uf5063 (view source)

Defined in: uf_curve.h

Overview

uf5063 create an arc through three points
replaced by UF_CURVE_create_arc_thru_3pts

Required License(s)

gateway

```
void uf5063
(
    int * ip1,
    double * rp2,
    double * rp3,
    double * rp4,
    tag_t * nr5
)
```

int *	**ip1**
double *	**rp2**
double *	**rp3**
double *	**rp4**
tag_t *	**nr5**

---

uf5070 (view source)

Defined in: uf_curve.h

Overview

uf5070 create conic
replaced by UF_CURVE_convert_conic_to_std and UF_CURVE_create_conic

Required License(s)

gateway

```
void uf5070
(
    double * rp1,
    double * rp2,
    double * rp3,
    double * rp4,
    double * rp5,
    double * rp6,
    double * ra7,
    double * ra8,
    double * rp9,
    tag_t * nr10
)
```

double *	**rp1**
double *	**rp2**
double *	**rp3**
double *	**rp4**
double *	**rp5**
double *	**rp6**
double *	**ra7**
double *	**ra8**
double *	**rp9**
tag_t *	**nr10**

---

uf5071 (view source)

Defined in: uf_curve.h

Overview

uf5071 read conic real data
replaced by UF_CURVE_ask_conic_data

Required License(s)

gateway

```
void uf5071
(
    tag_t * np1,
    double * rar2
)
```

tag_t *	**np1**
double *	**rar2**

---

uf5072 (view source)

Defined in: uf_curve.h

Overview

uf5072 edit conic real data
No replacement

Required License(s)
gateway

void uf5072
(
tag_t * np1,
double * ra2
)

tag_t *	**np1**
double *	**ra2**

---

uf5072x (view source)

Defined in: uf_curve.h

Overview
uf5072x edit conic real data without update
No Replacement

Required License(s)
gateway

void uf5072x
(
tag_t * np1,
double * ra2
)

tag_t *	**np1**
double *	**ra2**

---

uf5073 (view source)

Defined in: uf_curve.h

Overview
uf5073 read conic data
Replaced by UF_CURVE_ask_conic_data and UF_CURVE_convert_conic_to_gen

Required License(s)
gateway

void uf5073
(

```
    tag_t * np1,
    double * rr2,
    double * rr3,
    double * rr4,
    double * rr5,
    double * rr6,
    double * rr7,
    double * rar8,
    double * rar9,
    double * rr10
)
```

tag_t *	**np1**
double *	**rr2**
double *	**rr3**
double *	**rr4**
double *	**rr5**
double *	**rr6**
double *	**rr7**
double *	**rar8**
double *	**rar9**
double *	**rr10**

---

uf5074 (view source)

Defined in: uf_curve.h

Overview

uf5074 edit conic data
No Replacement

Required License(s)

gateway

```
void uf5074
(
    tag_t * np1,
    double * rp2,
    double * rp3,
    double * rp4,
    double * rp5,
    double * rp6,
    double * rp7,
    double * ra8,
    double * ra9,
    double * rp10
```

)

tag_t *	**np1**
double *	**rp2**
double *	**rp3**
double *	**rp4**
double *	**rp5**
double *	**rp6**
double *	**rp7**
double *	**ra8**
double *	**ra9**
double *	**rp10**

---

uf5074x (view source)

Defined in: uf_curve.h

Overview

uf5074x edit conic data without update
No Replacement

Required License(s)

gateway

void uf5074x
(
tag_t * np1,
double * rp2,
double * rp3,
double * rp4,
double * rp5,
double * rp6,
double * rp7,
double * ra8,
double * ra9,
double * rp10
)

tag_t *	**np1**
double *	**rp2**
double *	**rp3**

double *	**rp4**
double *	**rp5**
double *	**rp6**
double *	**rp7**
double *	**ra8**
double *	**ra9**
double *	**rp10**

---

uf5075 (view source)

Defined in: uf_curve.h

Overview

uf5075 create conic
Replaced by UF_CURVE_create_conic

Required License(s)

gateway

```
void uf5075
(
    int * ip1,
    double * rp2,
    tag_t * np3,
    tag_t * nr4
)
```

int *	**ip1**
double *	**rp2**
tag_t *	**np3**
tag_t *	**nr4**

---

uf5080 (view source)

Defined in: uf_curve.h

Overview

uf5080 create spline
Replace by UF_CURVE_create_spline_thru_pts

Required License(s)

gateway

void uf5080
(
int * ip1,
int * ip2,
double * rp3,
tag_t * nr4
)

int *	**ip1**
int *	**ip2**
double *	**rp3**
tag_t *	**nr4**

---

uf5081 (view source)

Defined in: uf_curve.h

Overview

uf5081 create spline using complete definition
Replaced by UF_CURVE_create_spline_thru_pts

Required License(s)

gateway

void uf5081
(
int * ip1,
int * ip2,
double * rp3,
double * rp4,
double * rp5,
tag_t * nr6
)

int *	**ip1**
int *	**ip2**
double *	**rp3**
double *	**rp4**
double *	**rp5**
tag_t *	**nr6**

---

uf5082 (view source)

Defined in: uf_curve.h

Overview

uf5082 read spline data
Replaced by UF_CURVE_ask_spline_thru_pts

Required License(s)

gateway

```
void uf5082
(
    tag_t * np1,
    int * ip2,
    double * rr3,
    double * rar4,
    double * rar5
)
```

tag_t *	**np1**
int *	**ip2**
double *	**rr3**
double *	**rar4**
double *	**rar5**

---

uf5083 (view source)

Defined in: uf_curve.h

Overview

uf5083 edit spline data
Replaced by UF_CURVE_edit_spline_thru_pts and
UF_CURVE_edit_move_mult_points

Required License(s)

gateway

```
void uf5083
(
    tag_t * np1,
    int * ip2,
    double * rp3,
    double * ra4,
    double * ra5
)
```

tag_t *	**np1**
int *	**ip2**
double *	**rp3**
double *	**ra4**
double *	**ra5**

---

uf5083x (view source)

Defined in: uf_curve.h

Overview

uf5083x edit spline data without update
No Replacement

Required License(s)

gateway

```
void uf5083x
(
    tag_t * np1,
    int * ip2,
    double * rp3,
    double * ra4,
    double * ra5
)
```

tag_t *	**np1**
int *	**ip2**
double *	**rp3**
double *	**ra4**
double *	**ra5**

---

uf5084 (view source)

Defined in: uf_curve.h

Overview

uf5084 add spline knot point
Replaced by UF_CURVE_ask_spline_thru_pts and UF_CURVE_edit_spline_thru_pts

Required License(s)

gateway

```
void uf5084
(
    tag_t * np1,
    double * rp2,
    double * ra3,
    double * ra4
)
```

tag_t *	**np1**
double *	**rp2**
double *	**ra3**
double *	**ra4**

---

uf5085 (view source)

Defined in: uf_curve.h

Overview

uf5085 count spline knot points
Replaced by UF_CURVE_ask_spine_thru_pts

Required License(s)

(solid_modeling or drafting)

```
int uf5085
(
    tag_t * np1
)
```

tag_t *	**np1**

---

uf5445 (view source)

Defined in: uf_curve.h

Overview

uf5445 create spline
Replaced by UF_MODL_create_spline

Required License(s)

gateway

```
void uf5445
(
```

```
    int * ip1,
    int * ip2,
    int * ip3,
    int * ip4,
    double * rp5,
    double * rp6,
    tag_t * nr7
)
```

int *	**ip1**
int *	**ip2**
int *	**ip3**
int *	**ip4**
double *	**rp5**
double *	**rp6**
tag_t *	**nr7**

---

uf5446 (view source)

Defined in: uf_curve.h

Overview

uf5446 read spline data
No Replacement

Required License(s)

gateway

```
void uf5446
(
    tag_t * np1,
    int * ir2,
    int * ir3,
    int * ir4,
    int * ir5,
    double * rr6,
    double * rr7
)
```

tag_t *	**np1**
int *	**ir2**
int *	**ir3**
int *	**ir4**
int *	**ir5**

double *	**rr6**
double *	**rr7**

---

uf5447 (view source)

Defined in: uf_curve.h

Overview

uf5447 create spline from points on curve or
Replace by UF_CURVE_create_spline_thru_pts and UF_CURVE_create_spline

Required License(s)

gateway

```
void uf5447
(
    int * ip1,
    int * ip2,
    int * ip3,
    int * ip4,
    double * rp5,
    double * rp6,
    tag_t * nr7,
    int * ir8
)
```

int *	**ip1**
int *	**ip2**
int *	**ip3**
int *	**ip4**
double *	**rp5**
double *	**rp6**
tag_t *	**nr7**
int *	**ir8**

---

uf5463 (view source)

Defined in: uf_curve.h

Overview

uf5463 reads the count of poles
No Replacement

Required License(s)
gateway

void uf5463
(
tag_t * np1,
int * ir2,
int * ir3
)

tag_t *	**np1**
int *	**ir2**
int *	**ir3**

---

uf5466 (view source)

Defined in: uf_curve.h

Overview
uf5466 create a spline from a curve
Replaced by UF_CURVE_create_joined_curve

Required License(s)
gateway

void uf5466
(
tag_t * np1,
int * ip2,
tag_t * nr3,
int * ir4
)

tag_t *	**np1**
int *	**ip2**
tag_t *	**nr3**
int *	**ir4**

---

uf5900 (view source)

Defined in: uf_curve.h

Overview

uf5900 evaluate point on a curve
Replaced by UF_CURVE_evaluate_curve or UF_MODL_ask_curve_props

Required License(s)

gateway

```
void uf5900
(
    tag_t * np1,
    double * rp2,
    double * rr3
)
```

tag_t *	**np1**
double *	**rp2**
double *	**rr3**

---

uf5901 (view source)

Defined in: uf_curve.h

Overview

uf5901 evaluate curve point data
Replaced by UF_CURVE_evaluate_curve or UF_MODL_ask_curve_props

Required License(s)

gateway

```
void uf5901
(
    tag_t * np1,
    double * rp2,
    double * rr3
)
```

tag_t *	**np1**
double *	**rp2**
double *	**rr3**

---

uf5903 (view source)

Defined in: uf_curve.h

Overview

uf5903 trim curve by a given arclength
Replaced by UF_CURVE_edit_length

Required License(s)

gateway

```
void uf5903
(
    tag_t * np1,
    int * ip2,
    double * rp3,
    int * ip4,
    double * rp5,
    int * ir6
)
```

tag_t *	**np1**
int *	**ip2**
double *	**rp3**
int *	**ip4**
double *	**rp5**
int *	**ir6**

---

uf5910 (view source)

Defined in: uf_curve.h

Overview

uf5910 evaluate point on a face
Replaced by UF_MODL_ask_face_props

Required License(s)

gateway

```
void uf5910
(
    int * np1,
    double * rp2,
    double * rr3
)
```

int *	**np1**
double *	**rp2**
double *	**rr3**

uf5911 (view source)

Defined in: uf_curve.h

Overview

uf5911 evaluate face point data
Replaced by UF_MODL_ask_face_props

Required License(s)

gateway

```
void uf5911
(
    tag_t * np1,
    double * rp2,
    double * rr3,
    double * rr4,
    double * rr5,
    double * rr6
)
```

tag_t *	**np1**
double *	**rp2**
double *	**rr3**
double *	**rr4**
double *	**rr5**
double *	**rr6**

---

uf5930 (view source)

Defined in: uf_curve.h

Overview

uf5930 intersect two curves
Replaced by UF_CURVE_intersect or UF_MODL_intersect_objects

Required License(s)

gateway

```
void uf5930
(
    tag_t * np1,
    tag_t * np2,
    double * rp3,
    double * rr4,
    double * rr5,
```

```
    double * rr6,
    int * ir7
)
```

tag_t *	**np1**
tag_t *	**np2**
double *	**rp3**
double *	**rr4**
double *	**rr5**
double *	**rr6**
int *	**ir7**

---

uf5931 (view source)

Defined in: uf_curve.h

Overview

uf5931 intersect curve and face/plane
Replaced by UF_CURVE_intersect or UF_MODL_intersect_objects

Required License(s)

gateway

```
void uf5931
(
    tag_t * np1,
    tag_t * np2,
    double * rp3,
    double * rr4,
    double * rr5,
    double * rr6,
    int * ir7
)
```

tag_t *	**np1**
tag_t *	**np2**
double *	**rp3**
double *	**rr4**
double *	**rr5**
double *	**rr6**
int *	**ir7**

uf5932 (view source)

Defined in: uf_curve.h

Overview

uf5932 intersection of two faces
Replaced by UF_MODL_intersect_objects

Required License(s)

gateway

```
void uf5932
(
    tag_t * jp1,
    tag_t * jp2,
    double * rp3,
    tag_t * jp4,
    double * rp5,
    tag_t * jp6,
    double * rp7,
    int * ip8,
    double * rp9,
    int * ip10,
    double * rp11,
    int * ip12,
    int * ir13,
    int * ir14,
    tag_t * jr15
)
```

tag_t *	**jp1**
tag_t *	**jp2**
double *	**rp3**
tag_t *	**jp4**
double *	**rp5**
tag_t *	**jp6**
double *	**rp7**
int *	**ip8**
double *	**rp9**
int *	**ip10**
double *	**rp11**
int *	**ip12**

int *	**ir13**
int *	**ir14**
tag_t *	**jr15**

UF_CURVE_add_faces_ocf_data (view source)

Defined in: uf_curve.h

Overview
Add a face collector to the offset data for the associated offset curve on face feature.
If the offset data already has a face collector associated with it, that existing face collector is ignored and the input face collector is used

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_add_faces_ocf_data
(
    tag_t face_tag,
    UF_CURVE_ocf_data_p_t uf_offset_data
)
```

tag_t	**face_tag**	Input	Identifier of the face collector to add to the defining date
UF_CURVE_ocf_data_p_t	**uf_offset_data**	Input	Pointer to the data defining the offset curve on face feature

---

UF_CURVE_add_string_to_ocf_data (view source)

Defined in: uf_curve.h

Overview
Add a string to an associated offset curve on face feature.
If the string is invalid, an error is returned.

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_add_string_to_ocf_data
(
    tag_t string_tag,
    int offset_direction,
    int num_offsets,
    UF_CURVE_ocf_values_p_t offset_distances,
    UF_CURVE_ocf_data_p_t uf_offset_data
)
```

tag_t	**string_tag**	Input	Identifier of a string
int	**offset_direction**	Input	direction along where the offset will be performed
int	**num_offsets**	Input	Number of offsets to create for the input string
UF_CURVE_ocf_values_p_t	**offset_distances**	Input	Offset values for each offset
UF_CURVE_ocf_data_p_t	**uf_offset_data**	Input / Output	Pointer to the structure containing data to define the offset curve on face operation

---

UF_CURVE_ask_arc_data (view source)

Defined in: uf_curve.h

Overview

Returns the data of an arc. The data is in the structure arc_coords pointed to by UF_CURVE_arc_p_t and includes the matrix tag, start and end angles (radian measure), the arc center coordinates, and the arc radius. Another function that provides arc data is UF_EVAL_ask_arc.

Environment

Internal and External

Required License(s)

gateway

```
int UF_CURVE_ask_arc_data
(
    tag_t arc,
    UF_CURVE_arc_p_t arc_coords
)
```

tag_t	**arc**	Input	Object identifier of arc to query
UF_CURVE_arc_p_t	**arc_coords**	Output	Data of arc in CSYS in which the arc resides.

UF_CURVE_ask_arc_length (view source)

Defined in: uf_curve.h

Overview
Computes the arc length of a curve between two input normalized parameters.
It will also compute the length of a solid edge.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_arc_length
(
    tag_t curve_tag,
    double start_param,
    double end_param,
    UF_MODL_units_t unit_flag,
    double* arc_length
)
```

tag_t	**curve_tag**	Input	Tag of curve to query for arc length
double	**start_param**	Input	Start parameter of the curve from which the arc length is to be calculated. (Normalized parameter)
double	**end_param**	Input	End parameter of the curve to which the arc length is to be calculated. (Normalized parameter)
UF_MODL_units_t	**unit_flag**	Input	Any one of the following enumerated constants: UF_MODL_UNITS_PART - same as parts units UF_MODL_INCH - inches UF_MODL_MMETER - millimeters UF_MODL_CMETER - centimeters UF_MODL_METER - meters
double*	**arc_length**	Output	arc length of the curve from start_param to end_param.

UF_CURVE_ask_bridge_feature (view source)

Defined in: uf_curve.h

Overview
Inquire a bridge curve feature

Environment
Internal and External

See Also
UF_CURVE_create_bridge_feature
UF_CURVE_edit_bridge_feature

History
Originally released in V16.0

Required License(s)
gateway

int UF_CURVE_ask_bridge_feature
(
tag_t bridge_feature,
UF_CURVE_bridge_data_p_t bridge_data
)

tag_t	**bridge_feature**	Input	Object identifier of the bridge curve feature
UF_CURVE_bridge_data_p_t	**bridge_data**	Input / Output	Parameters of bridge curve feature

---

UF_CURVE_ask_centroid (view source)

Defined in: uf_curve.h

Overview
Computes the centroid (center of gravity) of a curve or solid edge.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_CURVE_ask_centroid
(
tag_t curve_id,
double centroid [3]
)

tag_t	**curve_id**	Input	Object identifier of curve or edge
double	**centroid [3]**	Output	Centroid of curve or edge

UF_CURVE_ask_combine_curves (view source)

Defined in: uf_curve.h

Overview

Retrieves the parameters used to create parametric curve(s) by combining two curves along specified direction vectors.

Environment

Internal and External

See Also

Refer to the example
For structure refer
UF_CURVE_combine_curves_direction_t

Required License(s)

gateway

```
int UF_CURVE_ask_combine_curves
(
    tag_t combine_curve_feature,
    tag_t * first_curve_tag,
    UF_CURVE_combine_curves_direction_p_t first_dir,
    tag_t * second_curve_tag,
    UF_CURVE_combine_curves_direction_p_t second_dir,
    char * * tol,
    uf_list_p_t * curve_list
)
```

tag_t	**combine_curve_feature**	Input	Object tag for the combined curve feature.
tag_t *	**first_curve_tag**	Output	First curve tag projected for combine
UF_CURVE_combine_curves_direction_p_t	**first_dir**	Output	Pointer to projection direction info for first curve
tag_t *	**second_curve_tag**	Output	Second curve tag projected for combine
UF_CURVE_combine_curves_direction_p_t	**second_dir**	Output	Pointer to projection direction info for second curve
char * *	**tol**	Output to UF_*free*	String containing the value for curve approximation tolerance (distance tolerance). This parameter must be freed by calling UF_free.

uf_list_p_t *	**curve_list**	Output to UF_*free*	Address of pointer to list of curve tags created by combining two curves. Use UF_MODL_delete_list to free the space allocated for this list.

UF_CURVE_ask_conic_data (view source)

Defined in: uf_curve.h

Overview
Reads the data from the conic arc and returns it in the data structure provided.

Environment
Internal and External

History
Original release was in V13.0

Required License(s)
gateway

```
int UF_CURVE_ask_conic_data
(
    tag_t conic,
    UF_CURVE_conic_t * conic_data
)
```

tag_t	**conic**	Input	tag of conic for which to obtain data
UF_CURVE_conic_t *	**conic_data**	Output	pointer to a conic arc data structure to be filled with the data for the conic arc

---

UF_CURVE_ask_curve_fit_data (view source)

Defined in: uf_curve.h

Overview
Ask curve fit data from the curve feature

Environment
Internal and External

See Also
UF_CURVE_edit_by_curve_fit_data

History

This function is introduced in NX3.0.

Required License(s)
gateway

int UF_CURVE_ask_curve_fit_data
(
tag_t curve_feature,
UF_CURVE_curve_fit_data * curve_fit_data
)

tag_t	**curve_feature**	Input	tag of the curve feature
UF_CURVE_curve_fit_data *	**curve_fit_data**	Output	curve fit method, maximum degree, and maximum segments

---

UF_CURVE_ask_curve_inflections (view source)

Defined in: uf_curve.h

Overview
Calculates inflection points of a curves projection by inputting the curves tag, the projection matrix, and the relative range (in percentage) in which the inflection points are to be found.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_CURVE_ask_curve_inflections
(
tag_t curve_eid,
double proj_matrx [9] ,
double range [2] ,
int * num_infpts,
double * * inf_pts
)

tag_t	**curve_eid**	Input	Object identifier of the curve
double	**proj_matrx [9]**	Input	Projection matrix specified in a 1D array(proj_matrx[9])
double	**range [2]**	Input	Lower and upper limit of the relative range (in %) in which inflection points are searched range[0] = lower limit (umin) range[1] = upper limit (umax)
int *	**num_infpts**	Output	Number of inflection points found

double * *	**inf_pts**	Output to UF_*free*	The inflection points data. The size of the array is inf_pts[num_infpts 4]. (inf_pts+4i)[0] = u value of the ith inflection. [1-3]= x, y, z values of the ith inflections. This array must be freed by calling UF_free.

UF_CURVE_ask_curve_struct (view source)

Defined in: uf_curve.h

Overview

Gets the structure pointer corresponding to the specified curve id.

NOTE: This routine is to be permanently withdrawn in the near future. Use the routines in the UF_EVAL module instead.
Gets the structure pointer corresponding to the specified curve id.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

gateway

```
int UF_CURVE_ask_curve_struct
(
    tag_t curve_id,
    UF_CURVE_struct_p_t * curve_struct
)
```

tag_t	**curve_id**	Input	Object identifier of the curve
UF_CURVE_struct_p_t *	**curve_struct**	Output to UF_*free*	Pointer to pointer of type UF_CURVE_struct_p_t. This routine allocates the structure and fills in the data. This argument must be freed by calling UF_CURVE_free_curve_struct.

UF_CURVE_ask_curve_struct_data (view source)

Defined in: uf_curve.h

Overview

Allows you to get the curve type and data for a specified curve structure.

NOTE: This routine is to be permanently withdrawn in the near future. Use the routines in the UF_EVAL module instead.
Allows you to get the curve type and data for a specified curve structure. You may obtain the curve structure id using

UF_CURVE_ask_curve_struct.

Depending on the object type, curve_data may have the following data:
POINT curve_data[0 - 2]

LINE curve_data[0] = t0
curve_data[1] = tscale
curve_data[2] = periodicity
curve_data[3 - 5] = first point
curve_data[6 - 8] = second point

ARC curve_data[0] = t0
curve_data[1] = tscale
curve_data[2] = periodicity
curve_data[3] = start angle
curve_data[4] = end angle (radians)
curve_data[5] = radius
curve_data[6 - 8] = center
curve_data[9 - 11] = X axis of the
construction plane
curve_data[12 - 14] = Y axis of the
construction plane

CONIC curve_data[0] = t0
curve_data[1] = tscale
curve_data[2] = periodicity
curve_data[3] = T1
curve_data[4] = T2
curve_data[5] = K1
curve_data[6] = K2
curve_data[7] = conic type
curve_data[8 - 10] = center
curve_data[11 - 13] = axis1
curve_data[14 - 16] = axis2

B-CURVE curve_data[0] = t0
curve_data[1] = tscale
curve_data[2] = periodicity
curve_data[3] = number of poles
curve_data[4] = order
curve_data[] = knot sequence
(curve_data[3] +
curve_data[4])
curve_data[] = array of homogeneous
poles (wx, wy, wz, w), for
each pole.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

```
int UF_CURVE_ask_curve_struct_data
(
    UF_CURVE_struct_p_t curve_struct,
    int * type,
    double * * curve_data
)
```

UF_CURVE_struct_p_t	**curve_struct**	Input	Curve structure pointer.
int *	**type**	Output	Object type
double * *	**curve_data**	Output to UF_*free*	Array of object data. This array is allocated by this routine. The caller must free it by calling UF_free.

UF_CURVE_ask_curve_turn_angle (view source)

Defined in: uf_curve.h

Overview
Returns the angle made up by a curve as it wanders through space. The angle is taken on a projection plane defined by the orientation vector.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_curve_turn_angle
(
    tag_t curve,
    double orientation [ 3 ] ,
    double * angle
)
```

tag_t	**curve**	Input	Tag of the curve object.
double	**orientation [3]**	Input	A vector normal to the plane in which the curve should be evaluated.
double *	**angle**	Output	Angle in radians.

UF_CURVE_ask_feature_curves (view source)

Defined in: uf_curve.h

Overview
Returns the output curves of a curve feature.

Environment
Internal and External.

History
This function was originally released in V15.0

Required License(s)
gateway

```
int UF_CURVE_ask_feature_curves
(
    tag_t curve_feature_id,
    int * num_curves,
    tag_t * * feature_curves
)
```

tag_t	**curve_feature_id**	Input	The tag of the curve feature.
int *	**num_curves**	Output	The number of curves in the output array.
tag_t * *	**feature_curves**	Output to UF_*free*	The array of curves. This must be freed by calling UF_free.

UF_CURVE_ask_int_curve_parents (view source)

Defined in: uf_curve.h

Overview

Given an intersection curve feature, returns the Object identifier of the intersection curve object and the object identifiers of the two objects that produced the intersection curve. Returns an error if the intersection curve does not belong to an intersection curve object.

Environment

Internal and External

Required License(s)

gateway

```
int UF_CURVE_ask_int_curve_parents
(
    tag_t int_curve,
    tag_t * int_curve_object,
    tag_t input_objects [ 2 ]
)
```

tag_t	**int_curve**	Input	Feature tag of intersection curve to inquire.
tag_t *	**int_curve_object**	Output	Object identifier of intersection curve object.
tag_t	**input_objects [2]**	Output	Object identifier of the two objects that produced the intersection curve. Can be face, edge, or datum plane.

UF_CURVE_ask_int_curves (view source)

Defined in: uf_curve.h

Overview
Returns an array of intersection curve tags.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_int_curves
(
    tag_t int_curve_object,
    int * num_curves,
    tag_t * * intersection_curves
)
```

tag_t	**int_curve_object**	Input	Feature tag of intersection curve.
int *	**num_curves**	Output	Number of curves in array.
tag_t * *	**intersection_curves**	Output to UF_*free*	Array of intersection curves. This must be freed by calling UF_free.

UF_CURVE_ask_int_parms (view source)

Defined in: uf_curve.h

Overview
Reads the creation parameters of an intersection curve feature.
Returns array of tags to the sets of input intersection objects.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_int_parms
(
    tag_t int_curve_object,
    int * num_objects_set_1,
    tag_t * * object_set_1,
    int * num_objects_set_2,
    tag_t * * object_set_2
)
```

tag_t	**int_curve_object**	Input	Feature tag of intersection curve.
int *	**num_objects_set_1**	Output	Number of intersection objects in object_set_1 array.
tag_t * *	**object_set_1**	Output to UF_*free*	Array of tags of objects intersecting with objects in object_set_2 array. This must be freed by calling

			UF_free.
int *	**num_objects_set_2**	Output	Number of intersection objects in object_set_2 array.
tag_t * *	**object_set_2**	Output to UF_*free*	Array of tags of objects intersecting with objects in object_set_1 array. This must be freed by calling UF_free.

UF_CURVE_ask_int_parms_sc (view source)

Defined in: uf_curve.h

Overview

Reads the creation parameters of an intersection curve feature.
Returns array of tags to the sets of input intersection objects.
This also has the ability to return the tag of collector in case
the objects intersecting are collectors.

Environment

Internal and External

Required License(s)

gateway

```
int UF_CURVE_ask_int_parms_sc
(
    tag_t int_curve_object,
    int * num_objects_set_1,
    tag_t * * object_set_1,
    int * num_objects_set_2,
    tag_t * * object_set_2,
    logical * set1_is_collector,
    logical * set2_is_collector
)
```

tag_t	**int_curve_object**	Input	Feature tag of intersection curve.
int *	**num_objects_set_1**	Output	Number of intersection objects in object_set_1 array.
tag_t * *	**object_set_1**	Output to UF_*free*	Array of tags of objects intersecting with objects in object_set_2 array. This must be freed by calling UF_free.
int *	**num_objects_set_2**	Output	Number of intersection objects in object_set_2 array.
tag_t * *	**object_set_2**	Output to UF_*free*	Array of tags of objects intersecting with objects in object_set_1 array. This must be freed by calling UF_free.
logical *	**set1_is_collector**	Output	TRUE if set1 is a collector.
logical *	**set2_is_collector**	Output	TRUE if set2 is a collector.

UF_CURVE_ask_isocline (view source)

Defined in: uf_curve.h

Overview

Ask isocline curve feature data
Returns the parameters of a joined curve feature.

Environment

Internal and External

See Also

UF_CURVE_edit_isocline

Required License(s)

gateway

```
int UF_CURVE_ask_isocline
(
    tag_t isocline_feat,
    int * face_cnt,
    tag_p_t * faces,
    double direction [ 3 ] ,
    char * * start_angle,
    char * * end_angle,
    char * * step_angle,
    int * curve_cnt,
    tag_p_t * curves
)
```

tag_t	**isocline_feat**	Input	Tag of feature
int *	**face_cnt**	Output	Number of faces
tag_p_t *	**faces**	Output to UF_*free*	Array of faces used to calculate isocline curves. This must be freed by calling UF_free.
double	**direction [3]**	Output	Isocline direction vector
char * *	**start_angle**	Output to UF_*free*	Start angle. This must be freed by calling UF_free.
char * *	**end_angle**	Output to UF_*free*	End angle (NULL for single angle) This must be freed by calling UF_free.
char * *	**step_angle**	Output to UF_*free*	Step angle (NULL for single angle) This must be freed by calling UF_free.
int *	**curve_cnt**	Output	Number of isocline curves
tag_p_t *	**curves**	Output to UF_*free*	Array of isocline curves This must be freed by calling UF_free.

UF_CURVE_ask_joined_parms (view source)

Defined in: uf_curve.h

Overview
Returns the parameters of a joined curve feature.

Environment
Internal and External.

See Also
UF_STRING_p_t AKA string_list

History
This function was originally released in V15.0

Required License(s)
gateway

```
int UF_CURVE_ask_joined_parms
(
    tag_t joined_curve_feature,
    UF_STRING_p_t uf_curve_string,
    int * creation_method,
    double tols [ 3 ]
)
```

tag_t	**joined_curve_feature**	Input	CRV_JOIN feature
UF_STRING_p_t	**uf_curve_string**	Output to UF_*free*	String of input curves. This must be freed by calling UF_MODL_free_string_list.
int *	**creation_method**	Output	1 - Polynomial Cubic, 2 - General Spline
double	**tols [3]**	Output	Tolerances [0] - distance tol [1] - angle tol [2] - string tol

---

UF_CURVE_ask_line_arc_data (view source)

Defined in: uf_curve.h

Overview
Populates the supplied pointer to line_arc_data struct for the given associative line/arc feature.

Return
error code

Environment
Internal and External

See Also

UF_CURVE_create_line_arc
UF_CURVE_edit_line_arc
UF_CURVE_ask_line_arc_output

Required License(s)
gateway

int UF_CURVE_ask_line_arc_data
(
tag_t line_arc_feat_id,
UF_CURVE_line_arc_t * line_arc_data
)

tag_t	**line_arc_feat_id**	Input	Object identifier of associative line/arc feature.
UF_CURVE_line_arc_t *	**line_arc_data**	Output	Pointer to line/arc data structure to be filled up.

UF_CURVE_ask_line_arc_output (view source)

Defined in: uf_curve.h

Overview
Returns the ID of the line/arc created by the associative line/arc feature.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc
UF_CURVE_edit_line_arc
UF_CURVE_ask_line_arc_data

Required License(s)
gateway

int UF_CURVE_ask_line_arc_output
(
tag_t line_arc_feat_id,
tag_t * line_arc_id
)

tag_t	**line_arc_feat_id**	Input	Object identifier of associative line/arc feature.
tag_t *	**line_arc_id**	Output	Id of the line/arc created by the feature.

UF_CURVE_ask_line_data (view source)

Defined in: uf_curve.h

Overview
Returns the coordinates of a line with respect to the absolute coordinate system.

Environment
Internal and External

See Also
UF_CURVE_line_p_t

Required License(s)
gateway

```
int UF_CURVE_ask_line_data
(
    tag_t line,
    UF_CURVE_line_p_t line_coords
)
```

tag_t	**line**	Input	Object identifier of line
UF_CURVE_line_p_t	**line_coords**	Output	Coordinates of line in absolute space

---

UF_CURVE_ask_ocf_data (view source)

Defined in: uf_curve.h

Overview
Returns the offset data for a given associated offset curve on face feature. This data can be modified and then UF_CURVE_edit_ocf_feature invoked to modify a feature.
Memory is allocated when this data is returned. It is the responsibility of the caller to free that memory by calling UF_CURVE_free_ocf_data

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_ocf_data
(
    tag_t feature,
    UF_CURVE_ocf_data_p_t * offset_data
)
```

tag_t	**feature**	Input	Offset curve on face feature identifier

UF_CURVE_ocf_data_p_t *	**offset_data**	Output to UF_*free*	Pointer to the structure containing defining data for the offset curve on face feature

---

UF_CURVE_ask_offset_curves (view source)

Defined in: uf_curve.h

Overview
Returns an array of offset curve object identifiers.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_offset_curves
(
    tag_t offset_curve_object,
    int * num_curves,
    tag_t * * offset_curves
)
```

tag_t	**offset_curve_object**	Input	Object identifier of offset curve object.
int *	**num_curves**	Output	Number of curves in array.
tag_t * *	**offset_curves**	Output to UF_*free*	Array of offset curve object identifiers. This array must be freed by calling UF_free.

UF_CURVE_ask_offset_direction_2 (view source)

Defined in: uf_curve.h

Overview
Returns the offset direction vector and the draft direction vector for a string of curves. These vectors define the positive direction. If you want to offset in the opposite direction, then negate the values for distance, draft height, and/or draft angle. The input string of curves must be coplanar.

Environment
Internal and External

See Also
The following example is an internal Open API program that uses the new offset curve Open API routines. The intention of this Open API program is to give you and idea of how to set up the inputs and use the outputs from these routines. Thus, the data used in this example is very simple and is not representative of the capabilities of this function.

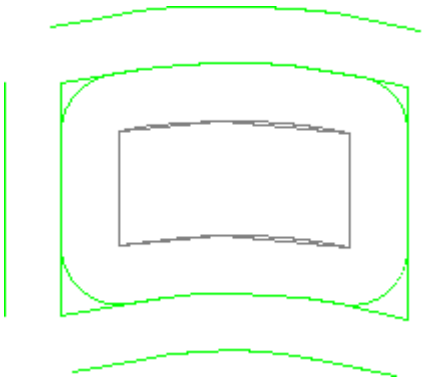
The flow of the code is as follows. First it creates the geometry needed as input. It creates two arcs and then a ruled surface from the arcs. A string is set up for the input curves from the arcs and two of the edges of the ruled surface.

UF_CURVE_ask_offset_direction_2
is called with the string as input to find the offset and draft direction vectors. UF_VEC3_dot is called to compare the offset direction vector to the +X axis to see if they are going in the same direction. If not, the distance string will be negated.

Structures are set up to create a UF_CURVE_OFFSET_DISTANCE_FILLET type offset curve. These curves will be non-associative. Next, UF_CURVE_create_offset_object is called to create a UF_CURVE_OFFSET_DISTANCE_TANGENT type associative offset curve object. Then UF_CURVE_ask_offset_curves
is called to retrieve the output offset curves. These curves will replace the previous input curves in the input curve string.

UF_CURVE_create_offset_object
is called again to create the next set of offset curves. UF_CURVE_ask_offset_parms is called to read the creation parameters of the object just created. Then the offset type is changed to UF_CURVE_OFFSET_DISTANCE_NO_TRIM and UF_CURVE_edit_offset_object
is called to update the last object. Finally, UF_CURVE_free_offset_parms is called to free the memory allocated by UF_CURVE_ask_offset_parms

The following is the output that is generated by the example program.
Input curves are in gray and resultant curves are in green.



Refer to the example code for this program.

Documentation for UF_STRING_p_t and UF_STRING_t can be found in the NX Open Reference Manual under uf_modl -> Types -> string_list.

History

This function was modified in V13.0 to add the base_point parameter.
Original function was UF_CURVE_ask_offset_direction, in NX6.0.2 this function was added to return correct offset direction.

Required License(s)

gateway

```
int UF_CURVE_ask_offset_direction_2
(
    UF_STRING_p_t input_curves,
    double offset_direction_vector [ 3 ] ,
```

```
    double draft_direction_vector [ 3 ] ,
    double base_point [ 3 ]
)
```

UF_STRING_p_t	**input_curves**	Input	Pointer to curves list structure int num Total number of primary strings (min=1,max=1) int string[] Total number of segment curves/ sketch id of each primary string (min=1,max=402) int dir[] Direction of each primary string 1 = Start to End -1 = End to Start tag_t id[] Identifier of primary curves Use UF_MODL_init_string_list and UF_MODL_create_string_list to create input strings. Use UF_MODL_free_string_list to free memory after the string is created.
double	**offset_direction_vector [3]**	Output	positive offset direction
double	**draft_direction_vector [3]**	Output	positive draft height direction
double	**base_point [3]**	Output	base point of vectors

UF_CURVE_ask_offset_parms (view source)

Defined in: uf_curve.h

Overview
Reads the creation parameters of an offset curve object.

Environment
Internal and External

See Also
UF_CURVE_free_offset_parms

Required License(s)
gateway

```
int UF_CURVE_ask_offset_parms
(
    tag_t offset_curve_object,
    UF_CURVE_offset_data_p_t offset_data_pointer
)
```

tag_t	**offset_curve_object**	Input	Object identifier of offset curve object to be interrogated.
UF_CURVE_offset_data_p_t	**offset_data_pointer**	Output to UF_*free*	Pointer to structure containing the defining data of the offset curve. The caller is responsible for allocating a UF_CURVE_offset_data_t

			structure, which this routine will fill with data which must be freed. The caller can free the data allocated inside this structure by calling UF_CURVE_free_offset_parms.

UF_CURVE_ask_parameterization (view source)

Defined in: uf_curve.h

Overview

Returns the parameter range and periodicity for any curve or edge. This function returns the "natural" parameter range. Arcs, ellipses and hyperbolas return the start and end angles.

Environment

Internal and External

History

This function was originally released in V15.0.

Required License(s)

gateway

```
int UF_CURVE_ask_parameterization
(
    tag_t object,
    double param_range [ 2 ] ,
    int * periodicity
)
```

tag_t	**object**	Input	curve or edge for which to return parameterization data
double	**param_range [2]**	Output	array into which to return parameter range of the curve
int *	**periodicity**	Output	One of: UF_CURVE_OPEN_CURVE UF_CURVE_CLOSED_PERIODIC_CURVE UF_CURVE_CLOSED_NON_PERIODIC_CURVE

UF_CURVE_ask_point_data (view source)

Defined in: uf_curve.h

Overview

Returns the coordinates for the specified object identifier of a point.

Environment

Internal and External

Required License(s)
gateway

int UF_CURVE_ask_point_data
(
tag_t point,
double point_coords [3]
)

tag_t	**point**	Input	Object identifier of point to inquire about
double	**point_coords [3]**	Output	Coordinates of point in absolute space

---

UF_CURVE_ask_proj_curve_parents (view source)

Defined in: uf_curve.h

Overview
Given an input projected curve, determines the feature identifier, the target object identifier (sheet_body/face/plane/datum_plane), and the defining curve identifier of the projected curves.

The projected curve must be a curve that was produced by UF_CURVE_create_proj_curves. You can get this curve by calling the function UF_CURVE_ask_proj_curves. An error is returned if the projected curve does not belong to a projected curves feature.
The following example creates a block at origin (0,0,0) with edge lengths of (300,25,150). The program creates and projects a line onto the faces of the block. Next the projected curves are found, and then the face the curves projected onto and the defining curves generating the projected curves are determined.

Environment
Internal and External

See Also
Refer to the example

Required License(s)
gateway

int UF_CURVE_ask_proj_curve_parents
(
tag_t proj_curve,
tag_t * defining_feature,
tag_t * defining_target,
tag_t * defining_curve
)

tag_t	**proj_curve**	Input	The curve identifier whose projected curve feature, defining sheet_body/face/plane/datum_plane, and defining curve are to be found.

tag_t *	**defining_feature**	Output	The projected curves feature identifier that produced the input curve.
tag_t *	**defining_target**	Output	The sheet_body/face/plane/datum_plane identifier of the defining target that the input curve was projected onto.
tag_t *	**defining_curve**	Output	The defining curve identifier of the projected curve feature that generated this projected curve.

UF_CURVE_ask_proj_curves (view source)

Defined in: uf_curve.h

Overview
Returns the curves associated with a projected curve feature.

Environment
Internal and External

See Also
UF_CURVE_create_proj_curves
UF_MODL_create_proj_curves
UF_MODL_ask_proj_curves

Required License(s)
gateway

```
int UF_CURVE_ask_proj_curves
(
    tag_t proj_curve_feature,
    int * n_curve_refs,
    tag_t * * curve_refs
)
```

tag_t	**proj_curve_feature**	Input	Projected curve feature identifier.
int *	**n_curve_refs**	Output	Number of curve identifiers.
tag_t * *	**curve_refs**	Output to UF_*free*	Pointer to array of curve identifiers. Use UF_free to deallocate memory.

UF_CURVE_ask_spline_data (view source)

Defined in: uf_curve.h

Overview
Reads the spline data and puts it in the data structure whose address is provided.

Environment

Internal and External

See Also
UF_CURVE_ask_parameterization

Required License(s)
gateway

int UF_CURVE_ask_spline_data
(
tag_t spline_tag,
UF_CURVE_spline_p_t spline_data
)

tag_t	**spline_tag**	Input	Tag of bspline curve for which to return data
UF_CURVE_spline_p_t	**spline_data**	Output to UF_*free*	The caller must allocate a UF_CURVE_spline_t structure. This routine will fill the structure in. The knots and poles members will be allocated by this routine, and must be freed by the caller using UF_free, e.g. UF_free(spline_data->knots); UF_free(spline_data->poles);

UF_CURVE_ask_spline_feature (view source)

Defined in: uf_curve.h

Overview
Inquire a general spline feature.
Output object id of the referenced smart spline.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_spline_feature
UF_CURVE_edit_spline_feature

History
Originally released in V18.0

Required License(s)
gateway

int UF_CURVE_ask_spline_feature
(
tag_t feature_id,
tag_t * spline
)

tag_t	**feature_id**	Input	object id of the spline feature
tag_t *	**spline**	Output	pointer to object id of the referenced smart spline

---

UF_CURVE_ask_spline_thru_pts (view source)

Defined in: uf_curve.h

Overview

Reads a splines defining data structure. The defining data returns regardless of whether the spline is in or out of synchronization with its defining data. However, if you pass the returned data structure to the routine UF_CURVE_edit_spline_thru_pts, the spline is recreated from the input data. Any previous change to the spline by modifying its pole data will be discarded. If there is no defining data stored with the spline, an error code will be returned.

The length of the array of parameters is num_points for non-periodic curves and (num_points+1) for periodic ones.

Environment

Internal and External

See Also

UF_CURVE_create_spline_thru_pts
UF_CURVE_edit_spline_thru_pts
UF_CURVE_pt_slope_crvatr_t
Refer to the example

History

Original release was in V13.0

Required License(s)

gateway

```
int UF_CURVE_ask_spline_thru_pts
(
    tag_t spline_tag,
    int * degree,
    int * periodicity,
    int * num_points,
    UF_CURVE_pt_slope_crvatr_t * * point_data,
    double * * parameters
)
```

tag_t	**spline_tag**	Input	tag of spline for which to return data
int *	**degree**	Output	degree of the spline
int *	**periodicity**	Output	periodicity of the spline
int *	**num_points**	Output	number of points and parameters in the following arrays

UF_CURVE_pt_slope_crvatr_t * *	**point_data**	Output to UF_*free*	Array of data defining points and slope/curvature control. This array must be freed by passing it to UF_free.
double * *	**parameters**	Output to UF_*free*	Parameters of the points defining the spline. This is a user specified parameterization for the spline definitions points, which needs to be monotonic increasing, but does not need to be normalized, if NULL then default parameterization will be used. This array need to be freed by calling UF_free.

UF_CURVE_ask_trim (view source)

Defined in: uf_curve.h

Overview
Retrieve the current parameters of an associative trim curve feature.

Environment
Internal and External

See Also
Refer to the example

History
Originally released in V16.0

Required License(s)
gateway

```
int UF_CURVE_ask_trim
(
    tag_t trim_feature,
    UF_CURVE_trim_p_t trim_info
)
```

tag_t	**trim_feature**	Input	Trim curve feature whose parameters are to be retrieved
UF_CURVE_trim_p_t	**trim_info**	Output to UF_*free*	Information defining the trim curve features current parameters, the caller should free allocated memory in this structure by calling UF_CURVE_free_trim

---

UF_CURVE_ask_wrap_curve_parents (view source)

Defined in: uf_curve.h

Overview
Reads the creation parameters pertaining to a wrap or unwrap curve.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_wrap_curve_parents
(
    tag_t curve_tag,
    tag_t * defining_face,
    tag_t * defining_plane,
    tag_t * defining_curve,
    tag_t * wrap_curve_object
)
```

tag_t	**curve_tag**	Input	Object Identifier of a wrap or unwrap curve.
tag_t *	**defining_face**	Output	Object Identifier of the Wrap Face of the input curve.
tag_t *	**defining_plane**	Output	Object Identifier of the Wrap Plane of the input curve
tag_t *	**defining_curve**	Output	Object Identifiers of the curve, edge, or face that produced the wrap_curve
tag_t *	**wrap_curve_object**	Output	Object Identifier of the wrap or unwrap feature.

UF_CURVE_ask_wrap_curves (view source)

Defined in: uf_curve.h

Overview
Reads the creation parameters of a wrap or unwrap curves feature.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_wrap_curves
(
    tag_t wrap_curve_object,
    int * num_output_curves,
    tag_t * * output_curves
)
```

tag_t	**wrap_curve_object**	Input	Object Identifier of the wrap or unwrap feature.

int *	**num_output_curves**	Output	Number of tags returned in the output_curves array.
tag_t * *	**output_curves**	Output to UF_*free*	Object Identifiers of curves produced by this wrap/unwrap. You are responsible for freeing the memory allocated for this array by calling UF_free.

UF_CURVE_ask_wrap_parms (view source)

Defined in: uf_curve.h

Overview
Reads the creation parameters of a wrap or unwrap curves feature.

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_ask_wrap_parms
(
    tag_t wrap_curve_object,
    UF_CURVE_wrap_data_p_t wrap_data
)
```

tag_t	**wrap_curve_object**	Input	Object Identifier of the wrap or unwrap feature.
UF_CURVE_wrap_data_p_t	**wrap_data**	Output to UF_*free*	Pointer to a structure to be filled in with the defining information for the input feature. The calling routine must allocate a UF_CURVE_wrap_data_t structure, and pass the pointer into this routine. This routine will fill the structure with allocated data. The calling routine must then call UF_CURVE_free_wrap_parms to free memory allocated to support this structure.

UF_CURVE_auto_join_curves (view source)

Defined in: uf_curve.h

Overview
Sorts through a set of unordered curves, joining curves where appropriate. Curves must meet the following criteria in order to join: The curve must be open. Closed curves such as a circle or ellipse cannot be joined.

Only curves that share a common end point can be joined.
Only valid curves can be joined (line, arc, b-spline, conics).

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_auto_join_curves
(
tag_t * crv_list,
int crv_num,
int join_type,
tag_t * join_list,
int * join_num
)

tag_t *	**crv_list**	Input	Array of curves to be joined
int	**crv_num**	Input	Number of curves in crv_list
int	**join_type**	Input	Join type 1 = Polynomial Cubic 2 = General Spline
tag_t *	**join_list**	Output	List of output curves (joined and not joined). The size of this list cannot exceed crv_num. Therefore you can declare this variable as: tag_t join_list[crv_num];
int *	**join_num**	Output	Number of curves in join_list

UF_CURVE_convert_conic_to_gen (view source)

Defined in: uf_curve.h

Overview
The conic in standard form is converted to its general form. The matrix of the general form is always the same as for the standard form.

Environment
Internal and External

See Also
UF_CURVE_ask_conic_data

History
Original release was in V13.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_convert_conic_to_gen
(
    UF_CURVE_conic_p_t conic_data,
    UF_CURVE_genconic_t * gen_conic_data
)
```

UF_CURVE_conic_p_t	**conic_data**	Input	Conic data in standard form
UF_CURVE_genconic_t *	**gen_conic_data**	Output	Conic data in general form

---

UF_CURVE_convert_conic_to_std (view source)

Defined in: uf_curve.h

Overview

Converts general form conic data to standard form data.

The matrix for each form defines the orientation, or construction space; the conic arc is parallel to its XY plane. The Z coordinate of the center point gives the offset from the XY plane. The matrix for the standard form is always the same as it was for the general form. In the standard form, there is no way to specify the direction of parameterization. The start and end points of the general form can define a parameterization direction opposite that of the standard form. When this is the case the sense argument is returned false.

For an ellipse, the process of extracting the angle of rotation from the general form coefficients does not reliably yield the same rotation angle with which it was created. The result may be a parameterization in standard form between p and 3p. For this reason many workers change the matrix of an ellipse (and possibly parabolas and hyperbolas too) to incorporate the rotation angle and make the conics rotation angle be zero. This produces a reliable parameterization for the general form.

Environment

Internal and External

History

Original release was in V13.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_convert_conic_to_std
(
    UF_CURVE_genconic_p_t gen_conic_data,
    UF_CURVE_conic_t * conic_data,
    logical * sense
)
```

UF_CURVE_genconic_p_t	**gen_conic_data**	Input	Conic data in general form
UF_CURVE_conic_t *	**conic_data**	Output	Conic data in standard form

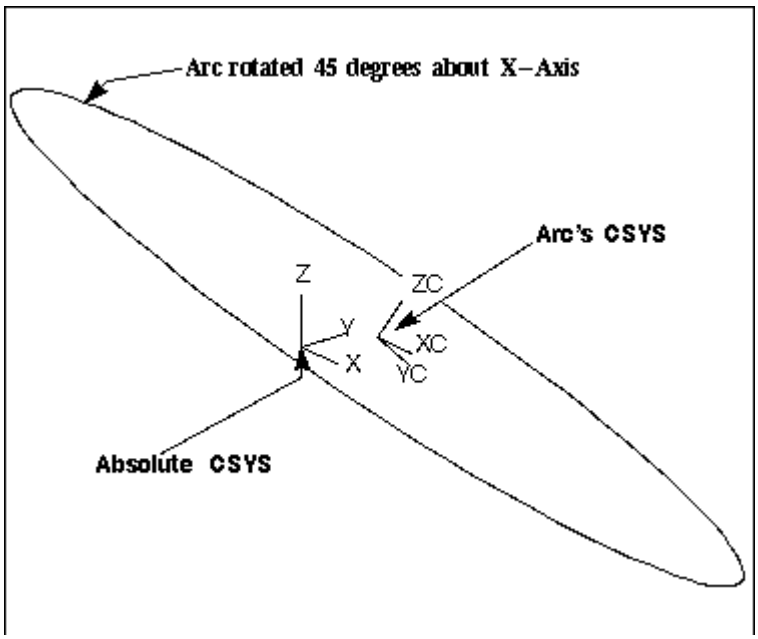
logical *	**sense**	Output	Sense of parameterization of the standard form with respect to the general form. True if they are the same direction.

UF_CURVE_create_arc (view source)

Defined in: uf_curve.h

Overview

Creates an arc. You input the matrix tag, start and end angles in radians, the coordinates of the arc center, and the radius by filling out the arc_coords data structure pointed to by UF_CURVE_arc_p_t. The arc is drawn counterclockwise from the start angle to the end angle as shown below.



This function returns an error if the absolute value of the difference between the start and end angle is greater than two pi (plus the system tolerance). An error is also returned if the start angle is greater than the end angle.

Environment

Internal and External

See Also

UF_CURVE_arc_p_t
Refer to the example

Required License(s)

gateway

```
int UF_CURVE_create_arc
(
    UF_CURVE_arc_p_t arc_coords,
    tag_t * arc
)
```

UF_CURVE_arc_p_t	**arc_coords**	Input	Pointer to arc data structure
tag_t *	**arc**	Output	Object identifier of new arc

UF_CURVE_create_arc_3point (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through three points.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_3point
(
    tag_t point1,
    tag_t point2,
    tag_t point3,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point1**	Input	tag of start point
tag_t	**point2**	Input	tag of end point
tag_t	**point3**	Input	tag of middle point
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_3tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through tangent to three curves.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_3tangent
(
    tag_t tangent_object1,
    tag_t tangent_object2,
    tag_t tangent_object3,
    UF_CURVE_help_data_p_t help_data_p [ 3 ] ,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**tangent_object1**	Input	tag of tangent object at start
tag_t	**tangent_object2**	Input	tag of tangent object at end
tag_t	**tangent_object3**	Input	tag of tangent object at middle
UF_CURVE_help_data_p_t	**help_data_p [3]**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_center_radius (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature of given radius and specific center

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_center_radius
(
    tag_t center,
    double radius,
    tag_t help_point,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**center**	Input	tag of center
double	**radius**	Input	value of radius
tag_t	**help_point**	Input	point to define the start orientation
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_center_tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature tangent to a curve and specific center.

Return
error code

Environment

Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_create_arc_center_tangent
(
tag_t center,
tag_t tangent,
UF_CURVE_help_data_p_t help_data_p,
UF_CURVE_limit_p_t limit_p [2] ,
tag_t support_plane,
logical is_asso,
tag_p_t arc_feature_id
)

tag_t	**center**	Input	tag of center
tag_t	**tangent**	Input	tag of tangent at start
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_center (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through a start point and specific center.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_center
(
    tag_t point,
    tag_t center,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point**	Input	tag of start point
tag_t	**center**	Input	tag of center
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_point_radius (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through two points and of specific radius.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_point_radius
(
    tag_t point1,
    tag_t point2,
    double radius,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point1**	Input	tag of start point
tag_t	**point2**	Input	tag of end point
double	**radius**	Input	value of radius
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_point_tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through two points and tangent to a curve.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_point_tangent
(
    tag_t point1,
    tag_t point2,
    tag_t tangent_object,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point1**	Input	tag of start point
tag_t	**point2**	Input	tag of end point
tag_t	**tangent_object**	Input	tag of tangent object at middle

UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_tangent_point (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through two points and tangent to a curve.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_tangent_point
(
    tag_t point1,
    tag_t tangent_object,
    tag_t point2,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point1**	Input	tag of start point
tag_t	**tangent_object**	Input	tag of tangent object at end
tag_t	**point2**	Input	tag of middle point
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits

tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_tangent_radius (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through one point, tangent to a curve and of specific radius.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_tangent_radius
(
    tag_t point,
    tag_t tangent_object,
    double radius,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point**	Input	tag of start point
tag_t	**tangent_object**	Input	tag of tangent object at end
double	**radius**	Input	value of radius
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative

tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_point_tangent_tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through a point and tangent to two curves.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_point_tangent_tangent
(
    tag_t point,
    tag_t tangent_object1,
    tag_t tangent_object2,
    UF_CURVE_help_data_p_t help_data_p [ 2 ] ,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**point**	Input	tag of start point
tag_t	**tangent_object1**	Input	tag of tangent object at end
tag_t	**tangent_object2**	Input	tag of tangent object at middle
UF_CURVE_help_data_p_t	**help_data_p [2]**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative

tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_tangent_point_point (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through two points and tangent to a curve.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_tangent_point_point
(
    tag_t tangent_object,
    tag_t point1,
    tag_t point2,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**tangent_object**	Input	tag of tangent object at start
tag_t	**point1**	Input	tag of end point
tag_t	**point2**	Input	tag of middle point
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

UF_CURVE_create_arc_tangent_point_tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through a point and tangent to two curves.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_tangent_point_tangent
(
    tag_t tangent_object1,
    tag_t point,
    tag_t tangent_object2,
    UF_CURVE_help_data_p_t help_data_p [ 2 ] ,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**tangent_object1**	Input	tag of tangent object at start
tag_t	**point**	Input	tag of end point
tag_t	**tangent_object2**	Input	tag of tangent object at middle
UF_CURVE_help_data_p_t	**help_data_p [2]**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

UF_CURVE_create_arc_tangent_tangent_point (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature through a point and tangent to two curves.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_tangent_tangent_point
(
    tag_t tangent_object1,
    tag_t tangent_object2,
    tag_t point,
    UF_CURVE_help_data_p_t help_data_p [ 2 ] ,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**tangent_object1**	Input	tag of tangent object at start
tag_t	**tangent_object2**	Input	tag of tangent object at end
tag_t	**point**	Input	tag of middle point
UF_CURVE_help_data_p_t	**help_data_p [2]**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_tangent_tangent_radius (view source)

Defined in: uf_curve.h

Overview
Create an associative arc feature tangent to two curves and of specific radius.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_arc_tangent_tangent_radius
(
    tag_t tangent_object1,
    tag_t tangent_object2,
    double radius,
    UF_CURVE_help_data_p_t help_data_p [ 2 ] ,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t arc_feature_id
)
```

tag_t	**tangent_object1**	Input	tag of tangent object at start
tag_t	**tangent_object2**	Input	tag of tangent object at end
double	**radius**	Input	value of radius
UF_CURVE_help_data_p_t	**help_data_p [2]**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the arc
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**arc_feature_id**	Output	if is_asso == TRUE - object identifier of new associative arc feature if is_asso == FALSE - object identifier of new associative arc

---

UF_CURVE_create_arc_thru_3pts (view source)

Defined in: uf_curve.h

Overview
Create an arc or a circle passing through the user specified input points.

Environment

Internal and External

History
Originally released in V16.0

Required License(s)
gateway

int UF_CURVE_create_arc_thru_3pts
(
int create_flag,
double first_point [3] ,
double second_point [3] ,
double third_point [3] ,
tag_t * arc_tag
)

int	**create_flag**	Input	flag indicating if an arc or a circle will be created = 1 -> arc will be created = 2 -> circle will be created
double	**first_point [3]**	Input	Coordinates of the first point the arc will go through.
double	**second_point [3]**	Input	Coordinates of the second point the arc will go through.
double	**third_point [3]**	Input	Coordinates of the third point the arc will go through.
tag_t *	**arc_tag**	Output	Identifier of the arc or circle that is created

---

UF_CURVE_create_bridge_curve (view source)

Defined in: uf_curve.h

Overview
Bridges two curves/edges by matching tangents or matching curvatures. You can control the connection of the bridge curve along the two curves/edges by specifying the parameter along each curve/edge. You have the ability to reverse the tangent direction vector at the parameter along the curves/edges. The routine returns the object identifier of the new bridge curve.

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_create_bridge_curve
(
int bridge_method,
tag_t curve_ids [2] ,
double parms [2] ,

```
    int reverse_tangent [ 2 ] ,
    tag_t * bridge_id
)
```

int	**bridge_method**	Input	UF_CURVE_TANGENT - match tangents UF_CURVE_CURVATURE - match curvatures
tag_t	**curve_ids [2]**	Input	Object identifiers of the two curves/edges to bridge.
double	**parms [2]**	Input	Parameters on curves/edges to connect bridge.
int	**reverse_tangent [2]**	Input	Indicates whether to reverse the tangent direction vector at the chosen parameter on the curve/edge. 0 = do not reverse not 0 means to reverse
tag_t *	**bridge_id**	Output	Object identifier of the bridge curve.

UF_CURVE_create_bridge_feature (view source)

Defined in: uf_curve.h

Overview

Creates a new bridge curve feature by bridging 2 input curves.
You can control the connection of the bridge curve along the two curves by specifying the parameter along each curve. You have the ability to reverse the tangent direction vector at the parameter along the curves.
You can control the shape of the curve by specifing a reference curve or by indicating shape control values.

The routine returns the object identifier of the new bridge curve feature.
The new bridge curve is associative to the input curves and any expressions used to determine shape or end point location.

Environment

Internal and External

See Also

UF_CURVE_edit_bridge_feature
UF_CURVE_ask_bridge_feature

History

Originally released in V16.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_bridge_feature
(
    UF_CURVE_bridge_data_p_t bridge_data,
    tag_p_t bridge_feature
)
```

UF_CURVE_bridge_data_p_t	**bridge_data**	Input	Input parameters

tag_p_t	**bridge_feature**	Output	Object identifier of the bridge curve feature.

UF_CURVE_create_combine_curves (view source)

Defined in: uf_curve.h

Overview

Creates parametric curve(s) by combining two curves along specified direction vectors. When "normal to plane of curve" is specified as the projection type, the curve must be planar. An error will be given if the above is not true.

Environment

Internal and External

See Also

UF_CURVE_combine_curves_direction_t
Refer to the example

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_combine_curves
(
    tag_t first_curve_tag,
    UF_CURVE_combine_curves_direction_p_t first_dir,
    tag_t second_curve_tag,
    UF_CURVE_combine_curves_direction_p_t second_dir,
    const char * curve_aprox_tol,
    tag_t * combine_curve_feature
)
```

tag_t	**first_curve_tag**	Input	First curve tag to be projected for combine
UF_CURVE_combine_curves_direction_p_t	**first_dir**	Input	Pointer to projection direction info for first curve
tag_t	**second_curve_tag**	Input	Second curve tag to be projected for combine
UF_CURVE_combine_curves_direction_p_t	**second_dir**	Input	Pointer to projection direction info for second curve
const char *	**curve_aprox_tol**	Input	String containing the value for the curve approximation tolerance (distance tolerance).
tag_t *	**combine_curve_feature**	Output	Object tag for the combined curve feature.

UF_CURVE_create_conic (view source)

Defined in: uf_curve.h

Overview

Creates a conic curve. See the description of the data structure for details on the interpretation of the data items.

Environment

Internal and External

History

Original release was in V13.0.

Required License(s)

gateway

```
int UF_CURVE_create_conic
(
    UF_CURVE_conic_p_t conic_data,
    tag_t * conic
)
```

UF_CURVE_conic_p_t	**conic_data**	Input	Data structure defining the conic to create
tag_t *	**conic**	Output	Returned tag of curve

UF_CURVE_create_fillet (view source)

Defined in: uf_curve.h

Overview

Creates a fillet between the specified curves. The curves can be any combination of points, lines, arcs, conics, or splines.

A two curve fillet is an arc generated in the counterclockwise direction from the first curve to the second. The fillet formed is tangent to both curves. The counterclockwise direction is determined based on the current views orientation, not the absolute or WCS coordinate systems.

NOTE: The ability to specify whether an input curve is to be trimmed is provided. If a "spline-type" curve is flagged to be trimmed, the defining points and associated dimensions will be deleted. If a trimmed curve has a length equal to zero and there is no associative connection to the curve, the curve will be deleted.

NOTE: The center point coordinates (for 2-curve fillet) will be projected along the Z-axis of the WCS, to the construction plane of the fillet if necessary.

There is no 3 curve fillet so the documentation for this option has been removed.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

(solid_modeling or drafting)

int UF_CURVE_create_fillet
(
int type,
tag_t curve_objs [3] ,
double center [3] ,
double radius,
int trim_opts [3] ,
int arc_opts [3] ,
tag_t * fillet_obj
)

int	**type**	Input	The type of fillet that is to be created: UF_CURVE_2_CURVE = create 2 curve fillet
tag_t	**curve_objs [3]**	Input	Object identifiers of the curves between which the fillet is to be created. [0] = identifier of first curve [1] = identifier of second curve [2] = not used in a 2 curve fillet
double	**center [3]**	Input	Approximate fillet center expressed as absolute coordinates
double	**radius**	Input	Radius of the fillet
int	**trim_opts [3]**	Input	Trimming options FOR 2 CURVE FILLET: [0] = TRUE -> trim first curve FALSE -> do not trim first curve [1] = TRUE -> trim second curve FALSE -> do not trim second curve
int	**arc_opts [3]**	Input	Not used
tag_t *	**fillet_obj**	Output	The object identifier associated to the newly created fillet NULL_TAG = fillet unable to be created

---

UF_CURVE_create_int_object (view source)

Defined in: uf_curve.h

Overview

Intersects two sets of objects. Objects to intersect must be 1 body, 1 datum plane, 1 face collector or multiple faces (from same body) per set. Datum planes cannot be input for both sets since the result is a non-associative line. If you wish to intersect two datum planes you

can use UF_MODL_intersect_objects.

Environment
Internal and External

See Also
UF_MODL_intersect_objects

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_create_int_object
(
int num_objects_set_1,
tag_t * object_set_1,
int num_objects_set_2,
tag_t * object_set_2,
tag_t * int_curve_object
)

int	**num_objects_set_1**	Input	Number of intersection objects in object_set_1 array.
tag_t *	**object_set_1**	Input	Array of tags of objects to intersect with objects in object_set_2 array.
int	**num_objects_set_2**	Input	Number of intersection objects in object_set_2 array.
tag_t *	**object_set_2**	Input	Array of tags of objects to intersect with objects in object_set_1 array.
tag_t *	**int_curve_object**	Output	Feature tag of intersection curve

---

UF_CURVE_create_isocline (view source)

Defined in: uf_curve.h

Overview
Creates a new joined curve feature that consists of a new spline curve that represents the input list of joined curves and edges. The curves can be sketch curves. The curves or edges that you join may not have gaps.

The Modeling distance and angle tolerance are used to create and validate the resulting spline curve. The modeling tolerance can be modified by UF_MODL_set_distance_tolerance or UF_MODL_set_angle_tolerance.

Environment
Internal and External

See Also
Refer to the example
UF_MODL_create_isocline_curves
UF_CURVE_ask_isocline
UF_CURVE_edit_isocline

Required License(s)

gateway

```
int UF_CURVE_create_isocline
(
    int face_cnt,
    tag_t faces [ ] ,
    double direction [ 3 ] ,
    const char * start_angle,
    const char * end_angle,
    const char * step_angle,
    tag_t * isocline_feat
)
```

int	**face_cnt**	Input	Number of faces
tag_t	**faces [ ]**	Input	Array of faces used to calculate isocline curves
double	**direction [3]**	Input	Isocline direction vector
const char *	**start_angle**	Input	Start angle (-90 to 90 degrees)
const char *	**end_angle**	Input	End angle (-90 to 90 degrees)
const char *	**step_angle**	Input	Step angle (NULL for single angle)
tag_t *	**isocline_feat**	Output	Tag of new feature

UF_CURVE_create_joined_curve (view source)

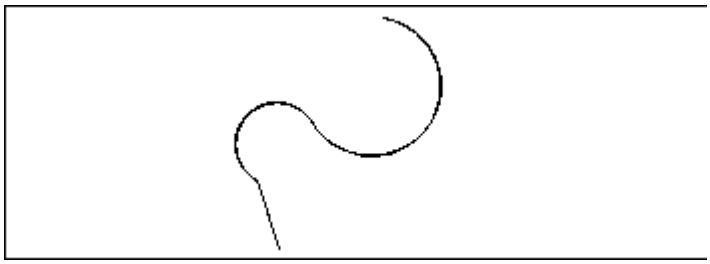
Defined in: uf_curve.h

Overview

Creates a new spline curve that represents the input list of joined curves and edges. The curves or edges that you join may not have gaps.

The Modeling distance and angle tolerance are used to create and validate the resulting spline curve. The modeling tolerance can be modified by UF_MODL_set_distance_tolerance or UF_MODL_set_angle_tolerance.

The curves you join must be input in order: either clockwise or counterclockwise.



Environment

Internal and External

See Also

Refer to the example

History
Creation method has the advanced option in NX3.0.

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_joined_curve
(
    uf_list_p_t uf_curve_list,
    int creation_method,
    tag_t * joined_curve,
    int * status
)
```

uf_list_p_t	**uf_curve_list**	Input	List of curves or edges to be joined
int	**creation_method**	Input	The method to join the list of curves: 1 = Polynomial Cubic 2 = General Spline 3 = polynomial quintic 4 = advanced
tag_t *	**joined_curve**	Output	The resulting joined curve spline
int *	**status**	Output	A flag indicating if the joined curve returned has corners: 0 = Curve has no corners 1 = Curve has corners

---

UF_CURVE_create_joined_feature (view source)

Defined in: uf_curve.h

Overview
Creates a new joined curve feature that consists of a new spline curve that represents the input list of joined curves and edges. The curves can be sketch curves. The curves or edges that you join may not have gaps.

The Modeling distance and angle tolerance are used to create and validate the resulting spline curve. The modeling tolerance can be modified by UF_MODL_set_distance_tolerance or UF_MODL_set_angle_tolerance.

Environment
Internal and External.

See Also
UF_MODL_set_distance_tolerance
UF_MODL_set_angle_tolerance
UF_STRING_p_t AKA string_list
UF_MODL_init_string_list
UF_MODL_create_string_list
UF_MODL_free_string_list

History

This function was originally released in V15.0.
Creation method has the advanced option in NX3.0.

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_joined_feature
(
    UF_STRING_p_t uf_curve_string,
    int creation_method,
    tag_t * joined_curve_feature,
    int * status
)
```

UF_STRING_p_t	**uf_curve_string**	Input	string of input curves. Use UF_MODL_init_string_list and UF_MODL_create_string_list to create input strings. Use UF_MODL_free_string_list to free memory after the string is created.
int	**creation_method**	Input	1 - Polynomial Cubic, 2 - General Spline 3 = polynomial quintic 4 = advanced
tag_t *	**joined_curve_feature**	Output	CRV_JOIN feature
int *	**status**	Output	0 = Curve has no corners 1 = Curve has corners

---

UF_CURVE_create_line (view source)

Defined in: uf_curve.h

Overview
Creates a line. You input the start and end points of the line by filling out the line_coords data structure pointed to by UF_CURVE_line_p_t.

Environment
Internal and External

See Also
UF_CURVE_line_p_t

Required License(s)
gateway

```
int UF_CURVE_create_line
(
    UF_CURVE_line_p_t line_coords,
    tag_t * line
)
```

UF_CURVE_line_p_t	**line_coords**	Input	Coordinates of line in absolute space
tag_t *	**line**	Output	Object identifier Of new line

UF_CURVE_create_line_arc (view source)

Defined in: uf_curve.h

Overview
Create an associative line/arc feature. You input a UF_CURVE_line_arc_t data structure, which specified the data needed for the line/arc to be created. Output is the object identifier of the associative line/arc feature. An error is returned if line/arc cannot be created with the specified data.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_edit_line_arc
UF_CURVE_ask_line_arc_data
UF_CURVE_ask_line_arc_output

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_line_arc
(
    UF_CURVE_line_arc_t * line_arc_data,
    tag_t * line_arc_feat_id
)
```

UF_CURVE_line_arc_t *	**line_arc_data**	Input	Pointer to line/arc data structure
tag_t *	**line_arc_feat_id**	Output	Object identifier of new associative line/arc feature. if is_asso == TRUE - object identifier of new associative line/arc feature. if is_asso == FALSE - object identifier of new line/arc

---

UF_CURVE_create_line_point_angle (view source)

Defined in: uf_curve.h

Overview
Create an associative line feature through a point and at an angle to a linear curve/edge.

Return

error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_create_line_point_angle
(
tag_t point,
tag_t angle_object,
double angle_value,
UF_CURVE_limit_p_t limit_p [2] ,
tag_t support_plane,
logical is_asso,
tag_p_t line_feature_id
)

tag_t	**point**	Input	tag of start point
tag_t	**angle_object**	Input	tag of angle object
double	**angle_value**	Input	value of angle in degree
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the line
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**line_feature_id**	Output	if is_asso == TRUE - object identifier of new associative line feature if is_asso == FALSE - object identifier of new associative line

---

UF_CURVE_create_line_point_point (view source)

Defined in: uf_curve.h

Overview
Create an associative line feature through two points.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_line_point_point
(
    tag_t point1,
    tag_t point2,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t line_feature_id
)
```

tag_t	**point1**	Input	tag of start point
tag_t	**point2**	Input	tag of end point
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the line
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**line_feature_id**	Output	if is_asso == TRUE - object identifier of new associative line feature if is_asso == FALSE - object identifier of new associative line

---

UF_CURVE_create_line_point_principal_axis (view source)

Defined in: uf_curve.h

Overview
Create an associative line feature through a point and along any one principal axis.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_line_point_principal_axis
(
    tag_t point,
    UF_CURVE_principal_axis_type_t principal_axis,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t line_feature_id
```

```
)
```

tag_t	**point**	Input	tag of start point
UF_CURVE_principal_axis_type_t	**principal_axis**	Input	one of the 3 principal axes : UF_CURVE_X_AXIS, UF_CURVE_Y_AXIS, UF_CURVE_Z_AXIS
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the line
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**line_feature_id**	Output	if is_asso == TRUE - object identifier of new associative line feature if is_asso == FALSE - object identifier of new associative line

---

UF_CURVE_create_line_point_tangent (view source)

Defined in: uf_curve.h

Overview
Create an associative line feature through point and tangent to curve.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_line_point_tangent
(
    tag_t point,
    tag_t tangent,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t line_feature_id
)
```

tag_t	**point**	Input	tag of start point
tag_t	**tangent**	Input	tag of tangent object
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent

UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the line
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**line_feature_id**	Output	if is_asso == TRUE - object identifier of new associative line feature if is_asso == FALSE - object identifier of new associative line

---

UF_CURVE_create_line_tangent_point (view source)

Defined in: uf_curve.h

Overview
Create an associative line feature tangent to a curve and through a point.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_line_tangent_point
(
    tag_t tangent,
    tag_t point,
    UF_CURVE_help_data_p_t help_data_p,
    UF_CURVE_limit_p_t limit_p [ 2 ] ,
    tag_t support_plane,
    logical is_asso,
    tag_p_t line_feature_id
)
```

tag_t	**tangent**	Input	tag of tangent object
tag_t	**point**	Input	tag of end point
UF_CURVE_help_data_p_t	**help_data_p**	Input	help data for tangent
UF_CURVE_limit_p_t	**limit_p [2]**	Input	extension limits
tag_t	**support_plane**	Input	tag of support plane of the line
logical	**is_asso**	Input	true - if associative, false - if not associative
tag_p_t	**line_feature_id**	Output	if is_asso == TRUE - object identifier of new associative line feature

if is_asso == FALSE - object identifier of new associative line

---

UF_CURVE_create_ocf_feature (view source)

Defined in: uf_curve.h

Overview

Create an associated offset curves on face feature using the input offset data.
A UF_CURVE_ocf_data_p_t is allocated and created. A feature tag along with an error code is returned.

Environment

Internal and External

See Also

Refer to the example
For examples using different selection intent rules to create section and face collectors view:
Refer to the example
Refer to the example

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_ocf_feature
(
    UF_CURVE_ocf_data_p_t offset_data,
    tag_t* feature
)
```

UF_CURVE_ocf_data_p_t	**offset_data**	Input	Pointer to the structure containing defining data for the offset curve on face operation The offset_data->string_data->string_tag must be created via UF_MODL_create_section. The valid string is a section created using a connected set of edges or curves. If the curves are used to create section, then they should lie on set of faces used to create the smart face container. The offset_data->face_data->face_tag must be created via UF_MODL_create_smart_face_container. The faces used to create smart container should be a connected set and come from a single body.
tag_t*	**feature**	Output	offset curve on face feature identifier

---

UF_CURVE_create_offset_curve (view source)

Defined in: uf_curve.h

Overview
Offsets a string of curves (non-associative).

Environment
Internal and External

See Also
UF_CURVE_offset_data_t

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_offset_curve
(
    UF_CURVE_offset_data_p_t offset_data_pointer,
    int * num_offset_curves,
    tag_t * * offset_curve_tags
)
```

UF_CURVE_offset_data_p_t	**offset_data_pointer**	Input	Pointer to structure containing the defining data of the offset curve.
int *	**num_offset_curves**	Output	Number of offset curves created
tag_t * *	**offset_curve_tags**	Output to UF_*free*	Object identifiers of offset curves created. Use UF_free to deallocate memory when done.

UF_CURVE_create_offset_object (view source)

Defined in: uf_curve.h

Overview
Offsets a string of curves. Returns a tag to the object which associates input curves and defining data to the output offset curves. Use UF_CURVE_ask_offset_curves to retrieve the offset curve tags.

Only the following types of offsets are allowed for offset_data_pointer:
UF_CURVE_OFFSET_DISTANCE_NO_TRIM
UF_CURVE_OFFSET_DISTANCE_TANGENT
UF_CURVE_OFFSET_DISTANCE_FILLET
UF_CURVE_OFFSET_DRAFT_NO_TRIM
UF_CURVE_OFFSET_DRAFT_TANGENT
UF_CURVE_OFFSET_DRAFT_FILLET
UF_CURVE_OFFSET_3D_AXIAL

Environment
Internal and External

Required License(s)

(solid_modeling or drafting)

int UF_CURVE_create_offset_object
(
UF_CURVE_offset_data_p_t offset_data_pointer,
tag_t * offset_curve_object
)

UF_CURVE_offset_data_p_t	**offset_data_pointer**	Input	Pointer to structure containing the defining data of the offset curve.
tag_t *	**offset_curve_object**	Output	Object identifier of offset curve object created.

UF_CURVE_create_point (view source)

Defined in: uf_curve.h

Overview
Creates a point in the absolute coordinate system.

Environment
Internal and External

Required License(s)
gateway

int UF_CURVE_create_point
(
double point_coords [3] ,
tag_t * point
)

double	**point_coords [3]**	Input	Coordinates of point in absolute space
tag_t *	**point**	Output	Object identifier of point

UF_CURVE_create_proj_curves (view source)

Defined in: uf_curve.h

Overview
Creates projection curves.

Objects to project may be points, curves, sketch feature identifiers, or other projected curve feature identifiers. The "number of curves and points" is the number of such object tags (and does not include the numbers of curves in any sketch or projection curve feature).

For copy_flag=1 or 2, a group identifier is returned. You can perform operations on the entire group or extract individual curves from the group and operate on them.

CAUTION: Because multiple projection curves can be returned for single defining curves, no attempt is made to assure that the order of the output curves in the group corresponds to the order of the input curves. If the order is important, for copy_flag = 1 or 2, the curves should be projected one at a time.

NOTE: The Along Vector option (proj_type = 3) projects the selected objects along a specified vector. You can project the curves in the direction indicated by the vector by setting the multiplicity to 1 (single), or in both directions by setting the multiplicity to 2 (both). The Along Vector option produces all possible images on the face.

CAUTION: For copy_flag = 2 (move), new curves/points are created and the original curves/points are deleted.
For a projected curve feature created using copy_flag=3, the feature can be deleted using UF_MODL_delete_feature. Individual projection curves of the feature can not be deleted.

UF_MODL_move_feature can be used to move the projected curve feature.

Points can not be projected using "angle and vector" (proj_type = 4) or "equal arc length" (proj_type =6).

NOTE: You can change the tolerance used to create a curve projection by using the subroutine UF_MODL_set_distance_tolerance. The projection along face normals or along a vector produces an exact projection when projecting onto a plane.

Environment

Internal and External

See Also

UF_MODL_create_proj_curves
UF_MODL_ask_proj_curves
UF_CURVE_ask_proj_curves

History

In V13.0, the proj_data argument was modified to add projection type 5 (toward a line), 6 (equal arc length), multiplicity, arcl_option, and the x_vector.

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_proj_curves
(
    int n_curve_refs,
    tag_t * curve_refs,
    int n_face_refs,
    tag_t * face_refs,
    int copy_flag,
    UF_CURVE_proj_p_t proj_data,
    tag_t * proj_curve_feature
)
```

int	**n_curve_refs**	Input	Number of curves and points

tag_t *	**curve_refs**	Input	Array of curve and point identifiers.
int	**n_face_refs**	Input	Number of sheet bodies, faces and planes
tag_t *	**face_refs**	Input	Array of plane, datum plane, face or sheet body identifiers.
int	**copy_flag**	Input	Copy flag: 1 = copy 2 = move 3= associate
UF_CURVE_proj_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj_s for entries.
tag_t *	**proj_curve_feature**	Output	Group identifier for copy_flag = 1 or 2, Projected curve feature identifier for copy_flag=3

UF_CURVE_create_proj_curves1 (view source)

Defined in: uf_curve.h

Overview

Creates projection curves.

Objects to project may be points, curves, sketch feature identifiers, or other projected curve feature identifiers. The "number of curves and points" is the number of such object tags (and does not include the numbers of curves in any sketch or projection curve feature).

For copy_flag=1 or 2, a group identifier is returned. You can perform operations on the entire group or extract individual curves from the group and operate on them.

CAUTION: Because multiple projection curves can be returned for single defining curves, no attempt is made to assure that the order of the output curves in the group corresponds to the order of the input curves. If the order is important, for copy_flag = 1 or 2, the curves should be projected one at a time.

NOTE: The Along Vector option (proj_type = 3) projects the selected objects along a specified vector. You can project the curves in the direction indicated by the vector by setting the multiplicity to 1 (single), or in both directions by setting the multiplicity to 2 (both). The Along Vector option produces all possible images on the face.

CAUTION: For copy_flag = 2 (move), new curves/points are created and the original curves/points are deleted.
For a projected curve feature created using copy_flag=3, the feature can be deleted using UF_MODL_delete_feature. Individual projection curves of the feature can not be deleted.

UF_MODL_move_feature can be used to move the projected curve feature.

Points can not be projected using "angle and vector" (proj_type = 4) or "equal arc length" (proj_type =6).

NOTE: You can change the tolerance used to create a curve projection by using the subroutine UF_MODL_set_distance_tolerance. The projection along face normals or along a vector produces an exact projection when projecting onto a plane.

Environment
Internal and External

See Also
UF_MODL_create_proj_curves
UF_MODL_ask_proj_curves
UF_CURVE_ask_proj_curves

History
Originally released in NX4.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_proj_curves1
(
    int n_curve_refs,
    tag_t * curve_refs,
    int n_face_refs,
    tag_t * face_refs,
    int copy_flag,
    UF_CURVE_proj1_p_t proj_data,
    tag_t * proj_curve_feature
)
```

int	**n_curve_refs**	Input	Number of curves and points
tag_t *	**curve_refs**	Input	Array of curve and point identifiers.
int	**n_face_refs**	Input	Number of sheet bodies, faces and planes
tag_t *	**face_refs**	Input	Array of plane, datum plane, face or sheet body identifiers.
int	**copy_flag**	Input	Copy flag: 1 = copy 2 = move 3= associate
UF_CURVE_proj1_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj1_s for entries.
tag_t *	**proj_curve_feature**	Output	Group identifier for copy_flag = 1 or 2, Projected curve feature identifier for copy_flag=3

UF_CURVE_create_shadow_curves (view source)

Defined in: uf_curve.h

Overview
Function: UF_CURVE_create_shadow_curves
Purpose: Create curves for the shadow of a group of bodies in a view

Environment
Internal and External
History Released in NX 5.0.5

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_shadow_curves
(
    int solid_count,
    tag_t * solid_array,
    tag_t view_tag,
    int * shadow_curve_count,
    tag_t * * shadow_curves
)
```

int	**solid_count**	Input	Number of solid bodies to shadow
tag_t *	**solid_array**	Input	Array of solid_body tags
tag_t	**view_tag**	Input	Tag of view for shadow
int *	**shadow_curve_count**	Output	Number of curves created
tag_t * *	**shadow_curves**	Output to UF_*free*	Array of curve tags. This must be freed by the caller.

UF_CURVE_create_shadow_outline (view source)

Defined in: uf_curve.h

Overview
Create shadow outline for a given array of solids. Solids passed to this routine must be on a selectable layer and visible. If created shadow curves can not form loops, please use UF_CURVE_create_shadow_curves instead to get shadow curves.

Environment
Internal and External
History Released in V15.0.3

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_shadow_outline
(
    int solid_count,
    tag_t * solid_array,
    tag_t view,
    int * loop_count,
```

```
    int * * count_array,
    tag_t * * * curve_array,
    double tol [ 2 ]
)
```

int	**solid_count**	Input	Number of solids to shadow
tag_t *	**solid_array**	Input	Array of Solids to Shadow
tag_t	**view**	Input	View tag to project to
int *	**loop_count**	Output	Number of shadow loops obtained
int * *	**count_array**	Output to UF_*free*	Number of curves in each loop - freed by user
tag_t * * *	**curve_array**	Output to UF_*free*	Array of curve loops. Each element of this array is a pointer to an array of tags which make up one of the shadow loops. So curve_array[0] is a pointer to a tag_t , which contains count_array[0] tags that make up the first shadow loop. This must be freed by the caller using prior freeing the entire array. Thus the caller must free the sequence of curves starting with curve_array[0] , curve_array[1] etc.
double	**tol [2]**	Input	tol[0] - the tolerance for loops tol[1] - tol for angle projection

---

UF_CURVE_create_silhouette (view source)

Defined in: uf_curve.h

Overview

Returns an array of line and arc identifiers which approximates curves and edges. The tolerance parameter determines the maximum distance between a curve or edge and its approximated arc segments.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_silhouette
(
    tag_t solid,
    tag_t view,
    int* count,
    tag_t* * curves
)
```

tag_t	**solid**	Input	Solid sheet or body to get silhouette curves from

tag_t	**view**	Input	View to generate curves in
int*	**count**	Output	Count of curves created and returned in the curves array.
tag_t* *	**curves**	Output to UF_*free*	Array of curves. You are responsible for freeing the memory allocated for this array. Use UF_free to deallocate memory when done.

UF_CURVE_create_simplified_curve (view source)

Defined in: uf_curve.h

Overview

Returns an array of line and arc identifiers which approximates curves and edges. The tolerance parameter determines the maximum distance between a curve or edge and its approximated arc segments. The segments returned are in random order and may not connect end to end in the order they are returned.

Environment

Internal and External

History

This function was originally released in V15.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_simplified_curve
(
    int curves_count,
    tag_t * curves,
    double tolerance,
    int * segments_count,
    tag_t * * segments
)
```

int	**curves_count**	Input	Count of curves and edges
tag_t *	**curves**	Input	Array of curve and edge identifiers
double	**tolerance**	Input	Maximum distance from arc segments to curve
int *	**segments_count**	Output	Pointer to count of line and arc identifiers
tag_t * *	**segments**	Output to UF_*free*	Pointer to the returned array of line and arc identifiers. This must be freed by calling UF_free.

UF_CURVE_create_spline (view source)

Defined in: uf_curve.h

Overview

A spline curve is created from the data in the spline_data structure.

The spline is a NURBS (NonUniform Rational B-Spline) curve. See the description of the structure for a basic description of the data items.

If the is_rational field is nonzero, the weights must be positive. If it is 0 (non-rational), the weights are ignored.

If the is_spcurve field is nonzero, the Z coordinates are ignored.

NURBS curves need not have Bezier end conditions (be clamped).

NX currently requires that the parameter range of a spline be 0 to 1. The User Function API normalizes the knot vector and returns a state indicating the parameter transformation thus induced.

The multiplicity of a knot is the number of times the same value appears in the sequence. The multiplicity of the end knots cannot exceed the order of the spline; the multiplicity of interior knots cannot exceed order-1. The knot sequence must be monotonically increasing. These problems and degeneracies are fixed using tolerances assigned by Open API.

Utilities are provided to assist in removing many of these conditions from your splines using larger tolerances.

Periodicity is determined directly from the spline data by Open API on create. A periodic spline is identified as follows:

1. Evaluate the location and tangent vectors at the start and end parameter values.
2. If the location vectors are the same, it is closed and is made periodic, but it might not be smooth at the closure.
3. If the tangent vectors have the same direction, it is smooth (G1). If the tangent vectors also have the same length, it is parameter smooth (C1).
4. If the spline is rational, and it is also C1 in homogeneous space, it can be unclamped. If the spline is not rational, Cartesian space is equivalent to homogeneous space and so it can be unclamped.
5. If it can be unclamped, the closure is made as high as the data allows (up to degree - 1). If the closure cannot be made C1, it is made G1. If it cannot be made G1, it is left closed, but not smooth. (Unclamping applies knot removal to both ends of the spline. NX chooses knot intervals for this process that yield wrap-around control points if the closure is C1.)

The period is determined from the parameter range as defined above, largest minus smallest allowed values.

If the closure for a periodic curve must be checked using tolerances larger than Open APIs defaults, use UF_CURVE_smooth_spline_data. If degeneracies and knot problems must be fixed using tolerances larger than Open APIs defaults, use UF_CURVE_fix_spline_data.

Specific errors: spline contains fixable problems, or unfixable problems. An unfixable overrides the fixable return code.

Environment

Internal and External

See Also
UF_CURVE_ask_parameterization
UF_CURVE_fix_spline_data
UF_CURVE_smooth_spline_data

History
This function was originally released in V15.0.

Required License(s)
gateway

```
int UF_CURVE_create_spline
(
    UF_CURVE_spline_p_t spline_data,
    tag_p_t spline_tag,
    int * num_states,
    UF_CURVE_state_p_t * states
)
```

UF_CURVE_spline_p_t	**spline_data**	Input	Address of spline structure describing the desired curve.
tag_p_t	**spline_tag**	Output	Tag of curve is returned
int *	**num_states**	Output	Number of entries in states array
UF_CURVE_state_p_t *	**states**	Output to UF_*free*	Array of states indicating what kinds of problems are present; if NULL no state return is needed. This array must be freed by calling UF_free.

UF_CURVE_create_spline_feature (view source)

Defined in: uf_curve.h

Overview
Create a general spline feature. Output the feature id.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_edit_spline_feature
UF_CURVE_ask_spline_feature

History
Originally released in V18.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_spline_feature
(
    tag_t spline,
    tag_t * feature_id
)
```

tag_t	**spline**	Input	object id of a smart spline
tag_t *	**feature_id**	Output	pointer to object id of the spline feature

---

UF_CURVE_create_spline_thru_pts (view source)

Defined in: uf_curve.h

Overview

Creates a spline curve from the input defining data specified by the UF_CURVE_pt_slope_crvatr_s data structure (See the description of the structure for a description of the data items). The spline passes through the input points and conforms to the specified slopes and curvatures.

The length of the array of parameters is num_points for non-periodic curves and (num_points+1) for periodic ones. The difference with periodic curves is that although we know the last point of the curve (it is coincident with the first point), we dont know the parameter range of the curve. Therefore, you need to provide the parameter for the last point (num_points +1) to specify the parameter range of the curve.

Environment

Internal and External

See Also

UF_CURVE_edit_spline_thru_pts
UF_CURVE_pt_slope_crvatr_t
Refer to the example

History

Original release was in V13.0

Required License(s)

gateway

```
int UF_CURVE_create_spline_thru_pts
(
    int degree,
    int periodicity,
    int num_points,
    UF_CURVE_pt_slope_crvatr_t point_data [ ] ,
    double parameters [ ] ,
    int save_def_data,
    tag_t* spline_tag
)
```

int	**degree**	Input	degree of the spline

int	**periodicity**	Input	periodicity of the spline: 0=non-periodic, 1=periodic
int	**num_points**	Input	number of points and parameters in the following arrays
UF_CURVE_pt_slope_crvatr_t	**point_data [ ]**	Input	array of data defining points and slope/curvature control
double	**parameters [ ]**	Input	parameters of input points. This is a user specified parameterization for the input points, which needs to be monotonic increasing (i.e. parameters(i) < parameters(i+1) for all i), but does not need to be normalized, if NULL then the default parameterization will be used.
int	**save_def_data**	Input	If save_def_data = 1, save input defining data with the created spline. Otherwise, no.
tag_t*	**spline_tag**	Output	tag of the created spline

UF_CURVE_create_trim (view source)

Defined in: uf_curve.h

Overview
Create an associative trim curve feature.

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_create_trim
(
    UF_CURVE_trim_p_t trim_info,
    UF_CURVE_trim_mult_p_t out_info,
    tag_t * trim_feature
)
```

UF_CURVE_trim_p_t	**trim_info**	Input	Information defining trim curve feature desired
UF_CURVE_trim_mult_p_t	**out_info**	Output to UF_*free*	If the return value is UF_CURVE_TRIM_MULT_PTS then this structure will contain information about multiple intersection points for each boundary. The caller

			can then pass the desired point in through the suggested_point field.
tag_t *	**trim_feature**	Output	Created trim curve feature

UF_CURVE_create_wrap_object (view source)

Defined in: uf_curve.h

Overview

Creates a wrap or unwrap curves feature.

If wrap_unwrap_sw is UF_CURVE_WRAP, the wrap_curves will be wrapped from the wrap_plane onto the wrap_face. The wrap_curves should lie on the wrap_plane, however the curves will internally be projected onto the wrap_plane along the normal to the wrap_plane before being wrapped. If this default projection is undesirable, be sure that the curves lie in the wrap_plane before invoking this function.

If wrap_unwrap_sw is UF_CURVE_UNWRAP, the wrap_curves will be unwrapped from the wrap_face onto the wrap_plane. The wrap_curves should lie on the wrap_face, however the selected curves will internally be projected onto the wrap_face along the face normals before being unwrapped. If this default projection is undesirable, be sure that the curves lie in the wrap_face before invoking this function.

Environment

Internal and External

See Also

Refer to the example

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_create_wrap_object
(
    UF_CURVE_wrap_data_p_t wrap_data,
    tag_t * wrap_curve_object
)
```

UF_CURVE_wrap_data_p_t	**wrap_data**	Input	Structure specifying the wrap /unwrap to be created.
tag_t *	**wrap_curve_object**	Output	Pointer to the Object Identifier of the new wrap or unwrap feature.

UF_CURVE_edit_arc_data (view source)

Defined in: uf_curve.h

Overview
Edit an existing arc. You can edit (change) csys matrix, the start and end angles, the coordinates of the arc center, and the radius by filling out the arc_coords data structure pointed to by UF_CURVE_arc_p_t. The arc is drawn counterclockwise from the start angle to the end angle. The start and end angles are expressed in radians. This function returns an error if the absolute value of the difference between the start and end angle is greater than two pi (plus the system tolerance). An error is also returned if the start angle is greater than the end angle.

Environment
Internal and External

See Also
UF_CURVE_arc_p_t

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_arc_data
(
    tag_t arc,
    UF_CURVE_arc_p_t arc_coords
)
```

tag_t	**arc**	Input	Object identifier of existing arc to edit
UF_CURVE_arc_p_t	**arc_coords**	Input	Edited coordinates of arc in absolute space

---

UF_CURVE_edit_bridge_feature (view source)

Defined in: uf_curve.h

Overview
Edit a bridge curve feature

Environment
Internal and External

See Also
UF_CURVE_create_bridge_feature
UF_CURVE_ask_bridge_feature

History
Originally released in V16.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_bridge_feature
(
    tag_t bridge_feature,
    UF_CURVE_bridge_data_p_t bridge_data
)
```

tag_t	**bridge_feature**	Input	Object identifier of the bridge curve feature
UF_CURVE_bridge_data_p_t	**bridge_data**	Input	Parameters that new bridge curve feature should have.

UF_CURVE_edit_by_curve_fit_data (view source)

Defined in: uf_curve.h

Overview

Save curve fit data to the curve feature and execute the editing process

In NX3, only the following three curve features are supported,
Projection curve features, Intersecion curve features and Join curve features.

From NX4 on, the following six curve features will be supported,
Projection curve features, Intersecion curve features, Join curve features.
Combined projection curve features, Isocline curve features and
Offset curve features

Environment

Internal and External

See Also

UF_CURVE_ask_curve_fit_data

History

This function is introduced in NX3.0.

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_by_curve_fit_data
(
    tag_t curve_feature,
    UF_CURVE_curve_fit_data curve_fit_data
)
```

tag_t	**curve_feature**	Input	tag of the curve feature
UF_CURVE_curve_fit_data	**curve_fit_data**	Input	curve fit method, maximum degree, and maximum segments

---

UF_CURVE_edit_combine_curves (view source)

Defined in: uf_curve.h

Overview

Edits parametric curve(s) by combining two curves along specified direction vectors. The parameters used to create the curve should be acquired by using UF_MODL_ask_combine_curves. Any of the original input parameters can be modified. When "normal to plane of curve" is specified as the projection type, the curve must be planar. An error occurs if the above is not true.

Environment
Internal and External

See Also
UF_CURVE_combine_curves_direction_t

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_combine_curves
(
    tag_t combine_curve_feature,
    tag_t first_curve_tag,
    UF_CURVE_combine_curves_direction_p_t first_dir,
    tag_t second_curve_tag,
    UF_CURVE_combine_curves_direction_p_t second_dir,
    const char * curve_aprox_tol
)
```

tag_t	**combine_curve_feature**	Input	Object tag for the combined curve feature to be edited.
tag_t	**first_curve_tag**	Input	First curve tag to be projected for combine.
UF_CURVE_combine_curves_direction_p_t	**first_dir**	Input	Pointer to projection direction info for first curve.
tag_t	**second_curve_tag**	Input	Second curve tag to be projected for combine.
UF_CURVE_combine_curves_direction_p_t	**second_dir**	Input	Pointer to projection direction info for second curve.
const char *	**curve_aprox_tol**	Input	String containing the value for the curve approximation tolerance (distance tolerance).

UF_CURVE_edit_conic_data (view source)

Defined in: uf_curve.h

Overview
Edits the defining data of a dumb conic. A dumb conic is no longer associative.

Environment

Internal & External

See Also
UF_CURVE_ask_conic_data

History
originally released in V15.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_conic_data
(
tag_t conic,
UF_CURVE_conic_p_t conic_data
)

tag_t	**conic**	Input	the dumb conic to edit
UF_CURVE_conic_p_t	**conic_data**	Input	defining data for the conic

---

UF_CURVE_edit_int_object (view source)

Defined in: uf_curve.h

Overview
Edits Intersection curves by allowing you to replace input intersection objects. Objects to intersect must be 1 body, 1 datum plane, or multiple faces (from same body) per set. Datum planes cannot be input for both sets since the result is a non-associative line.

Environment
Internal and External

History
Original release was in V13.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_int_object
(
int num_object_set_1,
tag_t * object_set_1,
int num_object_set_2,
tag_t * object_set_2,
tag_t int_curve_object
)

int	**num_object_set_1**	Input	Number of intersection objects in object_set_1 array.
tag_t *	**object_set_1**	Input	Array of tags of objects to intersect with objects in object_set_2 array.

int	**num_object_set_2**	Input	Number of intersection objects in object_set_2 array.
tag_t *	**object_set_2**	Input	Array of tags of objects to intersect with objects in object_set_1 array.
tag_t	**int_curve_object**	Input	Feature tag of intersection curve to edit.

UF_CURVE_edit_isocline (view source)

Defined in: uf_curve.h

Overview
Edit isocline curve feature
Modify an isocline curve feature.

Environment
Internal and External

See Also
UF_CURVE_create_isocline
UF_CURVE_ask_isocline

Required License(s)
gateway

```
int UF_CURVE_edit_isocline
(
    tag_t isocline_feat,
    int face_cnt,
    tag_t faces [ ] ,
    double direction [ 3 ] ,
    const char * start_angle,
    const char * end_angle,
    const char * step_angle
)
```

tag_t	**isocline_feat**	Input	Tag of feature
int	**face_cnt**	Input	Number of faces
tag_t	**faces [ ]**	Input	Array of faces used to calculate isocline curves
double	**direction [3]**	Input	Isocline direction vector
const char *	**start_angle**	Input	Start angle (-90 to 90 degrees)
const char *	**end_angle**	Input	End angle (-90 to 90 degrees)
const char *	**step_angle**	Input	Step angle (NULL for single angle)

UF_CURVE_edit_joined_feature (view source)

Defined in: uf_curve.h

Overview

Edits a joined curve feature.

Environment

Internal and External.

See Also

UF_MODL_set_distance_tolerance
UF_MODL_set_angle_tolerance
UF_STRING_p_t AKA string_list
UF_MODL_init_string_list
UF_MODL_create_string_list
UF_MODL_free_string_list

History

This function was originally released in V15.0.
Creation method has the advanced option in NX3.0.

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_joined_feature
(
    tag_t joined_curve_feature,
    UF_STRING_p_t uf_curve_string,
    int creation_method,
    double tols [ 3 ]
)
```

tag_t	**joined_curve_feature**	Input	CRV_JOIN feature
UF_STRING_p_t	**uf_curve_string**	Input	string of input curves. Use UF_MODL_init_string_list and UF_MODL_create_string_list to create input strings. Use UF_MODL_free_string_list to free memory after the string is created.
int	**creation_method**	Input	1 - Polynomial Cubic, 2 - General Spline 3 - polynomial quintic 4 - Advanced
double	**tols [3]**	Input	Tolerances [0] - distance tol [1] - angle tol [2] - string tol

---

UF_CURVE_edit_length (view source)

Defined in: uf_curve.h

Overview

Trim or extend the length of a user specified curve. If the method is 1, and the length is positive, the curve is extended. If the method is 1, and the length is negative, the curve is trimmed. If the method is 2, the curve will be extended or trimmed as needed to make the total curve length equal to the specified length.

Note: When this function is used on a regular spline, it will change the length of that particular spline. However, when it is used on a spline feature, it will create a new spline of the appropriate length, leaving the original intact.

Environment
Internal and External

History
Originally released in V16.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_length
(
tag_t curve,
int method,
double length,
int location,
int ext_type
)

tag_t	**curve**	Input	tag of curve to trim
int	**method**	Input	trim/extend method, increment or total curve length = 1 -> trim/extend the curve incrementally by the specified curve length = 2 -> trim/extend the curve by making the total curve length as the specified curve length
double	**length**	Input	incremental or total curve length
int	**location**	Input	Flag to indicate which end of the curve would be edited = 1 -> start of curve = 2 -> end of curve = 3 -> both sides of curve - if method = 1, it will extend/trim both ends by one half of the length - if method = 2, it will determing how much the curve needs to be extended/trimmed, and then apply half of that distance to both ends.
int	**ext_type**	Input	Shape of the extension of the curve. It can be either of the following: = UF_CURVE_NATURAL_SHAPE = UF_CURVE_LINEAR_SHAPE = UF_CURVE_CIRCULAR_SHAPE

UF_CURVE_edit_line_arc (view source)

Defined in: uf_curve.h

Overview

Edits an existing associative line/arc feature. An error is returned if line/arc cannot be created with the specified data.

Return
error code

Environment
Internal and External

See Also
UF_CURVE_create_line_arc
UF_CURVE_ask_line_arc_data
UF_CURVE_ask_line_arc_output

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_line_arc
(
    tag_t line_arc_feat_id,
    UF_CURVE_line_arc_t * line_arc_data
)
```

tag_t	**line_arc_feat_id**	Input	Object identifier of associative line/arc feature.
UF_CURVE_line_arc_t *	**line_arc_data**	Input	Pointer to line/arc data structure to be filled up.

UF_CURVE_edit_line_data (view source)

Defined in: uf_curve.h

Overview
Edits (changes) the coordinates of a line. You input the tag of the point that you would like to edit, then pass in the new coordinates (start and end) to the line_coords data structure pointed to by UF_CURVE_line_p_t

Environment
Internal and External

See Also
UF_CURVE_line_p_t

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_line_data
(
    tag_t line,
    UF_CURVE_line_p_t line_coords
)
```

tag_t	**line**	Input	Object identifier of line to edit

UF_CURVE_line_p_t	**line_coords**	Input	Edited coordinates of point in absolute space

---

UF_CURVE_edit_move_mult_points (view source)

Defined in: uf_curve.h

Overview

Applies the move multiple curve points method to the editing of a spline. The identifier of the curve to be edited is provided along with a structure containing the details of the edit. The first four fields of the structure are required. The two int fields are also required. The need for the remaining double elements is based on the method chosen.

There are three values for the displace_method element. They are:
1 = distance normal to the curve
2 = vector and distance
3 = direction point.

If displace_method1 = 1, then displace_method2 must = 1. For values of 1 and 2 the distance1 and distance2 fields are required. For method 3 they are not required. Vector1 and vector2 are only used by 2, the vector method. Dir_pt1 and dir_pt2 are only used by 3, the direction point method.

Environment

Internal and External

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_move_mult_points
(
    tag_t curve_tag,
    UF_CURVE_BCMMCP_t * mmcp_dat
)
```

tag_t	**curve_tag**	Input	tag of curve
UF_CURVE_BCMMCP_t *	**mmcp_dat**	Input	Structure containing spline edit data. The structure is defined in uf_curve.h

---

UF_CURVE_edit_ocf_feature (view source)

Defined in: uf_curve.h

Overview

Edits an associated offset curves on face feature using the input offset data
A UF_CURVE_ocf_data_p_t is allocated and filled.
The input UF_CURVE_ocf_data_p_t is used to update the feature and the feature is updated to reflect the changes

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_ocf_feature
(
    UF_CURVE_ocf_data_p_t offset_data,
    tag_t feature
)
```

UF_CURVE_ocf_data_p_t	**offset_data**	Input	Pointer to the structure containing defining data for the offset curve on face operation
tag_t	**feature**	Input	Offset curve on face identifier

UF_CURVE_edit_offset_object (view source)

Defined in: uf_curve.h

Overview
Edits the creation parameters of an offset curve object.

Only the following types of offsets are allowed for offset_data_pointer:
UF_CURVE_OFFSET_DISTANCE_NO_TRIM
UF_CURVE_OFFSET_DISTANCE_TANGENT
UF_CURVE_OFFSET_DISTANCE_FILLET
UF_CURVE_OFFSET_DRAFT_NO_TRIM
UF_CURVE_OFFSET_DRAFT_TANGENT
UF_CURVE_OFFSET_DRAFT_FILLET
UF_CURVE_OFFSET_3D_AXIAL

Environment
Internal & External

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_offset_object
(
    UF_CURVE_offset_data_p_t offset_data_pointer,
    tag_t offset_curve_object
)
```

UF_CURVE_offset_data_p_t	**offset_data_pointer**	Input	Pointer to structure containing the defining data of the offset curve.
tag_t	**offset_curve_object**	Input	Object identifier of offset curve object to be edited.

UF_CURVE_edit_point_data (view source)

Defined in: uf_curve.h

Overview

Edits (changes) the coordinates of a point. You input the tag of the point that you would like to edit, then pass in the new coordinate values.

Environment

Internal and External

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_point_data
(
    tag_t point,
    double point_coords [ 3 ]
)
```

tag_t	**point**	Input	Object identifier of point to edit
double	**point_coords [3]**	Input	Edited coordinates of point in absolute space

UF_CURVE_edit_proj_curves (view source)

Defined in: uf_curve.h

Overview

Edits projection curve features.

Objects to project may be points, curves, sketch feature identifiers, or other projected curve feature identifiers. The "number of curves and points" is the number of such object tags (and does not include the numbers of curves in any sketch or projection curve feature).

NOTE: The Along Vector option (proj_type = 3) projects the selected objects along a specified vector. You can project the curves in the direction indicated by the vector by setting the multiplicity to 1 (single), or in both directions by setting the multiplicity to 2 (both). The Along Vector option produces all possible images on the face.

Points can not be projected using "angle and vector" (proj_type = 4) or "equal arc length" (proj_type =6).

NOTE: You can change the tolerance used to create a curve projection by using the subroutine UF_MODL_set_distance_tolerance. The projection along face normals or along a vector produces an exact projection when projecting onto a plane.

Environment

Internal and External

See Also
UF_MODL_create_proj_curves
UF_MODL_ask_proj_curves
UF_CURVE_ask_proj_curves

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_proj_curves
(
    tag_t proj_curve_feature,
    int n_curve_refs,
    tag_t * curve_refs,
    int n_face_refs,
    tag_t * face_refs,
    UF_CURVE_proj_p_t proj_data
)
```

tag_t	**proj_curve_feature**	Input	Projected curve feature identifier
int	**n_curve_refs**	Input	Number of curves and points
tag_t *	**curve_refs**	Input	Array of curve and point identifiers.
int	**n_face_refs**	Input	Number of sheet bodies, faces and planes
tag_t *	**face_refs**	Input	Array of plane, datum plane, face or sheet body identifiers.
UF_CURVE_proj_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj_s for entries.

UF_CURVE_edit_proj_curves1 (view source)

Defined in: uf_curve.h

Overview
Edits projection curve features.

Objects to project may be points, curves, sketch feature identifiers, or other projected curve feature identifiers. The "number of curves and points" is the number of such object tags (and does not include the numbers of curves in any sketch or projection curve feature).

NOTE: The Along Vector option (proj_type = 3) projects the selected objects along a specified vector. You can project the curves in the direction indicated by the vector by setting the multiplicity to 1 (single), or in both directions by setting the multiplicity to 2 (both). The Along Vector option produces all possible images on the face.

Points can not be projected using "angle and vector" (proj_type = 4) or "equal arc length" (proj_type =6).

NOTE: You can change the tolerance used to create a curve projection by using the subroutine UF_MODL_set_distance_tolerance. The projection along face normals or along a vector produces an exact projection when projecting onto a plane.

Environment
Internal and External

See Also
UF_MODL_create_proj_curves
UF_MODL_ask_proj_curves
UF_CURVE_ask_proj_curves

History
Originally released in NX4.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_proj_curves1
(
    tag_t proj_curve_feature,
    int n_curve_refs,
    tag_t * curve_refs,
    int n_face_refs,
    tag_t * face_refs,
    UF_CURVE_proj1_p_t proj_data
)
```

tag_t	**proj_curve_feature**	Input	Projected curve feature identifier
int	**n_curve_refs**	Input	Number of curves and points
tag_t *	**curve_refs**	Input	Array of curve and point identifiers.
int	**n_face_refs**	Input	Number of sheet bodies, faces and planes
tag_t *	**face_refs**	Input	Array of plane, datum plane, face or sheet body identifiers.
UF_CURVE_proj1_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj1_s for entries.

UF_CURVE_edit_spline_feature (view source)

Defined in: uf_curve.h

Overview
Edit a general spline feature.

Return
error code

Environment

Internal and External

See Also
UF_CURVE_create_spline_feature
UF_CURVE_ask_spline_feature

History
Originally released in V18.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_spline_feature
(
tag_t spline,
tag_t feature_id
)

tag_t	**spline**	Input	object id of a new smart spline
tag_t	**feature_id**	Input	object id of the spline feature being edited

---

UF_CURVE_edit_spline_thru_pts (view source)

Defined in: uf_curve.h

Overview
Edits the defining data of a spline, which is replaced by the new input data as specified. If the spline has no or incomplete defining data, its shape is recreated from the input defining data.

The length of the array of parameters is num_points for non-periodic curves and (num_points+1) for periodic ones. The difference with periodic curves is that although we know the last point of the curve, it is the first point, we dont know the parameter range of the curve. You need to give us the parameter for the last point so we know the parameter range of the curve.

Environment
Internal and External

See Also
UF_CURVE_create_spline_thru_pts
UF_CURVE_edit_spline_thru_pts
UF_CURVE_pt_slope_crvatr_t

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_spline_thru_pts
(
tag_t spline_tag,
int degree,
int periodicity,
int num_points,

```
    UF_CURVE_pt_slope_crvatr_t * point_data,
    double * parameters,
    int save_def_data
)
```

tag_t	**spline_tag**	Input	tag of the spline to be edited
int	**degree**	Input	degree of the spline
int	**periodicity**	Input	periodicity of the spline
int	**num_points**	Input	number of points and parameters in the following arrays
UF_CURVE_pt_slope_crvatr_t *	**point_data**	Input	array of data defining points and slope/curvature control
double *	**parameters**	Input	parameters of input points. This is a user specified parameterization for the input points, which needs to be monotonic increasing, but does not need to be normalized, if NULL then default parameterization will be used
int	**save_def_data**	Input	If save_def_data = 1, save input defining data with the created spline. Otherwise, no.

UF_CURVE_edit_trim (view source)

Defined in: uf_curve.h

Overview
Change the parameters of an associative trim curve feature to the parameters supplied

Environment
Internal and External

See Also
Refer to the example

History
Originally released in V16.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_edit_trim
(
    tag_t trim_feature,
    UF_CURVE_trim_p_t trim_info
)
```

tag_t	**trim_feature**	Input	Trim curve feature whose parameters are to be changed
UF_CURVE_trim_p_t	**trim_info**	Input	Information defining the trim curve features desired parameters

---

UF_CURVE_edit_trim_curve (view source)

Defined in: uf_curve.h

Overview

Trims a curve to a bounding object. The bounding object can be a curve, edge, plane, face, or point. The reference point is used to determine which end of the curve to trim. The reference point is also used to start the iteration that determines the intersection point of the curve and the bounding object. The extension shape (ext_ind) parameter is used when the curve is extended.

Environment

Internal and External

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_trim_curve
(
    tag_t curve_tag,
    tag_t bounding_id,
    double ref_point [ 3 ] ,
    double int_point [ 3 ] ,
    int ext_ind
)
```

tag_t	**curve_tag**	Input	tag of curve to trim
tag_t	**bounding_id**	Input	tag of first bounding object (curve, edge, plane, face, point)
double	**ref_point [3]**	Input	ref_point[3] are the coordinates of a point used to specify which portion of the curve to trim also used as start point intersection point iteration (ACS)
double	**int_point [3]**	Input	int_point[3] is the start point for finding the intersection of the curve to trim and the bounding object.
int	**ext_ind**	Input	shape of the extension of the curve: UF_CURVE_NATURAL_SHAPE UF_CURVE_LINEAR_SHAPE UF_CURVE_CIRCULAR_SHAPE

UF_CURVE_edit_with_template (view source)

Defined in: uf_curve.h

Overview

Edit a B-curve with the spline order and knot vector taken from a template curve, when possible.

RESTRICTIONS -

1. In some cases the edit curve cannot be edited with the selected template curve. In these cases the spline order and knot vector of the edited curve are different from the template curve, and the edited curve is interpolated. These cases are distinguished by the error_data parameter.

Failure return code: TEMPLATE_CURVE_UNABLE_TO_EDIT

Environment

Internal and External

See Also

UF_MODL_create_curve_mesh1 and UF_MODL_create_thru_curves1
UF_MODL_create_curve_mesh1
UF_MODL_create_thru_curves1

History

Originally released in V16.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_edit_with_template
(
    tag_t edit_id,
    tag_t template_id,
    UF_CURVE_fit_error_p_t error_data
)
```

tag_t	**edit_id**	Input	Curve to edit
tag_t	**template_id**	Input	Template curve Spline order and knot vector are taken from template curve
UF_CURVE_fit_error_p_t	**error_data**	Output	Error data returned by edit with template.

UF_CURVE_edit_wrap_object (view source)

Defined in: uf_curve.h

Overview

Edits the creation parameters of a wrap or unwrap curves feature.

Environment

Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_edit_wrap_object
(
UF_CURVE_wrap_data_p_t wrap_data,
tag_t wrap_curve_object
)

UF_CURVE_wrap_data_p_t	**wrap_data**	Input	Structure specifying the new wrap/unwrap data for wrap_curve_object.
tag_t	**wrap_curve_object**	Input	Object Identifier of the wrap or unwrap feature.

UF_CURVE_evaluate_curve (view source)

Defined in: uf_curve.h

Overview
Returns the point on the curve and the requested derivatives.
This function differs from UF_MODL_evaluate_curve in two ways:

The param and deriv_flag are not pointers, and the param value is specified in the "natural" parameter range of the curve. For example, to evaluate an arc, UF_MODL_evaluate_curve takes a parameter value between 0 and 1, normalized to the parameter range of the arc when created. UF_CURVE_evaluate_curve takes a parameter in radians, in the parameter range given when the arc was created.

Environment
Internal and External

See Also
UF_MODL_evaluate_curve
UF_CURVE_ask_parameterization
UF_CURVE_evaluate_curve_structure

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_evaluate_curve
(
tag_t curve,
double param,
int deriv_flag,
double pos_and_deriv [ ]
)

tag_t	**curve**	Input	Tag of curve to evaluate
double	**param**	Input	Parameter value at which to evaluate
int	**deriv_flag**	Input	Number of derivatives to evaluate: UF_CURVE_LOC = return the point

			UF_CURVE_LOC_1STDERV = return the point and 1st derivative UF_CURVE_LOC_1STDERV_2NDDERV = return the point, 1st and 2nd derivatives
double	**pos_and_deriv [ ]**	Output	Position and derivatives. Dimension of the array = 3 (deriv_flag+1)

UF_CURVE_evaluate_curve_structure (view source)

Defined in: uf_curve.h

Overview

Returns the point on the curve and the requested derivatives.
This function differs from UF_MODL_evaluate_curve in two ways:

The param and deriv_flag are not pointers, and the param value is specified in the "natural" parameter range of the curve. For example, to evaluate an arc, UF_MODL_evaluate_curve takes a parameter value between 0 and 1, normalized to the parameter range of the arc when created. UF_CURVE_evaluate_curve takes a parameter in radians, in the parameter range given when the arc was created.

This function differs from UF_CURVE_evaluate_curve is that the input is a curve data structure instead of a curve tag.

Environment

Internal and External

See Also

UF_MODL_evaluate_curve
UF_CURVE_ask_parameterization
UF_CURVE_evaluate_curve

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_evaluate_curve_structure
(
    UF_CURVE_struct_t * curve_data_ptr,
    double param,
    int deriv_flag,
    double * pos_and_deriv
)
```

UF_CURVE_struct_t *	**curve_data_ptr**	Input	Data structure of curve to evaluate
double	**param**	Input	Parameter value at which to evaluate
int	**deriv_flag**	Input	Number of derivatives to evaluate: UF_CURVE_LOC = return the point UF_CURVE_LOC_1STDERV = return the point and 1st derivative UF_CURVE_LOC_1STDERV_2NDDERV = return the point, 1st and 2nd derivatives

double *	**pos_and_deriv**	Output	Position and derivatives. Dimension of the array = 3 (deriv_flag+1)

UF_CURVE_fix_spline_data (view source)

Defined in: uf_curve.h

Overview

This function scans the spline data for degeneracies, points that are or potentially are non-C0, and knot values too close together.

If any degeneracies are found, it attempts to fix them by moving one or more control points. If that is not successful, points and knots will be removed, reducing the degree if necessary.

If any non-C0 or potentially non-C0 control points are found they are coerced to C0.

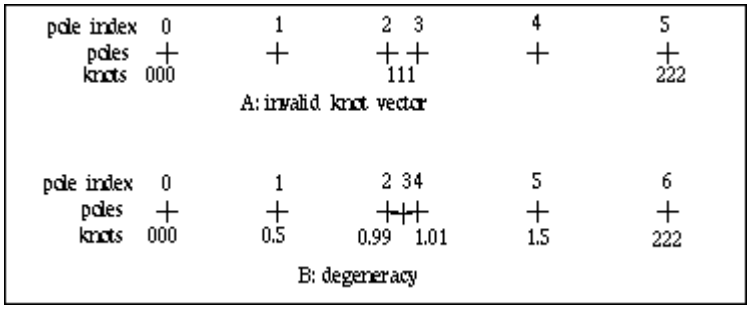
If any two adjacent knot values that are distinct are not separated by enough, they are fixed.

The fixed-up data are returned in the same memory as the supplied data.

If valid spline data cannot be returned, it means that the data cannot be represented by a linear line of nonzero length; it is too short. The flags in the states return with "WAS" in the name indicate the status of the input curve data; the output data, if there is no error, is always valid. The flag with "IS" in the name indicates the status of the output curve.

Removing degeneracies and fixing illegal knot vectors may introduce discontinuities.

Specific errors: spline is completely degenerate.



This Figure shows two possible problems with spline data. (Neither one is to scale.) In the first case (A), the spline has an illegal knot multiplicity at value 1. It is not C0: there is a gap between points 2 and 3. It is fixed. If poles 2 and 3 are coincident, the shape of the spline is not changed. If they are not, the spline is changed, and a state is returned.

In the second case (B), the spline is fixed by pushing poles 2 and 4 away from 3, if it is possible to do so without causing other problems. If that would introduce other problems, poles and knots are deleted

until the spline is no longer degenerate.

Environment
Internal and External

See Also
UF_BREP_attach_geometry
UF_CURVE_create_spline

History
This function was originally released in V15.0.

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_fix_spline_data
(
    UF_CURVE_spline_p_t spl,
    double toler,
    int * num_states,
    UF_CURVE_state_p_t * states
)
```

UF_CURVE_spline_p_t	**spl**	Input / Output	spline data in which to fix degeneracies and knot multiplcities
double	**toler**	Input	tolerance to use for determining degeneracies. zero indicates use NX modelling resolution.
int *	**num_states**	Output	number of returned states
UF_CURVE_state_p_t *	**states**	Output to UF_*free*	Pointer to state array pointer to receive state information. If a NULL pointer value is provided, no states are returned. The caller is responsible for freeing this by calling UF_free.

UF_CURVE_free_curve_struct (view source)

Defined in: uf_curve.h

Overview
Frees the curve structure pointer obtained from the function UF_CURVE_ask_curve_struct().

NOTE: This routine is to be permanently withdrawn in the near future. Use the routines in the UF_EVAL module instead.
Frees the curve structure pointer obtained from the function UF_CURVE_ask_curve_struct.

Environment
Internal and External

See Also

Refer to the example

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_free_curve_struct
(
UF_CURVE_struct_p_t curve_struct
)

UF_CURVE_struct_p_t	**curve_struct**	Input	Curve structure pointer

---

UF_CURVE_free_ocf_data (view source)

Defined in: uf_curve.h

Overview
Free the input offset data allocated during crate/edit of an associated offset curves on face feature.

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_free_ocf_data
(
UF_CURVE_ocf_data_p_t * offset_data
)

UF_CURVE_ocf_data_p_t *	**offset_data**	Input / Output	Free the data allocated to define the offset curve on face feature.

UF_CURVE_free_offset_parms (view source)

Defined in: uf_curve.h

Overview
This routine should be called after a call to UF_CURVE_ask_offset_parms in order to free the space allocated by this routine.

Environment
Internal and External

See Also
UF_CURVE_ask_offset_parms

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_free_offset_parms
(
UF_CURVE_offset_data_p_t offset_data_pointer
)

UF_CURVE_offset_data_p_t	**offset_data_pointer**	Input	Pointer to structure containing the defining data of the offset curve.

UF_CURVE_free_trim (view source)

Defined in: uf_curve.h

Overview
Free the memory within the UF_CURVE_trim_p_t structure that was allocated by a call to UF_CURVE_ask_trim

Environment
Internal and External

See Also
Refer to the example

History
Originally released in V16.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_free_trim
(
UF_CURVE_trim_p_t trim_info
)

UF_CURVE_trim_p_t	**trim_info**	Input	Information defining the trim curve features parameters that was generated by a call to UF_CURVE_ask_trim

---

UF_CURVE_free_wrap_parms (view source)

Defined in: uf_curve.h

Overview
Frees any data structures pointed to by the input structure. The existing pointers will be set to NULL after the associated memory is freed. The memory for the UF_CURVE_wrap_data_t structure is not

freed by this function.

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_free_wrap_parms
(
UF_CURVE_wrap_data_p_t wrap_data
)

UF_CURVE_wrap_data_p_t	**wrap_data**	Input	Structure specifying the wrap/unwrap to be freed.

UF_CURVE_init_ocf_data (view source)

Defined in: uf_curve.h

Overview
Initialize a offset data structure for an associated offset curve in face feature. This function assumes that memory has already been allocated

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_init_ocf_data
(
UF_CURVE_ocf_data_p_t uf_offset_data
)

UF_CURVE_ocf_data_p_t	**uf_offset_data**	Input	Init the data defining the offset curve on face feature

---

UF_CURVE_init_proj_curves_data (view source)

Defined in: uf_curve.h

Overview
Initialize a project curve (UF_CURVE_proj_p_t) data structure for an associated project curve feature. This function assumes that memory has already been allocated

Environment
Internal and External

History
Originally released in NX4.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_init_proj_curves_data
(
UF_CURVE_proj_p_t proj_data
)

UF_CURVE_proj_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj_s for entries.

UF_CURVE_init_proj_curves_data1 (view source)

Defined in: uf_curve.h

Overview
Initialize a project curve (UF_CURVE_proj1_p_t) data structure for an associated project curve feature. This function assumes that memory has already been allocated

Environment
Internal and External

History
Originally released in NX4.0

Required License(s)
(solid_modeling or drafting)

int UF_CURVE_init_proj_curves_data1
(
UF_CURVE_proj1_p_t proj_data
)

UF_CURVE_proj1_p_t	**proj_data**	Input	Pointer to projection curve data structure. See documentation of UF_CURVE_proj1_s for entries.

UF_CURVE_intersect (view source)

Defined in: uf_curve.h

Overview
Determine intersection between curve/edge and a curve/edge/face/single face body/plane/datum plane. The intersection will be done in the orientation of the current WCS when the entity is a curve/edge otherwise in 3D. If the intersection is an actual

3D intersection then that information will be output. Also the intersection between the naturally extended version of the curve or edge will be returned if needed, however, only a curve(the first calling parameter) will be extended when the entity(the second calling parameter) is something other than a curve/edge.

For curve (the first calling parameter) of UF_line_type or UF_spline_type, a return parameter greater than 0 or
less than 1, indicates that the itersection is on the natural extension of the curve.

Return
error code

Environment
Internal and External

History
Originally released in V18.0

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_intersect
(
    tag_t curve,
    tag_t entity,
    double ref_point [ 3 ] ,
    UF_CURVE_intersect_info_p_t out_info
)
```

tag_t	**curve**	Input	object id of the curve or edge
tag_t	**entity**	Input	object id of the curve, edge, face, single face body, plane or datum plane
double	**ref_point [3]**	Input	reference point to select from multiple intersections, intersection closest to this point will be output
UF_CURVE_intersect_info_p_t	**out_info**	Output	- intersection information

---

UF_CURVE_is_spline_in_sync (view source)

Defined in: uf_curve.h

Overview
Queries if a spline is in synchronization with its defining data. The shape of a spline is determined by its control pole data, which could be obtained from its defining data. If the control pole data of a spline is matched with its defining data, then the spline is in synchronization, otherwise, the spline is out of synchronization. For example, users can modify the control pole data of a spline after it is created from the defining data, in this case, it is out of synchronization.

Environment
Internal and External

See Also
UF_CURVE_create_spline_thru_pts
UF_CURVE_ask_spline_thru_pts
UF_CURVE_edit_spline_thru_pts

History
Original release was in V13.0

Required License(s)
gateway

```
int UF_CURVE_is_spline_in_sync
(
    tag_t spline_tag,
    logical * is_sync
)
```

tag_t	**spline_tag**	Input	tag of the spline
logical *	**is_sync**	Output	TRUE = the spline is in synchronization with its defining data. FALSE = the spline is out of synchronization with its defining data.

UF_CURVE_is_spline_self_int (view source)

Defined in: uf_curve.h

Overview
Determines whether a spline has self intersecting loops.

Environment
Internal and External

History
Original release was in V14.0.

Required License(s)
gateway

```
int UF_CURVE_is_spline_self_int
(
    tag_t spline_tag,
    logical * is_self_intersecting
)
```

tag_t	**spline_tag**	Input	Object identifier of spline to inquire.
logical *	**is_self_intersecting**	Output	TRUE - spline is self intersecting (has loops) FALSE - spline is not self intersecting.

UF_CURVE_LINE_ARC__is_arc_equal (view source)

Defined in: uf_curve.h

Overview
Test whethere two arcs are geometrically different

Return
0 - Two Arcs are geometrically identical
1 - Two Arcs are geometrically different

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_LINE_ARC__is_arc_equal
(
    tag_t arc1,
    tag_t arc2
)
```

tag_t	**arc1**	Input	tag of arc 1
tag_t	**arc2**	Input	tag of arc 2

UF_CURVE_LINE_ARC__is_line_equal (view source)

Defined in: uf_curve.h

Overview
Test whether two lines are geometrically different

Return
0 - Two Lines are geometrically identical
1 - Two Lines are geometrically different

Environment
Internal and External

Required License(s)
gateway

```
int UF_CURVE_LINE_ARC__is_line_equal
(
    tag_t line1,
    tag_t line2
)
```

tag_t	**line1**	Input	tag of line 1

tag_t	**line2**	Input	tag of line 2

UF_CURVE_modify_offsets_in_string (view source)

Defined in: uf_curve.h

Overview

Modify the offset values associated with a string in an associated offset curves on face feature.
If the string does not exist an error is returned.

Environment

Internal and External

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_modify_offsets_in_string
(
    tag_t string_tag,
    UF_CURVE_ocf_data_p_t uf_offset_data,
    int num_offsets,
    UF_CURVE_ocf_values_p_t offset_distance
)
```

tag_t	**string_tag**	Input	Identifier to string that has to be modified
UF_CURVE_ocf_data_p_t	**uf_offset_data**	Input / Output	Pointer to data defining the offset curve on face operation
int	**num_offsets**	Input	New number of offsets to create
UF_CURVE_ocf_values_p_t	**offset_distance**	Input	New offset distances

---

UF_CURVE_ocf_ask_curves (view source)

Defined in: uf_curve.h

Overview

Return all offset curves for the associated offset curves on face feature
The caller has to free the memory allocated when offset_curves is returned.

Environment

Internal and External

Required License(s)

gateway

```
int UF_CURVE_ocf_ask_curves
(
    tag_t feature_eid,
    int * num_curves,
    tag_t * * offset_curves
)
```

tag_t	**feature_eid**	Input	: Identifier to the offset curve on face feature
int *	**num_curves**	Output	: Number of offset curves created
tag_t * *	**offset_curves**	Output to UF_*free*	pointer to array containing the offset curve identifiers

---

UF_CURVE_ocf_offset_pt_direction (view source)

Defined in: uf_curve.h

Overview

Given a section tag and the face collector tag, get the default offset direction and a reference point on the section.
The offset point and the offset direction is returned along with an error code that indicates if the default offset direction was computed succesfully or not.

Environment

Internal and External

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_ocf_offset_pt_direction
(
    tag_t uf_string_tag,
    tag_t uf_face_collector_tag,
    double offset_point [ 3 ] ,
    double offset_direction [ 3 ]
)
```

tag_t	**uf_string_tag**	Input	Section tag
tag_t	**uf_face_collector_tag**	Input	Face Collector tag
double	**offset_point [3]**	Output	Reference Point
double	**offset_direction [3]**	Output	Default offset direction

---

UF_CURVE_remove_string_from_ocf_data (view source)

Defined in: uf_curve.h

Overview
Remove a string from the associated offset curves in face feature.

If the string is not found, an error is returned.

Environment
Internal and External

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_remove_string_from_ocf_data
(
    tag_t string_tag,
    UF_CURVE_ocf_data_p_t uf_offset_data
)
```

tag_t	**string_tag**	Input	Identifier of string
UF_CURVE_ocf_data_p_t	**uf_offset_data**	Input / Output	Pointer to the data defining the offset curve on face operation

---

UF_CURVE_section_ask_parallel_data (view source)

Defined in: uf_curve.h

Overview
Outputs the defining data structures for a section curves feature defined using a set of parallel planes.

Environment
Internal and External

See Also
See UF_CURVE_section_from_planes for a list of related functions.

History
Originally released in V17.0

Required License(s)
gateway

```
int UF_CURVE_section_ask_parallel_data
(
    tag_t section_curves_feature,
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_parallel_data_p_t parallel_data
)
```

tag_t	**section_curves_feature**	Input	The section curve feature to inquire about.
UF_CURVE_section_general_data_p_t	**general_data**	Output to UF_*free*	The general data describing this section curve feature. Use UF_free to free the array, general_data->objects
UF_CURVE_section_parallel_data_p_t	**parallel_data**	Output	The data defining the parallel planes for this feature.

UF_CURVE_section_ask_perpcrv_data (view source)

Defined in: uf_curve.h

Overview
Outputs the defining data structures for a section curves feature defined using a set of planes perpendicular to a curve.

Environment
Internal and External

See Also
See UF_CURVE_section_from_planes for a list of related functions.

History
Originally released in V17.0

Required License(s)
gateway

```
int UF_CURVE_section_ask_perpcrv_data
(
    tag_t section_curves_feature,
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_perpcrv_data_p_t perpcrv_data
)
```

tag_t	**section_curves_feature**	Input	The section curve feature to inquire about.
UF_CURVE_section_general_data_p_t	**general_data**	Output to UF_*free*	The general data describing this section curve feature. Use UF_free to free the array, general_data->objects
UF_CURVE_section_perpcrv_data_p_t	**perpcrv_data**	Output	The data defining the set of planes perpendicular to a curve for this feature.

UF_CURVE_section_ask_planes_data (view source)

Defined in: uf_curve.h

Overview
Outputs the defining data structures for a section curves feature defined using an array of planes.

Environment
Internal and External

See Also
See UF_CURVE_section_from_planes for a list of related functions.

History
Originally released in V17.0

Required License(s)
gateway

```
int UF_CURVE_section_ask_planes_data
(
    tag_t section_curves_feature,
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_planes_data_p_t planes_data
)
```

tag_t	**section_curves_feature**	Input	The section curve feature to inquire about.
UF_CURVE_section_general_data_p_t	**general_data**	Output to UF_*free*	The general data describing this section curve feature. Use UF_free to free the array, general_data->objects
UF_CURVE_section_planes_data_p_t	**planes_data**	Output to UF_*free*	The section planes data describing this feature. Use UF_free to free the array, planes_data->planes

UF_CURVE_section_ask_radial_data (view source)

Defined in: uf_curve.h

Overview
Outputs the defining data structures for a section curves feature defined using a set of radial planes.

Environment

Internal and External

See Also
See UF_CURVE_section_from_planes for a list of related functions.

History
Originally released in V17.0

Required License(s)
gateway

int UF_CURVE_section_ask_radial_data
(
tag_t section_curves_feature,
UF_CURVE_section_general_data_p_t general_data,
UF_CURVE_section_radial_data_p_t radial_data
)

tag_t	**section_curves_feature**	Input	The section curve feature to inquire about.
UF_CURVE_section_general_data_p_t	**general_data**	Output to UF_*free*	The general data describing this section curve feature. Use UF_free to free the array, general_data->objects
UF_CURVE_section_radial_data_p_t	**radial_data**	Output	The data defining the radial planes for this feature.

UF_CURVE_section_ask_type (view source)

Defined in: uf_curve.h

Overview
Given a section curves feature, this function returns
the type of planes used in the sectioning.

Return
0 if no error
WRONG_FEATURE_TYPE if not a section curves feature

Environment
Internal and External

See Also
See UF_CURVE_section_from_planes for a list of related functions.

History
Originally released in V17.0

Required License(s)
gateway

```
int UF_CURVE_section_ask_type
(
    tag_t section_curves_feature,
    int * plane_type
)
```

tag_t	**section_curves_feature**	Input	The tag of the section curves feature to inquire.
int *	**plane_type**	Output	The plane type: 0 for an array of planes 1 for parallel planes 2 for radial planes 3 for planes perpendicular to a curve

---

UF_CURVE_section_curve_ask_parents (view source)

Defined in: uf_curve.h

Overview

Given a section curve, this function returns the section curves feature, if any, and the defining object and sectioning objects.

Return
0 if no error
WRONG_FEATURE_TYPE if not a section curves feature

Environment

Internal and External

See Also

See UF_CURVE_section_from_planes for a list of related functions.

History

Originally released in V17.0

Required License(s)

gateway

```
int UF_CURVE_section_curve_ask_parents
(
    tag_t section_curve,
    tag_t * section_curves_feature,
    int * plane_type,
    tag_t * defining_object,
    tag_t sectioning_objects [ 2 ]
)
```

tag_t	**section_curve**	Input	The section curve to inquire.
tag_t *	**section_curves_feature**	Output	The section curves feature, or NULL_TAG, if none.
int *	**plane_type**	Output	The plane type: 0 for array of planes 1 for parallel planes

			2 for radial planes 3 for planes perpendicular to a curve.
tag_t *	**defining_object**	Output	The object sectioned.
tag_t	**sectioning_objects [2]**	Output	The sectioning objects For plane_type = 0: sectioning_objects[0] = plane eid sectioning_objects[1] = NULL_TAG For plane_type = 1: sectioning_objects[0] = base plane sectioning_objects[1] = NULL_TAG For plane_type = 2: sectioning_objects[0] = datum axis eid sectioning_objects[1] = point on plane For plane_type = 3: sectioning_objects[0] = curve eid sectioning_objects[1] = plane eid (NULL_TAG if none)

UF_CURVE_section_from_parallel_planes (view source)

Defined in: uf_curve.h

Overview

Creates an associative section curve feature or non-associative group of section curves using a set of parallel planes.

NOTE - If an object in the general data objects array is not fully loaded, the section curve will not be created for that object.

Environment

Internal and External

See Also

See UF_CURVE_section_from_planes for a list of related functions.

History

Originally released in V17.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_section_from_parallel_planes
(
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_parallel_data_p_t parallel_data,
    tag_t * section_curves
)
```

UF_CURVE_section_general_data_p_t	**general_data**	Input	Data provided by the user to to be

			used in creating the section curve feature.
UF_CURVE_section_parallel_data_p_t	**parallel_data**	Input	Data provided by the user describing the parallel planes.
tag_t *	**section_curves**	Output	Feature tag, if general_data->associate is 1, or Group tag of section curves if general_data->associate is 0 and general_data->grouping is 0, or Group tag of groups of curves if general_data->associate is 0 and general_data->grouping is 1

UF_CURVE_section_from_perpcrv_planes (view source)

Defined in: uf_curve.h

Overview

Creates an associative section curve feature or non-associative group of section curves using a set of planes perpendicular to a curve.

NOTE - If an object in the general data objects array is not fully loaded, the section curve will not be created for that object.

Environment

Internal and External

See Also

See UF_CURVE_section_from_planes for a list of related functions.

History

Originally released in V17.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_section_from_perpcrv_planes
(
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_perpcrv_data_p_t perpcrv_data,
    tag_t * section_curves
)
```

UF_CURVE_section_general_data_p_t	**general_data**	Input	Data provided by the user to to be used in creating the section curve feature.
UF_CURVE_section_perpcrv_data_p_t	**perpcrv_data**	Input	Data provided by the user describing the perpendicular curves.

tag_t *	**section_curves**	Output	Feature tag, if general_data->associate is 1, or Group tag of section curves if general_data->associate is 0 and general_data->grouping is 0, or Group tag of groups of curves if general_data->associate is 0 and general_data->grouping is 1

UF_CURVE_section_from_planes (view source)

Defined in: uf_curve.h

Overview

Creates an associative section curve feature or non-associative group of section curves using an array of planes.

NOTE - If an object in the general data objects array is not fully loaded, the section curve will not be created for that object.

See the example below for a sample execution of this function.

Environment

Internal and External

See Also

UF_CURVE_section_ask_planes_data
UF_CURVE_section_from_parallel_planes
UF_CURVE_section_ask_parallel_data
UF_CURVE_section_from_radial_planes
UF_CURVE_section_ask_radial_data
UF_CURVE_section_from_perpcrv_planes
UF_CURVE_section_ask_perpcrv_data
UF_CURVE_section_ask_type
UF_CURVE_section_curve_ask_parents
UF_CURVE_ask_feature_curves
UF_CURVE_create_int_object
UF_MODL_intersect_objects

Refer to the example

History

Originally released in V17.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_section_from_planes
(
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_planes_data_p_t planes_data,
    tag_t * section_curves
)
```

UF_CURVE_section_general_data_p_t	**general_data**	Input	Data provided by the user to to be

			used in creating the section curve feature.
UF_CURVE_section_planes_data_p_t	**planes_data**	Input	Sectioning planes and datum planes.
tag_t *	**section_curves**	Output	Feature tag, if general_data->associate is 1, or Group tag of section curves if general_data->associate is 0 and general_data->grouping is 0, or Group tag of groups of curves if general_data->associate is 0 and general_data->grouping is 1

UF_CURVE_section_from_radial_planes (view source)

Defined in: uf_curve.h

Overview

Creates an associative section curve feature or non-associative group of section curves using a set of radial planes.

NOTE - If an object in the general data objects array is not fully loaded, the section curve will not be created for that object.

Environment

Internal and External

See Also

See UF_CURVE_section_from_planes for a list of related functions.

History

Originally released in V17.0

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_section_from_radial_planes
(
    UF_CURVE_section_general_data_p_t general_data,
    UF_CURVE_section_radial_data_p_t radial_data,
    tag_t * section_curves
)
```

UF_CURVE_section_general_data_p_t	**general_data**	Input	Data provided by the user to to be used in creating the section curve feature.
UF_CURVE_section_radial_data_p_t	**radial_data**	Input	Data provided by the user describing the radial planes.
tag_t *	**section_curves**	Output	Feature tag, if general_data->associate is 1, or Group tag of section curves if

general_data->associate is 0
and general_data->grouping is 0, or
Group tag of groups of curves if
general_data->associate is 0
and general_data->grouping is 1

UF_CURVE_smooth_spline_data (view source)

Defined in: uf_curve.h

Overview

Makes a spline continuous to cont_order, if possible.

The order of continuity may be any positive integer up to
(but not including) the degree of the spline (the degree is order-1).

NOTE: cont_order values greater than 1 are not currently supported, but
may be specified.

Curves created interactively by NX, by creating a curve directly or
through modeling a solid body, do not need this function. Only if you
manage to create a curve with a corner, as by joining two splines, is
it possible to create a curve that would be affected by this function.

The spline must be C0 before calling this function, but it may have
degeneracies. C0 means continuous to order zero, which means that it
has no gaps. Splines that have end knots with a multiplicity greater than
the order of the spline, or have interior knots with a multiplicity greater
than the degree of the spline, may have gaps. Such splines are not of
general utility in CAD/CAM work. The creation function
(UF_CURVE_create_spline) will not create them and this function will
not smooth them. They can be fixed using UF_CURVE_fix_spline_data, and
then smoothed if necessary.

Specification of a given cont_order implies that all lower order
continuities are desired. The order of continuity is the
highest order derivative that is required to exist at all points of the
spline. At any point in the spline, if cont_order cannot be achieved,
the highest continuity within tolerance is achieved. If C1 cannot be
achieved, G1 is achieved within tolerance and a state code is returned.
If G1 cannot be achieved a state code is returned. Whatever can be
smoothed within tolerance is returned even if the entire spline cannot
be done. States are returned for all conditions found in the spline.

Any knot that has a multiplicity high enough to make it possible for
the spline to be discontinuous to the given order at the corresponding
pole is checked. If the pole can be treated in such a way as to make it
continuous to the given order without changing the shape more than
toler, it is made so.

Think of discontinuities as 'corners'. A discontinuity in the first
derivative is what one most often thinks of as a corner; it is readily
visualized. A discontinuity in the second derivative is similar to a
discontinuity in the first derivative, that is, it is a corner, but it
is a corner in the first derivative. Basically, a knot of multiplicity
order-n means the curve is potentially discontinuous in the nth derivative.
It now becomes apparent what C0 comes from: it means continuity in the
'zero-th derivative', referring to the condition introduced by a knot of
multiplicity order (or greater), which is a potential gap in the curve.
Open curves, and some closed curves, have knots of multiplicity order at

the ends, where they are not C0, because, naturally, they stop.

If the spline is closed the closure is also checked and smoothed if within tolerance.

Continuity is achieved using a mathematical process called knot removal. Geometric continuity (G1) is achieved by moving the poles.

Use this function conservatively. In particular, if simplification is to be done when the geometry is attached or edited on a body topology, be aware that smoothing may make the spline's equivalent curve (such as a circle) unrecognizable to the simplifier. One of two things might happen:

1. The curve may be modified in its point and knot content so as not to match the patterns that the simplifier uses to recognize simplifiable curves, or,

2. If the smoothing tolerance is large enough, it may be modified so that it is out of matching tolerance on the possible simplified curve.

Consult a textbook on NURBS curves (NX splines are NURBS curves) for more information on continuity and the interpretaion of the knot vector. One such book is "The NURBS Book," by Piegl and Tiller, Springer Verlag, 1995. The first 100 pages provide a good grounding.

Environment

Internal and External

See Also

UF_BREP_attach_geometry
UF_CURVE_smooth_spline_data_st
UF_CURVE_fix_spline_data
UF_CURVE_create_spline

History

This function was originally released in V15.0.

Required License(s)

(solid_modeling or drafting)

```
int UF_CURVE_smooth_spline_data
(
    UF_CURVE_spline_p_t spline_data,
    int cont_order,
    double distance_toler,
    double angle_toler,
    int * num_states,
    UF_CURVE_state_p_t * states
)
```

UF_CURVE_spline_p_t	**spline_data**	Input / Output	spline data to make continuous
int	**cont_order**	Input	order of continuity to achieve
double	**distance_toler**	Input	tolerance; smoothing is done if the deviation in shape introduced is less than this amout. Zero indicates use NX modelling resolution.
double	**angle_toler**	Input	tolerance; smoothing is done if the angle between tangents at the joint is less than this

			amount. Zero indicates use NX modelling resolution.
int *	**num_states**	Output	number of states in returned states array
UF_CURVE_state_p_t *	**states**	Output to UF_*free*	Pointer to array pointer for joints where continuity could not be achieved within tolerance. This must be freed by calling UF_free. May also contain knot fixup states.

UF_CURVE_smooth_spline_data_st (view source)

Defined in: uf_curve.h

Overview

See the documentation for UF_CURVE_smooth_spline_data for a description of smoothing. UF_CURVE_smooth_spline_data_st differs from it only in its tolerance argument, dist_toler and in the interpretation of the continuity state distance value.

3D curves are isotropic, that is, they occupy space with the same "density" in all three directions. If the curves you are smoothing are always isotropic, use UF_CURVE_smooth_spline_data.

UF_CURVE_smooth_spline_data_st smooths using a shear tolerance specification, meaning that it may have different values for x, y, and z. Use it for smoothing bcurves destined to become fin curves, which are used to trim surfaces. (See UF_BREP).

Fin curves are defined in the surface parameter, or SP, space, of some surface and thus are often called SP curves. Their existence in 3-space is a composition of their two-dimensional definition and the 3-dimensional existence of the surface. An SP curve defined on a surface has u,v coordinates, with the third coordinate (normally z) being zero, instead of x, y, z coordinates. In specifying the coordinates of the bcurves vertices, and the tolerances for smoothing, the x coordinate is the u value and the y coordinate is the v value.

SP curves for trimming surfaces are anisotropic, that is, not isotropic. They do not occupy space with the same "density" in both directions. Consider a cylinder, which has a parameterization of part units (mm or in) in u (along the axis) and angular units (radians) in v (around the axis). Thus u has the same dimensions as 3-space, but v does not, indeed the v parameter being angular causes a dependence on the radius for the correspondence between the v tolerance in 3D and in parameter space.

Suppose we have a cylinder of radius 10mm, and we want changes from smoothing restricted to .02mm. dist_toler[0] = .02 is the tolerance for u.
dist_toler[1] = .02/10 is the tolerance for v (it is not an approximation).
A curve along the surface in v is a circle, and so the distance along the curve (the 3D tolerance) is equal to the radius times the angle (the v tolerance). Working the algebra leads to the expression for dist_toler[1]. Similar methods work for other analytic surface types. Generally a method that yields a smaller rather than larger tolerance is preferable, like finding some approximation to the largest radius of a surface of revolution, or the radius of a cone at the larger part of where the face occupies the cone.

For a bsurface (which is anisotropic except in some very special cases), it may be necessary to select a u,v pair, get a point, offset the u,v pair by a small amount, get another point, and work the proportions to obtain the

u and v deltas that correspond to the desired 3D tolerance. Surfaces can vary in their parameterization, and that affects the tolerances obtained. You have to gain some knowledge about your surfaces; surfaces made by NX dont contain variations in parameterization from one part of the surface to another, so selecting 0.5, 0.5 as the evaluation point should be sufficient.

Note: cont_order values greater than 1 are not currently supported, but may be specified.

Environment
Internal and External

See Also
UF_BREP_attach_geometry
UF_CURVE_smooth_spline_data
UF_CURVE_fix_spline_data
UF_CURVE_create_spline

Required License(s)
(solid_modeling or drafting)

```
int UF_CURVE_smooth_spline_data_st
(
    UF_CURVE_spline_p_t spline_data,
    int cont_order,
    double dist_toler [ 3 ] ,
    double ang_toler,
    int * num_states,
    UF_CURVE_state_p_t * states
)
```

UF_CURVE_spline_p_t	**spline_data**	Input / Output	spline to smooth
int	**cont_order**	Input	order of continuity to get
double	**dist_toler [3]**	Input	dont exceed this, x, y, z
double	**ang_toler**	Input	dont exceed this
int *	**num_states**	Output	number of states in states array
UF_CURVE_state_p_t *	**states**	Output to UF_*free*	states found during smoothing. The caller is responsible for freeing this by calling UF_free.

UF_MODL_ask_curve_fit_data (view source)

Defined in: uf_curve.h

Overview
Ask curve fit data: curve fit method, maximum degree and maximum number of segments from the modeling preferences

Environment

Internal and External

See Also
UF_MODL_set_curve_fit_data

History
This function is introduced in NX3.0.

Required License(s)
gateway

int UF_MODL_ask_curve_fit_data
(
UF_MODL_curve_fit_data * curve_fit_data
)

UF_MODL_curve_fit_data *	**curve_fit_data**	Output	curve fit method, maximum degree, and maximum segments

---

UF_MODL_set_curve_fit_data (view source)

Defined in: uf_curve.h

Overview
Set curve fit data: curve fit method, maximum degree and maximum number of segments to the modeling preferences

Environment
Internal and External

See Also
UF_MODL_ask_curve_fit_data

History
This function is introduced in NX3.0.

Required License(s)
solid_modeling

int UF_MODL_set_curve_fit_data
(
UF_MODL_curve_fit_data curve_fit_data
)

UF_MODL_curve_fit_data	**curve_fit_data**	Input	curve fit method, maximum degree, and maximum segments

---