UF_EVAL_ask_arc (view source)

Defined in: uf_eval.h

Overview
Returns the arc data of the evaluator of a circular curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_arc

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_arc
(
    UF_EVAL_p_t evaluator,
    UF_EVAL_arc_p_t arc
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be an arc, or an error will be returned.
UF_EVAL_arc_p_t	**arc**	Output	arc data

UF_EVAL_ask_ellipse (view source)

Defined in: uf_eval.h

Overview
Returns the ellipse data of the evaluator of an elliptical curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_ellipse

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_ellipse
(
        UF_EVAL_p_t evaluator,
        UF_EVAL_ellipse_p_t ellipse
    )
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be an ellipse, or an error will be returned.
UF_EVAL_ellipse_p_t	**ellipse**	Output	ellipse data

UF_EVAL_ask_hyperbola (view source)

Defined in: uf_eval.h

Overview
Returns the hyperbola data of the evaluator of a hyperbolic curve.
There are no hyperbolic edges in NX.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_hyperbola

History
Originally released in V15.0

Required License(s)
gateway

```
    int UF_EVAL_ask_hyperbola
    (
        UF_EVAL_p_t evaluator,
        UF_EVAL_hyperbola_p_t hyperbola
    )
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be a hyperbola, or an error will be returned.
UF_EVAL_hyperbola_p_t	**hyperbola**	Output	hyperbola data

UF_EVAL_ask_limits (view source)

Defined in: uf_eval.h

Overview
Returns the curve limits of the evaluator of a curve or edge.

Environment

Internal and External

See Also

UF_EVAL_initialize

History

Originally released in V15.0

Required License(s)

gateway

int UF_EVAL_ask_limits
(
UF_EVAL_p_t evaluator,
double limits [2]
)

UF_EVAL_p_t	**evaluator**	Input	evaluator structure
double	**limits [2]**	Output	limits

UF_EVAL_ask_line (view source)

Defined in: uf_eval.h

Overview

Returns the line data of the evaluator of a linear curve or edge

Environment

Internal and External

See Also

UF_EVAL_initialize
UF_EVAL_is_line

History

Originally released in V15.0

Required License(s)

gateway

int UF_EVAL_ask_line
(
UF_EVAL_p_t evaluator,
UF_EVAL_line_p_t line
)

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve. This must be a line, or an error will be returned.
UF_EVAL_line_p_t	**line**	Output	The data for this line.

UF_EVAL_ask_parabola (view source)

Defined in: uf_eval.h

Overview
Returns the parabola data from the evaluator of a parabolic curve.
There are no parabolic edges in NX.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_parabola

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_parabola
(
    UF_EVAL_p_t evaluator,
    UF_EVAL_parabola_p_t parabola
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be a parabola, or an error will be returned.
UF_EVAL_parabola_p_t	**parabola**	Output	parabola data

UF_EVAL_ask_spline (view source)

Defined in: uf_eval.h

Overview
Returns the spline data of the evaluator of a spline curve or edge

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_spline
UF_EVAL_ask_spline_control_pts
UF_EVAL_ask_spline_knots

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_spline
(
    UF_EVAL_p_t evaluator,
    UF_EVAL_spline_p_t spline
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be a spline, or an error will be returned.
UF_EVAL_spline_p_t	**spline**	Output	spline data

UF_EVAL_ask_spline_control_pts (view source)

Defined in: uf_eval.h

Overview
Returns the spline control points of the evaluator of a spline curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_spline
UF_EVAL_ask_spline
UF_EVAL_ask_spline_knots

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_spline_control_pts
(
    UF_EVAL_p_t evaluator,
    int * n_points,
    double * * points
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be a spline, or an error will be returned.
int *	**n_points**	Output	count of control points
double * *	**points**	Output to UF_*free*	For each control point, four values are returned, (wx, wy, xz, w). This array must be freed by calling UF_free.

UF_EVAL_ask_spline_knots (view source)

Defined in: uf_eval.h

Overview
Returns the spline knots of the evaluator of a spline curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_is_spline
UF_EVAL_ask_spline
UF_EVAL_ask_spline_control_pts

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_ask_spline_knots
(
    UF_EVAL_p_t evaluator,
    int * n_knots,
    double * * knots
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve. This must be a spline, or an error will be returned.
int *	**n_knots**	Output	count of knots in sequence
double * *	**knots**	Output to UF_*free*	knot sequence points, the parameter values for each knot point in the spline. This array must be freed by calling UF_free. These parameters are not normalized.

UF_EVAL_copy (view source)

Defined in: uf_eval.h

Overview
Copies the evaluator of an curve or edge.
Return copy of evaluator structure.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_free

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_copy
(
    UF_EVAL_p_t evaluator,
    UF_EVAL_p_t * evaluator_copy
)
```

UF_EVAL_p_t	**evaluator**	Input	Evaluator structure to copy
UF_EVAL_p_t *	**evaluator_copy**	Output to UF_*free*	Copy of the evaluator structure. This must be freed by calling UF_EVAL_free.

UF_EVAL_evaluate (view source)

Defined in: uf_eval.h

Overview
Evaluates a point or the derivatives of an evaluator of a curve or edge

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_limits

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_evaluate
(
    UF_EVAL_p_t evaluator,
    int n_derivatives,
    double parm,
    double point [ 3 ] ,
    double derivatives [ ]
)
```

UF_EVAL_p_t	**evaluator**	Input	Evaluator structure for the curve.
int	**n_derivatives**	Input	number of derivative vectors

double	**parm**	Input	Parameter to evaluate the curve at. This parameter is not normalized. The parameter limits for a given curve can be found by calling UF_EVAL_ask_limits.
double	**point [3]**	Output	point on curve
double	**derivatives [ ]**	Output	Derivative vectors on the curve. The caller is responsible for providing space to return 3n_derivatives real numbers.

UF_EVAL_evaluate_closest_point (view source)

Defined in: uf_eval.h

Overview
Returns the closest parameter point on the curve or edge given a specified reference point. Note that parameters returned are not normalized.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_limits
UF_MODL_ask_point_along_curve_2 -- Handles curves with sharp corners better

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_evaluate_closest_point
(
    UF_EVAL_p_t evaluator,
    const double reference_point [ 3 ] ,
    double * parm,
    double closest_point [ 3 ]
)
```

UF_EVAL_p_t	**evaluator**	Input	The evaluator structure of the curve.
const double	**reference_point [3]**	Input	The reference point
double *	**parm**	Output	The parameter of the closest point on the curve to the reference point. This parameter is not normalized.
double	**closest_point [3]**	Output	The closest point on the curve to the reference point.

UF_EVAL_evaluate_unit_vectors (view source)

Defined in: uf_eval.h

Overview
Evaluates a point or the unit vectors of an evaluator of a curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_limits

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_evaluate_unit_vectors
(
    UF_EVAL_p_t evaluator,
    double parm,
    double point [ 3 ] ,
    double tangent [ 3 ] ,
    double normal [ 3 ] ,
    double binormal [ 3 ]
)
```

UF_EVAL_p_t	**evaluator**	Input	The evaluator structure for the curve.
double	**parm**	Input	The parameter to evaluate at. Note that the parameter is not normalized. You can get the parameters for the given evaluator structure by calling UF_EVAL_ask_limits.
double	**point [3]**	Output	The point on the curve
double	**tangent [3]**	Output	The tangent to the curve at that point.
double	**normal [3]**	Output	The normal to the curve at that point.
double	**binormal [3]**	Output	The binormal to the curve at that point.

UF_EVAL_free (view source)

Defined in: uf_eval.h

Overview
Frees the evaluator of a curve or edge.

Environment

Internal and External

See Also

UF_EVAL_initialize
UF_EVAL_copy

History

Originally released in V15.0

Required License(s)

gateway

```
int UF_EVAL_free
(
    UF_EVAL_p_t evaluator
)
```

UF_EVAL_p_t	**evaluator**	Input	evalauator structure to be freed

---

UF_EVAL_initialize (view source)

Defined in: uf_eval.h

Overview

Initialize an evaluator structure.

Returns a pointer to the evaluator of a curve or edge. The evaluator can be used to find points, derivatives and vectors of a curve or edge at a given parameter value. No longer use UF_CURVE_ask_curve_struct ( ), UF_CURVE_ask_curve_struct_data ( ) and UF_CURVE_free_curve_struct ( ).

Note that this API will approximate procedurally created edges (such as edges created by intersecting two surfaces) with a spline to the current Modeling distance tolerance.

Environment

Internal and External

See Also

UF_EVAL_initialize_2
UF_EVAL_free

History

Originally released in V15.0

Required License(s)

gateway

```
int UF_EVAL_initialize
(
    tag_t tag,
    UF_EVAL_p_t * evaluator
)
```

tag_t	**tag**	Input	Object identifier of a curve or edge
UF_EVAL_p_t *	**evaluator**	Output to UF_*free*	Evaluator structure for this curve. It must be freed by calling UF_EVAL_free.

UF_EVAL_initialize_2 (view source)

Defined in: uf_eval.h

Overview

Initialize an evaluator structure.

Returns a pointer to the evaluator of a curve or edge. The evaluator
can be used to find points, derivatives and vectors of a curve or edge
at a given parameter value. No longer use
UF_CURVE_ask_curve_struct ( ),
UF_CURVE_ask_curve_struct_data ( ) and
UF_CURVE_free_curve_struct ( ).

This API will not approximate procedurally created edges.

Environment

Internal and External

See Also

UF_EVAL_initialize
UF_EVAL_free

History

Originally released in NX7.0

Required License(s)

gateway

```
int UF_EVAL_initialize_2
(
    tag_t tag,
    UF_EVAL_p_t * evaluator
)
```

tag_t	**tag**	Input	Object identifier of a curve or edge
UF_EVAL_p_t *	**evaluator**	Output to UF_*free*	Evaluator structure for this curve. It must be freed by calling UF_EVAL_free.

UF_EVAL_is_arc (view source)

Defined in: uf_eval.h

Overview

Determines if the given evaluator is from a circular curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_arc

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_is_arc
(
    UF_EVAL_p_t evaluator,
    logical * is_arc
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve
logical *	**is_arc**	Output	TRUE if this is an arc, else FALSE

UF_EVAL_is_ellipse (view source)

Defined in: uf_eval.h

Overview
Determines if the evaluator is from an elliptical curve or edge

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_ellipse

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_is_ellipse
(
    UF_EVAL_p_t evaluator,
    logical * is_ellipse
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve.

logical *	**is_ellipse**	Output	TRUE if this is an ellipse, else FALSE

UF_EVAL_is_equal (view source)

Defined in: uf_eval.h

Overview
Determines if two evaluators are equal. This implies that the curve(s) and/or edge(s) are equal.

Environment
Internal and External

See Also
UF_EVAL_initialize

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_is_equal
(
    UF_EVAL_p_t evaluator1,
    UF_EVAL_p_t evaluator2,
    logical * is_equal
)
```

UF_EVAL_p_t	**evaluator1**	Input	evaluator structure for the first curve
UF_EVAL_p_t	**evaluator2**	Input	evaluator structure for the second curve
logical *	**is_equal**	Output	TRUE if the curves are equal, else FALSE

UF_EVAL_is_hyperbola (view source)

Defined in: uf_eval.h

Overview
Determines if the evaluator is from a hyperbolic curve. There are no hyperbolic edges in NX.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_hyperbola

History

Originally released in V15.0

Required License(s)

gateway

```
int UF_EVAL_is_hyperbola
(
    UF_EVAL_p_t evaluator,
    logical * is_hyperbola
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve.
logical *	**is_hyperbola**	Output	TRUE if the curve is a hyperbola, else FALSE

UF_EVAL_is_line (view source)

Defined in: uf_eval.h

Overview

Determines if the evaluator is from a linear curve or edge.

Environment

Internal and External

See Also

UF_EVAL_initialize
UF_EVAL_ask_line

History

Originally released in V15.0

Required License(s)

gateway

```
int UF_EVAL_is_line
(
    UF_EVAL_p_t evaluator,
    logical * is_line
)
```

UF_EVAL_p_t	**evaluator**	Input	The evaluator structure of the curve.
logical *	**is_line**	Output	TRUE if this is a line, else FALSE

UF_EVAL_is_parabola (view source)

Defined in: uf_eval.h

Overview
Determines if the evaluator is from a parabolic curve. There are no parabolic edges in NX.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_parabola

History
Originally released in V15.0

Required License(s)
gateway

int UF_EVAL_is_parabola
(
UF_EVAL_p_t evaluator,
logical * is_parabola
)

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve
logical *	**is_parabola**	Output	TRUE if the curve is a parabola, else FALSE

UF_EVAL_is_periodic (view source)

Defined in: uf_eval.h

Overview
Query if the input curve is periodic.
Determines if the evaluator is from a periodic curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize

History
Originally released in V15.0

Required License(s)
gateway

int UF_EVAL_is_periodic
(
UF_EVAL_p_t evaluator,
logical * is_periodic
)

UF_EVAL_p_t	**evaluator**	Input	evaluator structure for the curve.
logical *	**is_periodic**	Output	TRUE if the curve is periodic, else FALSE

UF_EVAL_is_spline (view source)

Defined in: uf_eval.h

Overview
Determines if the evaluator is from a spline curve or edge.

Environment
Internal and External

See Also
UF_EVAL_initialize
UF_EVAL_ask_spline
UF_EVAL_ask_spline_control_pts
UF_EVAL_ask_spline_knots

History
Originally released in V15.0

Required License(s)
gateway

```
int UF_EVAL_is_spline
(
    UF_EVAL_p_t evaluator,
    logical * is_spline
)
```

UF_EVAL_p_t	**evaluator**	Input	evaluator structure of the curve
logical *	**is_spline**	Output	TRUE if the curve is a spline, else FALSE

UF_EVALSF_ask_face_uv_minmax (view source)

Defined in: uf_evalsf.h

Overview
Computes the u,v parameter space min, max of a face

Environment
Internal and External

Required License(s)
gateway

```
int UF_EVALSF_ask_face_uv_minmax
(
    UF_EVALSF_pc_t evaluator,
    double uv_min_max [ 4 ]
)
```

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
double	**uv_min_max [4]**	Output	[0] - umin [1] - umax [2] - vmin [3] - vmax

---

UF_EVALSF_ask_minimum_face_dist (view source)

Defined in: uf_evalsf.h

Overview

Returns the minimum distance parameter and point on the surface of a face given a reference point.

The surface point is on the portion which is bounded by the edges of the face.

To find the closest point on the extended surface (not within bounding edges) use UF_EVALSF_find_closest_point

Environment

Internal and External

Required License(s)

gateway

```
int UF_EVALSF_ask_minimum_face_dist
(
    UF_EVALSF_pc_t evaluator,
    const double point [ 3 ] ,
    UF_EVALSF_pos3_p_t srf_pos3
)
```

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
const double	**point [3]**	Input	Given point in absolute space
UF_EVALSF_pos3_p_t	**srf_pos3**	Output	Address of point position on surface

---

UF_EVALSF_evaluate (view source)

Defined in: uf_evalsf.h

Overview

Evaluates a point and the derivatives of a face at the given parameter value.

Environment

Internal and External

See Also

UF_MODL_evaluate_face and UF_MODL_ask_face_props
Note that UF_EVALSF_evaluate may not properly handle parameter values returned by UF_EVALSF_find_closest_point or similar functions for surface types Blend or Trimmed B-surface. The above functions should be called instead.

Required License(s)

gateway

```
int UF_EVALSF_evaluate
(
    UF_EVALSF_pc_t evaluator,
    int deriv_flag,
    const double uv_pair [ 2 ] ,
    UF_MODL_SRF_VALUE_p_t surf_eval
)
```

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
int	**deriv_flag**	Input	order of the derivative to be computed: UF_MODL_EVAL - position UF_MODL_EVAL_DERIV1 - position and 1. partial UF_MODL_EVAL_DERIV2 - position, 1. and 2. partials UF_MODL_EVAL_DERIV3 - position, 1., 2. and 3. partials UF_MODL_EVAL_UNIT_NORMAL - position, 1. partials and unitized normal. UF_MODL_EVAL_NORMAL - position, 1. partials and the ununitized normal. UF_MODL_EVAL_ALL - position, normals and all the partials up to the third order.
const double	**uv_pair [2]**	Input	uv-parameter pair at which derivatives are to be computed. The parameter limits for a given face can be found by calling UF_EVALSF_ask_face_uv_minmax.
UF_MODL_SRF_VALUE_p_t	**surf_eval**	Output	Address of evaluation result structure containing position and derivatives: srf_pos[3] - position srf_du[3] - d/du srf_dv[3] - d/dv srf_unormal[3] - unit normal srf_d2u[3] - d2/du2 srf_dudv[3] - d2/dudv srf_d2v[3] - d2/dv2 srf_d3u[3] - d3/du3 srf_d2udv[3] - d3/du2dv srf_dud2v[3] - d3/dud2v srf_d3v[3] - d3/dv3 srf_normal[3] - d/du X d/dv

UF_EVALSF_evaluate_array (view source)

Defined in: uf_evalsf.h

Overview
Evaluates an array of points and derivatives of a face for a given array of parameter values.

Environment
Internal and External

See Also
UF_MODL_evaluate_face and UF_MODL_ask_face_props
Note that UF_EVALSF_evaluate_array may not properly handle parameter values
returned by UF_EVALSF_find_closest_point or similar functions for
surface types Blend or Trimmed B-surface. The above functions should be
called instead.

Required License(s)
gateway

```
int UF_EVALSF_evaluate_array
(
    UF_EVALSF_pc_t evaluator,
    int deriv_flag,
    int num_points,
    const double uv_pairs [ ] ,
    UF_MODL_SRF_VALUE_t surf_evals [ ]
)
```

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
int	**deriv_flag**	Input	order of the derivative to be computed: UF_MODL_EVAL - position UF_MODL_EVAL_DERIV1 - position and 1. partial UF_MODL_EVAL_DERIV2 - position, 1. and 2. partials UF_MODL_EVAL_DERIV3 - position, 1., 2. and 3. partials UF_MODL_EVAL_UNIT_NORMAL - position, 1. partials and unitized normal. UF_MODL_EVAL_NORMAL - position, 1. partials and the ununitized normal. UF_MODL_EVAL_ALL - position, normals and all the partials up to the third order.
int	**num_points**	Input	number of points to evaluate
const double	**uv_pairs [ ]**	Input	num_points uv-parameter pairs The parameter limits for a given face can be found by calling UF_EVALSF_ask_face_uv_minmax.
UF_MODL_SRF_VALUE_t	**surf_evals [ ]**	Output	Address of array of evaluation results The caller must provide space of size = sizeof(UF_MODL_SRF_VALUE_t) num_points Each structure element contains: srf_pos[3] - position srf_du[3] - d/du

srf_dv[3] - d/dv
srf_unormal[3] - unit normal
srf_d2u[3] - d2/du2
srf_dudv[3] - d2/dudv
srf_d2v[3] - d2/dv2
srf_d3u[3] - d3/du3
srf_d2udv[3] - d3/du2dv
srf_dud2v[3] - d3/dud2v
srf_d3v[3] - d3/dv3
srf_normal[3] - d/du X d/dv

UF_EVALSF_find_closest_point (view source)

Defined in: uf_evalsf.h

Overview

Finds on the surface the closest point and uv-parameter pair to a given point.

Note that the returned point will be on the underlying surface
but may not be on the portion which is bounded by the edges
of the face.

Note that in some cases, there is a potential risk that the returned point might
be adjusted incorrectly where the underlying surface has an extended UV range.
Please use UF_EVALSF_find_closest_point_2 instead.

To find the closest point within bounding edges, use
UF_EVALSF_ask_minimum_face_dist

Environment

Internal and External

Required License(s)

gateway

```
int UF_EVALSF_find_closest_point
(
    UF_EVALSF_pc_t evaluator,
    const double point [ 3 ] ,
    UF_EVALSF_pos3_p_t srf_pos3
)
```

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
const double	**point [3]**	Input	Given point in absolute space
UF_EVALSF_pos3_p_t	**srf_pos3**	Output	Address of point position on surface

---

UF_EVALSF_find_closest_point_2 (view source)

Defined in: uf_evalsf.h

Overview

Finds on the surface the closest point and uv-parameter pair to a given point.

Note that the returned point will be on the underlying surface
but may not be on the portion which is bounded by the edges
of the face.

To find the closest point within bounding edges, use
UF_EVALSF_ask_minimum_face_dist

Environment
Internal and External

Required License(s)
gateway

int UF_EVALSF_find_closest_point_2
(
UF_EVALSF_pc_t evaluator,
const double point [3] ,
UF_EVALSF_pos3_p_t srf_pos3
)

UF_EVALSF_pc_t	**evaluator**	Input	Address of evaluator structure
const double	**point [3]**	Input	Given point in absolute space
UF_EVALSF_pos3_p_t	**srf_pos3**	Output	Address of point position on surface

---

UF_EVALSF_free (view source)

Defined in: uf_evalsf.h

Overview
Frees the evaluator of a face

Environment
Internal and External

Required License(s)
gateway

int UF_EVALSF_free
(
UF_EVALSF_p_t * evaluator
)

UF_EVALSF_p_t *	**evaluator**	Output	address of evaluator pointer

---

UF_EVALSF_initialize (view source)

Defined in: uf_evalsf.h

Overview

Initializes a face evaluator structure

Environment

Internal and External

Required License(s)

gateway

```
int UF_EVALSF_initialize
(
    tag_t face_tag,
    UF_EVALSF_p_t * evaluator
)
```

tag_t	**face_tag**	Input	Object identifier of a face
UF_EVALSF_p_t *	**evaluator**	Output to UF_*free*	Evaluator structure for this face. It must be freed by calling UF_EVALFS_free.

UF_EVALSF_initialize_2 (view source)

Defined in: uf_evalsf.h

Overview

Initializes a face evaluator structure with correct parameter mapping

Environment

Internal and External

Required License(s)

gateway

```
int UF_EVALSF_initialize_2
(
    tag_t face_tag,
    UF_EVALSF_p_t * evaluator
)
```

tag_t	**face_tag**	Input	Object identifier of a face
UF_EVALSF_p_t *	**evaluator**	Output to UF_*free*	Evaluator structure for this face. It must be freed by calling UF_EVALFS_free.
