

Required License(s)
gateway

```
int UF_DRF_create_top_seam
(
    tag_t weld_symbol_tag,
    tag_t view_tag,
    int num_objects,
    tag_p_t objects,
    logical flip,
    UF_DRF_weld_symbols_p_t weld_symbol_data
)
```

tag_t	**weld_symbol_tag**	Input	tag of the weld symbol for which top seam has to be created
tag_t	**view_tag**	Input	tag of the view for which top seam has to be created
int	**num_objects**	Input	number of objects to be used for creating the top seam
tag_p_t	**objects**	Input	num_objects objects used as guide curves for creating top seam. For plug/slot and spot, these have to be points. For others these can be any curve or a solid edge

logical	**flip**	Input	flag for direction reversal 0 = no 1 = yes
UF_DRF_weld_symbols_p_t	**weld_symbol_data**	Input	pointer to weld symbol data which defines the shape and size of the top seam

---

UF_DRF_create_vertical_baseline_dimension (view source)

Defined in: uf_drf.h

Overview
Creates and displays a vertical baseline dimension.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form
UF_DRF_null_object_structure - if any object in the object set has null object structure
UF_DRF_null_object - if any object in the object set is null
UF_DRF_invalid_number_of_objects - if num_of_objects is less than 3
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
drafting

```
int UF_DRF_create_vertical_baseline_dimension
(
    UF_DRF_object_t * object_set,
    int num_of_objects,
    double dimension_3d_origin [ 3 ] ,
    tag_t * dimension_tag
)
```

UF_DRF_object_t *	**object_set**	Input	Array of associated objects to be dimensioned.
int	**num_of_objects**	Input	Number of associated objects in array.
double	**dimension_3d_origin [3]**	Input	3d dimension set origin in absolute coordinates.
tag_t *	**dimension_tag**	Output	Object tag of created dimension set.

UF_DRF_create_vertical_chain_dimension (view source)

Defined in: uf_drf.h

Overview
Creates and displays a vertical chain dimension.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form
UF_DRF_null_object_structure - if any object in the object set has null object structure
UF_DRF_null_object - if any object in the object set is null
UF_DRF_invalid_number_of_objects - if num_of_objects is less than 3
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
drafting

```
int UF_DRF_create_vertical_chain_dimension
(
    UF_DRF_object_t * object_set,
    int num_of_objects,
    double dimension_3d_origin [ 3 ] ,
    tag_t * dimension_tag
)
```

UF_DRF_object_t *	**object_set**	Input	Array of associated objects to be dimensioned.
int	**num_of_objects**	Input	Number of associated objects in array.
double	**dimension_3d_origin [3]**	Input	3d dimension set origin in absolute coordinates.
tag_t *	**dimension_tag**	Output	Object tag of created dimension set.

UF_DRF_create_vertical_dim (view source)

Defined in: uf_drf.h

Overview
Creates and displays a vertical dimension.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_create_vertical_dim
(
```

```
    UF_DRF_object_p_t object1,
    UF_DRF_object_p_t object2,
    UF_DRF_text_t* drf_text,
    double dimension_3d_origin [ 3 ] ,
    tag_t* dimension_tag
)
```

UF_DRF_object_p_t	**object1**	Input	Data of first object (see uf_drf_types.h) Valid object types: point, line, arc, conic, cubic spline, B curve, pattern, solid curve, utility symbol (centerline), planar face, or cylindrical face
UF_DRF_object_p_t	**object2**	Input	Data of second object
UF_DRF_text_t*	**drf_text**	Input	Associated text (see uf_drf_types.h)
double	**dimension_3d_origin [3]**	Input	3d dimension origin in wcs coordinates
tag_t*	**dimension_tag**	Output	Object tag of created vertical dimension

---

UF_DRF_create_weld_symbol (view source)

Defined in: uf_drf.h

Overview
Creates the weld symbol.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_NO_ERRORS - No error
UF_DRF_ERR_WELD_SYM_STD_NOT_SUPPORTED - The specified Weld Symbol Standard is not supported
UF_DRF_ERR_INVALID_EXTENSION - The specified Extension type is not supported
UF_err_program_not_initialized - Open API has not been initialized
UF_DRF_null_object_structure
UF_DRF_INVALID_WELD_TYPE The specified weld type is not supported in UF

Environment
Internal and External

See Also
UF_DRF_ weld_symbols_t
UF_DRF_object_t

History
Originally released in v18.0

Required License(s)
gateway

```
int UF_DRF_create_weld_symbol
(
    double origin_3d [ 3 ] ,
    UF_DRF_leader_attach_type_t leader_attach_type,
```

```
    UF_DRF_object_p_t object,
    double model_pos_3d [ 3 ] ,
    UF_DRF_weld_symbols_p_t weld_symbol_data,
    tag_p_t weld_symbol_tag
)
```

double	**origin_3d [3]**	Input	3d object origin in absolute coordinates
UF_DRF_leader_attach_type_t	**leader_attach_type**	Input	Leader attachment type UF_DRF_leader_attach_object = attached to object UF_DRF_leader_attach_screen = screen position
UF_DRF_object_p_t	**object**	Input	Data of object to attach leader (see uf_drf_types.h). Valid object types: point, line, arc, conic, cubic spline, B curve, and solid curve.
double	**model_pos_3d [3]**	Input	3d model space position if leader_attach_type = UF_DRF_leader_attach_object This position is used as an approximate point on the object to attach the leader if leader_attach_type = UF_DRF_leader_attach_screen, This position is the endpoint of the leader
UF_DRF_weld_symbols_p_t	**weld_symbol_data**	Input	pointer to the weld symbol structure
tag_p_t	**weld_symbol_tag**	Output	tag of the weld symbol created

---

UF_DRF_create_xhatch (view source)

Defined in: uf_drf.h

Overview
Create a crosshatching

Environment
Internal and External

Return
Void

See Also
UF_DRF_create_crosshatch

Required License(s)
drafting

```
void UF_DRF_create_xhatch
(
    int * op_type,
    int * nmbnds,
    int * numels,
    tag_t * elems,
    tag_t * xhat_eid,
    int * rtc
)
```

int *	**op_type**	Input	The type of operation: 1 = Cross hatching 2 = Area fill 3 = Solid fill
int *	**nmbnds**	Input	The number of crosshatch boundaries.
int *	**numels**	Input	An array of integer values, containing the number of objects in each boundary. The sum of all of the values in this array should equal the number of boundary objects passed in. The number of elements in this array is equal to the number of crosshatch boundaries above.
tag_t *	**elems**	Input	Array of curve or point tags the make up each of the boundaries. A boundary object tag may be included in the array.
tag_t *	**xhat_eid**	Output	The object identifier of the created crosshatch.
int *	**rtc**	Output	The return code from the operation.

UF_DRF_edit_dim_assoc (view source)

Defined in: uf_drf.h

Overview
Edits the associativity of a selected dimension. This is applicable to changing a retained dimension by reassociating the dimension to up-to-date geometry.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_edit_dim_assoc
(
    tag_t dimension_tag,
    double old_leader_position [ 3 ] ,
    double new_leader_position [ 3 ] ,
    int new_assoc_type,
    UF_DRF_object_p_t new_assoc_object
)
```

tag_t	**dimension_tag**	Input	Tag of dimension whose associativity is being edited.
double	**old_leader_position [3]**	Input	Position on associated object to be replaced (arrow location). - drawing coordinates if on drawing - model space coordinates if in model
double	**new_leader_position [3]**	Input	Position on new associated object to attach to.
int	**new_assoc_type**	Input	Type of new associativity horizontal, vertical, parallel, perpendicular, cylindrical, and folded radius:

			1 = end point 2 = arc center point 3 = arc tangent point 9 = centerline horizontal and vertical ordinate, and ordinate origin dimensions: 0 = none 1 = end point 2 = arc center point 3 = arc tangent point 9 = centerline angular dimensions: 1 = dimension to line 2 = dimension to extension line 5 = dimension to centerline arc length, radius, diameter, hole, and concentric circle dimensions: does not apply
UF_DRF_object_p_t	**new_assoc_object**	Input	Data of new object (see uf_drf_types.h) to associate to. Note that the only fields used are object_tag and object_view_tag. Valid object_tag types: point, line, arc, conic, cubic spline, B curve, pattern, and solid curve.

UF_DRF_edit_weld_symbol (view source)

Defined in: uf_drf.h

Overview

Given the symbol tag and the new symbol data, this function will edit the weld symbol. Note that the change in weld symbol standard during editing is not supported. This means that the current weld symbol standard and the new symbol standard should be the same.

Environment

Internal and External

See Also

UF_DRF_weld_symbols_t
UF_DRF_object_t
UF_DRF_create_weld_symbol

History

Originally released in v18.0

Required License(s)

gateway

```
int UF_DRF_edit_weld_symbol
(
    tag_t weld_symbol_tag,
    UF_DRF_weld_symbols_p_t weld_symbol_data
)
```

tag_t	**weld_symbol_tag**	Input	tag of the weld symbol to be edited
UF_DRF_weld_symbols_p_t	**weld_symbol_data**	Input	pointer to new weld symbol data

UF_DRF_flip_image_about_height (view source)

Defined in: uf_drf.h

Overview
Flip the image about the direction of its height.

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_flip_image_about_height
(
    tag_t image
)
```

tag_t	**image**	Input	Image to edit

UF_DRF_flip_image_about_width (view source)

Defined in: uf_drf.h

Overview
Flip the image about the direction of its width.

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data

UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_flip_image_about_width
(
    tag_t image
)
```

tag_t	**image**	Input	Image to edit

UF_DRF_frdim (view source)

Defined in: uf_drf.h

Overview
Create a folded radius

Environment
Internal and External

Return
Void

Required License(s)
drafting

```
void UF_DRF_frdim
(
    tag_t np1,
    tag_t np2,
    int ip3,
    int ip4,
    double * rp5,
    double rp6,
    const char * cp7,
    int ip8,
    const UF_DRF_one_apptext_line_t cp9 [ ] ,
    double rp10 [ 3 ] ,
    tag_t * nr11,
    int * error
)
```

tag_t	**np1**	Input	Object identifier of an Arc, bolt circle or circular CLINE.
tag_t	**np2**	Input	Object identifier of offset center point (point, line, solid edge, offse center point, cylindrical center line,

			symmetrical center line, target point or intersection point.
int	**ip3**	Input	Type of associativity: 1 = endpoint 2 = Arc center 3 = Tangency
int	**ip4**	Input	Associativity modifier, if the type of associativity is an endpoint: 1 = Associate to first endpoint 2 = Associate to second endpoint If the type of associativity is a tangency: 0-100 is the parameter percent to be used to compute the approximate tangency point.
double *	**rp5**	Input	Fold location in WCS coordinates.
double	**rp6**	Input	Fold angle in degrees.
const char *	**cp7**	Input	Dimension text string.
int	**ip8**	Input	Number of lines of appended text.
const UF_DRF_one_apptext_line_t	**cp9 [ ]**	Input	Array of appended text strings.
double	**rp10 [3]**	Input	X, Y and Z of object origin in WCS coordinates.
tag_t *	**nr11**	Output	Object identifier of the created folded radius dimension.
int *	**error**	Output	Error return, 0 for success.

UF_DRF_free_appended_text (view source)

Defined in: uf_drf.h

Overview
The following function will free the appended text data returned from UF_DRF_ask_appended_text

Environment
Internal & External

See Also
UF_DRF_ask_appended_text
UF_DRF_appended_text_t

History
Created in V17.0

Required License(s)
drafting

```
int UF_DRF_free_appended_text
(
    int num_text,
    UF_DRF_appended_text_p_t appended_text
```

)

int	**num_text**	Input	Number of appended text
UF_DRF_appended_text_p_t	**appended_text**	Input	Appended text

---

UF_DRF_free_centerline (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing centerline data.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
drafting

int UF_DRF_free_centerline
(
UF_DRF_centerline_info_t * * centerline_info
)

UF_DRF_centerline_info_t * *	**centerline_info**	Input	Centerline information (see uf_drf_types.h)

UF_DRF_free_comp_data (view source)

Defined in: uf_drf.h

Overview
This function will free the data returned from UF_DRF_record_draft_objects.

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX4.0

Required License(s)
drafting

int UF_DRF_free_comp_data
(
void * objs
)

void *	**objs**	Input	Drafting object data from UF_DRF_record_draft_objects

---

UF_DRF_free_dimension (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing the dimension information of the data object.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
drafting

int UF_DRF_free_dimension
(
UF_DRF_dim_info_t * * dim_info
)

UF_DRF_dim_info_t * *	**dim_info**	Input	Dimension information (see UF_DRF_dim_info_p_t)

UF_DRF_free_dimension_preferences1 (view source)

Defined in: uf_drf.h

Overview
The following function will free the dimension preferences data returned from UF_DRF_ask_dimension_preferences1

History
Originally released in NX7.5

Required License(s)
gateway

int UF_DRF_free_dimension_preferences1
(
UF_DRF_dimension_preferences1_p_t dimension_preferences
)

UF_DRF_dimension_preferences1_p_t	**dimension_preferences**	Input	pointer to preferences structure to be freed

---

UF_DRF_free_font (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing user symbol font information.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRF_free_font
(
    int num_fonts,
    UF_DRF_ud_symbol_font_info_t * * font_info
)
```

int	**num_fonts**	Input	Number of User Defined Symbol blocks
UF_DRF_ud_symbol_font_info_t * *	**font_info**	Input	Data structure which contains the User Defined Symbol font number, left and right connection points, factor, length, height and the count of strokes and stoke data

---

UF_DRF_free_gdtsymbol (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing the GD & T symbol information.

Environment
Internal and External

See Also
See the example code for UF_DRF_ask_gdt_symbol_info.
The code in the example queries the GD&T Symbol information and frees the memory after usage.

History
Original release was in V13.0.

Required License(s)

drafting

int UF_DRF_free_gdtsymbol
(
UF_DRF_gdt_symbol_info_t * * gdt_symbol_info
)

UF_DRF_gdt_symbol_info_t * *	**gdt_symbol_info**	Input	GD&T Symbol information (see uf_drf_types.h)

UF_DRF_free_idsymbol (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing the ID symbol information.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
drafting

int UF_DRF_free_idsymbol
(
UF_DRF_id_symbol_info_t * * id_symbol_info
)

UF_DRF_id_symbol_info_t * *	**id_symbol_info**	Input	ID Symbol information (see uf_drf_types.h)

UF_DRF_free_image_data (view source)

Defined in: uf_drf.h

Overview
Free the image data

Returns
UF_DRF_NO_ERRORS if the image data free was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image

UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_free_image_data
(
    UF_DRF_image_data_t * data
)
```

UF_DRF_image_data_t *	**data**	Input	Image data pointer to free

---

UF_DRF_free_label (view source)

Defined in: uf_drf.h

Overview
Frees label information memory.

Environment
Internal and External

See Also
See the example code for UF_DRF_ask_label
The code in the example queries the label information, and frees the label information memory after usage.

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRF_free_label
(
    UF_DRF_label_info_t * * label_info
)
```

UF_DRF_label_info_t * *	**label_info**	Input	Label information (see UF_DRF_label_info_p_t)

UF_DRF_free_leader_data (view source)

Defined in: uf_drf.h

Overview
This function will free a UF_DRF_leader_data_p_t

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX3.0.2

Required License(s)
gateway

```
int UF_DRF_free_leader_data
(
    UF_DRF_leader_data_p_t * leader_data
)
```

UF_DRF_leader_data_p_t *	**leader_data**	Input / Output	The leader data to free

---

UF_DRF_free_text (view source)

Defined in: uf_drf.h

Overview
Frees the memory used for storing text data information.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V13.0.

Required License(s)
drafting

```
int UF_DRF_free_text
(
    int num_text,
    UF_DRF_draft_aid_text_info_t * * text_info
)
```

int	**num_text**	Input	Number of text strings
UF_DRF_draft_aid_text_info_t * *	**text_info**	Input	Pointer to data structure which contains drafting aid text information (see uf_drf_types.h)

UF_DRF_get_symbol_divider (view source)

Defined in: uf_drf.h

Overview
Find the location of a symbol inside of a drafting text.

Environment
Internal and External

See Also
UF_DRF_count_text_substring
UF_DRF_ask_ann_data
UF_DRF_set_draft_common
UF_DRF_get_text_substring

Required License(s)
drafting

```
int UF_DRF_get_symbol_divider
(
    int * segment_number,
    int * ann_data,
    int * divider_instance,
    double start_point [ 2 ] ,
    double end_point [ 2 ]
)
```

int *	**segment_number**	Input	The segment of the text to get.
int *	**ann_data**	Input	The data defining the drafting object. See UF_DRF_ask_ann_data.
int *	**divider_instance**	Input	Count to the substring that is to be used.
double	**start_point [2]**	Output	X and Y coordinates of the start point.
double	**end_point [2]**	Output	X and Y coordinates of the end point.

UF_DRF_get_text_bars (view source)

Defined in: uf_drf.h

Overview
Return the location where a fraction bar or underline bar or overline bar should be drawn for text.

Environment
Internal and External

See Also
UF_DRF_count_text_substring
UF_DRF_ask_ann_data
UF_DRF_set_draft_common

Required License(s)
drafting

```
int UF_DRF_get_text_bars
(
    int * segment_number,
    int * ann_data,
    int * number_of_bars,
    int * bar_type,
    int * offset_to_bar,
    double bar_position [ 4 ]
)
```

int *	**segment_number**	Input	The text segment number to work on.
int *	**ann_data**	Input	The data defining the drafting object. See UF_DRF_ask_ann_data.
int *	**number_of_bars**	Output	The number of text bars in this drafting object.
int *	**bar_type**	Output	An array of types of text bars, one for each text bar. 1 = Under line 2 = Over line
int *	**offset_to_bar**	Output	An array of offsets to the text in the string where the bar text ends.
double	**bar_position [4]**	Output	An array of positions for the text bar. There are four elements for each text bar, the X, Y of the start of the text bar, and the X, Y of the end of the bar.

UF_DRF_get_text_substring (view source)

Defined in: uf_drf.h

Overview
Return the requested text substring of the drafting object.

Environment
Internal and External

See Also
UF_DRF_count_text_substring
UF_DRF_ask_ann_data
UF_DRF_set_draft_common

Required License(s)
drafting

```
int UF_DRF_get_text_substring
(
    int * segment_number,
    int * ann_data,
    int * substring_instance,
    int * text_type,
    int * number_of_substring,
    char * substring,
    double substring_position [ 2 ] ,
    double substring_characteristic [ 6 ]
)
```

int *	**segment_number**	Input	The text segment number to work on.
int *	**ann_data**	Input	The data defining the drafting object. See UF_DRF_ask_ann_data.
int *	**substring_instance**	Input	The substring number that we want to find.
int *	**text_type**	Output	The type of the substring: 0 = string 1 = superscript 2 = subscript 3 = symbol
int *	**number_of_substring**	Output	The number of bytes in the returned substring.
char *	**substring**	Output	The substring found.
double	**substring_position [2]**	Output	The x and y coordinates for the start of this substring.
double	**substring_characteristic [6]**	Output	Six word array of characteristics of this substring: [0] = The character size [1] = The physical length of the substring [2] = The scale from the original character size of the string. [3] = The scale from the original character gap. [4] = The character slant angle. [5] = The text angle.

UF_DRF_get_xhatch_parms (view source)

Defined in: uf_drf.h

Overview
Gets the crosshatch parameters.

Return
void

Environment
Internal and External

Required License(s)
drafting

```
void UF_DRF_get_xhatch_parms
(
    tag_t * xhat_eid,
    char mat_name [ UF_OBJ_NAME_BUFSIZE ] ,
    int int_parms [ 9 ] ,
    double real_parms [ 13 ] ,
    int * rtc
)
```

tag_t *	**xhat_eid**	Input	Object identifier of the desired crosshatch

char	**mat_name [UF_OBJ_NAME_BUFSIZE]**	Output	Material name of the crosshatch
int	**int_parms [9]**	Output	Integer parameters of the crosshatch. This an array of nine elements (int_parms[9]) [0] = entity site [1] = line density [2] = filled arrowhead control [3] = crosshatch symbol color [4] = section validity flag [5-8] = number of crosshatch lines in each of 4 directions
double	**real_parms [13]**	Output	Real parameters of the crosshatch. This is an array of 13 elements (real_parms[13]). [0] = Arrow size [1-4] = Crosshatch distance for each of 4 directions [5-8] = Crosshatch angle for each of 4 directions [9] = Crosshatch tolerance [10] = Arrowhead filled distance [11] = Arrowhead included angle [12] = Dot arrowhead diameter
int *	**rtc**	Output	Return code: 1 = xhat_eid is not a hatch.

UF_DRF_has_associative_origin (view source)

Defined in: uf_drf.h

Overview
The following function will query the associative origin information for the annotation.

Environment
Internal and External

See Also
UF_DRF_ask_associative_origin
UF_DRF_set_associative_origin

History
Created in V17.0

Required License(s)
drafting

```
int UF_DRF_has_associative_origin
(
    tag_t drafting_entity,
    logical * has_associative_origin
)
```

tag_t	**drafting_entity**	Input	Dimension or drafting object to query.
logical *	**has_associative_origin**	Output	Result of query

UF_DRF_inherit_feature_data (view source)

Defined in: uf_drf.h

Overview

This routine inherits feature annotation into user specified views on a drawing.

Environment

Internal and External

See Also

UF_DRF_inherit_type_t
UF_DRF_ask_callout_of_annotation
UF_DRF_ask_controlling_member_of_callout
UF_DRF_ask_number_of_rows_in_callout
UF_DRF_ask_callout_row_members

History

Originally released in v18.0

Required License(s)

drafting

```
int UF_DRF_inherit_feature_data
(
    int feature_pre_v13_sketch_count,
    tag_p_t features_pre_v13_sketches,
    tag_p_t part_occs,
    int view_count,
    tag_p_t views,
    UF_DRF_inherit_type_t inherit_type
)
```

int	**feature_pre_v13_sketch_count**	Input	number of features and pre-v13 sketches to be annotated
tag_p_t	**features_pre_v13_sketches**	Input	feature_pre_v13_sketch_count the features and pre-v13 sketches to be annotated. A pre-v13 sketch tag is the tag in the part of the sketch, not in the part occurrence of the sketch.
tag_p_t	**part_occs**	Input	feature_pre_v13_sketch_count the i-th feature or sketch to be annotated occurs in the occurence of the part given by part_occs[i]. NULL_TAG is valid.
int	**view_count**	Input	number of views to be annotated
tag_p_t	**views**	Input	views to be annotated
UF_DRF_inherit_type_t	**inherit_type**	Input	the types of annotation to be inherited Currently only UF_DRF_inherit_feature_parameters is supported

UF_DRF_init_associativity_data (view source)

Defined in: uf_drf.h

Overview
The following function initialized the associativity structure on the annotation object.

Environment
Internal or External

History
Originally released in V19.0

Required License(s)
drafting

```
int UF_DRF_init_associativity_data
(
    UF_DRF_object_assoc_data_p_t associativity_data
)
```

UF_DRF_object_assoc_data_p_t	**associativity_data**	Input	

UF_DRF_init_assortpart_arc (view source)

Defined in: uf_drf.h

Overview
Initialize an Assorted Parts Arc Data Structure.

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_init_assortpart_arc
(
    UF_DRF_assortpart_arc_t * assortpart_arc
)
```

UF_DRF_assortpart_arc_t *	**assortpart_arc**	Input / Output	assorted part arc

---

UF_DRF_init_assortpart_arrow (view source)

Defined in: uf_drf.h

Overview
Initializes an Assorted Parts Arrow Data Structure.

Environment
Internal and External

Required License(s)
drafting

int UF_DRF_init_assortpart_arrow
(
UF_DRF_assortpart_arrow_t * assortpart_arrow
)

UF_DRF_assortpart_arrow_t *	**assortpart_arrow**	Input / Output	assorted part arrow

---

UF_DRF_init_assortpart_line (view source)

Defined in: uf_drf.h

Overview
Initializes an Assorted Parts Line Data Structure.

Environment
Internal and External

Required License(s)
drafting

int UF_DRF_init_assortpart_line
(
UF_DRF_assortpart_line_t * assortpart_line
)

UF_DRF_assortpart_line_t *	**assortpart_line**	Input / Output	assorted part line

---

UF_DRF_init_assortpart_text (view source)

Defined in: uf_drf.h

Overview
Initializes an Assorted Parts Text Data Structure.

Environment
Internal and External

Required License(s)
drafting

int UF_DRF_init_assortpart_text
(
UF_DRF_assortpart_text_t * assortpart_text
)

UF_DRF_assortpart_text_t *	**assortpart_text**	Input / Output	assorted part text

UF_DRF_init_image_data (view source)

Defined in: uf_drf.h

Overview
Initialize the image data

Returns
UF_DRF_NO_ERRORS if the image data initialization was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_init_image_data
(
    UF_DRF_image_data_t * data
)
```

UF_DRF_image_data_t *	**data**	Output	Image data to initialize

UF_DRF_init_line_object (view source)

Defined in: uf_drf.h

Overview
Initializes a line object structure.

Environment
Internal and External

See Also
Please See
UF_DRF_init_object_structure
for information on the initialization of drafting objects.
Please see the example for
UF_DRAW_redefine_sxline_hinge
which also invokes a call to UF_DRF_init_line_object.

History
Original release was in V14.0.

Required License(s)
drafting

int UF_DRF_init_line_object
(
UF_DRF_line_object_t * line_object
)

UF_DRF_line_object_t *	**line_object**	Output	Line object initialized as follows: line_object->line_assoc_type = UF_DRF_ASSOC_LINE_TYPE_NONE line_object->object1 line_object->object2 initialized by UF_DRF_init_object_structure

---

UF_DRF_init_object_structure (view source)

Defined in: uf_drf.h

Overview
Initialize an UF_DRF_object_t structure (see uf_drf_types.h).

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_diameter_dim

Required License(s)
drafting

int UF_DRF_init_object_structure
(
UF_DRF_object_t * object
)

UF_DRF_object_t *	**object**	Input / Output	Object structure to be initialized. (Input) object - initialized object structure (Output) object.object_tag = NULL_TAG object.object_view_tag = NULL_TAG object.object_assoc_type = UF_DRF_end_point object.object_assoc_modifier = UF_DRF_first_end_point object.object2_tag = NULL_TAG object.assoc_dwg_pos[0] = 0.0 object.assoc_dwg_pos[1] = 0.0

UF_DRF_init_symbol_create_data (view source)

Defined in: uf_drf.h

Overview
Initializes a symbol data structure for creation.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_init_symbol_create_data
(
UF_DRF_symbol_create_data_t * symbol_data
)

UF_DRF_symbol_create_data_t *	**symbol_data**	Input / Output	symbol data (see uf_drf_types.h)

UF_DRF_initialize_custom_symbol_data (view source)

Defined in: uf_drf.h

Overview
This function will initialize the UF_DRF_custom_symbol_t structure. This routine should be called prior to filling this structure with valid data. Calling this routine will guarantee that the data structure does not contain garbage values. Calling this routine WILL NOT guarantee that the structure will contain valid values which can be processed by UF_DRF_create_custom_symbol_instance. Do not call this routine on a structure with valid pointers to allocated memory. Doing that will cause the pointers to be reset, causing the allocated memory to become inaccessible. Please free any dynamically allocated memory in this structure before overwriting the pointers with new data.

Return
Return values include the following:
UF_DRF_NO_ERRORS
Other UF error codes

Environment
Internal and External

See Also
UF_DRF_is_sbf_symbol, UF_DRF_create_custom_symbol_instance

History
Created in V18.0.3

Required License(s)
drafting

int UF_DRF_initialize_custom_symbol_data
(
UF_DRF_custom_symbol_p_t symbol_definition
)

UF_DRF_custom_symbol_p_t	**symbol_definition**	Input / Output	Structure to initialize

---

UF_DRF_initialize_custom_symbol_text_data (view source)

Defined in: uf_drf.h

Overview

This function will initialize the UF_DRF_custom_symbol_text_t structure. This routine should be called prior to filling this structure with valid data. Calling this routine will guarantee that the data structure does not contain garbage values. Calling this routine WILL NOT guarantee that the structure will contain valid values which can be processed by UF_DRF_create_custom_symbol_instance. Note that calling this routine on a valid structure that points to allocated memory may result in memory leaks. To prevent this, please free any dynamically allocated memory before reusing pointer variables.

Return

Return values include the following:
UF_DRF_NO_ERRORS
Other UF error codes

Environment

Internal and External

See Also

UF_DRF_create_custom_symbol_instance,

History

Created in V18.0.3

Required License(s)

drafting

```
int UF_DRF_initialize_custom_symbol_text_data
(
    UF_DRF_custom_symbol_text_p_t symbol_text
)
```

UF_DRF_custom_symbol_text_p_t	**symbol_text**	Input / Output	Structure to be initialized

---

UF_DRF_initialize_leader_data (view source)

Defined in: uf_drf.h

Overview

This function will initialize the UF_DRF_leader_t structure. This routine should be called prior to filling this structure with valid data. Calling this routine will guarantee that the data structure does not contain garbage values. Calling this routine WILL NOT guarantee that the structure will contain valid values. Note that calling this routine on a valid structure that points to allocated memory may result in memory leaks. If you are reusing an existing structure, please free your dynamically allocated memory first.

Return
Return values include:
UF_DRF_NO_ERRORS
Other UF error codes

Environment
Internal and External

History
Created in V18.0.3

Required License(s)
drafting

int UF_DRF_initialize_leader_data
(
UF_DRF_leader_p_t leader
)

UF_DRF_leader_p_t	**leader**	Input / Output	Structure to be initialized

---

UF_DRF_is_annotation_retained (view source)

Defined in: uf_drf.h

Overview
Gets a logical indicating whether the specified annotation is retained or not.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_DRF_null_object
UF_err_bad_parameter_number_1

Environment
Internal and External

Required License(s)
gateway

int UF_DRF_is_annotation_retained
(
tag_t annotation,
logical * is_retained
)

tag_t	**annotation**	Input	Tag of the annotation
logical *	**is_retained**	Output	Retained state of the annotation, TRUE if the annotation is retained

---

UF_DRF_is_block_centerline (view source)

Defined in: uf_drf.h

Overview
Checks if the input tag is a block centerline.

Environment
Internal and External

History
Originally released in v19.0

Required License(s)
gateway

```
int UF_DRF_is_block_centerline
(
    tag_t object_tag,
    logical * is_block_cline
)
```

tag_t	**object_tag**	Input	tag of the centerline
logical *	**is_block_cline**	Output	a logical value. This is true if object_tag is block centerline. This is false if object_tag is not block centerline

---

UF_DRF_is_chamfer_dimension (view source)

Defined in: uf_drf.h

Overview
UF_DRF_is_chamfer_dimension

Description -
ask if dimension is a chamfer dimension

PARAMETERS -
dim_tag, - <I> tag of object
is_cham_dim - <O> TRUE if object is cham dim
return - <O> return code:
0 = OK
if not 0 = error code

Environment
Internal and External

Required License(s)
gateway

```
int UF_DRF_is_chamfer_dimension
(
    tag_t dim_tag,
    logical * is_cham_dim
)
```

tag_t	**dim_tag**	Input	tag of object

logical *	**is_cham_dim**	Output	TRUE if object is cham dim

UF_DRF_is_inherited_pmi (view source)

Defined in: uf_drf.h

Overview

This function will return TRUE if the given annotation is an inherited PMI on a drawing sheet/member view.
Inherited PMI is the associative copy of the PMI display instance in modeling.

Input parameter annotation must be an Annotation object. Annotation object occurrences are not supported.
Use UF_ASSEM_is_occurrence and UF_ASSEM_ask_prototype_of_occ as appropriate to get the Annotation object from its occurrence.

Returns

UF_DRF_NO_ERRORS

Environment

Internal & External

History

Created in NX5.0

Required License(s)

drafting

```
int UF_DRF_is_inherited_pmi
(
    tag_t annotation,
    logical * inherited
)
```

tag_t	**annotation**	Input	The tag of the annotation object
logical *	**inherited**	Output	TRUE if the tag is an inherited PMI

UF_DRF_is_narrow_dimension (view source)

Defined in: uf_drf.h

Overview

Ask if a dimension is narrow dimension

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized

Environment

Internal and External

History
Originally released in V19.0

Required License(s)
gateway

int UF_DRF_is_narrow_dimension
(
tag_t dimension_tag,
logical * is_narrow_dimension
)

tag_t	**dimension_tag**	Input	Object tag of any object
logical *	**is_narrow_dimension**	Output	True if the dimension is narrow dimension

---

UF_DRF_is_object_out_of_date (view source)

Defined in: uf_drf.h

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

Environment
Internal & External:

See Also
UF_DRAW_ask_suppress_view_update
UF_DRAW_set_suppress_view_update
UF_DRAW_update_one_view
Refer to the example

Required License(s)
gateway

int UF_DRF_is_object_out_of_date
(
tag_t object,
logical * out_of_date
)

tag_t	**object**	Input	The view or drawing to query
logical *	**out_of_date**	Output	The status of the object, TRUE if the object is out of date.

UF_DRF_is_pmi_display_instance (view source)

Defined in: uf_drf.h

Overview
This function will return TRUE if the given annotation is a PMI display instance.

PMI display instances are those objects that are created from the PMI menu pulldown. It's important to note that the type and subtype of PMI display instances may coincide with other drafting annotations.

Input parameter object must be an Annotation object. Annotation object occurrences are not supported.
Use UF_ASSEM_is_occurrence and UF_ASSEM_ask_prototype_of_occ as appropriate to get the Annotation object from its occurrence.

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX4.0

Required License(s)
gateway

```
int UF_DRF_is_pmi_display_instance
(
    tag_t object,
    logical * is_display_instance
)
```

tag_t	**object**	Input	The tag of the annotation object
logical *	**is_display_instance**	Output	TRUE if the tag is a display instance

---

UF_DRF_is_sbf_symbol (view source)

Defined in: uf_drf.h

Overview
The following function will return TRUE if the input symbol is a user defined symbol based on the SBF file format. If the input symbol is a custom symbol with a standard part file format, this function will return FALSE.

Environment
Internal & External

History
Created in V18.0

Required License(s)
gateway

```
int UF_DRF_is_sbf_symbol
(
    tag_t symbol,
    logical * is_sbf
)
```

tag_t	**symbol**	Input	A user defined symbol tag

logical *	**is_sbf**	Output	A logical value indicating TRUE if this symbol uses the SBF definition, or FALSE if the symbol is a custom symbol

---

UF_DRF_margin_to_cline (view source)

Defined in: uf_drf.h

Overview
Convert all V9 margin objects to centerline objects

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_margin_to_cline
(
    tag_t part_tag
)
```

tag_t	**part_tag**	Input	The tag of an open part

UF_DRF_place_symbol (view source)

Defined in: uf_drf.h

Overview
Places a standalone symbol.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_place_symbol
(
    UF_DRF_symbol_create_data_t * symbol_data,
    logical is_inverted,
    logical is_mirrored,
    tag_t * symbol_tag
)
```

UF_DRF_symbol_create_data_t *	**symbol_data**	Input	symbol data (see uf_drf_types.h)
logical	**is_inverted**	Input	TRUE = inverted symbol

logical	**is_mirrored**	Input	TRUE = mirrored symbol
tag_t *	**symbol_tag**	Output	tag of standalone symbol object

---

UF_DRF_record_draft_objects (view source)

Defined in: uf_drf.h

Overview
This function will query drafting objects and store associated data. The data stored includes:
origin, text box lenght and width, text, terminator locations, line and arc data, retain state

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

See Also
UF_DRF_are_draft_objects_const

History
Created in NX4.0

Required License(s)
drafting

```
int UF_DRF_record_draft_objects
(
    void * * objs,
    logical record_view_data
)
```

void * *	**objs**	Output to UF_*free*	Drafting objects Must be freed by calling UF_DRF_free_comp_data
logical	**record_view_data**	Input	Should view data be recorded?

UF_DRF_remove_controlling_exp (view source)

Defined in: uf_drf.h

Overview
Remove the link between drafting object and controlling expression

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
drafting

```
int UF_DRF_remove_controlling_exp
(
    tag_t object
)
```

tag_t	**object**	Input	Drafting object

UF_DRF_render_arrowhead (view source)

Defined in: uf_drf.h

Overview

Calls the supplied "rendering functions for each line or arc of the arrowhead. Useful for functionality such as translation to other systems.

No NULL function pointers are permitted in the render table.

Return

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

Possible return codes can include the following
UF_err_program_not_initialized

Environment

Internal and External

History

Written in V16.0

Required License(s)

drafting

```
int UF_DRF_render_arrowhead
(
    tag_t part,
    tag_t ann,
    UF_DRF_arrow_info_p_t arrow_info,
    logical use_arrow_z,
    UF_DRF_render_table_p_t render_table,
    void * client
)
```

tag_t	**part**	Input	part for text
tag_t	**ann**	Input	CANNOT be NULL_TAG
UF_DRF_arrow_info_p_t	**arrow_info**	Input	arrowhead information
logical	**use_arrow_z**	Input	If TRUE then use the input z. Should only be used for GD&T arrows obtained from UF_GDT_ask_leader
UF_DRF_render_table_p_t	**render_table**	Input	table of rendering functions

void *	**client**	Input	application client data passed through to each function in the render table

---

UF_DRF_render_text (view source)

Defined in: uf_drf.h

Overview

Traverses lines of drafting text and calls the supplied "rendering" functions. Useful for functionality such as translation to other systems.

No NULL function pointers are permitted in the render table.

NOTE: The draw_char and draw_uds functions are not yet implemented. Text is always output as lines and arcs and user defined symbols are always output as lines.

Return

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

Possible return codes can include the following
UF_err_program_not_initialized

Environment

Internal and External

History

Written in V16.0

Required License(s)

drafting

```
int UF_DRF_render_text
(
    tag_t part,
    tag_t ann,
    int num_lines,
    char * * text,
    UF_DRF_draft_aid_text_info_p_t text_info,
    UF_DRF_render_table_p_t render_table,
    void * client
)
```

tag_t	**part**	Input	part for text
tag_t	**ann**	Input	can be NULL_TAG
int	**num_lines**	Input	number of lines of text
char * *	**text**	Input	text to "render"
UF_DRF_draft_aid_text_info_p_t	**text_info**	Input	parameters to be used for text

UF_DRF_render_table_p_t	**render_table**	Input	table of rendering functions
void *	**client**	Input	application client data, passed through to each function in the render table

---

UF_DRF_rotate_image (view source)

Defined in: uf_drf.h

Overview
Rotate the image

The angle that the image can be rotated is limited to 90-degree increments.

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_rotate_image
(
    tag_t image,
    double angle
)
```

tag_t	**image**	Input	Image to edit
double	**angle**	Input	Angle in degrees to rotate image by Must be in increments of 90-degrees

---

UF_DRF_set_annotation_template (view source)

Defined in: uf_drf.h

Overview
Set the template name which is to be used in UF_DRF_inherit_feature_data.

Environment
Internal and External

See Also
UF_DRF_inherit_type_t
UF_DRF_ask_callout_of_annotation
UF_DRF_ask_controlling_member_of_callout
UF_DRF_ask_number_of_rows_in_callout
UF_DRF_ask_callout_row_members
UF_DRF_inherit_feature_data
UF_DRF_ask_annotation_template

History
Originally released in v19.0

Required License(s)
drafting

```
int UF_DRF_set_annotation_template
(
    char * annotation_template_name
)
```

char *	**annotation_template_name**	Input	the name of the annotation template part which will be used by UF_DRF_inherit_feature_data. The name is the same as the name appearing in the Feature Parameters dialog and does not contain the directory in which the part resides or the extension .atp.prt, for example, ansi or iso_din

---

UF_DRF_set_appended_text (view source)

Defined in: uf_drf.h

Overview
The following function will set the appended text for a dimension.
The appended text will only be changed for those locations specified.
If num_lines is input as 0,
the appended text for that location will be deleted.

Environment
Internal & External

See Also
UF_DRF_ask_appended_text
UF_DRF_appended_text_t

History
Created in V17.0
In NX2.0.1 this function was enhanced to support appended text
for horizontal and vertical ordinate dimensions. However,
for ordinate dimensions, appended text is currently limited to

1 location or before and after or above and below.

Required License(s)
drafting

int UF_DRF_set_appended_text
(
tag_t dimension,
int num_text,
UF_DRF_appended_text_p_t appended_text
)

tag_t	**dimension**	Input	Dimension object to query
int	**num_text**	Input	Number of appended text
UF_DRF_appended_text_p_t	**appended_text**	Input	Appended text

---

UF_DRF_set_areafill_angle (view source)

Defined in: uf_drf.h

Overview
Sets the area fill angle for the specified area fill object.

Environment
Internal and External

See Also
UF_DRF_ask_areafill_data
UF_DRF_set_areafill_scale
The example in UF_DRF_set_areafill_material
sets the material type, scale, and angle for all Area Fills in the part.

Required License(s)
drafting

int UF_DRF_set_areafill_angle
(
tag_t areafill_id,
double angle
)

tag_t	**areafill_id**	Input	Area fill object identifier
double	**angle**	Input	Area fill angle in radians

UF_DRF_set_areafill_material (view source)

Defined in: uf_drf.h

Overview

Sets the area fill material for the specified area fill object. The area fill material type structure, UF_DRF_valid_material_t defines the type of material.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

int UF_DRF_set_areafill_material
(
tag_t areafill_id,
UF_DRF_valid_material_t material
)

tag_t	**areafill_id**	Input	Area Fill object identifier
UF_DRF_valid_material_t	**material**	Input	Area Fill material type

---

UF_DRF_set_areafill_scale (view source)

Defined in: uf_drf.h

Overview
Sets the area fill scale for the specified area fill object.

Environment
Internal and External

See Also
UF_DRF_ask_areafill_data
UF_DRF_set_areafill_angle
The example in UF_DRF_set_areafill_material sets the material type, scale, and angle for all Area Fills in the part.

Required License(s)
drafting

int UF_DRF_set_areafill_scale
(
tag_t areafill_id,
double scale
)

tag_t	**areafill_id**	Input	Area fill object identifier
double	**scale**	Input	Area fill scale

UF_DRF_set_associative_origin (view source)

Defined in: uf_drf.h

Overview
The following function will set the associative origin information for the annotation.

Note that UF_DRF_ORIGIN_ALIGNED_WITH_ARROWS is not supported by this function.

Environment
Internal and External

See Also
UF_DRF_has_associative_origin
UF_DRF_ask_associative_origin

History
Created in V17.0

Required License(s)
drafting

```
int UF_DRF_set_associative_origin
(
    tag_t drafting_entity,
    UF_DRF_associative_origin_p_t origin_data,
    double origin [ 3 ]
)
```

tag_t	**drafting_entity**	Input	Dimension or drafting object to query.
UF_DRF_associative_origin_p_t	**origin_data**	Input	Data used to define the associative origin.
double	**origin [3]**	Input	Origin of the annotation in absolute coords.

UF_DRF_set_associativity_data (view source)

Defined in: uf_drf.h

Overview
The following function edit all of the associativities for the annotation object.

Environment
Internal or External

History
Originally released in V19.0

Required License(s)
drafting

```
int UF_DRF_set_associativity_data
(
    tag_t object,
    int num_associativities,
    UF_DRF_object_assoc_data_p_t associativity_data
```

)

tag_t	**object**	Input	Drafting object or dimension to edit for this operation.
int	**num_associativities**	Input	The number of associativities on the object in which to edit for this operation.
UF_DRF_object_assoc_data_p_t	**associativity_data**	Input	Array of new associativity information.

UF_DRF_set_chamfer_dimension_data (view source)

Defined in: uf_drf.h

Overview

UF_DRF_set_chamfer_dimension_data

Description -
set chamfer dimension preferences

PARAMETERS -
cham_dim_tag, - <I> tag of cham dim
cham_dim_data - <I> cham dim prefs
return - <O> return code:
0 = OK
if not 0 = error code

Environment

Internal and External

Required License(s)

drafting

int UF_DRF_set_chamfer_dimension_data
(
tag_t cham_dim_tag,
UF_DRF_chamfer_dimension_data_t cham_dim_data
)

tag_t	**cham_dim_tag**	Input	tag of cham dim
UF_DRF_chamfer_dimension_data_t	**cham_dim_data**	Input	cham dim prefs

---

UF_DRF_set_custom_symbol_angle (view source)

Defined in: uf_drf.h

Overview

Set the angle for a custom symbol instance.

Environment

Internal and External

History

Originally released in v19.0

Required License(s)
drafting

int UF_DRF_set_custom_symbol_angle
(
tag_t symbol_tag,
double angle
)

tag_t	**symbol_tag**	Input	tag of a custom symbol instance
double	**angle**	Input	double representing the new angle for this instance

---

UF_DRF_set_custom_symbol_scale (view source)

Defined in: uf_drf.h

Overview
Set the symbol scale for a custom symbol instance.

Environment
Internal and External

History
Originally released in v19.0

Required License(s)
drafting

int UF_DRF_set_custom_symbol_scale
(
tag_t symbol_tag,
double scale
)

tag_t	**symbol_tag**	Input	tag of a custom symbol instance
double	**scale**	Input	double value of the symbol scale

UF_DRF_set_customer_sbf_file (view source)

Defined in: uf_drf.h

Overview
Sets as current the customer default symbol font definition file.

Environment
Internal and External

See Also

Refer to the example

Required License(s)
drafting

```
int UF_DRF_set_customer_sbf_file
(
    void
)
```

UF_DRF_set_cyl_dim (view source)

Defined in: uf_drf.h

Overview
Changes the double flag of a cylindrical dimension. Any cylindrical dimension may be modified with this function. It is up to the user to determine which dimensions to modify.

Environment
Internal and External

See Also
Refer to the example

History
Original release was in V14.0.

Required License(s)
drafting

```
int UF_DRF_set_cyl_dim
(
    tag_t dim_obj,
    int double_flag,
    int * status
)
```

tag_t	**dim_obj**	Input	Target cylindrical dimension object identifier.
int	**double_flag**	Input	Double flag to be used on the dimension 0 = sets dimension value to be the distance between the two associated objects. 1 = sets dimension value to be twice the value between associated objects.
int *	**status**	Output	Return status: 0 = Cylindrical dimension changed accordingly. 1 = Cylindrical dimension was not changed since the new double flag is the same as the one it has. 2 = dimension was not changed since it is not a cylindrical dimension.

UF_DRF_set_diameter_radius_preferences (view source)

Defined in: uf_drf.h

Overview
Sets preferences for the display of radial dimensions.

Environment
Internal & External

See Also
UF_DRF_ask_diameter_radius_preferences

History
Originally released in V16.0

Required License(s)
drafting

int UF_DRF_set_diameter_radius_preferences
(
UF_DRF_diameter_radius_preferences_p_t diameter_radius_preferences
)

UF_DRF_diameter_radius_preferences_p_t	**diameter_radius_preferences**	Input	pointer to preferences structure to be used to set the diameter/radius preferences

---

UF_DRF_set_dim_appended_text_space_factor (view source)

Defined in: uf_drf.h

Overview
Set the factor for the space between the dimension and the appended text for the specified dimension. This factor controls the spacing between the before appended text and the next piece of dimension text to its right,
and the spacing between the after appended text and the next piece of dimension text to its left.
This factor is applied to the appended text character size to determine the spacing.

Environment
Internal and External

History
NX 2.0 release.

Required License(s)
drafting

int UF_DRF_set_dim_appended_text_space_factor
(
tag_t dimension,
double space_factor
)

tag_t	**dimension**	Input	dimension tag
double	**space_factor**	Input	factor for spacing between dimension and appended text

UF_DRF_set_dim_dim_line_space_factor (view source)

Defined in: uf_drf.h

Overview

Set the factor for the space between the dimension and the dimension line for the specified dimension.
This factor controls the spacing between the bottom most piece of dimension text and the dimension line only when the text orientation is UF_DRF_DIMENSION_TEXT_OVER_DIMENSION_LINE.
This factor is applied to the dimension text character size to determine the spacing.

Environment

Internal and External

History

NX 2.0 release.

Required License(s)

drafting

```
int UF_DRF_set_dim_dim_line_space_factor
(
    tag_t dimension,
    double space_factor
)
```

tag_t	**dimension**	Input	dimension tag
double	**space_factor**	Input	factor for spacing between dimension and dimension line

---

UF_DRF_set_dim_inspection_type (view source)

Defined in: uf_drf.h

Overview

Description -
Set the inspection preference for the given dimension object.

Input -
inspection_type - inspection type for the given dimension object.

Output - None

Return -
= 0 UF_DRF_NO_ERRORS
Error code if not zero.
UF_DRF_invalid_object -- if given tag is invalid.
UF_err_program_not_initialized

Required License(s)

drafting

```
int UF_DRF_set_dim_inspection_type
(
```

```
    tag_t dim_tag,
    UF_DRF_inspection_type_t inspection_type
)
```

tag_t	**dim_tag**	Input	dimension tag
UF_DRF_inspection_type_t	**inspection_type**	Input	inspection type for the give dimension

---

UF_DRF_set_dim_reference_type (view source)

Defined in: uf_drf.h

Overview

Description -
Set the reference preference for the given dimension object.

Input -
ref_type - reference type for the given dimension object.

Output - None

Return -
= 0 UF_DRF_NO_ERRORS
Error code if not zero.
UF_DRF_invalid_object -- if given tag is invalid.
UF_err_program_not_initialized

Required License(s)

drafting

```
int UF_DRF_set_dim_reference_type
(
    tag_t dim_tag,
    UF_DRF_reference_symbol_type_t ref_type
)
```

tag_t	**dim_tag**	Input	dimension tag
UF_DRF_reference_symbol_type_t	**ref_type**	Input	reference type to be set to dimension

---

UF_DRF_set_dim_tolerance_text_space_factor (view source)

Defined in: uf_drf.h

Overview

Set the factor for the space between the dimension and the tolerance text for the specified dimension.
This factor controls the spacing between the dimension text and the after tolerance text.
This factor is applied to the tolerance text character size to determine the spacing.

Environment

Internal and External

History

NX 2.0 release.

Required License(s)
drafting

int UF_DRF_set_dim_tolerance_text_space_factor
(
tag_t dimension,
double space_factor
)

tag_t	**dimension**	Input	dimension tag
double	**space_factor**	Input	factor for spacing between dimension and tolerance text

---

UF_DRF_set_dimension_preferences (view source)

Defined in: uf_drf.h

Overview
Sets dimension preferences for arrow and line formatting, type of placement, tolerance and precision, and text orientation.

Environment
Internal & External

See Also
UF_DRF_ask_dimension_preferences
UF_DRF_dimension_preferences_t

History
Originally released in V16.0

Required License(s)
drafting

int UF_DRF_set_dimension_preferences
(
UF_DRF_dimension_preferences_p_t dimension_preferences
)

UF_DRF_dimension_preferences_p_t	**dimension_preferences**	Input	pointer to preferences structure to be used to set dimension preferences

---

UF_DRF_set_dimension_preferences1 (view source)

Defined in: uf_drf.h

Overview
Sets dimension preferences for arrow and line formatting, type of placement, tolerance and precision, and text orientation.

Environment
Internal & External

See Also

UF_DRF_ask_dimension_preferences
UF_DRF_dimension_preferences_t

History
Originally released in V8.0.1

Required License(s)
drafting

int UF_DRF_set_dimension_preferences1
(
UF_DRF_dimension_preferences1_p_t dimension_preferences
)

UF_DRF_dimension_preferences1_p_t	**dimension_preferences**	Input	pointer to preferences1 structure to be used to set dimension preferences

---

UF_DRF_set_dimension_set_offset (view source)

Defined in: uf_drf.h

Overview
Set the offset of the dimension in a dimension set or the offset of all dimensions in the dimension set.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1 - if the dimension tag is invalid

Environment
Internal and External

History
Originally released in v18.0

Required License(s)
drafting

int UF_DRF_set_dimension_set_offset
(
tag_t dimension,
double offset
)

tag_t	**dimension**	Input	Tag of the given dimension. If this tag is a tag of dimension set, it will set offset for all dimensions in the set.
double	**offset**	Input	Dimension set offset of the given dimension.

UF_DRF_set_draft_common (view source)

Defined in: uf_drf.h

Overview
Set the drafting common values for a particular drafting object.
This is used to initialize prior to calling UF_DRF_count_text_substring, UF_DRF_get_symbol_divider and UF_DRF_get_text_substring

Environment
Internal and External

See Also
UF_DRF_count_text_substring
UF_DRF_ask_ann_data
UF_DRF_get_text_substring
UF_DRF_get_symbol_divider

Required License(s)
drafting

```
int UF_DRF_set_draft_common
(
    tag_t * object
)
```

tag_t *	**object**	Input	Tag of the object to set the drafting common for.

UF_DRF_set_hatch_fill_preferences (view source)

Defined in: uf_drf.h

Overview
Sets the preferences for crosshatching and area fill

Environment
Internal & External

See Also
UF_DRF_ask_preferences

History
Originally released in V16.0

Required License(s)
drafting

```
int UF_DRF_set_hatch_fill_preferences
(
    UF_DRF_hatch_fill_preferences_p_t hatch_fill_preferences
)
```

UF_DRF_hatch_fill_preferences_p_t	**hatch_fill_preferences**	Input	pointer to structure to be used to set the hatch/fill preferences

---

UF_DRF_set_image_align_position (view source)

Defined in: uf_drf.h

Overview
Edit the image alignment position

The image alignment position controls the placement of the image's origin with respect to the box created by the image width and height.

The image angle is limited to 90 degree increments.

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

```
int UF_DRF_set_image_align_position
(
    tag_t image,
    UF_DRF_align_position_t align_position
)
```

tag_t	**image**	Input	Image to edit
UF_DRF_align_position_t	**align_position**	Input	New image alignment position

---

UF_DRF_set_image_aspect_ratio_lock (view source)

Defined in: uf_drf.h

Overview
Edit the image's aspect ratio locked status.

The aspect ratio locked status controls whether or not the ratio of the image's width to its height can change when the image size changes.

Returns

UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_height
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

int UF_DRF_set_image_aspect_ratio_lock
(
tag_t image,
logical lock_aspect_ratio
)

tag_t	**image**	Input	Image to edit
logical	**lock_aspect_ratio**	Input	TRUE to not allow the image's aspect ratio to change

---

UF_DRF_set_image_height (view source)

Defined in: uf_drf.h

Overview
Edit the image height

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_width
UF_DRF_rotate_image
UF_DRF_flip_image_about_height

UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

int UF_DRF_set_image_height
(
tag_t image,
double height
)

tag_t	**image**	Input	Image to edit
double	**height**	Input	New image height in drawing sheet units

---

UF_DRF_set_image_width (view source)

Defined in: uf_drf.h

Overview
Edit the image width

Returns
UF_DRF_NO_ERRORS if the image edit was successful

Environment
Internal and External

See Also
Refer to the example
UF_DRF_create_image_from_file
UF_DRF_create_image
UF_DRF_init_image_data
UF_DRF_ask_image_data
UF_DRF_free_image_data
UF_DRF_set_image_align_position
UF_DRF_set_image_aspect_ratio_lock
UF_DRF_set_image_height
UF_DRF_rotate_image
UF_DRF_flip_image_about_height
UF_DRF_flip_image_about_width

History
This function was originally released in NX2.0.

Required License(s)
drafting

int UF_DRF_set_image_width
(
tag_t image,
double width
)

tag_t	**image**	Input	Image to edit

double	**width**	Input	New image width in drawing sheet units

---

UF_DRF_set_lettering_preferences (view source)

Defined in: uf_drf.h

Overview
Sets the lettering preferences for dimension, appended, tolerance and general (notes, id symbols, etc,) text.

Environment
Internal & External

See Also
UF_DRF_ask_lettering_preferences
UF_DRF_lettering_preferences_t

History
Originally released in V16.0
Horizontal text justification preference and GD&T frame height factor preference added in V17.0.
Dimension /Appended Text Space Factor,
Dimension/ Tolerance Text Space Factor, and
Dimension/Dimension Line Space Factor added in NX 2.0.

Required License(s)
drafting

```
int UF_DRF_set_lettering_preferences
(
    UF_DRF_lettering_preferences_p_t lettering_preferences
)
```

UF_DRF_lettering_preferences_p_t	**lettering_preferences**	Input	pointer to preferences structure which will be used to set the lettering preferences

---

UF_DRF_set_line_arrow_preferences (view source)

Defined in: uf_drf.h

Overview
Sets preferences that apply to leaders, arrows and extension lines for both dimensions and other annotations

Environment
Internal & External

See Also
UF_DRF_ask_line_arrow_preferences
UF_DRF_line_arrow_preferences_t

History

Originally released in V16.0

Required License(s)
drafting

int UF_DRF_set_line_arrow_preferences
(
UF_DRF_line_arrow_preferences_t * line_arrow_preferences
)

UF_DRF_line_arrow_preferences_t *	**line_arrow_preferences**	Input	pointer to preferences structure to be used to set the line/arrow preferences

---

UF_DRF_set_narrow_dimension_data (view source)

Defined in: uf_drf.h

Overview
Set narrow dimension parameters for a linear dimension.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_invalid_linear_dim_form - if the dimension given is not linear dimension.
UF_DRF_INVALID_NARROW_DIMENSION_DISPLAY_TYPE - if dimension display type is not correct.
UF_DRF_INVALID_NARROW_DIMENSION_LEADER_ANGLE - if leader angle is not greater than 0.0 or not less than 180.0
UF_DRF_INVALID_NARROW_DIMENSION_TEXT_ORIENTATION - if text orientation is not valid.
UF_err_program_not_initialized

Environment
Internal and External

History
Originally released in V19.0

Required License(s)
drafting

int UF_DRF_set_narrow_dimension_data
(
tag_t dimension_tag,
UF_DRF_narrow_dimension_info_p_t narrow_data
)

tag_t	**dimension_tag**	Input	Object tag of a linear dimension
UF_DRF_narrow_dimension_info_p_t	**narrow_data**	Input	Data of narrow dimension preferences

---

UF_DRF_set_object_preferences (view source)

Defined in: uf_drf.h

Overview

Sets the object preferences for a specified annotation.

Return

Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following:
UF_DRF_NO_ERRORS - No error
UF_DRF_NOT_DRAFTING_OBJECT
UF_DRF_invalid_object
UF_DRF_crosshatch_file_not_found
UF_DRF_material_definition_not_found
UF_DRF_unable_to_create_crosshatching

Environment

Internal and External

See Also

UF_DRF_ask_object_preferences
UF_DRF_ask_preferences
UF_DRF_set_preferences
UF_DRF_ask_ang_obj_units_format
Refer to the example
See drafting paramters

History

Original release was in V13.0. This function replaces uc5551.
Updated in V15 to increase the size of the character strings.
Updated in V16.0 to return UF_get_fail_message error codes
mpr[51], area fill tolerance, is now obsolete; use mpr[13] instead.
In V17.0, a separate preference controls the angular nominal and tolerance units format. This routine will set both nominal and tolerance format with nominal format value when angular nominal format gets changed by user.

Required License(s)

drafting

```
int UF_DRF_set_object_preferences
(
    tag_t drf_object_tag,
    int mpi [ 100 ] ,
    double mpr [ 70 ] ,
    const char * radius_val,
    const char * diameter_val
)
```

tag_t	**drf_object_tag**	Input	Drafting object Identifier
int	**mpi [100]**	Input	MPI Array [100 elements] The size of this array is defined by NUM_INT_PARAMS
double	**mpr [70]**	Input	MPR Array [70 elements] The size of this array is defined by NUM_REAL_PARAMS
const char *	**radius_val**	Input	Radius Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char radius_val[27];

const char *	**diameter_val**	Input	Diameter Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char diameter_val[27];

UF_DRF_set_origin (view source)

Defined in: uf_drf.h

Overview
Set the new origin of the annotation object.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1

Environment
Internal and External

History
Written in V16.0

Required License(s)
drafting

```
int UF_DRF_set_origin
(
    tag_t annotation,
    double new_origin [ 3 ]
)
```

tag_t	**annotation**	Input	Tag of the annotation to modify.
double	**new_origin [3]**	Input	New origin in model coordinates for the annotation.

UF_DRF_set_plot_drawing_images (view source)

Defined in: uf_drf.h

Overview
Indicate whether or not to plot raster images on drawing sheets when drawing sheets are plotted.

Returns
UF_DRF_NO_ERRORS if the preference edit was successful

Environment
Internal and External

See Also
UF_DRF_create_image_from_file
UF_DRF_create_image

History
This function was originally released in NX3.0.

Required License(s)
drafting

int UF_DRF_set_plot_drawing_images
(
logical plot_images
)

logical	**plot_images**	Input	TRUE to plot raster images on drawing sheets FALSE to not plot raster images on drawing sheets

---

UF_DRF_set_preferences (view source)

Defined in: uf_drf.h

Overview
Uses the arrays and character strings to set the drafting parameters.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following:
UF_DRF_NO_ERRORS - No error
UF_DRF_crosshatch_file_not_found
UF_DRF_INVALID_CROSSHATCH_FILE_FORMAT
UF_DRF_material_definition_not_found

Before NX5 there was only one preference for dimension text orientation. In NX5, a separate preference for Ordinate dimension text orientation was added. The Open C API was not enhanced. UF_DRF_set_preferences will set both the general text orientation and the ordinate text orientation with the same value. This was done so legacy programs would continue to work.

Environment
Internal and External

See Also
UF_DRF_ask_preferences
UF_DRF_ask_object_preferences
UF_DRF_set_object_preferences
Refer to the example
See drafting paramters

History
Original release was in V13.0. This function replaces uc/uf5521.
Updated in V15 to increase the size of the character strings.
Updated in V16.0 to return UF_get_fail_message error codes.
In V16.0, there are separate text angle preferences for dimensions and drafting aid. UF_DRF_set_preferences sets both text angle preferences.
If the input material index (mpi[31]) is input as -999, the crosshatch material preferences will not be changed.
mpr[51], area fill tolerance, is now obsolete; use mpr[13] instead.
In V17.0, a separate preference controls the angular nominal and tolerance units format. This routine will set both nominal and tolerance format with

nominal format value when angular nominal format gets changed by user.

Required License(s)
drafting

```
int UF_DRF_set_preferences
(
    int mpi [ 100 ] ,
    double mpr [ 70 ] ,
    const char * radius_val,
    const char * diameter_val
)
```

int	**mpi [100]**	Input	MPI Array [100 elements] The size of this array is defined by NUM_INT_PARAMS
double	**mpr [70]**	Input	MPR Array [70 elements] The size of this array is defined by NUM_REAL_PARAMS
const char *	**radius_val**	Input	Radius Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char radius_val[27];
const char *	**diameter_val**	Input	Diameter Symbol String. This can be at most six characters, however due to internal requirements, the buffer must be allocated as char diameter_val[27];

UF_DRF_set_retain_color_font_width (view source)

Defined in: uf_drf.h

Overview
Sets the color, font and widths used for retained annotations.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1
UF_err_bad_parameter_number_2
UF_err_bad_parameter_number_3

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_set_retain_color_font_width
(
    int color,
    int font,
    int width
)
```

int	**color**	Input	Color to use for retained annotations
int	**font**	Input	Font to use for retained annotations
int	**width**	Input	Line width to use for retained annotations

---

UF_DRF_set_retained_state (view source)

Defined in: uf_drf.h

Overview
Sets the retained state for annotations whose dependencies have expired.
If UF_DRF_KEEP_RETAINED_ANNOTATIONS, annotations are retained.
If UF_DRF_DELETE_RETAINED_ANNOTATIONS, annotations are deleted.

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_err_program_not_initialized
UF_err_bad_parameter_number_1

Environment
Internal and External

Required License(s)
drafting

```
int UF_DRF_set_retained_state
(
    UF_DRF_retained_state_t state
)
```

UF_DRF_retained_state_t	**state**	Input	Behavior state for retained annotations. Either 'UF_DRF_KEEP_RETAINED_ANNOTATIONS' or 'UF_DRF_DELETE_RETAINED_ANNOTATIONS'

UF_DRF_set_specified_sbf_file (view source)

Defined in: uf_drf.h

Overview
Sets as current the specified symbol font definition file.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_set_specified_sbf_file
(
    const char sbf_name [ UF_CFI_MAX_FILE_NAME_BUFSIZE ]
)
```

const char	**sbf_name [UF_CFI_MAX_FILE_NAME_BUFSIZE]**	Input	file to be set

---

UF_DRF_set_suppress_pre_zeros (view source)

Defined in: uf_drf.h

Overview
Sets the global preference for suppress preceding zeros.

Environment
Internal and External

See Also
UF_DRF_set_units_format_preferences
Refer to the example

History
This function was originally released in V15.0.

Required License(s)
drafting

```
int UF_DRF_set_suppress_pre_zeros
(
    logical option
)
```

logical	**option**	Input	Suppress preceding zeros mode

UF_DRF_set_suppress_view_update (view source)

Defined in: uf_drf.h

Overview
Change the value of the Suppress View Update preference. This preference is an environment setting and is not specific to a part. If the preference is TRUE, then functions which perform implicit drawing updates, will not update the drawing member views.

Environment
Internal & External:

See Also
UF_DRF_ask_suppress_view_update

Required License(s)
drafting

```
int UF_DRF_set_suppress_view_update
(
    logical suppress_view_update
)
```

logical	**suppress_view_update**	Input	the new setting of view update suppression

---

UF_DRF_set_symbol_preferences (view source)

Defined in: uf_drf.h

Overview
Sets preferences that apply to ID, User Defined, Centerline, Intersection, Target and GD&T symbols.

Environment
Internal & External

See Also
UF_DRF_ask_symbol_preferences
UF_DRF_symbol_preferences_t

History
Originally released in V16.0

Required License(s)
drafting

```
int UF_DRF_set_symbol_preferences
(
    UF_DRF_symbol_preferences_p_t symbol_preferences
)
```

UF_DRF_symbol_preferences_p_t	**symbol_preferences**	Input	pointer to preferences structure to be used to set the symbol preferences

---

UF_DRF_set_text_above_leader (view source)

Defined in: uf_drf.h

Overview
Sets the global value for the text above the leader attribute. This attribute controls displaying text above the leader stub when creating labels and some dimensions.

If the attribute is set to UF_DRF_NO_TEXT_ABOVE_LEADER, the text will not be displayed above the leader stub. The vertical text justification of Top, Middle or Bottom applies.

For all the other options, the text will be displayed above the leader stub. The BOTTOM options display the leader stub under the

bottom line of the text. The TOP options display the leader stub under the top line of the text. The UNDERLINE options will underline all other lines of text. The MAX options will extend the stub and optional underlines to the maximum text length. Without MAX, the stub and optional underlines will extend to the length of the individual text line it is under.

For dimensions, the text above leader attribute only applies to radial type dimensions (hole, diameter, radius, concentric circle, and folded radius) and only applies if the text alignment is Horizontal or Angle. Currently, for dimensions, only the option UF_DRF_LEADER_BOTTOM_TEXT_MAX is supported.

Environment
Internal and External

History
This function was originally released in V15.0.

Required License(s)
drafting

int UF_DRF_set_text_above_leader
(
UF_DRF_text_above_leader_t option
)

UF_DRF_text_above_leader_t	**option**	Input	Text above leader attribute UF_DRF_NO_TEXT_ABOVE_LEADER UF_DRF_LEADER_BOTTOM_TEXT_MAX UF_DRF_LEADER_BOTTOM_TEXT_MAX_UNDERLINE UF_DRF_LEADER_BOTTOM_TEXT UF_DRF_LEADER_BOTTOM_TEXT_UNDERLINE UF_DRF_LEADER_TOP_TEXT_MAX UF_DRF_LEADER_TOP_TEXT_MAX_UNDERLINE UF_DRF_LEADER_TOP_TEXT UF_DRF_LEADER_TOP_TEXT_UNDERLINE

UF_DRF_set_tolerance (view source)

Defined in: uf_drf.h

Overview
This function will set the tolerance used when compare the real data values of two drafting entities whose data was recorded using the UF_DRF_record_draft_objects function

Returns
UF_DRF_NO_ERRORS

Environment
Internal & External

History
Created in NX4.0

Required License(s)
drafting

```
int UF_DRF_set_tolerance
(
    float tolerance
)
```

float	**tolerance**	Input	Tolerance to use when comparing real data

---

UF_DRF_set_uds_size (view source)

Defined in: uf_drf.h

Overview
The following function will set the global preferences for the user defined symbol length / height or scale / aspect ratio.

Environment
Internal or External

History
Originally released in V19.0

Required License(s)
drafting

```
int UF_DRF_set_uds_size
(
    UF_DRF_uds_size_p_t uds_size
)
```

UF_DRF_uds_size_p_t	**uds_size**	Input	User defined symbol scale / aspect ratio or length / height parameters.

UF_DRF_set_ugdefault_sbf_file (view source)

Defined in: uf_drf.h

Overview
Sets as current the NX default symbol font definition file.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
drafting

```
int UF_DRF_set_ugdefault_sbf_file
(
    void
)
```

UF_DRF_set_units_format_preferences (view source)

Defined in: uf_drf.h

Overview
Sets preferences for the display of linear and angular dimensions values as well as dual dimension format.

Environment
Internal & External

See Also
UF_DRF_ask_units_format_preferences
UF_DRF_units_format_preferences_t

History
Originally released in V16.0. In v17.0, the data structure has changed.
An element controlling the units of the tolerance of angular dimension and an element controlling the zero suppression for angular dimension have been added.
The display leading zeros option has been removed.

Required License(s)
drafting

```
int UF_DRF_set_units_format_preferences
(
    UF_DRF_units_format_preferences_p_t units_format_preferences
)
```

UF_DRF_units_format_preferences_p_t	**units_format_preferences**	Input	pointer to preferences structure to used to set the units/ format preferences

UF_DRF_set_vertical_note (view source)

Defined in: uf_drf.h

Overview
This function will set a note to be either horizontal or vertical.

Environment
Internal and External

History
Created in V16.0

Required License(s)
drafting

```
int UF_DRF_set_vertical_note
(
    tag_t note,
    logical is_vertical
```

)

tag_t	**note**	Input	Tag of the note.
logical	**is_vertical**	Input	If TRUE, set the note to be vertical, if FALSE set the note horizontal. If the object is not a note, no action is taken, and the return code will be zero.

UF_DRF_set_weld_symbol_standard (view source)

Defined in: uf_drf.h

Overview
Set the weld symbol standard

Return
Return code:
0 = No error
not 0 = Error code
Possible return codes can include the following
UF_DRF_NO_ERRORS - No error
UF_err_program_not_initialized - Open C API has not been initialized

Environment
Internal and External

History
Originally released in vNX1.0.2

Required License(s)
gateway

int UF_DRF_set_weld_symbol_standard
(
char * standard
)

char *	**standard**	Input	standard of the weld symbol

UF_DRF_set_xhatch_mat (view source)

Defined in: uf_drf.h

Overview
Sets the crosshatch material using the given name.

Return
void

Environment
Internal and External

Required License(s)
drafting

```
void UF_DRF_set_xhatch_mat
(
    const char * file_name,
    const char * material_name,
    int * util,
    int * error
)
```

const char *	**file_name**	Input	The crosshatch definition file name
const char *	**material_name**	Input	The desired crosshatch material name
int *	**util**	Input	Flag indicating whether to search the utility directory first or the user's working directory. 1 = Search utility directory first. 2 = Search home directory first then utility directory.
int *	**error**	Output	Error flag. Value > 0 indicates an error has occurred.

UF_DRF_transfer_to_drawing (view source)

Defined in: uf_drf.h

Overview
This function will transfer an annotation view dependent in a drawing member view or a model view to the drawing sheet specified.

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

Environment
Internal and External

History
Created in V16.0

Required License(s)
drafting

```
int UF_DRF_transfer_to_drawing
(
    tag_t annotation,
    tag_t member_view,
    tag_t drawing,
    logical in_drawing_plane
)
```

tag_t	**annotation**	Input	Annotation to transfer
tag_t	**member_view**	Input	Tag of the member view in which the associativities should be related.

tag_t	**drawing**	Input	Tag of the drawing on which the annotation should now lie.
logical	**in_drawing_plane**	Input	If TRUE, the annotation will lie in the drawing plane, otherwise, the orientation will be retained.

UF_DRF_update_views (view source)

Defined in: uf_drf.h

Overview

Update one or more drawing member views on a drawing. The view update process includes resectioning section views, regenerating silhouettes, and, if applicable, updating hidden line display. This routine will:
1) force update one view (method == UF_DRF_UPDATE_NAMED & view_name),
2) update all automatic views (method == UF_DRF_UPDATE_AUTO) on the specified drawing,
3) update all out-of-date views (method == UF_DRF_UPDATE_ALL) on the specified drawing, or
4) force update all views (method == UF_DRF_UPDATE_FORCE) on the specified drawing.

If not current, the specified drawing will be retrieved and made current.
If no drawing is specified, then the current drawing is assumed.

Environment

Internal and External

Required License(s)

drafting

```
int UF_DRF_update_views
(
    char * drawing_name,
    int method,
    char * view_name
)
```

char *	**drawing_name**	Input	name of drawing whose views are to be updated
int	**method**	Input	one of 'UF_DRF_UPDATE_AUTO', 'UF_DRF_UPDATE_FORCE', 'UF_DRF_UPDATE_ALL', or 'UF_DRF_UPDATE_NAMED'
char *	**view_name**	Input	if method = 'UF_DRF_UPDATE_NAMED', the name of the view to update

---