UF_SF_add_material_fatigue (view source)

Defined in: uf_sf.h

Overview

UF_SF_add_material_fatigue:

Add fatigue properties to the material.

int UF_SF_add_material_fatigue
(
UF_SF_material_fatigue_prop_p_t property_values,
tag_t material_tag
)

UF_SF_material_fatigue_prop_p_t	**property_values**	Input	Structure containing material strength properties. The structure should be allocated by the user, but data within the structure is dynamically allocated. Call html#UF_SF_free_matl_strength_prop to free the storage allocated within the structure.
tag_t	**material_tag**	Input	Tag of the material.

UF_SF_add_material_strength (view source)

Defined in: uf_sf.h

Overview

UF_SF_add_material_strength:

Add plastic range properties to the material.

int UF_SF_add_material_strength
(
UF_SF_material_strength_prop_p_t property_values,
tag_t material_tag
)

UF_SF_material_strength_prop_p_t	**property_values**	Input	Structure containing material strength properties. The structure should be allocated by the user, but data within the structure is dynamically allocated. Call html#UF_SF_free_matl_strength_prop to free the storage allocated within the structure.
tag_t	**material_tag**	Input	Tag of the material.

UF_SF_add_to_solution_nx (view source)

Defined in: uf_sf.h

Overview

Adds loads or boundary conditions to the active solution. This function needs to be called after creating load or boundary condition.

Environment

Internal and External

History
Originally released in NX3.0

int UF_SF_add_to_solution_nx
(
int num_of_lbc,
tag_t * lbc
)

int	**num_of_lbc**	Input	Number of loads or boundary conditions to add to the active solution.
tag_t *	**lbc**	Input	pointer to array of tags for loads or boundary conditions.

UF_SF_add_to_step_nx (view source)

Defined in: uf_sf.h

Overview
Adds loads or boundary conditions to the active step. This function needs to be called after creating load or boundary condition.

Functions to return load tags are UF_SF_locate_solution_loads_nx and UF_SF_locate_step_loads_nx.

Environment
Internal and External

History
Originally released in NX3.0

int UF_SF_add_to_step_nx
(
int num_of_lbc,
tag_t * lbc
)

int	**num_of_lbc**	Input	Number of loads or boundary conditions to add to the active step.
tag_t *	**lbc**	Input	Pointer to array of tags for loads or boundary conditions.

UF_SF_alloc_defeature_parms (view source)

Defined in: uf_sf.h

Overview
This routine allocates an instance of UF_SF_defeature_parms_t structure used to hold all information needed to create a defeature feature.

The structure is allocated and the retained_faces, removed_faces fields are set from the corresponding input parameters.
The region substructure will be allocated and initialized to the values of the input parameters that define the region.

Environment
Internal and External

History

Originally released in NX3.0

```
int UF_SF_alloc_defeature_parms
(
    int num_ret_faces,
    tag_t * ret_faces_a,
    int num_rem_faces,
    tag_t * rem_faces_a,
    tag_t seed_face,
    int num_bnd_faces,
    tag_t * bnd_faces,
    tag_t angle_exp,
    UF_SF_defeature_parms_p_t * defeature_parms_p
)
```

int	**num_ret_faces**	Input	Number of retained faces in the retained_faces array. Can be 0
tag_t *	**ret_faces_a**	Input	Array of retained faces. Can be NULL if num_bnd_faces == 0
int	**num_rem_faces**	Input	Number of removed faces in the removed_faces array. Can be 0
tag_t *	**rem_faces_a**	Input	Array of retained faces. Can be NULL if num_bnd_faces == 0
tag_t	**seed_face**	Input	Seed face for the region (required to derive the body)
int	**num_bnd_faces**	Input	Number of boundary faces in the bnd_faces array. Can be 0
tag_t *	**bnd_faces**	Input	Array of boundary faces. Can be NULL if num_bnd_faces == 0
tag_t	**angle_exp**	Input	Expression to be used for the tagential edge angle which is used for the region boundary, Can be NULL_TAG
UF_SF_defeature_parms_p_t *	**defeature_parms_p**	Output to UF_*free*	Idealize region entity created. Use UF_SF_free_defeature_parms to free.

UF_SF_alloc_idealize_parms (view source)

Defined in: uf_sf.h

Overview

This routine allocates a parameter object for an idealize feature

The parameter object is used to hold all information needed to create an idealize feature. All expressions associated with this object will be initialized to NULL_TAG. All face sets associated with this object will be initalize to contain 0 faces

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_alloc_idealize_parms
(
    UF_SF_idealize_parms_p_t * parms_p
)
```

UF_SF_idealize_parms_p_t *	**parms_p**	Output to UF_*free*	Allocated idealize parameter entity Use UF_SF_free_idealize_parms to free memory associated to this entity

---

UF_SF_alloc_idealize_region (view source)

Defined in: uf_sf.h

Overview

This routine allocates a region object for an idealize feature

The region object is used to hold all information needed to create an idealize feature associated to a body region.

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_alloc_idealize_region
(
    tag_t seed_face,
    int num_bnd_faces,
    tag_t * bnd_faces,
    tag_t angle_exp,
    UF_SF_idealize_region_p_t * region_p
)
```

tag_t	**seed_face**	Input	Seed face for the region (required)
int	**num_bnd_faces**	Input	Number of boundary faces in the bnd_faces array. Can be 0
tag_t *	**bnd_faces**	Input	Array of boundary faces. Can be NULL if num_bnd_faces == 0
tag_t	**angle_exp**	Input	Expression to be used for the tagential edge angle which is used for the region boundary, Can be NULL_TAG
UF_SF_idealize_region_p_t *	**region_p**	Output to UF_*free*	Idealize region entity created Use UF_SF_free_idealize_region to free

---

UF_SF_apply_beam_end_mass (view source)

Defined in: uf_sf.h

Overview

Applies end mass values to entities selected.

Environment

Internal and External

```
int UF_SF_apply_beam_end_mass
(
    char* end_a_nsm_exp,
    char* end_b_nsm_exp,
    int num_items,
    tag_p_t items_array
```

```
)
```

char*	**end_a_nsm_exp**	Input	Beam end A mass expression
char*	**end_b_nsm_exp**	Input	Beam end B mass expression
int	**num_items**	Input	Number of items to associate mass.
tag_p_t	**items_array**	Input	Tags of items to associate end mass Items can be edges or 1D mesh

---

UF_SF_ask_active_solution_nx (view source)

Defined in: uf_sf.h

Overview

Returns the tag of the active solution.

Valid only when the work part is a Simulation.

Use UF_SF_ask_num_solutions_nx, UF_SF_ask_nth_solution_nx or UF_SF_solution_ask_name_nx to see what solutions are in the part. A call to UF_SF_set_active_solution_and_step_nx will make the solution active.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_ask_active_solution_nx
(
    tag_t * active_solution
)
```

tag_t *	**active_solution**	Output	Pointer to the tag of the active solution for the scenario part.

UF_SF_ask_active_step_nx (view source)

Defined in: uf_sf.h

Overview

Returns the tag of the active step.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_ask_active_step_nx
(
    tag_t * active_step
)
```

tag_t *	**active_step**	Output	Pointer to the tag of the active step for the scenario part.

UF_SF_ask_all_polygon_bodies (view source)

Defined in: uf_sf.h

Overview
This function will get all polygon bodies in the part.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_ask_all_polygon_bodies
(
    int * num_polygon_bodies,
    tag_t * * polygon_bodies
)
```

int *	**num_polygon_bodies**	Output	Number of polygon bodies in the fem
tag_t * *	**polygon_bodies**	Output to UF_*free*	Tags of polygon bodies

---

UF_SF_ask_beam_end_mass (view source)

Defined in: uf_sf.h

Overview
Given a mesh tag, this function gets end A and end B mass values associated with the object.

Environment
Internal and External

```
int UF_SF_ask_beam_end_mass
(
    tag_t object_tag,
    double* end_a_nsm,
    double* end_b_nsm
)
```

tag_t	**object_tag**	Input	Tag to mesh or mesh recipe
double*	**end_a_nsm**	Output	End A nonstructural mass value
double*	**end_b_nsm**	Output	End B nonstructural mass value

UF_SF_ask_closest_point (view source)

Defined in: uf_sf.h

Overview

Given a point and a Polygon geometry (i.e. Edge, Face, Body), Returns the projected point and distance.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_ask_closest_point
(
    double point [ 3 ] ,
    tag_t entity,
    double closest_point [ 3 ] ,
    double * min_dist
)
```

double	**point [3]**	Input	Input point
tag_t	**entity**	Input	Polygon edge/face/body
double	**closest_point [3]**	Output	Closest point on entity
double *	**min_dist**	Output	Minimum distance

UF_SF_ask_combined_load_case (view source)

Defined in: uf_sf.h

Overview
Retrieves the definition of combined load case.

Environment
Internal and External

```
int UF_SF_ask_combined_load_case
(
    tag_t clc_tag,
    char* * clc_name,
    int* num_comps,
    tag_t* * lbc_tags,
    double* * lbc_scales
)
```

tag_t	**clc_tag**	Input	Tag of the combined load case.
char* *	**clc_name**	Output to UF_*free*	Name of the combined load case. This must be freed by calling UF_free.
int*	**num_comps**	Output	Number of load cases in the combined load case definition.
tag_t* *	**lbc_tags**	Output to UF_*free*	pointer to the pointer to the array of the tags for load case components. This must be freed by calling UF_free.
double* *	**lbc_scales**	Output to UF_*free*	pointer to the pointer to the array of the scale factors for load case components. This must be freed by calling UF_free.

UF_SF_ask_combined_load_cases (view source)

Defined in: uf_sf.h

Overview

Finds all the combined load cases in the given scenario part. If passed NULL_TAG for part_tag then current work part will be traversed.

Environment

Internal and External

```
int UF_SF_ask_combined_load_cases
(
    tag_t part_tag,
    int* num_clcs,
    tag_t* * clc_tags
)
```

tag_t	**part_tag**	Input	part to retrieve from, NULL_TAG for current work part
int*	**num_clcs**	Output	Number of combined load case sets found in the current scenario part.
tag_t* *	**clc_tags**	Output to UF_*free*	pointer to the pointer to the array of the tag for the combined load cases. pass in NULL if no allocation is desired. If NULL is not passed in, this array must be freed by calling UF_free.

UF_SF_ask_dom_elm_type (view source)

Defined in: uf_sf.h

Overview

Finds the dominant element type of the input mesh recipe. For example if the dimension of a mesh is 1D, the dominant element type can be a beam or bar or a spring. In case of 2D meshes the dominant element type is UF_SF_QUAD4, though couple of triangular elements may get created in creating 2D mesh.

Environment

Internal and External

```
int UF_SF_ask_dom_elm_type
(
    tag_t mesh_recipe,
    UF_SF_element_type_t* elm_type
)
```

tag_t	**mesh_recipe**	Input	Tag to mesh recipe or mesh
UF_SF_element_type_t*	**elm_type**	Output	The dominant element type of this mesh_recipe.

UF_SF_ask_dursol_nx (view source)

Defined in: uf_sf.h

Overview

This function will ask a Durability Solution's values.
From NX7.5, this function will return the metasolution name, followed by

parameters from the first static event. If the static event is not available, an error will be returned.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_ask_dursol_nx
(
    tag_t dursol_tag,
    char dursol_name [ MAX_LINE_BUFSIZE ] ,
    tag_t * solution_tag,
    UF_SF_dursol_stress_criterion_t * stress_criterion,
    UF_SF_dursol_stress_type_t * stress_type,
    UF_SF_dursol_design_life_criterion_t * design_life_criterion,
    int * fatigue_cycles,
    double * k_factor,
    UF_SF_dursol_fatigue_life_criterion_t * fatigue_life_criterion,
    int * design_cycles
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag.
char	**dursol_name [MAX_LINE_BUFSIZE]**	Output	Name of Durability Solution. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters. If NULL, value will not be returned.
tag_t *	**solution_tag**	Output	Tag of the solution If NULL, value will not be returned.
UF_SF_dursol_stress_criterion_t *	**stress_criterion**	Output	Stress criterion If NULL, value will not be returned.
UF_SF_dursol_stress_type_t *	**stress_type**	Output	Stress type If NULL, value will not be returned.
UF_SF_dursol_design_life_criterion_t *	**design_life_criterion**	Output	Design life criterion If NULL, value will not be returned.
int *	**fatigue_cycles**	Output	Number of fatigue duty cycles If NULL, value will not be returned.
double *	**k_factor**	Output	Fatigue strength factor (Kf) If NULL, value will not be returned.
UF_SF_dursol_fatigue_life_criterion_t *	**fatigue_life_criterion**	Output	Fatigue life criterion If NULL, value will not be returned.
int *	**design_cycles**	Output	Number of design cycles desired for fatigue strength. If NULL, value will not be returned.

UF_SF_ask_edge_density (view source)

Defined in: uf_sf.h

Overview

Gets a number of edge densities given an input object. The edge density is defined as the number of elements to be created along the input object. There are two cases: If elements exist, this function

returns the current number of elements. If elements do not exist, this function gets the edge density attribute for this object and returns the number of elements which would be created. The valid types of input objects are UF_line_type, UF_circle_type, UF_conic_type, UF_spcurve_type, UF_caegeom_edge_subtype.

Environment
Internal and External

```
int UF_SF_ask_edge_density
(
    tag_t object_tag,
    UF_SF_edge_density_data_t* edge_density_data_ptr
)
```

tag_t	**object_tag**	Input	Tag of object to query.
UF_SF_edge_density_data_t*	**edge_density_data_ptr**	Output	pointer to UF_SF_edge_density_data_t where the edge_density data will be stored.

UF_SF_ask_element (view source)

Defined in: uf_sf.h

Overview
Finds an element entity's attributes. These are the element id, adaptivity, dimension and neutral type.

Environment
Internal and External

```
int UF_SF_ask_element
(
    tag_t elm_tag,
    int* elm_id,
    UF_SF_elm_adaptivity_type_t* adaptivity,
    UF_SF_elm_dimension_type_t* dimension,
    UF_SF_element_type_t* element_type
)
```

tag_t	**elm_tag**	Input	Element tag.
int*	**elm_id**	Output	ID number of element
UF_SF_elm_adaptivity_type_t*	**adaptivity**	Output	Element adaptivity.
UF_SF_elm_dimension_type_t*	**dimension**	Output	Dimension of element (0D, 1D, 2D or 3D)
UF_SF_element_type_t*	**element_type**	Output	Type of element .

UF_SF_ask_element_edges (view source)

Defined in: uf_sf.h

Overview
Given an element tag outputs the number of edges this element has and the nodes on those edges.

If given an element occurrence tag then node occurrences in the owning Simulation part are returned. Similarly if the given an element tag then nodes in the owning FEM part are returned.

Environment
Internal and External

```
int UF_SF_ask_element_edges
(
    tag_t elem_tag,
    int* num_edges,
    tag_t* * start_nodes,
    tag_t* * end_nodes,
    tag_t* * mid_nodes
)
```

tag_t	**elem_tag**	Input	The tag of the element
int*	**num_edges**	Output	Number of edges the element has
tag_t* *	**start_nodes**	Output to UF_*free*	Array of start nodes on these edges. This must be freed by calling UF_free.
tag_t* *	**end_nodes**	Output to UF_*free*	Array of end nodes on these edges. This must be freed by calling UF_free.
tag_t* *	**mid_nodes**	Output to UF_*free*	Array of mid side nodes on these edges, if any. This must be freed by calling UF_free.

UF_SF_ask_element_type_names (view source)

Defined in: uf_sf.h

Overview
This user function is used to ask the elemnt type name in the current solver language.

Environment
Internal and External

```
int UF_SF_ask_element_type_names
(
    UF_SF_element_type_t element_type,
    char* * element_name_array
)
```

UF_SF_element_type_t	**element_type**	Input	Parameters which control the required tet Mesh
char* *	**element_name_array**	Output	tag of the created mesh

---

UF_SF_ask_geom_data (view source)

Defined in: uf_sf.h

Overview

Returns geometry data from given mesh or mesh recipe.

Environment
Internal and External

```
int UF_SF_ask_geom_data
(
    tag_t mesh_object,
    UF_SF_mesh_geom_usage_t usage_type,
    int* num_geom_items,
    tag_t* * geom_items_p
)
```

tag_t	**mesh_object**	Input	Tag of the mesh
UF_SF_mesh_geom_usage_t	**usage_type**	Input	Type of usage (UF_SF_mesh_geom_usage_t) - refer to uf_sf_types.h
int*	**num_geom_items**	Output	Number of geometry items.
tag_t* *	**geom_items_p**	Output to UF_*free*	pointer to list of tags to geometry items. If the number of geometry items is > 0, then this array must be freed by calling UF_free.

UF_SF_ask_idealize_parm_exp (view source)

Defined in: uf_sf.h

Overview
This routine gets a specified expression from a parameter object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_ask_idealize_parm_exp
(
    UF_SF_idealize_parms_p_t parms_p,
    UF_SF_idealize_parm_exp_t parm_exp_t,
    tag_t * exp_tag
)
```

UF_SF_idealize_parms_p_t	**parms_p**	Input	Idealize parameter entity to query
UF_SF_idealize_parm_exp_t	**parm_exp_t**	Input	Specific expression to retrieve from idealize parameter entity
tag_t *	**exp_tag**	Output	Tag of the expression used for the specific idealize parameter Can be NULL_TAG which means that this parameter is not used

---

UF_SF_ask_idealize_parm_faces (view source)

Defined in: uf_sf.h

Overview
This routine gets a specified set of faces from a parameter object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_ask_idealize_parm_faces
(
    UF_SF_idealize_parms_p_t parms_p,
    UF_SF_idealize_parm_face_t parm_face_t,
    int * num_faces,
    tag_t * * faces
)
```

UF_SF_idealize_parms_p_t	**parms_p**	Input	Idealize parameter entity to query
UF_SF_idealize_parm_face_t	**parm_face_t**	Input	Specific face set parameter to query
int *	**num_faces**	Output	Number of faces returned in faces array
tag_t * *	**faces**	Output to UF_*free*	Array of face tags. This should be freed by calling UF_free.

UF_SF_ask_idealize_parms (view source)

Defined in: uf_sf.h

Overview
This routine returns the definition of an idealize feature

This routine takes the feature tag of an idealize feature and returns the parameters associated to it.

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_ask_idealize_parms
(
    tag_t feature_tag,
    logical * is_region,
    void * body_region,
    UF_SF_idealize_parms_p_t idealize_parms
)
```

tag_t	**feature_tag**	Input	The idealize feature
logical *	**is_region**	Output	Flag indicating whether the idealize feature is from a region versus a body is_region = TRUE -> Feature is from a region is_region = FALSE -> Feature is from a body
void *	**body_region**	Output to UF_*free*	If NULL on input only the idealize_parms will be returned. Pointer to a UF_SF_idealize_region_p_t or to a tag_p_t for the body depending on the returned value of the flag is_region. If is_region = TRUE -> output is a pointer to an instance of UF_SF_idealize_region_data_p_t

			If is_region = FALSE -> output is a pointer to a tag_p_t that identifies the body. Use UF_SF_idealize_free_region if a region Use UF_free if a body
UF_SF_idealize_parms_p_t	**idealize_parms**	Output to UF_*free*	The parameters of the idealize feature. Use UF_SF_idealize_free_parms to free

---

UF_SF_ask_idealize_region (view source)

Defined in: uf_sf.h

Overview

This routine returns the information in a region object for an idealize feature

The region object is used to hold all information needed to create an idealize feature associated to a body region.

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_ask_idealize_region
(
    UF_SF_idealize_region_p_t region_p,
    tag_t * seed_face,
    int * num_bnd_faces,
    tag_t * * bnd_faces,
    tag_t * angle_exp
)
```

UF_SF_idealize_region_p_t	**region_p**	Input	Idealize region entity in which to query
tag_t *	**seed_face**	Output	Seed face for the region Pass NULL if return parameter is not wanted
int *	**num_bnd_faces**	Output	Number of boundary faces in the bnd_faces array. Can be 0 Pass NULL if return parameter is not wanted If NULL is passed in, bnd_faces will be returned as NULL
tag_t * *	**bnd_faces**	Output to UF_*free*	Array of boundary faces. Is NULL if num_bnd_faces == 0 Pass NULL if return parameter is not wanted
tag_t *	**angle_exp**	Input	Expression used for the tagential edge angle which is used for the region boundary, Can be NULL_TAG

UF_SF_ask_language (view source)

Defined in: uf_sf.h

Overview

Given the input language_tag, finds its name in language_name. If language_tag == NULL_TAG, return the name of the current language. If there is is no current language and language_tag is NULL_TAG, return "". Pass NULL against any argument if

corresponding output is not desired.

Environment

Internal and External

```
int UF_SF_ask_language
(
    tag_t language_tag,
    char language_name [ UF_SF_LANG_MAX_NAME_BUFSIZE ] ,
    int* version,
    UF_SF_LANG_analysis_type_t* analysis_type,
    UF_SF_LANG_problem_abstract_t* problem_abstraction,
    UF_SF_LANG_linearity_t* linearity,
    UF_SF_LANG_time_depend_t* time_dependency,
    char* * solver_name
)
```

tag_t	**language_tag**	Input	Tag of the desired language or NULL_TAG. When NULL_TAG, finds the properties of current language. If there is no current language, returns "" as language name.
char	**language_name [UF_SF_LANG_MAX_NAME_BUFSIZE]**	Output	pointer to string which will contain the found name or "". pe that this is not dynamic space. A string at least UF_SF_LANG_MAX_NAME_BUFSIZE long is to be passed in.
int*	**version**	Output	The version of the language. If input == NULL, then no output is desired or returned.
UF_SF_LANG_analysis_type_t*	**analysis_type**	Output	The type of analysis this language represents. If input == NULL, then no output is desired or returned.
UF_SF_LANG_problem_abstract_t*	**problem_abstraction**	Output	The problem abstraction this language represents. If input == NULL, then no output is desired or returned.
UF_SF_LANG_linearity_t*	**linearity**	Output	Linear or non-linear. If input == NULL, then no output is desired or returned.
UF_SF_LANG_time_depend_t*	**time_dependency**	Output	Steady-state or transient. If input == NULL, then no output is desired or returned.
char* *	**solver_name**	Output to UF_*free*	The name of the solver to use. If input == NULL, then no output is desired or returned. If a NULL is not passed in for the solver_name, then the returned value must be freed by calling UF_free.

UF_SF_ask_library_materials (view source)

Defined in: uf_sf.h

Overview

Query material librefs and names from the NX Material library.

The output is optionally limited by material type, category and name filters.

Each filter is a character string containing a regular expression.
To disable a filter, enter "".
To specify a filter,
code, for example, "IRON" to find all materials named with a prefix of "IRON". The usual material types are "ISO" "ORTHO" "ANISO" "FLUID", filters are case insensitive.

The outputs of this function are the material count and 3 arrays. The libref field is used to retrive the actual material from the library. The name is intended to be the identifier that the user will see displayed. The material type is passed back as a convenience to allow post query filtering.

The 3 lists are arrays for allocated character strings. Use UF_free_string_array to free each array.

Environment

Internal and External

See Also

UF_free_string_array

History

Originally released in NX2

```
int UF_SF_ask_library_materials
(
    char* type_filter,
    char* category_filter,
    char* name_filter,
    int* material_count,
    char* * * material_librefs,
    char* * * material_names,
    char* * * material_types
)
```

char*	**type_filter**	Input	A regular expression string which filters the material type. To find all materials, enter "". To find all isotropic materials, enter "ISO". To find all orthotropic materials, enter "ORTHO". To find all anisotropic materials, enter "ANISO". To find all fluid materials, enter "FLUID".
char*	**category_filter**	Input	A regular expression string which filters the material category. To find all materials, enter "".
char*	**name_filter**	Input	A regular expression string which filters the material name. To find all materials, enter "".
int*	**material_count**	Output	The number of materials listed in the arrays.
char* * *	**material_librefs**	Output to UF_*free*	An array of characters strings containing the library reference field of each material. Call UF_free_string_array to free the storage allocated within the structure.
char* * *	**material_names**	Output to UF_*free*	An array of characters strings containing the name field of each material. Call UF_free_string_array to free the storage allocated within the structure.
char* * *	**material_types**	Output to UF_*free*	An array of characters strings containing the type field of each material. Call UF_free_string_array to free the storage allocated within the structure.

UF_SF_ask_lv_nx (view source)

Defined in: uf_sf.h

Overview

This function will ask a Durability load variation's values.
From NX7.5, the number of cycles will be the number of occurrences on the parent static event.
Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_ask_lv_nx
(
    tag_t lv_tag,
    char lv_name [ MAX_LINE_BUFSIZE ] ,
    double * scaling_factor,
    int * count,
    UF_SF_lv_function_mode_t * function_type,
    tag_t * solution_step_tag
)
```

tag_t	**lv_tag**	Input	Load Variation tag
char	**lv_name [MAX_LINE_BUFSIZE]**	Output	Name The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters. If NULL, value will not be returned.
double *	**scaling_factor**	Output	Scaling factor If NULL, value will not be returned.
int *	**count**	Output	Number of cycles If NULL, value will not be returned.
UF_SF_lv_function_mode_t *	**function_type**	Output	Scaling function type UF_SF_HALF_UNIT_CYCLE UF_SF_FULL_UNIT_CYCLE If NULL, value will not be returned.
tag_t *	**solution_step_tag**	Output	Solution Step tag If NULL, value will not be returned.

UF_SF_ask_material (view source)

Defined in: uf_sf.h

Overview

Gets all the material properties such as Young's Modulus, poisson's Ratio, Mass Density, Thermal Coefficient, Thermal Conduct, Heat Capacity, Thermal Density and Convection Coefficient. If any one of the above properties has not been applied on the material then the corresponding value will be UF_SF_FREE_VALUE.

The UF_SF_material_prop_t, property_values must be initialized using UF_SF_init_matl_prop.

Environment

Internal and External

See Also

UF_SF_init_matl_prop and
UF_SF_free_matl_prop

int UF_SF_ask_material
(
tag_t material_tag,
char* * material_name,
UF_SF_material_prop_t* property_values
)

tag_t	**material_tag**	Input	Tag of the material.
char* *	**material_name**	Output to UF_*free*	Material name string. pass in NULL if not required. This must be freed by calling UF_free.
UF_SF_material_prop_t*	**property_values**	Output to UF_*free*	Structure containing material properties. Pass in NULL if not required. The structure must be initialized using UF_SF_init_matl_prop. Call UF_SF_free_matl_prop to free the storage allocated within the structure.

UF_SF_ask_material_fatigue (view source)

Defined in: uf_sf.h

Overview
Gets the fatigue material properties; fatigue strength coefficient, fatigue strength exponent, fatigue ductility coefficient, fatigue ductility exponent. If any one of the above properties has not been applied on the material then the corresponding value will be UF_SF_FREE_VALUE.

Environment
Internal and External

See Also
html#UF_SF_free_fatigue_prop
html#UF_SF_init_fatigue_prop

History
This function was originally released in V20.0

int UF_SF_ask_material_fatigue
(
tag_t material_tag,
UF_SF_material_fatigue_prop_p_t property_values
)

tag_t	**material_tag**	Input	Tag of the material.
UF_SF_material_fatigue_prop_p_t	**property_values**	Output to UF_*free*	Structure containing material fatigue properties. The structure should be allocated by the user, but data within the structure is dynamically allocated. Call html#UF_SF_free_fatigue_prop to free the storage allocated within the structure.

UF_SF_ask_material_formability (view source)

Defined in: uf_sf.h

Overview
Gets the formability material properties; Work Hardening, Flow and Stress Strain. If any one of the above properties has not been applied on the material then the corresponding value will be UF_SF_FREE_VALUE.

Environment
Internal and External

See Also
html#UF_SF_free_formability_prop
html#UF_SF_init_formability_prop

History
This function was originally released in V19.1

int UF_SF_ask_material_formability
(
tag_t material_tag,
UF_SF_material_formability_prop_p_t property_values
)

tag_t	**material_tag**	Input	Tag of the material.
UF_SF_material_formability_prop_p_t	**property_values**	Output to UF_*free*	Structure containing material formability properties. The structure should be allocated by the user, but data within the structure is dynamically allocated. Call html#UF_SF_free_formability_prop to free the storage allocated within the structure.

UF_SF_ask_material_strength (view source)

Defined in: uf_sf.h

Overview
Gets the plastic range material properties; Yield Strength and Ultimate Tensile Strength. If either one of the above properties has not been applied on the material then the corresponding value will be UF_SF_FREE_VALUE.

Environment
Internal and External

See Also
html#UF_SF_free_matl_strength_prop
html#UF_SF_init_matl_strength_prop

History
This function was originally released in V19.1

int UF_SF_ask_material_strength
(
tag_t material_tag,
UF_SF_material_strength_prop_p_t property_values
)

tag_t	**material_tag**	Input	Tag of the material.
UF_SF_material_strength_prop_p_t	**property_values**	Output to UF_*free*	Structure containing material strength properties. The structure should be allocated by the user, but data within

	the structure is dynamically allocated. Call html#UF_SF_free_matl_strength_prop to free the storage allocated within the structure.

UF_SF_ask_material_type (view source)

Defined in: uf_sf.h

Overview

This routine looks up the type of a material. The following types are supported in NX:

UF_SF_MATERIAL_ISOTROPIC - isotropic.
UF_SF_MATERIAL_ORTHOTROPIC - orthotropic.
UF_SF_MATERIAL_ANISOTROPIC - anisotropic.
UF_SF_MATERIAL_FLUID - fluid.

Environment

Internal and External

History

Originally released in V16.0

```
int UF_SF_ask_material_type
(
    tag_t material_tag,
    UF_SF_neutral_material_types_p_t material_type
)
```

tag_t	**material_tag**	Input	Tag of the material
UF_SF_neutral_material_types_p_t	**material_type**	Output	Material Type

---

UF_SF_ask_mesh_dimension (view source)

Defined in: uf_sf.h

Overview

Finds the dimension of the input mesh or mesh recipe namely UF_SF_DIMENSION_0D for 0D mesh, UF_SF_DIMENSION_1D for 1D mesh, UF_SF_DIMENSION_2D for shell mesh or UF_SF_DIMENSION_3D for Solid mesh.

Environment

Internal and External

```
int UF_SF_ask_mesh_dimension
(
    tag_t mesh_recipe,
    UF_SF_mesh_dimension_t* dimension
)
```

tag_t	**mesh_recipe**	Input	Tag to mesh recipe or mesh
UF_SF_mesh_dimension_t*	**dimension**	Output	The dimensionality of this mesh_recipe

---

UF_SF_ask_mesh_mating_condition (view source)

Defined in: uf_sf.h

Overview

Given an assembly mesh tag, obtain the infomation about the assembly mesh.
If some information is unneeded, the input pointer can be set to NULL.

Environment

Internal and External

History

This function was originally released in V16.0

```
int UF_SF_ask_mesh_mating_condition
(
    tag_t assembly_mesh_tag,
    tag_t* assembling_geom,
    tag_t* target_geom,
    char* * assembly_name,
    int* seed_mesh,
    UF_SF_assembly_type_t* assembly_type,
    double* merge_tolerance,
    double* match_tolerance
)
```

tag_t	**assembly_mesh_tag**	Input	Assembly mesh tag
tag_t*	**assembling_geom**	Output	Assembling region geometry tag
tag_t*	**target_geom**	Output	Target region geometry tag
char* *	**assembly_name**	Output to UF_*free*	Assembly mesh name
int*	**seed_mesh**	Output	Flag of seed mesh
UF_SF_assembly_type_t*	**assembly_type**	Output	Assembly mesh type
double*	**merge_tolerance**	Output	Node merge tolerance
double*	**match_tolerance**	Output	match tolerance to define if an independent node matchs with a master element

---

UF_SF_ask_mesh_visuals (view source)

Defined in: uf_sf.h

Overview

Given a mesh tag returns its visual properties.

Environment

Internal and External

```
int UF_SF_ask_mesh_visuals
(
    tag_t mesh_tag,
    UF_SF_mesh_visuals_t* mesh_vis
)
```

tag_t	**mesh_tag**	Input	The tag of the mesh

UF_SF_mesh_visuals_t*	**mesh_vis**	Output	Structure to mesh visual properties.

UF_SF_ask_midsrf_frec_type (view source)

Defined in: uf_sf.h

Overview
This function determines the type of a midsurface feature

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_ask_midsrf_frec_type
(
    tag_t feature_tag,
    int * midsrf_type
)
```

tag_t	**feature_tag**	Input	Tag of given feature
int *	**midsrf_type**	Output	Type of midsurface if object is a midsurface = UF_SF_MIDSRF_FACEPAIR_METHOD = UF_SF_MIDSRF_OFFSET_METHOD = UF_SF_MIDSRF_USER_DEF_METHOD = UF_SF_MIDSRF_NON_EXIST -> If feature is not a midsurface

---

UF_SF_ask_node (view source)

Defined in: uf_sf.h

Overview
Finds a node entity's attributes. These are the node id, boundary type, element type, and absolute position.

Environment
Internal and External

```
int UF_SF_ask_node
(
    tag_t node_tag,
    int* node_id,
    UF_SF_node_btype_t* b_type,
    UF_SF_mid_node_type_t* e_type,
    double abspos [ 3 ]
)
```

tag_t	**node_tag**	Input	Node tag.
int*	**node_id**	Output	ID number of node.
UF_SF_node_btype_t*	**b_type**	Output	Type of node boundary .
UF_SF_mid_node_type_t*	**e_type**	Output	Type of mid node.
double	**abspos [3]**	Output	node location in absolute coordinates.

UF_SF_ask_node_pgeoms (view source)

Defined in: uf_sf.h

Overview

Given a tag to a node entity, locates the parent geometry features linked to the node.

If given a node occurrence from a Simulation part then polygonal (CAE) geometry occurrences from that Simulation part are returned. Similarly if given a node from a FEM part then polygonal (CAE) geometry from that FEM part are returned.

Environment

Internal and External

```
int UF_SF_ask_node_pgeoms
(
    tag_t node_tag,
    int* parent_cnt,
    tag_p_t* parent_list,
    UF_SF_mesh_geometry_types_p_t* geom_types
)
```

tag_t	**node_tag**	Input	Tag to a node object.
int*	**parent_cnt**	Output	Number of parent geometry tags related to the node.
tag_p_t*	**parent_list**	Output to UF_*free*	pointer to an array of parent geometry tags found or NULL if not found. This array must be freed by calling UF_free.
UF_SF_mesh_geometry_types_p_t*	**geom_types**	Output to UF_*free*	pointer to an array of parent geometry types found or NULL if not found. geom_types is 1:1 with objects list. This array must be freed by calling UF_free.

UF_SF_ask_nth_dursol_nx (view source)

Defined in: uf_sf.h

Overview

This function will return the tag of the specified Durability Solution.

UF_SF_ask_nth_dursol_nx

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_ask_nth_dursol_nx
(
    int index,
    tag_t * dursol_tag
)
```

int	**index**	Input	Index of the Durability Solution. Valid range is 0 : (number of Durability Solutions)-1
tag_t *	**dursol_tag**	Output	Durability Solution tag.

UF_SF_ask_nth_mesh_recipe (view source)

Defined in: uf_sf.h

Overview
Query nth mesh recipe in a given FEM part.

Environment
Internal and External

History
Originally released in NX4.01

```
int UF_SF_ask_nth_mesh_recipe
(
    tag_t fem_tag,
    int index,
    tag_p_t mesh_recipe_tag
)
```

tag_t	**fem_tag**	Input	FEM part
int	**index**	Input	Index into list of mesh recipes. Valid indices are from 0 to (number of mesh recipes-1). The number of mesh recipes can be obtained from function UF_SF_ask_num_mesh_recipes.
tag_p_t	**mesh_recipe_tag**	Output	Mesh recipe

UF_SF_ask_nth_solution_nx (view source)

Defined in: uf_sf.h

Overview
Given a index into a list of solutions, returns the tag of the solution. This function can be used to loop over all solutions.

The number of solutions can be obtained from function UF_SF_ask_num_solutions_nx.

The index is just a counter.
Valid indices are from 0 to (number of solutions-1).

Valid only when the work part is a Simulation.

Environment
Internal and External

See Also
UF_SF_ask_num_solutions_nx

History
Originally released in NX3.0

```
int UF_SF_ask_nth_solution_nx
(
    int solution,
    tag_t * solution_tag
)
```

int	**solution**	Input	Index of the solution to query. Valid indices are from 0 to (number of solutions-1).
tag_t *	**solution_tag**	Output	Tag of the solution.

UF_SF_ask_num_dursols_nx (view source)

Defined in: uf_sf.h

Overview

This function will return the number of Durability Solutions.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_ask_num_dursols_nx
(
    int * num_dursols
)
```

int *	**num_dursols**	Output	Number of Durability Solutions.

UF_SF_ask_num_mesh_recipes (view source)

Defined in: uf_sf.h

Overview

Query number of mesh recipes in a given FEM part.

Environment

Internal and External

History

Originally released in NX4.01

```
int UF_SF_ask_num_mesh_recipes
(
    tag_t fem_tag,
    int * num_mesh_recipes
)
```

tag_t	**fem_tag**	Input	FEM part
int *	**num_mesh_recipes**	Output	Number of mesh recipes

UF_SF_ask_num_solutions_nx (view source)

Defined in: uf_sf.h

Overview
Query the number of solutions.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_ask_num_solutions_nx
(
    int * solution_count
)
```

int *	**solution_count**	Output	Number of solutions.

UF_SF_ask_offset_midsrf_thickness (view source)

Defined in: uf_sf.h

Overview

Query the thickness of an offset midsurface. Please note that this function will not return the thickness if the midsurface is created by a method other than the offset method (e.g., Facepair method).

Environment
Internal and External

History
Originally released in V16.0

```
int UF_SF_ask_offset_midsrf_thickness
(
    tag_t midsrf_tag,
    double * thickness
)
```

tag_t	**midsrf_tag**	Input	Tag of the midsurface.
double *	**thickness**	Output	The thickness of the midsurface. If the tag is not an offset midsurface, then the thickness will be returned as 0.0.

UF_SF_ask_solution_nx (view source)

Defined in: uf_sf.h

Overview
Given a load or boundary condition entity, this function finds the solutions that use the load or boundary condition entity.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_ask_solution_nx
(
    tag_t lbc_tag,
    int * num_members,
    tag_t* * solution_pp
)
```

tag_t	**lbc_tag**	Input	Tag of the load or boundary condition.
int *	**num_members**	Output	Number of loads and bc's in the solution.
tag_t* *	**solution_pp**	Output to UF_*free*	The pointer to the array of the tags of the solutions. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_ask_step_nx (view source)

Defined in: uf_sf.h

Overview
Given a load or boundary condition entity, this function finds the steps that use the load or boundary condition entity.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_ask_step_nx
(
    tag_t lbc_tag,
    int * num_members,
    tag_t* * step_pp
)
```

tag_t	**lbc_tag**	Input	Tag of the load or boundary condition.
int *	**num_members**	Output	Number of loads and bc's in the step.
tag_t* *	**step_pp**	Output to UF_*free*	The pointer to the array of the tags of the steps. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_auto_create_mesh_mating_condition (view source)

Defined in: uf_sf.h

Overview
Creates automatically mesh mating conditions from an array of polygon bodies.

Environment
Internal and External

```
int UF_SF_auto_create_mesh_mating_condition
(
    int entity_num,
    tag_t entity_array [ ] ,
    double merge_tolerance,
    int make_mesh_coincident,
    int coincident_face_only,
    int mesh_mating_type,
    int * num_assembly_meshes,
    tag_t * * assembly_meshes
)
```

int	**entity_num**	Input	Number of entities in array
tag_t	**entity_array [ ]**	Input	Array of tags of the entities (polygon bodies)
double	**merge_tolerance**	Input	Merge tolerance for assembly mesh
int	**make_mesh_coincident**	Input	Make Mesh Coincident switch (0 or 1)
int	**coincident_face_only**	Input	Coincident face switch (0 or 1)
int	**mesh_mating_type**	Input	Mesh mating type = 0 --> GLUE = 3 --> FREE NOTE: There is no 1 or 2 Type
int *	**num_assembly_meshes**	Output	Number of assembly mesh tags created
tag_t * *	**assembly_meshes**	Output to UF_*free*	Array of assembly mesh tags created Free when done

---

UF_SF_auto_create_surface_contact_mesh (view source)

Defined in: uf_sf.h

Overview
Auto creates surface contact meshes. For a given capture distance, a mesh is created for every pair of meshes created. A mesh has only one source polygon face and one target polygon face.

Environment
Internal and External

History

```
int UF_SF_auto_create_surface_contact_mesh
(
    double capture_distance,
    void * property,
    int * num_meshes,
    tag_p_t * mesh_recipes
)
```

double	**capture_distance**	Input	The mimimum distance between two polygon faces of different polygon bodies that form a contact pair.

void *	**property**	Input	Structure to define mesh properties. The valid structures are UF_SF_SURFACE_CONTACT_ANS_data_p_t, UF_SF_SURFACE_CONTACT_NAS_data_p_t, UF_SF_SURFACE_CONTACT_UGFEA_data_p_t
int *	**num_meshes**	Output	Pointer to number of meshes
tag_p_t *	**mesh_recipes**	Output to UF_*free*	Pointer to the number of mesh recipe tags created.

UF_SF_body_ask_bounding_box (view source)

Defined in: uf_sf.h

Overview
This function will get bounding box of Polygon body

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_body_ask_bounding_box
(
    tag_t cae_tag,
    double pad_bounding_volume [ 6 ]
)
```

tag_t	**cae_tag**	Input	Polygon body
double	**pad_bounding_volume [6]**	Output	Bounding box

UF_SF_body_ask_edges (view source)

Defined in: uf_sf.h

Overview
This function will get polygon edges related to polygon body.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_body_ask_edges
(
    tag_t cae_tag,
    int * num_edges,
    tag_p_t * edges
)
```

tag_t	**cae_tag**	Input	Polygon body
int *	**num_edges**	Output	Number of polygon edges
tag_p_t *	**edges**	Output to UF_*free*	Tags of polygon edges

---

UF_SF_body_ask_faces (view source)

Defined in: uf_sf.h

Overview
This function will get polygon faces related to polygon body.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_body_ask_faces
(
    tag_t cae_tag,
    int * num_faces,
    tag_p_t * faces
)
```

tag_t	**cae_tag**	Input	Polygon body
int *	**num_faces**	Output	Number of polygon faces
tag_p_t *	**faces**	Output to UF_*free*	Polygon faces

UF_SF_body_ask_modl_body (view source)

Defined in: uf_sf.h

Overview
This function will get MODL body related to Polygon body.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_body_ask_modl_body
(
    tag_t cae_tag,
    tag_p_t modl_body_p
)
```

tag_t	**cae_tag**	Input	Polygon body
tag_p_t	**modl_body_p**	Output	CAD MODL body

UF_SF_body_ask_volume_and_centroid (view source)

Defined in: uf_sf.h

Overview

This function will get volume and centroid of Polygon body

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_body_ask_volume_and_centroid
(
    tag_t cae_tag,
    double * pd_volume,
    double centroid [ 3 ]
)
```

tag_t	**cae_tag**	Input	Polygon body
double *	**pd_volume**	Output	Volume
double	**centroid [3]**	Output	Centroid

UF_SF_cae_information (view source)

Defined in: uf_sf.h

Overview
runs a duplicate nodes model check on the current scenario model.

ENVIRONMENT
Internal and External

```
int UF_SF_cae_information
(
    UF_SF_scenario_info_t info_type,
    int num_entities,
    tag_p_t entities_tags,
    const char * output_file_with_path
)
```

UF_SF_scenario_info_t	**info_type**	Input	The type of information that is requested
int	**num_entities**	Input	The number of entities to get the requested information for.
tag_p_t	**entities_tags**	Input	The tags of the meshes, sections, loads, etc. that we need info for. Type of entitiy is determined by info_type. UF_SF_fea_summary_info, num_entities should be 0 and entities_tags should be NULL. UF_SF_mesh_info, specify mesh tags UF_SF_load_info, specify load tags UF_SF_load_case_info, specify a load tag in each desired load case UF_SF_dur_event_info, event information UF_SF_boundary_condition_info, specify bc tags UF_SF_material_info, specify solid or sheet body or mesh tags to display material info for. UF_SF_section_info, specify the tag of the curve, edge, point, or mesh which is

			associated to the section you want information about. UF_SF_mesh_mating_condition_info specify the tags of the assembly mesh that you want info about. This tag is returned from UF_SF_locate_mesh_mating_condition_by_name
const char *	**output_file_with_path**	Input	Full path including filename of file where results of model check should be written. Caller is responsible for allocating and deallocating this array.

UF_SF_check_model_comprehensive (view source)

Defined in: uf_sf_model_checker.h

Overview

FUNCTION
UF_SF_check_model_comprehensive

DESCRIPTION
This function will run a comprehensive model check on the
current scenario model.

INPUT
detailed_message TRUE for a more detailed output written
to output_file_with_path.
output_file_with_path Full path including filename of file
where results of model check should be
written. Caller is responsible for
allocating and deallocating this array.

OUTPUT:
none

RETURN
error == 0.
>0 for error

```
int UF_SF_check_model_comprehensive
(
    logical detailed_message,
    const char * ouput_file_with_path
)
```

logical	**detailed_message**	Input
const char *	**ouput_file_with_path**	Input

UF_SF_check_model_duplicate_nodes (view source)

Defined in: uf_sf_model_checker.h

Overview

FUNCTION
UF_SF_check_model_duplicate_nodes

DESCRIPTION
This function will run a duplicate node model check on the

current scenario model.

INPUT
num_meshes number of meshes to check. 0 if all
existing meshes should be checked.
mesh_tags Array of mesh tags to check. NULL if
all existing meshes should be checked.
merge_duplicates TRUE if duplicates should be merged.
Otherwise FALSE
tolerance tolerance to decide if nodes are duplicate.

OUTPUT:
num_duplicates number of duplicates found and/or merged.

RETURN
error == 0.
>0 for error

```
int UF_SF_check_model_duplicate_nodes
(
    int num_meshes,
    tag_p_t mesh_tags,
    logical merge_duplicates,
    double tolerance,
    int * num_duplicates
)
```

int	**num_meshes**	Input
tag_p_t	**mesh_tags**	Input
logical	**merge_duplicates**	Input
double	**tolerance**	Input
int *	**num_duplicates**	Output

UF_SF_check_model_element_shapes (view source)

Defined in: uf_sf.h

Overview

FUNCTION
UF_SF_check_model_element_shapes

DESCRIPTION
This function performs element shape check for the
meshes passed in and writes the results to a file
or appends at the end of an existing file.

RETURN
error == 0.
>0 for error

```
int UF_SF_check_model_element_shapes
(
    int num_meshes,
    tag_t mesh_tags [ ] ,
    logical list_all_elems,
    const char * output_file_with_path,
    FILE * file_to_append_to
)
```

int	**num_meshes**	Input	Length of the mesh_tags array.

tag_t	**mesh_tags [ ]**	Input	Array of mesh tags
logical	**list_all_elems**	Input	TRUE/FALSE whether to list shape check result for all elements in respective meshes.
const char *	**output_file_with_path**	Input	Name with path of output file. NULL if file_to_append_to is specified in arguement below.
FILE *	**file_to_append_to**	Input	Append output to end of the alteady opened file. The file must be opened in append mode. NULL if output_file_with_path specified.

UF_SF_clean_mshvld_error_container (view source)

Defined in: uf_sf_mshvld.h

Overview

Function: UF_SF_clean_mshvld_error_container

DESCRIPTION

This user function is used to clean the error container
of mesh validation. It's prototype is available in uf_sf_mshvld.h

INPUT/OUTPUT

UF_SF_mesh_error_container_p_t error_container :

the structure holding the mesh errors found in mesh validation.

RETURN

void

```
void UF_SF_clean_mshvld_error_container
(
    UF_SF_mesh_error_container_p_t container
)
```

UF_SF_mesh_error_container_p_t	**container**

UF_SF_clone_scenario (view source)

Defined in: uf_sf.h

Overview

This routine clone a Scenario

Environment

Internal and External

History

```
int UF_SF_clone_scenario
(
    tag_t master_part_tag,
    const char * orig_scen,
```

```
    const char * cloned_scen
)
```

tag_t	**master_part_tag**	Input	master part tag
const char *	**orig_scen**	Input	original scenario
const char *	**cloned_scen**	Input	name of the clone

UF_SF_close_scenario (view source)

Defined in: uf_sf.h

Overview
This routine closes the current Scenario which is opened when using UF_SF_open_scenario

Environment
Internal and External

History

```
int UF_SF_close_scenario
(
    void
)
```

UF_SF_count_all_nodes_of_valid_elements (view source)

Defined in: uf_sf.h

Overview
Counts number of all nodes of valid elements of current solver in the current part.

Environment
Internal and External

History
Originally released in NX11.0

```
int UF_SF_count_all_nodes_of_valid_elements
(
    int* number_of_all_valid_nodes
)
```

int*	**number_of_all_valid_nodes**	Output	Number of all valid nodes found.

UF_SF_count_all_valid_elements (view source)

Defined in: uf_sf.h

Overview
Counts number of all valid elements of current solver in the current part.

Environment

Internal and External

History
Originally released in NX11.0

---

int UF_SF_count_all_valid_elements
(
int* number_of_all_valid_elements
)

int*	**number_of_all_valid_elements**	Output	Number of valid elements found.

UF_SF_count_elements (view source)

Defined in: uf_sf.h

Overview
Counts number of elements in a mesh. If mesh_tag is NULL_TAG, counts all elements in the current part.

Environment
Internal and External

History
---

int UF_SF_count_elements
(
tag_t mesh,
int* number_of_elements
)

tag_t	**mesh**	Input	The tag of mesh. If passed NULL_TAG then all meshes in the current part will be traversed
int*	**number_of_elements**	Output	Number of elements found.

UF_SF_count_nodes (view source)

Defined in: uf_sf.h

Overview
Counts nodes in a mesh. If mesh_tag is NULL_TAG, counts all nodes in the current part.

Environment
Internal and External

int UF_SF_count_nodes
(
tag_t mesh,
int* number_of_nodes
)

tag_t	**mesh**	Input	The tag of mesh. If passed NULL_TAG then all meshes in the current part will be

			traversed
int*	**number_of_nodes**	Output	Number of nodes found.

UF_SF_create_0d_mesh (view source)

Defined in: uf_sf.h

Overview

Creates a 0-D mesh (zero dimensional elements) over selected geometry objects. Concentrated Mass is the only element type available for 0-D meshing.

Return

Return code:
0 = No error
not 0 = Error code
UF_SF_INVALID_INPUT
UF_SF_NO_MESH_RECIPE
UF_SF_UPDATE_FAILURE

Environment

Internal and External

```
int UF_SF_create_0d_mesh
(
    int num_geom_objects,
    tag_t* geom_array,
    UF_SF_0D_element_type_t element_type,
    double default_density,
    UF_SF_0D_density_type_t default_density_type,
    double mass_value,
    tag_t* mesh_tag
)
```

int	**num_geom_objects**	Input	The number of geometry objects to be meshed
tag_t*	**geom_array**	Input	pointer to an array of tags of geometry objects
UF_SF_0D_element_type_t	**element_type**	Input	Type of element UF_SF_0D_CONMASS = Concentrated Mass Element
double	**default_density**	Input	The element density.
UF_SF_0D_density_type_t	**default_density_type**	Input	Default edge density to the element UF_SF_0D_EDGE_DENSITY_SIZE = Default size of the element UF_SF_0D_EDGE_DENSITY_NUMBER = Default number of elements
double	**mass_value**	Input	Mass value of the element
tag_t*	**mesh_tag**	Output	Tag of the mesh created

UF_SF_create_0d_mesh_dist_mass (view source)

Defined in: uf_sf.h

Overview

Creates a 0-D mesh (zero dimensional elements) over selected geometry objects. Concentrated Mass is the only element type available for 0-D meshing. This function is different from the above in that this function takes into account the distribute mass parameter

Return

Return code:
0 = No error
not 0 = Error code
UF_SF_INVALID_INPUT
UF_SF_NO_MESH_RECIPE
UF_SF_UPDATE_FAILURE

Environment

Internal and External

```
int UF_SF_create_0d_mesh_dist_mass
(
    int num_geom_objects,
    tag_t* geom_array,
    UF_SF_0D_element_type_t element_type,
    double default_density,
    UF_SF_0D_density_type_t default_density_type,
    double mass_value,
    int distribute_mass,
    tag_t* mesh_tag
)
```

int	**num_geom_objects**	Input	The number of geometry objects to be meshed
tag_t*	**geom_array**	Input	pointer to an array of tags of geometry objects
UF_SF_0D_element_type_t	**element_type**	Input	Type of element UF_SF_0D_CONMASS = Concentrated Mass Element
double	**default_density**	Input	The element density.
UF_SF_0D_density_type_t	**default_density_type**	Input	Default edge density to the element UF_SF_0D_EDGE_DENSITY_SIZE = Default size of the element UF_SF_0D_EDGE_DENSITY_NUMBER = Default number of elements
double	**mass_value**	Input	Mass value of the element
int	**distribute_mass**	Input	Whether the mass is distributed among the elements
tag_t*	**mesh_tag**	Output	Tag of the mesh created

UF_SF_create_1d_connection_mesh (view source)

Defined in: uf_sf.h

Overview

Creates a one dimensional (1D) mesh over two groups of selected geometry objects. One dimensional elements are defined as two noded elements. The element types available for 1-D meshing are: Bar, Beam, Rod, Spring, and Rigid Link.

NOTE: This is the same as UF_SF_create_1d_mesh except for the additional direction arguments. This allows the user to create connection meshes between edges and specify the correct directions. These additional arguments are not needed if num_group2_objects == 0, or if the items in group2 are points.

Environment
Internal and External

```
int UF_SF_create_1d_connection_mesh
(
    UF_SF_1d_mesh_data_t* mesh_data,
    UF_SF_orientation_data_t* orient_data,
    int num_group1_objects,
    tag_t* group1_array,
    UF_SF_mesh_geom_meshdir_p_t group1_direction_array,
    int num_group2_objects,
    tag_t* group2_array,
    UF_SF_mesh_geom_meshdir_p_t group2_direction_array,
    tag_t* mesh_tag
)
```

UF_SF_1d_mesh_data_t*	**mesh_data**	Input	pointer to the structure UF_SF_1d_mesh_data_t
UF_SF_orientation_data_t*	**orient_data**	Input	pointer to the structure UF_SF_orientation_data_t. Pass NULL for default orientation.
int	**num_group1_objects**	Input	Number of geometry objects in group1
tag_t*	**group1_array**	Input	pointer to an array of tags to the group1 objects
UF_SF_mesh_geom_meshdir_p_t	**group1_direction_array**	Input	pointer to an array of directions for group1_array. Must be either UF_SF_MG_MESHDIR_FROM_START or UF_SF_MG_MESHDIR_FROM_END. Not needed if num_group2_objects == 0
int	**num_group2_objects**	Input	Number of geometry objects in group2
tag_t*	**group2_array**	Input	pointer to an array of tags to group2 objects
UF_SF_mesh_geom_meshdir_p_t	**group2_direction_array**	Input	pointer to an array of directions for group2_array. Must be either UF_SF_MG_MESHDIR_FROM_START or UF_SF_MG_MESHDIR_FROM_END
tag_t*	**mesh_tag**	Output	Tag of the mesh created

UF_SF_create_1d_mesh (view source)

Defined in: uf_sf.h

Overview
Creates a one dimensional (1D) mesh over selected geometry objects. One dimensional elements are defined as two noded elements. The element types available for 1-D meshing are: Bar, Beam, Rod, Spring, and Rigid Link.

Environment
Internal and External

```
int UF_SF_create_1d_mesh
(
    UF_SF_1d_mesh_data_t* mesh_data,
    UF_SF_orientation_data_t* orient_data,
    int num_group1_objects,
    tag_t* group1_array,
    int num_group2_objects,
    tag_t* group2_array,
    tag_t* mesh_tag
```

```
)
```

UF_SF_1d_mesh_data_t*	**mesh_data**	Input	pointer to the structure UF_SF_1d_mesh_data_t
UF_SF_orientation_data_t*	**orient_data**	Input	pointer to the structure UF_SF_orientation_data_t. Pass NULL for default orientation.
int	**num_group1_objects**	Input	Number of geometry objects in group1
tag_t*	**group1_array**	Input	pointer to an array of tags to the group1 objects
int	**num_group2_objects**	Input	Number of geometry objects in group2
tag_t*	**group2_array**	Input	pointer to an array of tags to group2 objects
tag_t*	**mesh_tag**	Output	Tag of the mesh created

UF_SF_create_auto_face_subdiv (view source)

Defined in: uf_sf.h

Overview

Given an array of solid bodies or faces (but not both). this function automatically creates the information of the pairs of faces that are mated within the given 'distance_tolerance'. It then uses this information to subdivide the overlapping face pairs using the opposite edges of each face pair. Finally, it outputs the faces pairs that became coincident as a result of subdivision. In addition, it also outputs the face pairs that are already coincident with or without an equal number of edges. However, there are two options as described below to control the output.

1. subdivision_needed - The user can specify whether the subdivision is needed or not. If it is not needed, only those face pairs that are found to be already coincident will be output. (1 for needed, 0 otherwise)
2. face_search_option - It gives the caller an option to specify whether all types of face pairs needs to be searched or only coincident pairs are of interest. (1 for search all faces, 0 for search only coincident ones)

When face pairs are coincident with an equal number of edges, the relative_status in the output structure 'resulting_pairs' is UF_SF_IDENTICAL_FACES_EQ_EDGES. Further, when the face pairs are coincident with unequal number of edges, the relative_status in the output structure is UF_SF_IDENTICAL_FACES_UNEQ_EDGES.

Caller of this function is responsible for freeing up the memory of the output structure array 'resulting_pairs'.

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_create_auto_face_subdiv
(
    tag_t * objects,
    int obj_count,
    double distance_tolerance,
    int subdivision_needed,
    int face_search_option,
    UF_SF_resulting_face_pairs_p_t * resulting_pairs,
    int * resulting_pairs_count
)
```

tag_t *	**objects**	Input	An array of solid bodies or faces

int	**obj_count**	Input	Count of the above array
double	**distance_tolerance**	Input	Tolerance for face pairing
int	**subdivision_needed**	Input	Whether subdivision needed 1 for YES, 0 for NO
int	**face_search_option**	Input	1 for all faces 0 for coincident faces only
UF_SF_resulting_face_pairs_p_t *	**resulting_pairs**	Output to UF_*free*	Array of face pair structures. This must be freed by calling UF_free.
int *	**resulting_pairs_count**	Output	Count of the above structures

---

UF_SF_create_combined_load_case (view source)

Defined in: uf_sf.h

Overview

Creates a combined load case with a given name and definition.

Valid only when the work part is a Simulation.

Environment

Internal and External

```
int UF_SF_create_combined_load_case
(
    char* clc_name,
    int num_comps,
    tag_t* lbc_tags,
    double* lbc_scales,
    tag_t* clc_tag
)
```

char*	**clc_name**	Input	Name of the combined load case.
int	**num_comps**	Input	Number of load cases in the combined load case definition.
tag_t*	**lbc_tags**	Input	pointer to the array of the tags for load case components.
double*	**lbc_scales**	Input	pointer to the array of the scale factors for load case components.
tag_t*	**clc_tag**	Output	Tag of the combined load case.

UF_SF_create_defeature_body (view source)

Defined in: uf_sf.h

Overview

This routine creates a defeature feature for specified set of faces of the body that either define a region or are explicitly selected, or both.

This routine takes a region object and an idealize parms object that is set to the desired operations and creates an idealize feature.
Since removal of all specified faces and edges is not guarenteed, returned are the edges of all failing wounds (edges of the face or hole

edge in a sheet body).

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_create_defeature_body
(
    UF_SF_defeature_parms_p_t defeature_parms_p,
    tag_t * feature_tag,
    int * n_failing_wound_edges,
    tag_t * * failing_wound_edges
)
```

UF_SF_defeature_parms_p_t	**defeature_parms_p**	Input	Parameters that define a set of faces on a body to be simplified. The body is derived from a seed face. The faces may be extracted automatically based on the parameters that define a region, or they can be specified explicitly, or both.
tag_t *	**feature_tag**	Output	The feature created
int *	**n_failing_wound_edges**	Output	Number of edges of failing wounds
tag_t * *	**failing_wound_edges**	Output to UF_*free*	Array of edges of failing wounds. This must be freed by calling UF_free.

UF_SF_create_dursol_nx (view source)

Defined in: uf_sf.h

Overview
This function will create a Durability Solution.
From NX7.5, this function will create a durability metasolution and a default static event referring to the solution tag.

Richer features are available in NXOpen.
Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_create_dursol_nx
(
    const char* name,
    tag_t solution_tag,
    tag_t * dursol_tag
)
```

const char*	**name**	Input	Durability Solution name.
tag_t	**solution_tag**	Input	Tag of the solution.
tag_t *	**dursol_tag**	Output	Durability Solution tag.

UF_SF_create_fem (view source)

Defined in: uf_sf.h

Overview
Creates fem part file

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_create_fem
(
    const char* fem_name,
    tag_t cad_part_tag,
    const char* idealized_part_name,
    logical use_all_bodies_flag,
    int num_bodies,
    const tag_t* body_tags,
    const char* solver_name,
    const char* analysis_type_name,
    int num_desc_lines,
    const char* * description,
    tag_t* new_fem_tag
)
```

const char*	**fem_name**	Input	Name of FEM part file to create.
tag_t	**cad_part_tag**	Input	Tag of the CAD part to associate the FEM to. Pass NULL_TAG to create a stand-alone FEM part
const char*	**idealized_part_name**	Input	Name of the Idealized part. Pass NULL for no Idealized part
logical	**use_all_bodies_flag**	Input	Flag indicating whether to use all bodies or not
int	**num_bodies**	Input	Number of bodies to use in creating the FEM Valid only when use_all_bodies_flag == false
const tag_t*	**body_tags**	Input	List of body tags to use in creating the FEM Valid only when use_all_bodies_flag == false
const char*	**solver_name**	Input	The name of the solver. Available solvers are: "NX NASTRAN", "MSC NASTRAN", "ANSYS", "ABAQUS", "NX THERMAL / FLOW" "NX NASTRAN DESIGN"
const char*	**analysis_type_name**	Input	The name of the analysis type. Available analysis types are: For "NX NASTRAN", "MSC NASTRAN", "ANSYS" & "ABAQUS": "Structural", "Thermal", "Axisymmetric Structural", "Axisymmetric Thermal" For "NX THERMAL / FLOW": "Thermal" "Flow" "Coupled Thermal-Flow" For "NX NASTRAN DESIGN": "Structural", "Thermal"
int	**num_desc_lines**	Input	Number of lines in the FEM description.

const char* *	**description**	Input	Description of this FEM part.
tag_t*	**new_fem_tag**	Output	Tag of created FEM part file.

UF_SF_create_fem_with_geom_opts (view source)

Defined in: uf_sf.h

Overview
Creates fem part file with geometry options

Environment
Internal and External

History
Originally released in NX4.0.4

```
int UF_SF_create_fem_with_geom_opts
(
    const char* fem_name,
    tag_t cad_part_tag,
    const char* idealized_part_name,
    logical use_all_bodies_flag,
    int num_bodies,
    const tag_t* body_tags,
    const char* solver_name,
    const char* analysis_type_name,
    int num_desc_lines,
    const char* * description,
    UF_SF_geom_options_t geom_options,
    tag_t* new_fem_tag
)
```

const char*	**fem_name**	Input	Name of FEM part file to create.
tag_t	**cad_part_tag**	Input	Tag of the CAD part to associate the FEM to. Pass NULL_TAG to create a stand-alone FEM part
const char*	**idealized_part_name**	Input	Name of the Idealized part. Pass NULL for no Idealized part
logical	**use_all_bodies_flag**	Input	Flag indicating whether to use all bodies or not
int	**num_bodies**	Input	Number of bodies to use in creating the FEM Valid only when use_all_bodies_flag == true
const tag_t*	**body_tags**	Input	List of body tags to use in creating the FEM Valid only when use_all_bodies_flag == true
const char*	**solver_name**	Input	The name of the solver. Available solvers are: "NX NASTRAN", "MSC NASTRAN", "ANSYS", "ABAQUS", "NX THERMAL / FLOW" "NX NASTRAN DESIGN"
const char*	**analysis_type_name**	Input	The name of the analysis type. Available analysis types are: For "NX NASTRAN", "MSC NASTRAN", "ANSYS" & "ABAQUS": "Structural", "Thermal", "Axisymmetric Structural", "Axisymmetric Thermal" For "NX THERMAL / FLOW": "Thermal"

			"Flow" "Coupled Thermal-Flow" For "NX NASTRAN DESIGN": "Structural", "Thermal"
int	**num_desc_lines**	Input	Number of lines in the FEM description.
const char* *	**description**	Input	Description of this FEM part.
UF_SF_geom_options_t	**geom_options**	Input	Geometry options
tag_t*	**new_fem_tag**	Output	Tag of created FEM part file.

UF_SF_create_hardpoint_on_geom (view source)

Defined in: uf_sf.h

Overview
Creates a hardpoint associated to either a face, an edge or curve. This hardpoint will then be honored by any meshing operation performed on that edge or face.

Environment
Internal or External

History
Released in NX3

```
int UF_SF_create_hardpoint_on_geom
(
    tag_t geom_tag,
    double ref_point [ 3 ] ,
    tag_p_t hardpoint_tag
)
```

tag_t	**geom_tag**	Input	The tag of the geometry that the hardpoint should be associated to. This geometry can be either a face, an edge or a curve.
double	**ref_point [3]**	Input	The 3d location of reference point in absolute csys. The actual hardpoint location will be determined by projecting this point on to a face or computing minimum distance of this point to an edge or curve.
tag_p_t	**hardpoint_tag**	Output	The tag of the hardpoint created

---

UF_SF_create_idealize_body (view source)

Defined in: uf_sf.h

Overview
This routine creates an idealize feature for an entire body

This routine takes the tag of a body and an idealize parms object that is set to the desired operation and creates an idealize feature. Since removal of all specified faces and edges is not guaranteed, returned are the edges of all failing wounds (edges of the face or hole edge in a sheet body) idealize feature associated to a body region.

Environment
Internal and External

History

Originally released in V17.0

```
int UF_SF_create_idealize_body
(
    tag_t body_tag,
    UF_SF_idealize_parms_p_t parms_p,
    tag_t * feature_tag,
    int * n_failing_wound_edges,
    tag_t * * failing_wound_edges
)
```

tag_t	**body_tag**	Input	Body to be simplified (Solid or Sheet)
UF_SF_idealize_parms_p_t	**parms_p**	Input	Parameters of the new idealize feature
tag_t *	**feature_tag**	Output	The feature created
int *	**n_failing_wound_edges**	Output	Number of edges of failing wounds
tag_t * *	**failing_wound_edges**	Output to UF_*free*	Array of edges of failing wounds. This must be freed by calling UF_free.

UF_SF_create_idealize_region (view source)

Defined in: uf_sf.h

Overview

This routine creates an idealize feature for specified region of a body

This routine takes a region object and an idealize parms object that is set to the desired operations and creates an idealize feature. Since removal of all specified faces and edges is not guaranteed, returned are the edges of all failing wounds (edges of the face or hole edge in a sheet body) idealize feature associated to a body region.

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_create_idealize_region
(
    UF_SF_idealize_region_p_t region_p,
    UF_SF_idealize_parms_p_t parms_p,
    tag_t * feature_tag,
    int * n_failing_wound_edges,
    tag_t * * failing_wound_edges
)
```

UF_SF_idealize_region_p_t	**region_p**	Input	Parameters that define the region of the body to be simplified.
UF_SF_idealize_parms_p_t	**parms_p**	Input	Parameters of the new idealize feature
tag_t *	**feature_tag**	Output	The feature created
int *	**n_failing_wound_edges**	Output	Number of edges of failing wounds
tag_t * *	**failing_wound_edges**	Output to UF_*free*	Array of edges of failing wounds. This must be freed by calling UF_free.

UF_SF_create_lv_nx (view source)

Defined in: uf_sf.h

Overview

This function will create a Durability load variation.
From NX7.5, this function will create a durability load variation to the first available static event. In case the event is not available, an error will be returned.
Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_create_lv_nx
(
    const char * lv_name,
    double scaling_factor,
    int count,
    UF_SF_lv_function_mode_t function_type,
    tag_t solution_step_tag,
    tag_t dursol_tag,
    tag_t * lv_tag
)
```

const char *	**lv_name**	Input	Load variation name If lv_name = NULL, a default name will be assigned.
double	**scaling_factor**	Input	Scaling factor
int	**count**	Input	Number of cycles
UF_SF_lv_function_mode_t	**function_type**	Input	Scaling function type UF_SF_HALF_UNIT_CYCLE UF_SF_FULL_UNIT_CYCLE
tag_t	**solution_step_tag**	Input	Solution Step tag
tag_t	**dursol_tag**	Input	Durablity Solution tag
tag_t *	**lv_tag**	Output	Load Variation tag

UF_SF_create_material (view source)

Defined in: uf_sf.h

Overview

Creates either an isotropic, orthotropic, anisotropic or fluid material, with, or without table properties, with, or without read-only constraints.

Initialize the UF_SF_material_prop_p_t, property_values structure using UF_SF_init_matl_prop before setting new values.

Environment

Internal and External

See Also

html#UF_SF_init_matl_prop and
html#UF_SF_free_matl_prop

```
int UF_SF_create_material
(
    char* name,
    char* category,
    UF_SF_neutral_material_types_t material_type,
    UF_SF_material_prop_p_t property_values,
    UF_SF_library_material_t library_material,
    tag_p_t material_tag
)
```

char*	**name**	Input	Name of material. The maximum number of allowable characters is 80.
char*	**category**	Input	Category of material, pass NULL if it is not required
UF_SF_neutral_material_types_t	**material_type**	Input	The type of material to create. UF_SF_MATERIAL_ISOTROPIC = Isotropic material. UF_SF_MATERIAL_ORTHOTROPIC = Orthotropic material, UF_SF_MATERIAL_ANISOTROPIC = Anisotropic material. UF_SF_MATERIAL_FLUID = Fluid material.
UF_SF_material_prop_p_t	**property_values**	Input	Input structure with Youngs modulus, poissons ratio, Mass density, Thermal conductivity, and Thermal Coefficient. Call "UF_SF_init_matl_prop" to initialize this structure before setting new values.
UF_SF_library_material_t	**library_material**	Input	UF_SF_MATL_IS_READ_ONLY UF_SF_MATL_IS_EDITABLE
tag_p_t	**material_tag**	Output	Created material tag.

UF_SF_create_mesh_mating_condition (view source)

Defined in: uf_sf.h

Overview
Create a SFEM assembly_mesh object.

Environment
Internal and External

History
This function was originally released in V16.0

```
int UF_SF_create_mesh_mating_condition
(
    tag_t assembling_geom,
    tag_t target_geom,
    char* assembly_name,
    int seed_mesh,
    UF_SF_assembly_type_t assembly_type,
    double merge_tolerance,
    double match_tolerance,
    tag_t * assembly_mesh
)
```

tag_t	**assembling_geom**	Input	Assembling region polygon geometry tag
tag_t	**target_geom**	Input	Target region polygon geometry tag

char*	**assembly_name**	Input	Assembly mesh name
int	**seed_mesh**	Input	Flag of seed mesh
UF_SF_assembly_type_t	**assembly_type**	Input	Assembly mesh type
double	**merge_tolerance**	Input	Node merge tolerance
double	**match_tolerance**	Input	match tolerance to define if an independent node matchs with a master element
tag_t *	**assembly_mesh**	Output	Tag of the newly created assembly mesh

---

UF_SF_create_offset_midsrf (view source)

Defined in: uf_sf.h

Overview

This function creates a midsurface using offset method. This creation requires the user to specify the following:

1) A Seed Face - This is the face on the body from which the function propagates and collects all the faces based on the following criteria:

a) Cliff Angle: The propagation halts and does not take the face concerned in to the collection if the angle between the normals of the current face and this face is greater than the cliff angle. Please note that this is the only criterion which is given as input.

NOTE: There are other criteria which the system computes such as the thickness of the body at the seed face, thickness of the body adjacent to the seed face and also the set of edges which belong to the set of faces opposite to the propagation.

2) The percentage distance is the percentage of the thickness (distance) at which the midsurface will be placed from the seed face, after creation within the body.

Environment

Internal and External

History

Originally released in V16.0

```
int UF_SF_create_offset_midsrf
(
    tag_t seed_face,
    double cliff_angle,
    double percentage_dist,
    tag_t * midsurface_tag
)
```

tag_t	**seed_face**	Input	The tag of the seed face on the solid body.
double	**cliff_angle**	Input	The user input cliff angle to define the boundary
double	**percentage_dist**	Input	Percentage location of midsurface
tag_t *	**midsurface_tag**	Output	Midsurface sheet body tag that was created.

UF_SF_create_report (view source)

Defined in: uf_sf.h

Overview
Creates the Scenario report objects based on a template file.
Valid only when a Simulation part is the work part

Environment
Internal or External

History
Released in NX3

```
int UF_SF_create_report
(
    void
)
```

UF_SF_create_scenario_nx (view source)

Defined in: uf_sf.h

Overview
Creates scenario part file from parent part (non scenario part file).
The input name should be shorter than 26 characters and should not include the path name. This is required because the scenario subdirectory is associated to the master model part.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_create_scenario_nx
(
    const char* scenario_name,
    tag_t* new_scenario_tag
)
```

const char*	**scenario_name**	Input	Name of Scenario part file to create.
tag_t*	**new_scenario_tag**	Output	Tag of created Scenario part file.

UF_SF_create_simulation (view source)

Defined in: uf_sf.h

Overview
Creates Simulation part file

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_create_simulation
(
    const char* simulation_name,
    tag_t fem_tag,
    int num_desc_lines,
    const char* * description,
    tag_t* new_simulation_tag
)
```

const char*	**simulation_name**	Input	Name of Simulation part file to create.
tag_t	**fem_tag**	Input	Tag of the FEM part to create the Simulation on.
int	**num_desc_lines**	Input	Number of lines in the Simulation description.
const char* *	**description**	Input	Description of this Simulation part.
tag_t*	**new_simulation_tag**	Output	Tag of created Simulation part file.

UF_SF_create_solution_nx (view source)

Defined in: uf_sf.h

Overview
Creates a solution with a given name.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_create_solution_nx
(
    tag_t solution_desc,
    tag_t solver_desc,
    const char* solution_name,
    tag_t* solution_tag
)
```

tag_t	**solution_desc**	Input	Tag for the solution descriptor.
tag_t	**solver_desc**	Input	Tag for the solver descriptor.
const char*	**solution_name**	Input	Name for the solution.
tag_t*	**solution_tag**	Output	Tag for the solution.

UF_SF_create_step_nx (view source)

Defined in: uf_sf.h

Overview
Creates a step with a given name.

Steps are subdivisions of a CAE solution.
UF_SFL_solution_ask_step_descriptor_nx - Given a solution descriptor tag and allowable step descriptor name, returns the tag of the step descriptor.

If the user does not know the step descriptor name, they can use the following:
UF_SFL_solution_ask_num_allowable_step_descriptors_nx

UF_SFL_solution_ask_nth_allowable_step_descriptor_nx
UF_SFL_step_descriptor_ask_name_nx

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_create_step_nx
(
    tag_t step_desc,
    const char* step_name,
    tag_t solution_tag,
    tag_t* step_tag
)
```

tag_t	**step_desc**	Input	Tag for the step descriptor.
const char*	**step_name**	Input	Name for the step.
tag_t	**solution_tag**	Input	Tag for the parent solution.
tag_t*	**step_tag**	Output	Tag for the step.

UF_SF_create_surface_contact_mesh (view source)

Defined in: uf_sf.h

Overview
Creates a surface contact mesh for a given source and target polygon face. The source and target polygon face should belong to a different polygon body.

Environment
Internal and External

History

```
int UF_SF_create_surface_contact_mesh
(
    tag_t source_face,
    tag_t target_face,
    void * property,
    tag_p_t mesh_recipe
)
```

tag_t	**source_face**	Input	The source side polygon face on which contact need to be created.
tag_t	**target_face**	Input	The target side polygon face on which contact need to be created.
void *	**property**	Input	Structure to define mesh properties. The valid structures are UF_SF_SURFACE_CONTACT_ANS_data_p_t, UF_SF_SURFACE_CONTACT_NAS_data_p_t, UF_SF_SURFACE_CONTACT_UGFEA_data_p_t
tag_p_t	**mesh_recipe**	Output	Pointer to the mesh recipe tag created.

UF_SF_create_swept_hex_mesh (view source)

Defined in: uf_sf.h

Overview
This function will create a 3d Hex mesh in a solid body.

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_create_swept_hex_mesh
(
    tag_t solid_body,
    tag_t source_face,
    logical midnodes,
    double elem_size,
    tag_p_t mesh_tag
)
```

tag_t	**solid_body**	Input	The tag of the polygon solid body to mesh
tag_t	**source_face**	Input	The tag of the face to use as the source polygon face for sweeping.
logical	**midnodes**	Input	If TRUE, Hex20s will be created. If FALSE, Hex8s will be created.
double	**elem_size**	Input	Desired element size. This will be used to mesh the source polygon face and determine the number of sweeping layers.
tag_p_t	**mesh_tag**	Output	The tag of the mesh created.

UF_SF_create_userdef_midsrf (view source)

Defined in: uf_sf.h

Overview
This function creates a midsurface feature using the user defined method. This creation requires the user to specify the following via an instance of UF_SF_midsef_userdef_parms_t structure:

1) A target solid body

2) A collection of one or more sheet bodies.

3) The thickness of the midsurface which will be used only for the thickness extraction at formatting time only if the automatic method fails if at least one of the specified sheet bodies is not fully contained in the target solid.

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_create_userdef_midsrf
(
    UF_SF_midsrf_user_parms_p_t parms_p,
    tag_t * feature_tag
)
```

UF_SF_midsrf_user_parms_p_t	**parms_p**	Input	Parameters of the new user defined midsurface feature
tag_t *	**feature_tag**	Output	Tag of user defined midsurface feature.

UF_SF_create_weld_mesh (view source)

Defined in: uf_sf.h

Overview
Creates weld (1D) mesh over selected geometry objects.
weld elements are defined as two noded elements. The
element types available for 1-D meshing are: Bar, Beam, Rod,
Spring, and Rigid Link.

Environment
Internal and External

```
int UF_SF_create_weld_mesh
(
    UF_SF_1d_mesh_data_t* mesh_data,
    UF_SF_orientation_data_t* orient_data,
    int num_objects,
    tag_t* obj_array,
    int num_top_faces,
    tag_t* top_face_array,
    int num_bot_faces,
    tag_t* bot_face_array,
    tag_t* mesh_tag
)
```

UF_SF_1d_mesh_data_t*	**mesh_data**	Input	UF_SF_1d_mesh_data_t
UF_SF_orientation_data_t*	**orient_data**	Input	Pass NULL for default orientation not required.
int	**num_objects**	Input	Number of points/curves.
tag_t*	**obj_array**	Input	Array of tags to the points/curves.
int	**num_top_faces**	Input	Number of top polygon faces.
tag_t*	**top_face_array**	Input	Array of tags to the top polygon faces.
int	**num_bot_faces**	Input	Number of bottom polygon faces.
tag_t*	**bot_face_array**	Input	Array of tags to the bottom polygon faces.
tag_t*	**mesh_tag**	Output	Tag of new mesh.

UF_SF_CURVE_create_arc_3point (view source)

Defined in: uf_sf_curve.h

Overview
Create a smart/dumb arc through three points in fem/sim part.

Return
error code

Environment

Internal and External

```
int UF_SF_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_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_id**	Output	if is_asso == TRUE - smart arc if is_asso == FALSE - dumb arc arc

---

UF_SF_CURVE_create_arc_center_radius (view source)

Defined in: uf_sf_curve.h

Overview

Create a smart/dumb arc of given radius and specific center in fem/sim part.

Return

error code

Environment

Internal and External

```
int UF_SF_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_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_id**	Output	if is_asso == TRUE - object identifier of new smart arc if is_asso == FALSE - object identifier of new dumb arc

---

UF_SF_CURVE_create_arc_point_center (view source)

Defined in: uf_sf_curve.h

Overview
Create a smart/dumb arc through a start point and specific center in fem/sim part.

Return
error code

Environment
Internal and External

```
int UF_SF_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_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_id**	Output	if is_asso == TRUE - object identifier of new smart arc if is_asso == FALSE - object identifier of new dumb arc

---

UF_SF_CURVE_create_arc_point_point_radius (view source)

Defined in: uf_sf_curve.h

Overview
Create a smart/dumb arc through two points and of specific radius in fem/sim part.

Return
error code

Environment
Internal and External

```
int UF_SF_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 smart arc feature if is_asso == FALSE - object identifier of new dumb arc

---

UF_SF_CURVE_create_line_arc (view source)

Defined in: uf_sf_curve.h

Overview

Create a smart/dumb line/arc in fem/sim part. 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 smart/dumb line/arc. An error is returned if line/arc cannot be created with the specified data.

Return

error code

Environment

Internal and External

```
int UF_SF_CURVE_create_line_arc
(
    UF_CURVE_line_arc_t * line_arc_data,
    tag_t * curve_id
)
```

UF_CURVE_line_arc_t *	**line_arc_data**	Input	Pointer to line/arc data structure
tag_t *	**curve_id**	Output	Object identifier of new associative line/arc in fem/sim. if is_asso == TRUE - smart line/arc. if is_asso == FALSE - dumb line/arc

---

UF_SF_CURVE_create_line_point_point (view source)

Defined in: uf_sf_curve.h

Overview

Create a smart/dumb line through two points in fem/sim part.

Return

error code

Environment

Internal and External

```
int UF_SF_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 smart line if is_asso == FALSE - object identifier of new dumb line

---

UF_SF_delete_disp_results (view source)

Defined in: uf_sf_ugopenint.h

Overview
Deletes the fringe display of the results associated with a mesh and optionally the associated legend.

Environment
Internal and External

History
Originally released in V19.0

```
int UF_SF_delete_disp_results
(
    int delete_legend_sw
)
```

int	**delete_legend_sw**	Input	Switch indicating whether to delete legend along with the fringe display of the results: = 0 --> Do not delete legend = 1 --> Delete legend

---

UF_SF_delete_dursol_nx (view source)

Defined in: uf_sf.h

Overview
This function will delete a Durability solution.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_delete_dursol_nx
(
    tag_t dursol_tag
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag.

UF_SF_delete_lv_nx (view source)

Defined in: uf_sf.h

Overview
This function will delete a Durability load variation.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_delete_lv_nx
(
    tag_t dursol_tag,
    tag_t lv_tag
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
tag_t	**lv_tag**	Input	Load Variation tag

UF_SF_delete_mesh_mating_condition (view source)

Defined in: uf_sf.h

Overview
Delete a SFEM assembly_mesh object.

Environment
Internal and External

```
int UF_SF_delete_mesh_mating_condition
(
    int num_mmc,
    tag_t mmc_array [ ]
)
```

int	**num_mmc**	Input	Number of MMCs to be deleted
tag_t	**mmc_array [ ]**	Input	Array of MMC tags to be deleted

---

UF_SF_delete_scenario (view source)

Defined in: uf_sf.h

Overview
This routine deletes the specified Scenario

Environment
Internal and External

History

```
int UF_SF_delete_scenario
(
    tag_t master_tag,
    const char * scenario_name
)
```

tag_t	**master_tag**	Input	master part tag
const char *	**scenario_name**	Input	scenario to be deleted

UF_SF_delete_solution_nx (view source)

Defined in: uf_sf.h

Overview
Delete a solution.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_delete_solution_nx
(
    tag_t solution
)
```

tag_t	**solution**	Input	Tag for the solution.

UF_SF_delete_step_nx (view source)

Defined in: uf_sf.h

Overview
Delete a step.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_delete_step_nx
(
    tag_t step
)
```

tag_t	**step**	Input	Tag for the step.

UF_SF_display_mesh (view source)

Defined in: uf_sf.h

Overview
Displays a mesh object.

Environment
Internal and External

History
Originally released in V19.0

```
int UF_SF_display_mesh
(
    tag_t mesh_tag
)
```

tag_t	**mesh_tag**	Input	Tag of mesh object to be displayed.

UF_SF_dursol_add_load_nx (view source)

Defined in: uf_sf.h

Overview
This function will add a Durability Load to a Durability Solution.
Valid only when the work part is a Simulation.
From NX7.5, this command will do nothing. A load when created has been automatically added to the static event it belongs to.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_dursol_add_load_nx
(
    tag_t dursol_tag,
    tag_t lv_tag
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
tag_t	**lv_tag**	Input	Load Variation tag

UF_SF_dursol_ask_load_count_nx (view source)

Defined in: uf_sf.h

Overview

This function returns the number of Load Variations defined for the specific Durability Solution.
From NX7.5, this function will return the number of load variations on the first available static event in the durability solution.
Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_dursol_ask_load_count_nx
(
    tag_t dursol_tag,
    int * num_lv
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
int *	**num_lv**	Output	Number of load variations found.

UF_SF_dursol_ask_load_nx (view source)

Defined in: uf_sf.h

Overview

This function returns the tag of the specified Load Variation in the Durability Solution.
From NX7.5, this function will return the nth load variation on the first available static event in the durability solution.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_dursol_ask_load_nx
(
    tag_t dursol_tag,
    int lv_index,
    tag_t * lv_tag
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
int	**lv_index**	Input	The index of the Load Variation. Valid range is 0 : (number of Load Variations)-1
tag_t *	**lv_tag**	Output	Tag of Load Variation

UF_SF_dursol_locate_all_members_nx (view source)

Defined in: uf_sf.h

Overview

This function will return all the load variation object tags and the total count for the given Durability Solution.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_dursol_locate_all_members_nx
(
    tag_t dursol_tag,
    int* num_lv,
    tag_p_t* lv_pp
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
int*	**num_lv**	Output	Number of Load Variations
tag_p_t*	**lv_pp**	Output to UF_*free*	List of Load Variation tags If lv_pp != NULL, must be freed in the calling routine by calling UF_free.

UF_SF_dursol_remove_load_nx (view source)

Defined in: uf_sf.h

Overview

This function will remove a Durability Load from a Durability Solution.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_dursol_remove_load_nx
(
    tag_t dursol_tag,
    tag_t lv_tag
)
```

tag_t	**dursol_tag**	Input	Durability Solution tag
tag_t	**lv_tag**	Input	Load Variation tag

UF_SF_edge_ask_adjacent_edges (view source)

Defined in: uf_sf.h

Overview
This function will get adjacent polygon edges of input polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_adjacent_edges
(
    tag_t cae_tag,
    int * num_edges,
    tag_p_t * edges
)
```

tag_t	**cae_tag**	Input	Polygon edge
int *	**num_edges**	Output	Number of edges
tag_p_t *	**edges**	Output to UF_*free*	Polygon edges

UF_SF_edge_ask_body (view source)

Defined in: uf_sf.h

Overview
This function will polygon body related to polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_body
(
    tag_t cae_tag,
    tag_p_t body_p
)
```

tag_t	**cae_tag**	Input	Polygon edge
tag_p_t	**body_p**	Output	Polygon body

UF_SF_edge_ask_bounding_box (view source)

Defined in: uf_sf.h

Overview
This function will get bounding box of polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_bounding_box
(
    tag_t cae_tag,
    double pad_bounding_box [ 6 ]
)
```

tag_t	**cae_tag**	Input	Polygon edge
double	**pad_bounding_box [6]**	Output	Bounding box

UF_SF_edge_ask_end_points (view source)

Defined in: uf_sf.h

Overview
This function will end points and tangents of polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_end_points
(
    tag_t cae_tag,
    double start_pt [ 3 ] ,
    double end_pt [ 3 ] ,
    double start_tangent [ 3 ] ,
    double end_tangent [ 3 ]
)
```

tag_t	**cae_tag**	Input	Polygon edge
double	**start_pt [3]**	Output	Start point location
double	**end_pt [3]**	Output	End point location
double	**start_tangent [3]**	Output	Start tanget
double	**end_tangent [3]**	Output	End tangent

UF_SF_edge_ask_faces (view source)

Defined in: uf_sf.h

Overview
This function will get polygon faces related to polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_faces
(
    tag_t cae_tag,
    int * num_faces,
    tag_p_t * faces
```

```
)
```

tag_t	**cae_tag**	Input	Polygon edge
int *	**num_faces**	Output	Number of faces
tag_p_t *	**faces**	Output to UF_*free*	Polygon faces

UF_SF_edge_ask_length (view source)

Defined in: uf_sf.h

Overview
This function will find the length of polygon edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_ask_length
(
    tag_t cae_tag,
    double * pd_length
)
```

tag_t	**cae_tag**	Input	Polygon edge
double *	**pd_length**	Output	Length of edge

UF_SF_edge_evaluate_closest_point (view source)

Defined in: uf_sf.h

Overview
This function will project passed in point and finds param location on edge.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_evaluate_closest_point
(
    tag_t cae_tag,
    double ad_point [ 3 ] ,
    double ad_out_point [ 3 ] ,
    double * pd_param
)
```

tag_t	**cae_tag**	Input	Polygon edge
double	**ad_point [3]**	Input	Input point to project
double	**ad_out_point [3]**	Output	Projected point

double *	**pd_param**	Output	Param value between 0 to 1

UF_SF_edge_evaluate_param_location (view source)

Defined in: uf_sf.h

Overview
This function will find 3d location for given param value.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_edge_evaluate_param_location
(
    tag_t cae_tag,
    double d_param,
    double ad_out_point [ 3 ]
)
```

tag_t	**cae_tag**	Input	Polygon edge
double	**d_param**	Input	Param value between 0 to 1
double	**ad_out_point [3]**	Output	Corresponding point

UF_SF_edit_beam_orientation (view source)

Defined in: uf_sf.h

Overview
Given a bar/beam mesh and the orientation data this function updates
the mesh to the new orientation.

Environment
Internal and External

```
int UF_SF_edit_beam_orientation
(
    tag_t mesh_tag,
    UF_SF_orientation_data_t* orient_data
)
```

tag_t	**mesh_tag**	Input	Tag to a bar/beam mesh to update orientation.
UF_SF_orientation_data_t*	**orient_data**	Input	Mesh orientation data.

UF_SF_edit_defeature_parms (view source)

Defined in: uf_sf.h

Overview

This routine edits a defeature feature

This routine takes the feature tag of a defeature feature and edits it according to the new parmater object definition. Since removal of all specified faces and edges is not guarenteed, returned are the edges of all failing wounds (edges of the face or hole edge in a sheet body). idealize feature associated to a body region.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_edit_defeature_parms
(
    tag_t feature_tag,
    UF_SF_defeature_parms_p_t defeature_parms_p,
    int * n_failing_wound_edges,
    tag_t * * failing_wound_edges
)
```

tag_t	**feature_tag**	Input	The defeature feature
UF_SF_defeature_parms_p_t	**defeature_parms_p**	Input	Parameters that define a set of faces on a body to be simplified. The body is derived from a seed face. The faces may be extracted automatically based on the parameters that define a region, or they can be specified explicitly, or both.
int *	**n_failing_wound_edges**	Output	The number of failing wound edges
tag_t * *	**failing_wound_edges**	Output to UF_*free*	Array of failing wound edges. This must be freed by calling UF_free.

UF_SF_edit_dursol_nx (view source)

Defined in: uf_sf.h

Overview
This function will edit a Durability Solution's values.
From NX7.5, this function will edit the metasolution name, followed by parameters of the first static event. If the static event is not available, an error will be returned.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_edit_dursol_nx
(
    tag_t dursol_tag,
    const char* dursol_name,
    UF_SF_dursol_stress_criterion_t * stress_criterion,
    UF_SF_dursol_stress_type_t * stress_type,
    UF_SF_dursol_design_life_criterion_t * design_life_criterion,
    int * fatigue_cycles,
    double * k_factor,
    UF_SF_dursol_fatigue_life_criterion_t * fatigue_life_criterion,
    int * design_cycles
```

)

tag_t	**dursol_tag**	Input	Durability Solution tag.
const char*	**dursol_name**	Input	Name of Durability Solution. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters. If NULL, value will not be modified.
UF_SF_dursol_stress_criterion_t *	**stress_criterion**	Input	Stress criterion If NULL, value will not be modified.
UF_SF_dursol_stress_type_t *	**stress_type**	Input	Stress type If NULL, value will not be modified.
UF_SF_dursol_design_life_criterion_t *	**design_life_criterion**	Input	Design life criterion If NULL, value will not be modified.
int *	**fatigue_cycles**	Input	Number of fatigue duty cycles If NULL, value will not be modified.
double *	**k_factor**	Input	Fatigue strength factor (Kf) If NULL, value will not be modified.
UF_SF_dursol_fatigue_life_criterion_t *	**fatigue_life_criterion**	Input	Fatigue life criterion If NULL, value will not be returned.
int *	**design_cycles**	Input	Number of design cycles desired for fatigue strength. If NULL, value will not be returned.

UF_SF_edit_idealize_parms (view source)

Defined in: uf_sf.h

Overview

This routine edits idealize feature

This routine takes the feature tag of an idealize feature and edits it according to the new parameter object definition. Since removal of all specified faces and edges is not guaranteed, returned are the edges of all failing wounds (edges of the face or hole edge in a sheet body) idealize feature associated to a body region.

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_edit_idealize_parms
(
    tag_t feature_tag,
    UF_SF_idealize_parms_p_t parms_p,
    int * n_failing_wound_edges,
    tag_t * * failing_wound_edges
)
```

tag_t	**feature_tag**	Input	The idealize feature
UF_SF_idealize_parms_p_t	**parms_p**	Input	New parameters of the idealize feature
int *	**n_failing_wound_edges**	Output	The number of failing wound edges

tag_t * *	**failing_wound_edges**	Output to UF_*free*	Array of failing wound edges. This must be freed by calling UF_free.

UF_SF_edit_lv_nx (view source)

Defined in: uf_sf.h

Overview

This function will edit a Durability load variation's values.
From NX7.5, this function will modify the number of Occurrences on the parent static event using count.
Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_edit_lv_nx
(
    tag_t lv_tag,
    const char* lv_name,
    double * scaling_factor,
    int * count,
    UF_SF_lv_function_mode_t * function_type
)
```

tag_t	**lv_tag**	Input	Load Variation tag
const char*	**lv_name**	Input	The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters. If NULL, value will not be modified.
double *	**scaling_factor**	Input	Scaling factor If NULL, value will not be modified.
int *	**count**	Input	Number of cycles If NULL, value will not be modified.
UF_SF_lv_function_mode_t *	**function_type**	Input	Scaling function type UF_SF_HALF_UNIT_CYCLE UF_SF_FULL_UNIT_CYCLE If NULL, value will not be modified.

UF_SF_edit_mesh_mating_condition (view source)

Defined in: uf_sf.h

Overview

Edit a SFEM assembly_mesh object.

Environment

Internal and External

```
int UF_SF_edit_mesh_mating_condition
(
    int num_mmc,
    tag_t mmc_array [ ] ,
    int make_mesh_coincident,
    int mesh_mating_type
```

)

int	**num_mmc**	Input	Number of MMCs to be deleted
tag_t	**mmc_array [ ]**	Input	Array of MMC tags to be deleted
int	**make_mesh_coincident**	Input	coincident or non-coincident
int	**mesh_mating_type**	Input	Glue or Free = 0 --> GLUE = 3 --> FREE NOTE: There is no 1 or 2 Type

---

UF_SF_edit_offset_midsrf (view source)

Defined in: uf_sf.h

Overview

This function edits a midsurface which was created using offset method. This requires the user to specify the following:

Seed Face: Optional. This has to be specified if the user wants the seed face to be different from the original seed face of the midsurface. Otherwise, this parameter can be set to NULL.

Cliff Angle: If a different cliff angle has to be chosen, other than the one with which the midsurface was created. Otherwise, this can either be zero or it can be the original Cliff Angle.

NOTE: There are other criteria which the system computes such as the thickness of the body at the seed face, thickness of the body adjacent to the seed face and also the set of edges which belong to the set of faces opposite to the propagation.

percentage distance: is the percentage of the thickness (distance) at which the midsurface will be placed from the seed face, after creation within the body.

The output is the tag of the modified midsurface.

Environment

Internal and External

History

Originally released in V16.0

```
int UF_SF_edit_offset_midsrf
(
    tag_t seed_face,
    double cliff_angle,
    double percentage_dist,
    tag_t * midsurface_tag
)
```

tag_t	**seed_face**	Input	A tag of a seed face on the solid body. This may be different than the original seed face.
double	**cliff_angle**	Input	The user input cliff angle to define the boundary
double	**percentage_dist**	Input	Percentage location of midsurface
tag_t *	**midsurface_tag**	Input / Output	Midsurface sheet body tag

UF_SF_edit_userdef_midsrf (view source)

Defined in: uf_sf.h

Overview

This function edits a midsurface feature which was created using the user defined method.
This requires the user to specify the following via an instance of UF_SF_midsef_userdef_parms_t structure:

1) A target solid body. If no change set this parameter to NULL.

2) A collection of one or more sheet bodies.
The specified sheet bodies will redefine the midsurface.
If the pointer is NULL then the midsurface will remain the same.

3) The thickness of the midsurface which will be used only for the thickness extraction at formatting time only if the automatic method fails if at least one of the specified sheet bodies is not fully contained in the target solid. If a different thickness has to be chosen, other than the one with which the midsurface was created. Otherwise, this can either be zero or it can be the original thickness. If no change set this parameter to 0.

Environment

Internal and External

History

Originally released in NX2

```
int UF_SF_edit_userdef_midsrf
(
    UF_SF_midsrf_user_parms_p_t parms_p,
    tag_t * feature_tag
)
```

UF_SF_midsrf_user_parms_p_t	**parms_p**	Input	Parameters of the user defined midsurface feature to be edited.
tag_t *	**feature_tag**	Input / Output	Tag of user defined midsurface feature.

UF_SF_export_expression (view source)

Defined in: uf_sf.h

Overview

function to export the expressions to the master part.

```
int UF_SF_export_expression
(
    void
)
```

UF_SF_face_ask_adjacent_faces (view source)

Defined in: uf_sf.h

Overview

This function will get adjacent polygon faces of input polygon face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_ask_adjacent_faces
(
    tag_t cae_tag,
    int * num_faces,
    tag_p_t * faces
)
```

tag_t	**cae_tag**	Input	Polygon face
int *	**num_faces**	Output	Number of faces
tag_p_t *	**faces**	Output to UF_*free*	Polygon faces

UF_SF_face_ask_area (view source)

Defined in: uf_sf.h

Overview
This function will find area of polygon face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_ask_area
(
    tag_t cae_tag,
    double * pd_area
)
```

tag_t	**cae_tag**	Input	Polygon face
double *	**pd_area**	Output	Face Area

UF_SF_face_ask_body (view source)

Defined in: uf_sf.h

Overview
This function will get polygon body related to polygon face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_ask_body
(
    tag_t cae_tag,
    tag_p_t body_p
)
```

tag_t	**cae_tag**	Input	Polygon face
tag_p_t	**body_p**	Output	Polygon body

UF_SF_face_ask_bounding_box (view source)

Defined in: uf_sf.h

Overview
This function will get bounding box of polygon face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_ask_bounding_box
(
    tag_t cae_tag,
    double pad_bounding_box [ 6 ]
)
```

tag_t	**cae_tag**	Input	Polygon face
double	**pad_bounding_box [6]**	Output	Bounding box

UF_SF_face_ask_edges (view source)

Defined in: uf_sf.h

Overview
This function will get polygon edges related to polygon face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_ask_edges
(
    tag_t cae_tag,
    int * num_edges,
    tag_p_t * edges
)
```

tag_t	**cae_tag**	Input	Polygon face
int *	**num_edges**	Output	Number of polygon edges
tag_p_t *	**edges**	Output to UF_*free*	Polygon edges

UF_SF_face_evaluate_closest_point (view source)

Defined in: uf_sf.h

Overview
This function will project passed in points and finds required data on face.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_evaluate_closest_point
(
    tag_t cae_tag,
    int num_evaluations,
    double xyz [ ] ,
    double xyz_cl [ ] ,
    double uv_cl [ ] ,
    double nrml [ ] ,
    double dist [ ] ,
    int in_out [ ]
)
```

tag_t	**cae_tag**	Input	Polygon face
int	**num_evaluations**	Input	number of evaluations in U parameter
double	**xyz [ ]**	Input	Input 3d points (can be NULL)
double	**xyz_cl [ ]**	Output	Output closest 3d points (can be NULL)
double	**uv_cl [ ]**	Output	uv values (can be NULL)
double	**nrml [ ]**	Output	normals (can be NULL)
double	**dist [ ]**	Output	distance between input and projection
int	**in_out [ ]**	Output	0 -> inside, 1 -> outside, 2 -> boundary

---

UF_SF_face_evaluate_param_location (view source)

Defined in: uf_sf.h

Overview
This function will find 3d location for given param value.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_face_evaluate_param_location
(
    tag_t cae_tag,
    double ad_param [ 2 ] ,
    double ad_out_point [ 3 ]
)
```

tag_t	**cae_tag**	Input	Polygon face
double	**ad_param [2]**	Input	UV Param values
double	**ad_out_point [3]**	Output	Corresponding point

UF_SF_facepair_ask_midsrf_frec (view source)

Defined in: uf_sf.h

Overview
This function determines the parent midsurface feature of a given facepair feature.

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_facepair_ask_midsrf_frec
(
    tag_t facepair_feature_tag,
    tag_t * midsrf_feature_tag
)
```

tag_t	**facepair_feature_tag**	Input	facepair_feature_tag Tag of given facepair feature
tag_t *	**midsrf_feature_tag**	Output	Tag of parent feature of given facepair feature.

UF_SF_fem_ask_cad_part (view source)

Defined in: uf_sf.h

Overview
Returns the cad part for the given Fem (NULL_TAG if no cad part exists)

Environment
Internal and External

History
Originally released in NX4.0

```
tag_t UF_SF_fem_ask_cad_part
(
    tag_t fem_tag,
    logical* is_idealized_part
)
```

tag_t	**fem_tag**	Input	Tag of a Fem
logical*	**is_idealized_part**	Output	True if cad part returned is an idealized part

---

UF_SF_find_mesh (view source)

Defined in: uf_sf.h

Overview

Locates all meshes related to the input object.

Environment

Internal and External

```
int UF_SF_find_mesh
(
    tag_t object_tag,
    UF_SF_mesh_dimension_t dimension,
    int* num_of_meshes,
    tag_t* * mesh_items_p
)
```

tag_t	**object_tag**	Input	Tag to input entity, group, load, boundary condition, mesh_recipe, element or node.
UF_SF_mesh_dimension_t	**dimension**	Input	UF_SF_mesh_dimension_t of desired mesh recipe when object_tag is a mesh or element,This argument should be UF_SF_DIMENSION_ANY since only a single recipe is possible.
int*	**num_of_meshes**	Output	The number of meshes being returned. Could be zero.
tag_t* *	**mesh_items_p**	Output to UF_*free*	pointer to an array of mesh tags or NULL. This array must be freed by calling UF_free.

UF_SF_find_minimum_distance (view source)

Defined in: uf_sf.h

Overview

Given two entities (atleast one should be polygon geometry), gives minimum distance and points on the entities.

Environment

Internal and External

History

Originally released in NX4.0

```
int UF_SF_find_minimum_distance
(
    tag_t entity_1,
    tag_t entity_2,
    double * min_dist,
    double point1 [ 3 ] ,
    double point2 [ 3 ]
)
```

tag_t	**entity_1**	Input	Polygon edge/face/body
tag_t	**entity_2**	Input	Polygon edge/face/body
double *	**min_dist**	Output	Minimum distance
double	**point1 [3]**	Output	Point on entity_1

double	**point2 [3]**	Output	Point on entity_2

UF_SF_free_defeature_parms (view source)

Defined in: uf_sf.h

Overview
This routine frees memory associated with an allocated instance of the UF_SF_defeature_parms_s structure used to define the parameters of a defeature feature

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_free_defeature_parms
(
    UF_SF_defeature_parms_p_t def_parms_p
)
```

UF_SF_defeature_parms_p_t	**def_parms_p**	Input	Defeature parameter entity to free

---

UF_SF_free_fatigue_prop (view source)

Defined in: uf_sf.h

Overview
Cleans up dynamic storage associated to a UF_SF_material_fatigue_prop_t. The structure UF_SF_material_fatigue_prop_t must have been fully initialized using UF_SF_init_fatigue_prop.

Environment
Internal and External

Returns
There is no return code.

See Also
html#UF_SF_init_fatigue_prop

History
This function was originally released in V19.1

```
void UF_SF_free_fatigue_prop
(
    UF_SF_material_fatigue_prop_p_t property_values
)
```

UF_SF_material_fatigue_prop_p_t	**property_values**	Input	pointer to the UF_SF_material_fatigue_prop_t structure

---

UF_SF_free_formability_prop (view source)

Defined in: uf_sf.h

Overview
Cleans up dynamic storage associated to a UF_SF_material_prop_t.
The structure UF_SF_material_prop_t must have been fully initialized using UF_SF_init_formability_prop.

Environment
Internal and External

Returns
There is no return code.

See Also
html#UF_SF_init_formability_prop

History
This function was originally released in V19.1

```
void UF_SF_free_formability_prop
(
    UF_SF_material_formability_prop_p_t property_values
)
```

UF_SF_material_formability_prop_p_t	**property_values**	Input	pointer to the UF_SF_material_formability_prop_t structure

---

UF_SF_free_idealize_parms (view source)

Defined in: uf_sf.h

Overview
This routine frees memory associated to a parameter object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_free_idealize_parms
(
    UF_SF_idealize_parms_p_t parms_p
)
```

UF_SF_idealize_parms_p_t	**parms_p**	Input	Idealize parameter entity to freed

---

UF_SF_free_idealize_region (view source)

Defined in: uf_sf.h

Overview
This routine frees a region object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

int UF_SF_free_idealize_region
(
UF_SF_idealize_region_p_t region_p
)

UF_SF_idealize_region_p_t	**region_p**	Input	Idealize region entity in which to free

---

UF_SF_free_matl_prop (view source)

Defined in: uf_sf.h

Overview
Cleans up dynamic storage associated to a UF_SF_material_prop_t.
The structure UF_SF_material_prop_t must have been fully initialized using UF_SF_init_matl_prop.

Environment
Internal and External

Returns
There is no return code.

See Also
html#UF_SF_init_matl_prop

void UF_SF_free_matl_prop
(
UF_SF_material_prop_t* property_values
)

UF_SF_material_prop_t*	**property_values**	Input	pointer to the UF_SF_material_prop_t structure

---

UF_SF_free_matl_strength_prop (view source)

Defined in: uf_sf.h

Overview
Cleans up dynamic storage associated to a UF_SF_material_prop_t.
The structure UF_SF_material_prop_t must have been fully initialized using UF_SF_init_matl_strength_prop.

Environment
Internal and External

Returns
There is no return code.

See Also
html#UF_SF_init_matl_strength_prop

History
This function was originally released in V19.1

void UF_SF_free_matl_strength_prop
(

UF_SF_material_strength_prop_p_t property_values
)

UF_SF_material_strength_prop_p_t	**property_values**	Input	pointer to the UF_SF_material_strength_prop_t structure

---

UF_SF_free_midsrf_user_parms (view source)

Defined in: uf_sf.h

Overview
Frees an instance of UF_SF_midsrf_user_parms_t structure.

Environment
Internal and External

History
Originally released in NX2

int UF_SF_free_midsrf_user_parms
(
UF_SF_midsrf_user_parms_p_t parms_p
)

UF_SF_midsrf_user_parms_p_t	**parms_p**	Input	Midsurface user defined parms to free.

UF_SF_get_auto_element_size (view source)

Defined in: uf_sf.h

Overview
This user function is used to compute auto-element size for polygon bodies.

Environment
Internal and External

int UF_SF_get_auto_element_size
(
int num_objects,
tag_p_t objects_p,
double * esize
)

int	**num_objects**	Input	No of bodies in objects_p array
tag_p_t	**objects_p**	Input	list of bodies from which to calculate auto size
double *	**esize**	Output	Auto size value

UF_SF_idealized_part_ask_master_part (view source)

Defined in: uf_sf.h

Overview

Returns the master part for the given idealized part (NULL_TAG if no master part exists)

Environment
Internal and External

History
Originally released in NX4.0

tag_t UF_SF_idealized_part_ask_master_part
(
tag_t idealized_part_tag
)

tag_t	**idealized_part_tag**	Input	Tag of a Idealized Part

UF_SF_init_fatigue_prop (view source)

Defined in: uf_sf.h

Overview
Initializes pointers associated to the structure UF_SF_material_fatigue_prop_t.
Should be called before using the structure to make sure the structure is properly initialized.
Note that the type of each member of the structure will be defaulted to expression.

Environment
Internal and External

See Also
html#UF_SF_free_fatigue_prop

History
This function was originally released in V19.1

int UF_SF_init_fatigue_prop
(
UF_SF_material_fatigue_prop_p_t property_values
)

UF_SF_material_fatigue_prop_p_t	**property_values**	Input / Output	Structure whose members are to be initialized

---

UF_SF_init_formability_prop (view source)

Defined in: uf_sf.h

Overview
Initializes pointers associated to the structure UF_SF_material_formability_prop_t.
Should be called before using the structure to make sure the structure is properly initialized.
Note that the type of each member of the structure will be defaulted to expression.

Environment
Internal and External

See Also

html#UF_SF_free_formability_prop

History
This function was originally released in V19.1

int UF_SF_init_formability_prop
(
UF_SF_material_formability_prop_p_t property_values
)

UF_SF_material_formability_prop_p_t	**property_values**	Input / Output	Structure whose members are to be initialized

---

UF_SF_init_matl_prop (view source)

Defined in: uf_sf.h

Overview
Initializes pointers associated to the structure UF_SF_material_prop_t. Should be called before using the structure to make sure the structure is properly initialized. Note that the type of each member of the structure will be defaulted to expression.

Environment
Internal and External

History
This function was originally released in V16.0

int UF_SF_init_matl_prop
(
UF_SF_material_prop_p_t property_values
)

UF_SF_material_prop_p_t	**property_values**	Input / Output	Structure, whose members are to be initialized

---

UF_SF_init_matl_strength_prop (view source)

Defined in: uf_sf.h

Overview
Initializes pointers associated to the structure UF_SF_material_strength_prop_t. Should be called before using the structure to make sure the structure is properly initialized.
Note that the type of each member of the structure will be defaulted to expression.

Environment
Internal and External

See Also
html#UF_SF_free_matl_strength_prop

History
This function was originally released in V19.1

int UF_SF_init_matl_strength_prop
(

UF_SF_material_strength_prop_p_t property_values
)

UF_SF_material_strength_prop_p_t	**property_values**	Input / Output	Structure whose members are to be initialized

---

UF_SF_init_mshvld_error_container (view source)

Defined in: uf_sf_mshvld.h

Overview

Function: UF_SF_init_mshvld_error_container

DESCRIPTION

This user function is used to initialize the error container
of mesh validation. It's prototype is available in uf_sf_mshvld.h

INPUT/OUTPUT

UF_SF_mesh_error_container_p_t error_container :

the structure holding the mesh errors found in mesh validation.

RETURN

void

void UF_SF_init_mshvld_error_container
(
UF_SF_mesh_error_container_p_t container
)

UF_SF_mesh_error_container_p_t	**container**	

UF_SF_is_fem (view source)

Defined in: uf_sf.h

Overview

Returns true if the input tag is a Fem

Environment

Internal and External

History

Originally released in NX4.0

logical UF_SF_is_fem
(
tag_t fem_tag
)

tag_t	**fem_tag**	Input	Tag of a fem

UF_SF_is_idealized_part (view source)

Defined in: uf_sf.h

Overview
Returns true if the input tag is a idealized part

Environment
Internal and External

History
Originally released in NX4.0

```
logical UF_SF_is_idealized_part
(
    tag_t idealized_part_tag
)
```

tag_t	**idealized_part_tag**	Input	Tag of a Idealized Part

UF_SF_is_midsrf (view source)

Defined in: uf_sf.h

Overview
This function determines if a given object is a midsurface

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_is_midsrf
(
    tag_t sheet_body_tag,
    logical * is_midsrf,
    int * midsrf_type,
    tag_t * feature_tag
)
```

tag_t	**sheet_body_tag**	Input	Tag of given object
logical *	**is_midsrf**	Output	Flag signifying the outcome: = TRUE --> object is a Midsurface = FALSE --> object is NOT a Midsurface
int *	**midsrf_type**	Output	Tag of the midsurface feature if the object is associated with such a feature.
tag_t *	**feature_tag**	Output	Tag of the midsurface feature if the object is associated with such a feature. NOTE: If the midsurface is facepair type the feature of the facepair has as midsurface body the given sheet body.

UF_SF_is_offset_midsrf (view source)

Defined in: uf_sf.h

Overview

Query a sheet body to see if it is an offset midsurface.
This function may not be used for general midsurface. If the input is a midsurface created by Facepair method, this function will return FALSE.

Environment

Internal and External

History

Originally released in V16.0

```
int UF_SF_is_offset_midsrf
(
    tag_t sheet_body_tag,
    logical * is_midsrf
)
```

tag_t	**sheet_body_tag**	Input	Tag of the sheet body.
logical *	**is_midsrf**	Output	TRUE if the sheet body is an offset midsurface, FALSE otherwise.

UF_SF_is_scenario_part (view source)

Defined in: uf_sf.h

Overview

Checks whether the current part file is a Scenario type part or not.

Return

Return code:
0 = Scenario part
UF_SF_NON_SCENARIO_PART =not a Scenario part.

Environment

Internal and External

```
int UF_SF_is_scenario_part
(
    void
)
```

UF_SF_is_simulation (view source)

Defined in: uf_sf.h

Overview

Returns true if the input tag is a Simulation

Environment

Internal and External

History

Originally released in NX4.0

```
logical UF_SF_is_simulation
(
    tag_t simulation_tag
)
```

tag_t	**simulation_tag**	Input	Tag of a Simulation

UF_SF_is_userdef_midsrf (view source)

Defined in: uf_sf.h

Overview
This function determines if a given object is a user defined midsurface

Environment
Internal and External

History
Originally released in NX2

```
int UF_SF_is_userdef_midsrf
(
    tag_t sheet_body_tag,
    logical * is_midsrf,
    tag_t * feature_tag
)
```

tag_t	**sheet_body_tag**	Input	Tag of given object
logical *	**is_midsrf**	Output	Flag signifying the outcome: = TRUE --> object is a user defined Midsurface = FALSE --> object is NOT a user defined Midsurface
tag_t *	**feature_tag**	Output	Tag of the user defined midsurface feature if the object is associated with such a feature.

UF_SF_LEGEND_set_colors (view source)

Defined in: uf_sf_ugopenint.h

Overview
Sets the color attributes of the scalar-to-color mapping that is used in rendering fringe displays.

Environment
Internal and External

History
Originally released in V19.0

```
int UF_SF_LEGEND_set_colors
(
    UF_SF_COLOR_attr_p_t color_attr_p
)
```

UF_SF_COLOR_attr_p_t	**color_attr_p**	Input	Color attribute structure defining parameters of the scalar-to-color mapping that is used in rendering fringe displays.

UF_SF_link_material (view source)

Defined in: uf_sf.h

Overview

Given a tag to a material property, and a tag to a mesh or geometry, this function checks the mesh or geometry for an existing material property. If there is any material already present then it unlinks the old material and links the input material. The valid objects are only meshes or mesh recipes.

Environment

Internal and External

```
int UF_SF_link_material
(
    tag_t material_tag,
    tag_t object_tag
)
```

tag_t	**material_tag**	Input	The tag of material.
tag_t	**object_tag**	Input	Tag of the object to which material is to be attached.

UF_SF_link_section (view source)

Defined in: uf_sf.h

Overview

Given a tag to a UF_SF_property of subtype section, and a tag to a edge/curve or point, finds the mesh_geometry and looks at the mesh_geometry for an existing section property. If it is not the input section, unlinks the old section and links the input section to the mesh_recipe.

Environment

Internal and External

```
int UF_SF_link_section
(
    tag_t section_tag,
    tag_t mesh_geom_tag
)
```

tag_t	**section_tag**	Input	The tag of section property.
tag_t	**mesh_geom_tag**	Input	Tag of the the 1D mesh, edge/ curve or point to which section is to be attached.

UF_SF_locate_all_meshes (view source)

Defined in: uf_sf.h

Overview

-

This routine finds all the mesh entities in the work part.

Environment
Internal & External

History
Written in v18.0

```
int UF_SF_locate_all_meshes
(
    tag_t mesh_tag,
    int * mesh_count,
    tag_p_t* mesh_pointer
)
```

tag_t	**mesh_tag**	Input	Mesh tag, pass in NULL_TAG to locate all tags in current part.
int *	**mesh_count**	Output	mesh count
tag_p_t*	**mesh_pointer**	Output to UF_*free*	mesh array. This array must be freed by calling UF_free.

UF_SF_locate_element (view source)

Defined in: uf_sf.h

Overview

Locates the elements on a mesh if object is mesh tag. If object is a node tag then parent element of node will be given as output. If there are no elements , then returns num_of_elements = 0 and element_tags_not = NULL.

If the given mesh or node occurrence is from the Simulation part then element occurrences associated with that mesh or node occurrence in the Simulation part are returned. Similarly if the given mesh or node is from the FEM part then the elements associated with that mesh or node in the FEM part are returned.

Environment
Internal and External

History

```
int UF_SF_locate_element
(
    tag_t mesh_tag,
    int* num_of_elements,
    tag_t* * element_tags_p
)
```

tag_t	**mesh_tag**	Input	Mesh tag or Node tag
int*	**num_of_elements**	Output	The number of elements. Could be zero.
tag_t* *	**element_tags_p**	Output to UF_*free*	pointer to an array of element tags or NULL if the number of elements is zero. This array must be freed by calling UF_free.

UF_SF_locate_element_by_id (view source)

Defined in: uf_sf.h

Overview

Locates and returns the the list element tags which match the list of input element ids. If there are no elements, then returns num_of_elements = 0 and element_tags_not = NULL.

Environment

Internal and External

History

```
int UF_SF_locate_element_by_id
(
    tag_t mesh_tag,
    int num_of_ids,
    int element_ids [ ] ,
    int* num_of_elements,
    tag_t* * element_tags_p
)
```

tag_t	**mesh_tag**	Input	Mesh tag, pass in NULL_TAG to locate all elements in current part.
int	**num_of_ids**	Input	The number of element ids to look for.
int	**element_ids [ ]**	Input	An array of element ids for which element tags are to be returned.
int*	**num_of_elements**	Output	The number of elements found. Could be zero.
tag_t* *	**element_tags_p**	Output to UF_*free*	pointer to an array of element tags or NULL. This array must be freed by calling UF_free.

UF_SF_locate_hpt_mg_parents (view source)

Defined in: uf_sf.h

Overview

Given a tag to a smart point, locate the parents whose mesh geometry has to be updated.

Environment

Internal and External

```
int UF_SF_locate_hpt_mg_parents
(
    tag_t point_tag,
    int* parent_cnt,
    tag_p_t* parent_list
)
```

tag_t	**point_tag**	Input	Tag to a hard point.
int*	**parent_cnt**	Output	Number of parents tags related to the point.

tag_p_t*	**parent_list**	Output to UF_*free*	pointer to an array of parent tags. This array must be freed if parent_cnt is greater than 0.

UF_SF_locate_material (view source)

Defined in: uf_sf.h

Overview
Given a tag to an object, this function returns the material tag associated to the object.

Environment
Internal and External

```
int UF_SF_locate_material
(
    tag_t object_tag,
    tag_t* material_tag
)
```

tag_t	**object_tag**	Input	The tag of object.
tag_t*	**material_tag**	Output	Tag of the material attached to the object.

UF_SF_locate_mesh_mating_condition_by_name (view source)

Defined in: uf_sf.h

Overview
Locate the assembly mesh object from an assembly mesh name.

Environment
Internal and External

History
This function was originally released in V16.0

```
int UF_SF_locate_mesh_mating_condition_by_name
(
    char* assembly_name,
    tag_t* assembly_mesh
)
```

char*	**assembly_name**	Input	Assembly mesh name
tag_t*	**assembly_mesh**	Output	Tag of the assembly mesh object

---

UF_SF_locate_mesh_mating_condition_list (view source)

Defined in: uf_sf.h

Overview
Return a list of all the assembly_mesh objects.

Environment

Internal and External

History
This function was originally released in V16.0

int UF_SF_locate_mesh_mating_condition_list
(
int* assembly_mesh_cnt,
tag_t* * assembly_mesh_tags
)

int*	**assembly_mesh_cnt**	Output	Count of Assembly mesh objects returned
tag_t* *	**assembly_mesh_tags**	Output to UF_*free*	Pointer to an array of tags. If not desired, input a NULL pointer. This array must be freed by calling UF_free.

UF_SF_locate_named_dursol_nx (view source)

Defined in: uf_sf.h

Overview
This function will return the tag of a Durability Solution, given its name.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

int UF_SF_locate_named_dursol_nx
(
const char* dursol_name,
tag_t* dursol_tag
)

const char*	**dursol_name**	Input	Name of the durability solution The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.
tag_t*	**dursol_tag**	Output	Tag of the durability solution with the specified name.

UF_SF_locate_named_material (view source)

Defined in: uf_sf.h

Overview

Given a name of material, this function returns the material tag associated to the name.

Environment
Internal and External

int UF_SF_locate_named_material
(

```
    char* material_name,
    tag_p_t material_tag_ptr
)
```

char*	**material_name**	Input	Material name string. The maximum number of allowable characters is 80.
tag_p_t	**material_tag_ptr**	Output	Tag of the material attached to object.

UF_SF_locate_named_section (view source)

Defined in: uf_sf.h

Overview

Given a name of a section, returns the section tag associated to the name in the work part.

Environment

Internal and External

```
int UF_SF_locate_named_section
(
    char* section_name,
    tag_p_t section_tag_ptr
)
```

char*	**section_name**	Input	Section name string
tag_p_t	**section_tag_ptr**	Output	Tag of the section attached to object.

UF_SF_locate_named_solution_nx (view source)

Defined in: uf_sf.h

Overview

Given a name of a solution, this function finds the tag of the solution.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_locate_named_solution_nx
(
    const char* solution_name,
    tag_t* solution_tag
)
```

const char*	**solution_name**	Input	Name of the solution.
tag_t*	**solution_tag**	Output	Tag of the solution.

UF_SF_locate_named_step_nx (view source)

Defined in: uf_sf.h

Overview
Given the name of a step and the tag of its parent solution, this function finds the step.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_locate_named_step_nx
(
    tag_t solution_tag,
    const char* step_name,
    tag_t* step_tag
)
```

tag_t	**solution_tag**	Input	Tag of the solution.
const char*	**step_name**	Input	Name of the step. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.
tag_t*	**step_tag**	Output	Tag of the step.

UF_SF_locate_node_by_id (view source)

Defined in: uf_sf.h

Overview

Locates and returns the the list node tags which match the list of input node ids. If there are no nodes , then returns num_of_nodes = 0 and node_tags_p = NULL.

Environment
Internal and External

History

```
int UF_SF_locate_node_by_id
(
    tag_t mesh_tag,
    int num_of_ids,
    int node_ids [ ] ,
    int* num_of_node,
    tag_t* * node_tags_p
)
```

tag_t	**mesh_tag**	Input	Mesh tag, pass in NULL_TAG to locate all nodes in current part.
int	**num_of_ids**	Input	The number of node ids to look for.
int	**node_ids [ ]**	Input	An array of node ids for which element tags are to be returned.

int*	**num_of_node**	Output	The number of nodes found. Could be zero.
tag_t* *	**node_tags_p**	Output to UF_*free*	pointer to an array of node tags or NULL. This must be freed by calling UF_free.

UF_SF_locate_nodes_on_element (view source)

Defined in: uf_sf.h

Overview

Locates the nodes on a element. If there are no nodes, then returns num_of_nodes = 0 and node_tags_not = NULL.

If given an element occurrence tag then node occurrences in the owning Simulation part are returned. Similarly if the given an element tag then nodes in the owning FEM part are returned.

Environment

Internal and External

History

```
int UF_SF_locate_nodes_on_element
(
    tag_t element_tag,
    int* num_of_nodes,
    tag_t* * node_tags_p
)
```

tag_t	**element_tag**	Input	Element tag.
int*	**num_of_nodes**	Output	The number of node. Could be zero.
tag_t* *	**node_tags_p**	Output to UF_*free*	pointer to an array of node tags or NULL. This array must be freed by calling UF_free.

UF_SF_locate_nodes_on_geometry (view source)

Defined in: uf_sf.h

Overview

Given a tag to a geometry, find all nodes on it

If given a polygon geometry (CAE geometry) occurrence then node occurrences in the owning Simulation part are returned. Similarly if given a ploygon geometry then nodes in the owning FEM part are returned.

Environment

Internal and External

```
int UF_SF_locate_nodes_on_geometry
(
    tag_t geom_tag,
    UF_SF_node_switch_t type_sw,
    int * nodes_cnt,
    tag_p_t * nodes_list
)
```

tag_t	**geom_tag**	Input	Tag of CAE geometry.
UF_SF_node_switch_t	**type_sw**	Input	switch indicating face/solid interior nodes are desired, SFOM_node_switch_t It can be one of the three choices: UF_SF_SWITCH_ON_BOUNDARY: boundary nodes only UF_SF_SWITCH_IN_INTERIOR: internal nodes only UF_SF_SWITCH_ALL: all nodes
int *	**nodes_cnt**	Output	number of nodes on the geometry
tag_p_t *	**nodes_list**	Output to UF_*free*	list of node tags on the geometry

---

UF_SF_locate_nodes_on_mesh (view source)

Defined in: uf_sf.h

Overview

Locates all nodes in a mesh.

If the given mesh tag is from a Simulation part or given a NULL_TAG and the current work part is a Simulation part then node occurrences from that Simulation part are returned. Similarly if the given mesh tag is from a FEM part or given a NULL_TAG and the current work part is a FEM part then nodes from that FEM part are returned.

Environment

Internal and External

History

```
int UF_SF_locate_nodes_on_mesh
(
    tag_t mesh_tag,
    int* num_of_nodes,
    tag_t* * node_tags_p
)
```

tag_t	**mesh_tag**	Input	Mesh tag, pass in NULL_TAG to locate all nodes in current part.
int*	**num_of_nodes**	Output	The number of nodes. Could be zero.
tag_t* *	**node_tags_p**	Output to UF_*free*	pointer to an array of node tags or NULL if the number of nodes is zero. This array must be freed by calling UF_free.

UF_SF_locate_scenarios (view source)

Defined in: uf_sf.h

Overview

Locates all scenario parts and finds names of Scenario parts in the current working part.

Environment

Internal and External

```
int UF_SF_locate_scenarios
(
    int* number_of_scenarios,
    char* * * scenario_names
)
```

int*	**number_of_scenarios**	Output	Number of Scenario part files found.
char* * *	**scenario_names**	Output to UF_*free*	pointer to array of names of Scenario part files. Maximum name size of each Scenario part file is 24 characters. This array must be freed by calling UF_free_string_array.

UF_SF_locate_section (view source)

Defined in: uf_sf.h

Overview

Returns sections associated to the input mesh recipe or geometry object such as a curve/edge or point.

The input mesh_object has to belong to a FEM part

Environment

Internal and External

```
int UF_SF_locate_section
(
    tag_t mesh_object,
    tag_t* section_tag
)
```

tag_t	**mesh_object**	Input	Tag to mesh recipe or geometry object. Can be a mesh_recipe, a curve/edge, or a point.
tag_t*	**section_tag**	Output	Section tag or NULL_TAG if no section found. This must be freed by calling UF_SF_free_section_data

---

UF_SF_locate_solution_boundary_conditions_nx (view source)

Defined in: uf_sf.h

Overview

Locates the boundary conditions in a solution. Does not locate boundary conditions in steps contained by the solution (see: UF_SF_locate_step_boundary_conditions_nx).
The membs_pp pointer is undefined if num_members is zero.

Environment

Internal and External

See Also

UF_SF_locate_step_boundary_conditions_nx

History

Originally released in NX3.0

```
int UF_SF_locate_solution_boundary_conditions_nx
(
```

```
    tag_t solution,
    int * num_members,
    tag_t* * membs_pp
)
```

tag_t	**solution**	Input	Tag of the solution.
int *	**num_members**	Output	Number of boundary conditions in the solution.
tag_t* *	**membs_pp**	Output to UF_*free*	The pointer to the array of the tags of the boundary conditions. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_locate_solution_loads_nx (view source)

Defined in: uf_sf.h

Overview

Locates the loads in a solution. Does not locate loads
in steps contained by the solution (see: UF_SF_locate_step_loads_nx).
The membs_pp pointer is undefined if num_members is zero.

Environment

Internal and External

See Also

UF_SF_locate_step_loads_nx

History

Originally released in NX3.0

```
int UF_SF_locate_solution_loads_nx
(
    tag_t solution,
    int * num_members,
    tag_t* * membs_pp
)
```

tag_t	**solution**	Input	Tag of the solution.
int *	**num_members**	Output	Number of loads in the solution.
tag_t* *	**membs_pp**	Output to UF_*free*	The pointer to the array of the tags of the loads. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_locate_step_boundary_conditions_nx (view source)

Defined in: uf_sf.h

Overview

Locates the boundary conditions in a step.
The membs_pp pointer is undefined if num_members is zero.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_locate_step_boundary_conditions_nx
(
    tag_t step,
    int * num_members,
    tag_t* * membs_pp
)
```

tag_t	**step**	Input	Step tag.
int *	**num_members**	Output	Number of boundary conditions in the step.
tag_t* *	**membs_pp**	Output to UF_*free*	The pointer to the array of the tags of the boundary conditions. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_locate_step_loads_nx (view source)

Defined in: uf_sf.h

Overview

Locates the loads in a step.
The membs_pp pointer is undefined if num_members is zero.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_locate_step_loads_nx
(
    tag_t step,
    int * num_members,
    tag_t* * membs_pp
)
```

tag_t	**step**	Input	Step tag.
int *	**num_members**	Output	Number of loads in the step.
tag_t* *	**membs_pp**	Output to UF_*free*	The pointer to the array of the tags of the loads. If num_members is zero, this pointer should not be used. If num_members is > 0, then this array must be freed by calling UF_free.

UF_SF_map_object_to_current_part (view source)

Defined in: uf_sf.h

Overview

Given an input object this function returns the corresponding object in the
current work part

This function was written to aid users who have developed applications based
on the Pre-NX4 data model that had a single scenario part the referenced a master cad
part. When Pre-NX4 structures scenario parts are migrated to NX4 the data in the scenario
will be moved into one of the following 3 parts. assumes the following part hierachy:
Sim part (loads, contraints & solution data)
|

V
Fem Part (CAE Geometry, Meshes, Elements & node)
|
V
Idealized Part (Modeling data, idealize feature, midsurfaces .....) Optional layer
|
V
Master Cad Part

At this time the only mappings we will support are:

Part where
input object Current work
exists part type
========== ========== ==========
Idealized part ==> Fem part Returns mapped object if possible
Idealized part ==> Sim part Returns mapped object if possible

Fem part ==> Sim part Returns mapped object if possible

Cad part ==> Cad part Return input tag
Idealized part ==> Idealized part Return input tag
Fem part ==> Fem part Return input tag
Sim part ==> Sim part Return input tag

Environment
Internal and External

History
Originally released in NX4.0

```
tag_t UF_SF_map_object_to_current_part
(
    tag_t object_tag
)
```

tag_t	**object_tag**	Input	Tag of a object to be mapped

UF_SF_mc_2d_angle (view source)

Defined in: uf_sf_element_check.h

Overview
UF_SF_mc_2d_angle :

This function will calculate an element's angle between sides

Note: result_format is a dummy input on NX85

Environment
Internal and External

History
released in v18.0

```
int UF_SF_mc_2d_angle
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t results_format,
    double min_threshold,
    double max_threshold,
    double* min_angle,
    double* max_angle,
```

```
    UF_SFMC_result_t* min_status_ptr,
    UF_SFMC_result_t* max_status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**results_format**	Input	format of result
double	**min_threshold**	Input	minimum threshold value
double	**max_threshold**	Input	maximum threshold value
double*	**min_angle**	Output	The Elements minimim angle
double*	**max_angle**	Output	The Elements maximum angle
UF_SFMC_result_t*	**min_status_ptr**	Output	minimum result status
UF_SFMC_result_t*	**max_status_ptr**	Output	maximum result status

---

UF_SF_mc_aspect_ratio (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_aspect_ratio :

This function will check an element's aspect ratio value against threshold value

Note: result_format is a dummy input on NX85

Environment

Internal & External

History

released in v18.0

```
int UF_SF_mc_aspect_ratio
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double* aspect_ratio,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**result_format**	Input	format of results
double	**threshold**	Input	threshold value

double*	**aspect_ratio**	Output	aspect ratio for the element.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_element_check (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_element_check :

This function will check an element's all qualities, such as warp, taper, skew, aspect ratio, jacobian ratio, against the threshold values .

Environment

Internal & External

History

released in v18.0

```
int UF_SF_mc_element_check
(
    tag_t mesh_tag,
    int * failed_elm_count
)
```

tag_t	**mesh_tag**	Input	tag of mesh which needs to be checked
int *	**failed_elm_count**	Output	

UF_SF_mc_jacobian_ratio (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_jacobian_ratio:

This function will check an element's jacobian ratio against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_jacobian_ratio
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t results_format,
    double threshold,
    double* jacobian_ratio,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**results_format**	Input	format of result
double	**threshold**	Input	threshold value
double*	**jacobian_ratio**	Output	The Elements jacobian ratio.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_jacobian_zero (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_jacobian_zero:

This function will check an element's jacobian zero against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_jacobian_zero
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t results_format,
    double threshold,
    double* jacobian_zero,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**results_format**	Input	format of result
double	**threshold**	Input	threshold value
double*	**jacobian_zero**	Output	The Elements jacobian zero
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_skew (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_skew:

This function will check an element's skew against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_skew
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double* skew,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**result_format**	Input	format of results
double	**threshold**	Input	threshold value
double*	**skew**	Output	skew for the element.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_taper (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_taper:

This function will check an element's taper against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_taper
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
```

```
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double* taper,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**result_format**	Input	format of results
double	**threshold**	Input	threshold value
double*	**taper**	Output	taper for the element.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_tet_collapse (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_tet_collapse

This function will check elements' tet collapse. An element tet collapse is defined as the ratio of its longest edge to its shortest altitude. This check only applies to the Tet elements.

Note: result_format is a dummy input on NX85

Input:
element_type element type
abs_node_coordinates[] Array of PNT3_t which contain absolute node coordinates.

node_cnt The number of nodes in an array.

results_format Indicates whether how results are to be presented. (i.e normalized or not)

threshold threshold aspect_ratio value.

Output:
tet_collapse The Elements tet collapse

status_ptr status whether passed or failed check.

Returns

int = 0 - successful completion
> 0 - error

```
int UF_SF_mc_tet_collapse
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double * tet_collapse,
    UF_SFMC_result_t * status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input
double	**abspos [3]**	Input
int	**node_count**	Input
UF_SF_mc_result_format_t	**result_format**	Input
double	**threshold**	Input
double *	**tet_collapse**	Output
UF_SFMC_result_t *	**status_ptr**	Output

UF_SF_mc_tetra_angle (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_tetra_angle:

This function will calculate an element's tet angle

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_tetra_angle
(
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t results_format,
    double min_threshold,
    double max_threshold,
    double* min_angle,
    double* max_angle,
    UF_SFMC_result_t* min_status_ptr,
    UF_SFMC_result_t* max_status_ptr
)
```

double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**results_format**	Input	format of result
double	**min_threshold**	Input	minimum threshold value
double	**max_threshold**	Input	maximum threshold value
double*	**min_angle**	Output	The Elements minimim angle
double*	**max_angle**	Output	The Elements maximum angle
UF_SFMC_result_t*	**min_status_ptr**	Output	minimum result status
UF_SFMC_result_t*	**max_status_ptr**	Output	maximum result status

UF_SF_mc_twist (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_twist:

This function will check an element's twist against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal and External

History

released in v18.0

```
int UF_SF_mc_twist
(
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double* twist,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**result_format**	Input	format of results
double	**threshold**	Input	threshold value
double*	**twist**	Output	twist for the element.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_mc_warp (view source)

Defined in: uf_sf_element_check.h

Overview

UF_SF_mc_warp:

This function will check an element's warp against the threshold value

Note: result_format is a dummy input on NX85

Environment

Internal

History

released in v18.0

```
int UF_SF_mc_warp
(
```

```
    UF_SF_element_type_t element_type,
    double abspos [ 3 ] ,
    int node_count,
    UF_SF_mc_result_format_t result_format,
    double threshold,
    double* warp,
    UF_SFMC_result_t* status_ptr
)
```

UF_SF_element_type_t	**element_type**	Input	element type.
double	**abspos [3]**	Input	pointer to the node location in absolute coordinates.
int	**node_count**	Input	number of nodes.
UF_SF_mc_result_format_t	**result_format**	Input	format of results
double	**threshold**	Input	threshold value
double*	**warp**	Output	warp for the element.
UF_SFMC_result_t*	**status_ptr**	Output	result status

UF_SF_modl_body_ask_body (view source)

Defined in: uf_sf.h

Overview

This function will get Polygon body related to MODL body.

Valid only when the work part is a FEM

Environment

Internal and External

History

Originally released in NX4.0

```
int UF_SF_modl_body_ask_body
(
    tag_t modl_body,
    tag_p_t cae_body_p
)
```

tag_t	**modl_body**	Input	CAD MODL body
tag_p_t	**cae_body_p**	Output	Polygon body

UF_SF_open_scenario (view source)

Defined in: uf_sf.h

Overview

This routine initializes Scenario

Environment

Internal and External

History

```
int UF_SF_open_scenario
(
    const char* scenario_name,
    tag_t master_part_tag
)
```

const char*	**scenario_name**	Input	scenario to be opened
tag_t	**master_part_tag**	Input	master part tag

UF_SF_partition_body (view source)

Defined in: uf_sf.h

Overview
This function will partition solid bodies into 2 solid bodies.

Environment
Internal and External

See Also
.
.

```
int UF_SF_partition_body
(
    int num_solid_bodies,
    tag_p_t solid_body_tags,
    tag_t tool_body,
    int * num_partitioned_bodies,
    tag_p_t * partitioned_bodies
)
```

int	**num_solid_bodies**	Input	The number of solid bodies to partition.
tag_p_t	**solid_body_tags**	Input	Array of solid bodies to partition.
tag_t	**tool_body**	Input	tag of the tool to use to define the partition.
int *	**num_partitioned_bodies**	Output	Number of bodies generated after partitioning
tag_p_t *	**partitioned_bodies**	Output to UF_*free*	

UF_SF_partition_body_nx5 (view source)

Defined in: uf_sf.h

Overview
This function will partition solid bodies into 2 solid bodies.
The method of partitioning will be based on the "Associate" toggle
as follows ...
1. Associate toggle ON implies that the existing behavior (the
two resulting bodies are fully associated to the CAD master part)
will be maintained
2. Associate toggle OFF implies that the two resulting bodies become

unparameterized and not associated to the CAD master part.

Environment
Internal and External

See Also
.
.

```
int UF_SF_partition_body_nx5
(
    int associate,
    int num_solid_bodies,
    tag_p_t solid_body_tags,
    tag_t tool_body,
    int * num_partitioned_bodies,
    tag_p_t * partitioned_bodies
)
```

int	**associate**	Input	Associative toggle state (1 implies resultant bodies are associated to CAD part 0 implies resultant bodies are not associated to CAD part .. they become unparameterized)
int	**num_solid_bodies**	Input	The number of solid bodies to partition.
tag_p_t	**solid_body_tags**	Input	Array of solid bodies to partition.
tag_t	**tool_body**	Input	tag of the tool to use to define the partition.
int *	**num_partitioned_bodies**	Output	Number of bodies generated after partitioning
tag_p_t *	**partitioned_bodies**	Output to UF_*free*	

UF_SF_polygon_body_ask_type (view source)

Defined in: uf_sf.h

Overview
This function will get polygon body type : i.e, Sheet body or Solid body.

Environment
Internal and External

History
Originally released in NX4.0

```
int UF_SF_polygon_body_ask_type
(
    tag_t polygon_body,
    int * body_type
)
```

tag_t	**polygon_body**	Input	Polygon body tag
int *	**body_type**	Output	Polygon body type; 0=>Solid; 1=>Sheet not mid-surface; 2=>sheet mid surface

---

UF_SF_property_ask_name_nx (view source)

Defined in: uf_sf.h

Overview
Given a property tag, returns the name of the property.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_property_ask_name_nx
(
    tag_t property_tag,
    char* * property_name_pp
)
```

tag_t	**property_tag**	Input	The tag of the property.
char* *	**property_name_pp**	Output to UF_*free*	The name of the property. NOTE: Caller must UF_free the string.

UF_SF_property_ask_type_nx (view source)

Defined in: uf_sf.h

Overview
Given a property tag, returns the property type.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_property_ask_type_nx
(
    tag_t property_tag,
    UF_SF_fem_value_type_t * property_type
)
```

tag_t	**property_tag**	Input	The tag of the property.
UF_SF_fem_value_type_t *	**property_type**	Output	The property type.

UF_SF_property_ask_value_nx (view source)

Defined in: uf_sf.h

Overview
Given a property tag, returns the property value.

Environment
Internal and External

History

Originally released in NX3.0

```
int UF_SF_property_ask_value_nx
(
    tag_t property_tag,
    logical * bool_value,
    int * int_value,
    double * scalar_value,
    char* * text_value,
    int * num_lines,
    char* * * multi_text_value
)
```

tag_t	**property_tag**	Input	The tag of the property.
logical *	**bool_value**	Output	Property value, if type SFOM_FEM_VALUE_BOOL.
int *	**int_value**	Output	Property value, if type is one of the following: UF_SF_FEM_VALUE_INT UF_SF_FEM_VALUE_ENUM
double *	**scalar_value**	Output	Property value, if type UF_SF_FEM_VALUE_SCALAR.
char* *	**text_value**	Output to UF_*free*	Property value, if type is UF_SF_FEM_VALUE_TEXT. NOTE: Caller must UF_free the string.
int *	**num_lines**	Output	If type is UF_SF_FEM_VALUE_MULTI_STRING, this is the number of text lines in the property.
char* * *	**multi_text_value**	Output to UF_*free*	If type is UF_SF_FEM_VALUE_MULTI_STRING, this is the array of text lines in the property. NOTE: Call UF_free_string_array to free the storage allocated within the structure.

UF_SF_property_set_value_nx (view source)

Defined in: uf_sf.h

Overview
Sets a property value.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_property_set_value_nx
(
    tag_t property_tag,
    logical bool_value,
    int int_value,
    double scalar_value,
    char* const text_value,
    int num_lines,
    char* * const multi_text_value
)
```

tag_t	**property_tag**	Input	The tag of the property.
logical	**bool_value**	Input	Property value, if type SFOM_FEM_VALUE_BOOL.

int	**int_value**	Input	Property value, if type is one of the following: UF_SF_FEM_VALUE_INT UF_SF_FEM_VALUE_ENUM
double	**scalar_value**	Input	Property value, if type is UF_SF_FEM_VALUE_SCALAR.
char* const	**text_value**	Input	Property value, if type is UF_SF_FEM_VALUE_TEXT.
int	**num_lines**	Input	If type is UF_SF_FEM_VALUE_MULTI_STRING, this is the number of text lines in the property.
char* * const	**multi_text_value**	Input	If type is UF_SF_FEM_VALUE_MULTI_STRING, this is the array of text lines in the property.

UF_SF_remove_from_solution_nx (view source)

Defined in: uf_sf.h

Overview
Removes the specified number of loads and boundary conditions from the active solution.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_remove_from_solution_nx
(
    int num_of_lbc,
    tag_t* lbc
)
```

int	**num_of_lbc**	Input	Number of loads and boundary conditions.
tag_t*	**lbc**	Input	Pointer to the array of tags for loads or boundary conditions.

UF_SF_remove_from_step_nx (view source)

Defined in: uf_sf.h

Overview
Removes the specified number of loads and boundary conditions from the active step.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_remove_from_step_nx
(
    int num_of_lbc,
    tag_t* lbc
)
```

int	**num_of_lbc**	Input	Number of loads and boundary conditions.
tag_t*	**lbc**	Input	Pointer to the array of tags for loads or boundary conditions.

UF_SF_rename_scenario (view source)

Defined in: uf_sf.h

Overview
This routine renames the current Scenario

Environment
Internal and External

History

```
int UF_SF_rename_scenario
(
    tag_t master_tag,
    const char * old_scenario,
    const char * new_scenario
)
```

tag_t	**master_tag**	Input	master part tag
const char *	**old_scenario**	Input	old scenario name
const char *	**new_scenario**	Input	new scenario name

UF_SF_reset_element_ids (view source)

Defined in: uf_sf.h

Overview
This function resets all element IDs, starting at a specified ID.
Each time you update a Scenario model, the mesher numbers elements starting from the highest available ID when it remeshes the model.
After multiple updates, element ID numbers can become too high.
Use this function to reset the element IDs.

This call works only when a FEM part is the work part

Environment
Internal and External

History
Originally released in NX3

```
int UF_SF_reset_element_ids
(
    int start_id
)
```

int	**start_id**	Input	Start ID

UF_SF_reset_node_ids (view source)

Defined in: uf_sf.h

Overview

This function resets all node IDs, starting at a specified ID.
Each time you update a Scenario model, the mesher numbers nodes starting from the highest available ID when it remeshes the model.
After multiple updates, node ID numbers can become too high.
Use this function to reset the node IDs.

This call works only when a FEM part is the work part

Environment

Internal and External

History

Originally released in NX3

```
int UF_SF_reset_node_ids
(
    int start_id
)
```

int	**start_id**	Input	Start ID

UF_SF_retrieve_library_material (view source)

Defined in: uf_sf.h

Overview

Given a material libref from the NX Material library,
retrieve the material from the library into the context part.

Use UF_SF_ask_library_materials to read the librefs.

Use UF_SF_ask_material to read the properties of the new material.

If a material of the same name or libref already exists in the context part file, then an error is returned.

Environment

Internal and External

See Also

UF_SF_ask_library_materials
UF_SF_ask_material

History

Originally released in NX2

```
int UF_SF_retrieve_library_material
(
    char* libref,
    tag_p_t material_tag
)
```

char*	**libref**	Input	The library reference string read from the NX Material Library.
tag_p_t	**material_tag**	Output	Tag of the retrieved material, or NULL_TAG if an error.

UF_SF_save_scenario (view source)

Defined in: uf_sf.h

Overview
This routine saves the current Scenario (Not valid for FEMs or Simulation parts)

Environment
Internal and External

History
---

int UF_SF_save_scenario
(
void
)

UF_SF_set_active_solution_and_step_nx (view source)

Defined in: uf_sf.h

Overview
Set the active solution and step. The step must belong to the solution.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

int UF_SF_set_active_solution_and_step_nx
(
tag_t active_solution,
tag_t active_step
)

tag_t	**active_solution**	Input	Tag of the active solution for the scenario part.
tag_t	**active_step**	Input	Tag of the active step which belongs to the above solution.

UF_SF_set_edge_density (view source)

Defined in: uf_sf.h

Overview
This function will set the edge density given an input
object.
The edge density is defined as the number of elements to be created
along the input object. There are several cases:

```
int UF_SF_set_edge_density
(
    tag_t object_tag,
    UF_SF_edge_density_data_t edge_density_data
)
```

tag_t	**object_tag**	Input	- Tag of object to query.
UF_SF_edge_density_data_t	**edge_density_data**	Input	UF_SF_edge_density_data_t where the edge_density data will be stowed.

UF_SF_set_idealize_parm_exp (view source)

Defined in: uf_sf.h

Overview
This routine sets a specified expression from a parameter object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_set_idealize_parm_exp
(
    UF_SF_idealize_parms_p_t parms_p,
    UF_SF_idealize_parm_exp_t parm_exp_t,
    tag_t exp_tag
)
```

UF_SF_idealize_parms_p_t	**parms_p**	Input	Idealize parameter entity to set
UF_SF_idealize_parm_exp_t	**parm_exp_t**	Input	Specific expression to set in the idealize parameter entity
tag_t	**exp_tag**	Input	Tag of the expression to be used for the specific idealize parameter Can be NULL_TAG which means that this parameter will not be used

---

UF_SF_set_idealize_parm_faces (view source)

Defined in: uf_sf.h

Overview
This routine sets a specified set of faces from a parameter object for an idealize feature

Environment
Internal and External

History
Originally released in V17.0

```
int UF_SF_set_idealize_parm_faces
(
    UF_SF_idealize_parms_p_t parms_p,
    UF_SF_idealize_parm_face_t parm_face_t,
```

```
    int num_faces,
    tag_t * faces
)
```

UF_SF_idealize_parms_p_t	**parms_p**	Input	Idealize parameter entity to query
UF_SF_idealize_parm_face_t	**parm_face_t**	Input	Specific face set parameter to query
int	**num_faces**	Input	Number of faces in the faces array
tag_t *	**faces**	Input	Array of face tags

UF_SF_set_mesh_visuals (view source)

Defined in: uf_sf.h

Overview

Sets a mesh's visual attributes.

Environment

Internal and External

```
int UF_SF_set_mesh_visuals
(
    tag_t mesh_tag,
    UF_SF_mesh_visuals_t* mesh_vis
)
```

tag_t	**mesh_tag**	Input	The tag of the mesh
UF_SF_mesh_visuals_t*	**mesh_vis**	Output	Structure to mesh visual properties. Set members to UF_SF_MESH_VISUAL_NO_CHANGE whose attribute you do not want to change.

UF_SF_set_shell_thickness (view source)

Defined in: uf_sf.h

Overview

Applies shell thickness to 2D meshes.

Valid only when the work part is a FEM

Environment

Internal and External

```
int UF_SF_set_shell_thickness
(
    const char* shell_thick_exp,
    int num_items,
    tag_t* items_array
)
```

const char*	**shell_thick_exp**	Input	Shell thickness value
int	**num_items**	Input	Number of items to associate shell thickness.
tag_t*	**items_array**	Input	pointer to the array of 2D mesh tags.

UF_SF_simulation_ask_fem (view source)

Defined in: uf_sf.h

Overview
Returns the Fem for the given Simulation (NULL_TAG if no fem exists)

Environment
Internal and External

History
Originally released in NX4.0

```
tag_t UF_SF_simulation_ask_fem
(
    tag_t simulation_tag
)
```

tag_t	**simulation_tag**	Input	Tag of a Simulation

UF_SF_solution_ask_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Get the descriptor for a existing solution.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_descriptor_nx
(
    tag_t solution,
    tag_t* descriptor
)
```

tag_t	**solution**	Input	Tag of the solution.
tag_t*	**descriptor**	Output	Tag of the descriptor.

UF_SF_solution_ask_language_nx (view source)

Defined in: uf_sf.h

Overview
Get the language of a solution.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_language_nx
(
    tag_t solution,
    tag_t* language
)
```

tag_t	**solution**	Input	Tag for the solution.
tag_t*	**language**	Output	Tag for the language.

UF_SF_solution_ask_name_nx (view source)

Defined in: uf_sf.h

Overview
Given the tag of a solution, this function finds the name of the solution.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_name_nx
(
    tag_t solution,
    char solution_name [ MAX_LINE_BUFSIZE ]
)
```

tag_t	**solution**	Input	Tag of the solution.
char	**solution_name [MAX_LINE_BUFSIZE]**	Output	Name of the solution. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.

UF_SF_solution_ask_nth_step_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution and an index into the list of steps for the solution, returns the tag of the step. This function can be used to loop over all steps in a solution.

The number of steps can be obtained from function UF_SF_solution_ask_num_steps_nx.

The index is just a counter.
Valid indices are from 0 to (number of steps-1).

Environment
Internal and External

See Also
UF_SF_solution_ask_num_steps_nx

History
Originally released in NX3.0

```
int UF_SF_solution_ask_nth_step_nx
(
    tag_t solution_tag,
    int step,
    tag_t * step_tag
)
```

tag_t	**solution_tag**	Input	Tag of the solution.
int	**step**	Input	Index of the step to query. Valid indices are from 0 to (number of steps-1).
tag_t *	**step_tag**	Output	Tag of the step.

UF_SF_solution_ask_num_steps_nx (view source)

Defined in: uf_sf.h

Overview
Query the number of steps.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_num_steps_nx
(
    tag_t solution_tag,
    int * step_count
)
```

tag_t	**solution_tag**	Input	Tag of the solution.
int *	**step_count**	Output	Number of steps.

UF_SF_solution_ask_property_by_index_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution and index into the solution's property list, returns the tag of the property descriptor. This function can be used to loop over all property descriptors in the solution.

The number of property descriptors can be obtained from function UF_SF_solution_ask_property_count_nx.

The index is just a counter.
Valid indices are from 0 to (number of property descriptors-1).

Environment
Internal and External

See Also
UF_SF_solution_ask_property_count_nx

History
Originally released in NX3.0

```
int UF_SF_solution_ask_property_by_index_nx
(
    tag_t solution,
    int property,
    tag_t* property_tag
)
```

tag_t	**solution**	Input	Tag for the solution.
int	**property**	Input	Index into list of properties. Valid indices are from 0 to (number of properties-1). The number of properties can be obtained from function UF_SF_solution_ask_property_count_nx.
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_solution_ask_property_count_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution, query the number of properties that it references.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_property_count_nx
(
    tag_t solution,
    int* num_props
)
```

tag_t	**solution**	Input	Tag for the solution.
int*	**num_props**	Output	Number of properties.

UF_SF_solution_ask_property_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution and the name of a property that it references, returns
the tag of the property.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_property_nx
(
    tag_t solution,
    const char* property_name,
    tag_t* property_tag
)
```

tag_t	**solution**	Input	Tag for the solution.
const char*	**property_name**	Input	Property name.
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_solution_ask_solver_property_by_index_nx (view source)

Defined in: uf_sf.h

Overview

Given a solution and index into the solution's solver property list, returns the tag of the solver property descriptor. This function can be used to loop over all solver property descriptors in the solution.

The number of solver property descriptors can be obtained from function UF_SF_solution_ask_solver_property_count_nx.

The index is just a counter.
Valid indices are from 0 to (number of solver property descriptors-1).

See Also

UF_SF_solution_ask_solver_property_count_nx

History

Originally released in NX3.0

```
int UF_SF_solution_ask_solver_property_by_index_nx
(
    tag_t solution,
    int property,
    tag_t* property_tag
)
```

tag_t	**solution**	Input	Tag for the solution.
int	**property**	Input	Index into list of solver properties for the solution. Valid indices are from 0 to (number of solver properties-1). The number of solver properties can be obtained from function UF_SF_solution_ask_solver_property_count_nx.
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_solution_ask_solver_property_count_nx (view source)

Defined in: uf_sf.h

Overview

Given a solution, query the number of solver properties that it references.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_solution_ask_solver_property_count_nx
(
    tag_t solution,
    int* num_props
```

```
)
```

tag_t	**solution**	Input	Tag for the solution.
int*	**num_props**	Output	Number of properties.

UF_SF_solution_ask_solver_property_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution and the name of a solver property that it references, returns the tag of the solver property.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solution_ask_solver_property_nx
(
    tag_t solution,
    const char* property_name,
    tag_t* property_tag
)
```

tag_t	**solution**	Input	Tag for the solution.
const char*	**property_name**	Input	Property name.
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_solution_set_name_nx (view source)

Defined in: uf_sf.h

Overview
Given the tag of a solution and a new name, this function sets the name of the solution associated to the given tag.

Returns
0(success) OR > 1(failure):
UF_NAME_CONFLICT - solution name name not unique
UF_NAME_INVALID_CHAR - invalid character in name
UF_RESULT_FILE_NAME_CONFLICT - solution name has potential conflict with existing result files.
UF_NAME_TOO_LONG - solution name too long

Environment
Internal and External

```
int UF_SF_solution_set_name_nx
(
    tag_t solution,
    const char* solution_name,
    logical rename_result_file
)
```

tag_t	**solution**	Input	Tag of the solution attached to name.
const char*	**solution_name**	Input	Name of the solution. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.
logical	**rename_result_file**	Input	Flag for renaming any associated result file. If set to TRUE, the solution and result file will be renamed the part will automatically be saved in order to maintain the association between the solution and the result file. If set to FALSE, the solution will be renamed, but the result file will retain its current name. The association between the solution and the result file will be lost.

UF_SF_solve_active_solution_nx (view source)

Defined in: uf_sf.h

Overview
Initiates solve of the active solution.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_solve_active_solution_nx
(
    int format_choice
)
```

int	**format_choice**	Input	Format/Solve option 0 (for format and solve) 1 (for format only)

UF_SF_step_ask_name_nx (view source)

Defined in: uf_sf.h

Overview
Given the tag of a step, this function finds the name of the step.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_step_ask_name_nx
(
    tag_t step,
    char step_name [ MAX_LINE_BUFSIZE ]
)
```

tag_t	**step**	Input	Step tag.

char	**step_name [MAX_LINE_BUFSIZE]**	Output	Name of the step. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.

UF_SF_step_ask_property_by_index_nx (view source)

Defined in: uf_sf.h

Overview

Given a solution step and index into the step's property list, returns the tag of the property descriptor. This function can be used to loop over all property descriptors in the step.

The number of property descriptors can be obtained from function UF_SF_step_ask_property_count_nx.

The index is just a counter.
Valid indices are from 0 to (number of property descriptors-1).

Environment

Internal and External

See Also

UF_SF_step_ask_property_count_nx

History

Originally released in NX3.0

```
int UF_SF_step_ask_property_by_index_nx
(
    tag_t step,
    int property,
    tag_t* property_tag
)
```

tag_t	**step**	Input	Tag for the step.
int	**property**	Input	Index into list of properties for the step. Valid indices are from 0 to (number of property descriptors-1).
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_step_ask_property_count_nx (view source)

Defined in: uf_sf.h

Overview

Given a step, query the number of properties that it references.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SF_step_ask_property_count_nx
(
    tag_t step,
    int* num_props
)
```

tag_t	**step**	Input	Tag for the step.
int*	**num_props**	Output	Number of properties.

UF_SF_step_ask_property_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution step and the name of a property that it references, returns the tag of the property.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SF_step_ask_property_nx
(
    tag_t step,
    const char* property_name,
    tag_t* property_tag
)
```

tag_t	**step**	Input	Tag for the step.
const char*	**property_name**	Input	Property name.
tag_t*	**property_tag**	Output	Tag for the property.

UF_SF_step_set_name_nx (view source)

Defined in: uf_sf.h

Overview
Given the tag of a step and a new name, this function sets the name of the step associated to the given tag.

Returns
0(success) OR > 1(failure).
UF_NAME_CONFLICT - step name name not unique
UF_NAME_INVALID_CHAR - invalid character in name
UF_NAME_TOO_LONG - step name too long

Environment
Internal and External

```
int UF_SF_step_set_name_nx
(
    tag_t solution,
    tag_t step,
    const char* step_name
)
```

tag_t	**solution**	Input	Tag of the parent solution.
tag_t	**step**	Input	Tag of the step attached to name.

const char*	**step_name**	Input	Name of the step. The buffer must be declared using MAX_LINE_BUFSIZE so it can hold MAX_LINE_NCHARS characters.

UF_SF_switch_scenarios (view source)

Defined in: uf_sf.h

Overview
Switches between two scenarios.

Environment
Internal and External

History
Written in V17.0

```
int UF_SF_switch_scenarios
(
    char * scenario_1_name,
    char * scenario_2_name
)
```

char *	**scenario_1_name**	Input	Name of first scenario
char *	**scenario_2_name**	Input	Name of second scenario

UF_SF_temp_disp_results (view source)

Defined in: uf_sf_ugopenint.h

Overview
Generates the temporary display of a mesh and the associated resutst.

Environment
Internal

History
Originally released in V19.0

```
int UF_SF_temp_disp_results
(
    UF_SF_COLOR_attr_p_t color_att_p,
    UF_SF_LEGEND_attr_p_t legend_attr_p
)
```

UF_SF_COLOR_attr_p_t	**color_att_p**	Input	Color attribute structure.
UF_SF_LEGEND_attr_p_t	**legend_attr_p**	Input	Legend attribute structure or NULL. If NULL a default legend will be created and displayed.

UF_SF_temp_display_element (view source)

Defined in: uf_sf_ugopenint.h

Overview
Generates the temporary display of an element.

Environment
Internal

```
int UF_SF_temp_display_element
(
    tag_t node_tag,
    int color,
    double height,
    logical display_edges,
    logical display_id,
    logical display_nodes,
    UF_DISP_poly_marker_t node_marker,
    logical display_orientation
)
```

tag_t	**node_tag**	Input	Tag of the element to be displayed.
int	**color**	Input	Color to be used in temporary display. See uf_obj.h for color constants.
double	**height**	Input	Size of display for ID and symbol display.
logical	**display_edges**	Input	Indicates whether element edges are to be displayed.
logical	**display_id**	Input	Indicates whether element label is to be displayed.
logical	**display_nodes**	Input	Indicates whether element nodes are to be displayed.
UF_DISP_poly_marker_t	**node_marker**	Input	node_marker (uf_disp.h)
logical	**display_orientation**	Input	Indicates whether element orientation is to be displayed. (some 1d and 2d elms only)

UF_SF_temp_display_node (view source)

Defined in: uf_sf_ugopenint.h

Overview
Generates the temporary display of a node.

Environment
Internal

```
int UF_SF_temp_display_node
(
    tag_t node_tag,
    int color,
    double height,
    logical display_id,
    const char* object_symbol,
    UF_DISP_poly_marker_t node_marker
)
```

tag_t	**node_tag**	Input	Tag of the node to be displayed.
int	**color**	Input	Color to be used in temporary display. See uf_obj.h for color constants.

double	**height**	Input	Height of display for ID and symbol display.
logical	**display_id**	Input	Indicates whether element label is to be displayed.
const char*	**object_symbol**	Input	Object Symbol or NULL to concatenate preceding to display id.
UF_DISP_poly_marker_t	**node_marker**	Input	node_marker (uf_disp.h)

UF_SF_test_gpe (view source)

Defined in: uf_sf.h

Overview
Test the GPE module

Environment
Internal and External

History
---

int UF_SF_test_gpe
(
FILE * fp
)

FILE *	**fp**	Input	file pointer

UF_SF_unlink_material (view source)

Defined in: uf_sf.h

Overview
Given and a tag to a mesh or geometry, this function unlinks the material from the object.

Environment
Internal and External

int UF_SF_unlink_material
(
tag_t object_tag
)

tag_t	**object_tag**	Input	Tag of the object to which material is attached.

UF_SF_unlink_section (view source)

Defined in: uf_sf.h

Overview
Unlinks the section from the object.

Environment
Internal and External

int UF_SF_unlink_section
(
tag_t section_tag,
tag_t mesh_geom_tag
)

tag_t	**section_tag**	Input	The tag of section property.
tag_t	**mesh_geom_tag**	Input	Tag of the the 1D mesh, edge/ curve or point to which section is to be attached.

UF_SF_update_fatigue (view source)

Defined in: uf_sf.h

Overview
Similar to UF_SF_update_material but updates the fatigue material object only.

int UF_SF_update_fatigue
(
tag_t material_tag,
UF_SF_material_fatigue_prop_p_t property_values
)

tag_t	**material_tag**	Input	The tag of material to be updated. May be the PHYS_MAT or SFMAT object.
UF_SF_material_fatigue_prop_p_t	**property_values**	Input	Contains the material strength property values.

UF_SF_update_material (view source)

Defined in: uf_sf.h

Overview
Given a material tag and optionally name, categary, lib_reference and property_values, this function will update a material.

Note that the type of the material cannot be updated.
Rather, delete and create a new material.

Returns
Completion status.
=0 successful completion
= SFMAT_INVALID_MATERIAL_NAME
= SFMAT_INVALID_DUPLICATE_NAME
= SFMAT_INVALID_TYPE
= SFMAT_MATERIAL_FAILED_TO_UPDATED
>0 error

int UF_SF_update_material
(
tag_t material_tag,
char* name_str,
char* category_str,

```
    char* lib_reference,
    UF_SF_material_prop_p_t property_values
)
```

tag_t	**material_tag**	Input	The tag of material to be updated.
char*	**name_str**	Input	Pointer to string representing the the name to be given to the material.
char*	**category_str**	Input	Pointer to string representing the the Category to be given to the material.
char*	**lib_reference**	Input	Pointer to string representing the the library reference to be given to the material.
UF_SF_material_prop_p_t	**property_values**	Input	Contains the material property values.

UF_SF_update_scenario (view source)

Defined in: uf_sf.h

Overview

This function updates a structures part file.

This call works only when a FEM part is the work part

Returns
int error = 0 --> Success
= UF_SF_NON_SCENARIO_PART -> Not a scenario file

Environment

Internal and External

History

Originally released in V17.0

```
int UF_SF_update_scenario
(
    void
)
```

UF_SF_update_strength (view source)

Defined in: uf_sf.h

Overview

Similar to UF_SF_update_material but updates the strength material object only.

```
int UF_SF_update_strength
(
    tag_t material_tag,
    UF_SF_material_strength_prop_p_t property_values
)
```

tag_t	**material_tag**	Input	The tag of material to be updated. May be the PHYS_MAT or SFMAT object

UF_SF_material_strength_prop_p_t	**property_values**	Input	Contains the material strength property values.

UF_SF_validate_meshes (view source)

Defined in: uf_sf_mshvld.h

Overview

Function: UF_SF_validate_meshes

DESCRIPTION

This user function is used to perform mesh validation.
It's prototype is available in uf_sf_mshvld.h

INPUT

tag_p_t mesh_ents_p : the list of mesh/ mesh_recipe tags
If mesh_ents_tag = NULL, mesh validation
is performed on all mesh recipes/meshes.
For this case, we do not care about the
value of num_mesh_ents.
int num_mesh_ents : The number of tags in the list mesh_ents_p.

unsigned int option_mask : the mesh validation options are defined in
the head file uf_sf_mshvld.h.

OUTPUT

UF_SF_mesh_error_container_p_t error_container :

the structure holding the mesh errors found.

The caller is required to free the memory of all mesh_errors of type
UF_SF_mesh_error_p_t in each mesh error list in this container.

RETURN

0 for success
Error code for failure

```
int UF_SF_validate_meshes
(
    tag_p_t mesh_ents_p,
    int num_mesh_ents,
    unsigned int option_mask,
    UF_SF_mesh_error_container_p_t error_container
)
```

tag_p_t	**mesh_ents_p**
int	**num_mesh_ents**
unsigned int	**option_mask**
UF_SF_mesh_error_container_p_t	**error_container**

UF_SF_write_report (view source)

Defined in: uf_sf.h

Overview

Exports html files comprising the report generated by UF_SF_create_report.
Valid only when a Simulation part is the work part

Environment
Internal or External

History
Released in NX3

```
int UF_SF_write_report
(
    void
)
```

UF_SFL_ask_bc_descriptor_name_nx (view source)

Defined in: uf_sf.h

Overview
Given an boundary condition descriptor tag, returns the boundary condition descriptor name.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_bc_descriptor_name_nx
(
    tag_t bc_desc_tag,
    char* * name_pp
)
```

tag_t	**bc_desc_tag**	Input	The tag of the bc descriptor.
char* *	**name_pp**	Output to UF_*free*	The name of the bc descriptor. NOTE: Caller must UF_free the string.

UF_SFL_ask_bc_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag and boundary condition descriptor name, returns the tag of the boundary condition descriptor.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_bc_descriptor_nx
(
```

```
    tag_t language_tag,
    const char* name,
    tag_t * bc_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
const char*	**name**	Input	Name of the bc descriptor.
tag_t *	**bc_descriptor_tag**	Output	The tag of the bc descriptor.

UF_SFL_ask_cur_language_nx (view source)

Defined in: uf_sf.h

Overview

Returns tag of the current language. Valid only when a FEM or Simulation is the work part. When a Simulation is active, he active Solution defines the current language. If no Solution is active, then the default environment defines the current language. When a FEM is active, the default language of the FEM is the current language.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_cur_language_nx
(
    tag_t * language_tag
)
```

tag_t *	**language_tag**	Output	The tag of the language.

UF_SFL_ask_element_descriptor_name_nx (view source)

Defined in: uf_sf.h

Overview

Given an element descriptor tag, returns the element descriptor name. Valid only when the work part is a FEM or a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_element_descriptor_name_nx
(
    tag_t elem_desc_tag,
    char* * name_pp
)
```

tag_t	**elem_desc_tag**	Input	The tag of the element descriptor.
char* *	**name_pp**	Output to UF_*free*	The name of the element descriptor. NOTE: Caller must UF_free the string.

UF_SFL_ask_element_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag and element descriptor name, returns the tag of the element descriptor. Valid only when the work part is a FEM or a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_element_descriptor_nx
(
    tag_t language_tag,
    const char* name,
    tag_t * elem_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
const char*	**name**	Input	Name of the element descriptor.
tag_t *	**elem_descriptor_tag**	Output	The tag of the element descriptor.

UF_SFL_ask_language_nx (view source)

Defined in: uf_sf.h

Overview
Given a solver descriptor tag and language name, returns the tag of the language.

Valid only if the work part is a Simulation

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_language_nx
(
    tag_t solver_desc_tag,
    const char* lang_name,
    tag_t * language_tag
)
```

tag_t	**solver_desc_tag**	Input	Solver descriptor tag.
const char*	**lang_name**	Input	Name of the language.
tag_t *	**language_tag**	Output	The tag of the language.

UF_SFL_ask_load_descriptor_name_nx (view source)

Defined in: uf_sf.h

Overview
Given a load descriptor tag, returns the load descriptor name.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_load_descriptor_name_nx
(
    tag_t load_desc_tag,
    char* * name_pp
)
```

tag_t	**load_desc_tag**	Input	The tag of the load descriptor.
char* *	**name_pp**	Output to UF_*free*	The name of the load descriptor. NOTE: Caller must UF_free the string.

UF_SFL_ask_load_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag and load descriptor name, returns the tag of the load descriptor.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_load_descriptor_nx
(
    tag_t language_tag,
    const char* name,
    tag_t * load_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
const char*	**name**	Input	Name of the load descriptor.
tag_t *	**load_descriptor_tag**	Output	The tag of the load descriptor.

UF_SFL_ask_nth_bc_descriptor_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag and index into the boundary condition (bc) descriptor list, returns the tag of the bc descriptor. This function can be used to loop over all bc descriptors in the language.

The number of bc descriptors can be obtained from function UF_SFL_ask_num_bc_descriptors_nx.

The index is just a counter.
Valid indices are from 0 to (number of bc descriptors-1).

Valid only when the work part is a Simulation.

Environment

Internal and External

See Also

UF_SFL_ask_num_bc_descriptors_nx

History

Originally released in NX3.0

```
int UF_SFL_ask_nth_bc_descriptor_nx
(
    tag_t language_tag,
    int index,
    tag_t * bc_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
int	**index**	Input	Index into list of bc descriptors. Valid indices are from 0 to (number of bc descriptors-1). The number of bc descriptors can be obtained from function UF_SFL_ask_num_bc_descriptors_nx.
tag_t *	**bc_descriptor_tag**	Output	The tag of the bc descriptor.

UF_SFL_ask_nth_element_descriptor_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag and index into the element descriptor list, returns the tag of the element descriptor. This function can be used to loop over all element descriptors in the language.

The number of element descriptors can be obtained from function UF_SFL_ask_num_element_descriptors_nx.

The index is just a counter.
Valid indices are from 0 to (number of element descriptors-1).

Valid only when the work part is a FEM or a Simulation.

Environment

Internal and External

See Also

UF_SFL_ask_num_element_descriptors_nx

History

Originally released in NX3.0

```
int UF_SFL_ask_nth_element_descriptor_nx
(
    tag_t language_tag,
    int index,
    tag_t * elem_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
int	**index**	Input	Index into list of element descriptors. Valid indices are from 0 to (number of element descriptors-1). The number of element descriptors can be obtained from function UF_SFL_ask_num_element_descriptors_nx.
tag_t *	**elem_descriptor_tag**	Output	The tag of the element descriptor.

UF_SFL_ask_nth_language_nx (view source)

Defined in: uf_sf.h

Overview

Given a solver descriptor tag and index into the solver's language list, returns the tag of the language. This function can be used to loop over all languages that are valid for the solver.

The number of languages for the solver can be obtained from function UF_SFL_ask_num_languages_nx.

The index is just a counter.
Valid indices are from 0 to (number of languages-1).

Valid only if the work part is a Simulation

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_nth_language_nx
(
    tag_t solver_desc_tag,
    int index,
    tag_t * language_tag
)
```

tag_t	**solver_desc_tag**	Input	Solver descriptor tag.
int	**index**	Input	Index into list of languages. Valid indices are from 0 to (number of languages-1). The number of languages can be obtained from function UF_SFL_ask_num_languages_nx.
tag_t *	**language_tag**	Output	The tag of the language.

UF_SFL_ask_nth_load_descriptor_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag and index into the load descriptor list, returns the tag of the load descriptor. This function can be used to loop over all load descriptors in the language.

The number of load descriptors can be obtained from function UF_SFL_ask_num_load_descriptors_nx.

The index is just a counter.
Valid indices are from 0 to (number of load descriptors-1).

Valid only when the work part is a Simulation.

Environment
Internal and External

See Also
UF_SFL_ask_num_load_descriptors_nx

History
Originally released in NX3.0

```
int UF_SFL_ask_nth_load_descriptor_nx
(
    tag_t language_tag,
    int index,
    tag_t * load_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
int	**index**	Input	Index into list of load descriptors. Valid indices are from 0 to (number of load descriptors-1). The number of load descriptors can be obtained from function UF_SFL_ask_num_load_descriptors_nx.
tag_t *	**load_descriptor_tag**	Output	The tag of the load descriptor.

UF_SFL_ask_nth_solution_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag and index into the solution descriptor list, returns the tag of the solution descriptor. This function can be used to loop over all solution descriptors in the language.

The number of solution descriptors can be obtained from function UF_SFL_ask_num_solution_descriptors_nx.

The index is just a counter.
Valid indices are from 0 to (number of solution descriptors-1).

Valid only when the work part is a Simulation.

Environment
Internal and External

See Also
UF_SFL_ask_num_solution_descriptors_nx

History
Originally released in NX3.0

```
int UF_SFL_ask_nth_solution_descriptor_nx
(
```

```
    tag_t language_tag,
    int index,
    tag_t * solution_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
int	**index**	Input	Index into list of solution descriptors. Valid indices are from 0 to (number of solution descriptors-1). The number of solution descriptors can be obtained from function UF_SFL_ask_num_solution_descriptors_nx.
tag_t *	**solution_descriptor_tag**	Output	The tag of the solution descriptor.

UF_SFL_ask_nth_solver_nx (view source)

Defined in: uf_sf.h

Overview

Given a index into the list of solvers, returns the tag of the solver descriptor. This function can be used to loop over all solver descriptors.

The number of solver descriptors can be obtained from function UF_SFL_ask_num_solvers_nx.

The index is just a counter.
Valid indices are from 0 to (number of solver descriptors-1).

Valid only if the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_nth_solver_nx
(
    int index,
    tag_t * solver_desc_tag
)
```

int	**index**	Input	Index into list of solvers. Valid indices are from 0 to (number of solvers-1). The number of solvers can be obtained from function UF_SFL_ask_num_solvers.
tag_t *	**solver_desc_tag**	Output	The tag of the solver descriptor.

UF_SFL_ask_num_bc_descriptors_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag, returns the number of bc descriptors.
The bc descriptors define what boundary condition types are valid for the language.

Valid only when the work part is a Simulation.

Environment

Internal and External

History
Originally released in NX3.0

int UF_SFL_ask_num_bc_descriptors_nx
(
tag_t language_tag,
int* num_bc_descriptors
)

tag_t	**language_tag**	Input	Language tag.
int*	**num_bc_descriptors**	Output	The number of bc descriptors.

UF_SFL_ask_num_element_descriptors_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag, returns the number of element descriptors.
The element descriptors define what elements are valid for the language.
Valid only when the work part is a FEM or a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

int UF_SFL_ask_num_element_descriptors_nx
(
tag_t language_tag,
int* num_elem_descriptors
)

tag_t	**language_tag**	Input	Language tag.
int*	**num_elem_descriptors**	Output	The number of element descriptors.

UF_SFL_ask_num_languages_nx (view source)

Defined in: uf_sf.h

Overview
Given a solver descriptor, returns number of languages that are valid for the solver.

Valid only if the work part is a Simulation

Environment
Internal and External

History
Originally released in NX3.0

int UF_SFL_ask_num_languages_nx
(

```
    tag_t solver_desc_tag,
    int * num_languages
)
```

tag_t	**solver_desc_tag**	Input	Solver descriptor tag.
int *	**num_languages**	Output	The number of languages.

UF_SFL_ask_num_load_descriptors_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag, returns the number of load descriptors.
The load descriptors define what load types are valid for the language.
Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_num_load_descriptors_nx
(
    tag_t language_tag,
    int* num_load_descriptors
)
```

tag_t	**language_tag**	Input	Language tag.
int*	**num_load_descriptors**	Output	The number of load descriptors.

UF_SFL_ask_num_solution_descriptors_nx (view source)

Defined in: uf_sf.h

Overview

Given a language tag, returns the number of solution descriptors.
The solution descriptors define what solution types are valid for the language.

Valid only when the work part is a Simulation.

Environment

Internal and External

History

Originally released in NX3.0

```
int UF_SFL_ask_num_solution_descriptors_nx
(
    tag_t language_tag,
    int* num_solution_descriptors
)
```

tag_t	**language_tag**	Input	Language tag.
int*	**num_solution_descriptors**	Output	The number of solution descriptors.

UF_SFL_ask_num_solvers_nx (view source)

Defined in: uf_sf.h

Overview
Returns number of solvers.

Valid only when the work part is a Simulation

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_num_solvers_nx
(
    int * num_solvers
)
```

int *	**num_solvers**	Output	The number of solvers.

UF_SFL_ask_solution_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a language tag and solution descriptor name, returns the tag of the solution descriptor. The solution descriptor name appears as "Analysis Type" on the "Create Solution" dialog.

Alternatively, one can call UF_SFL_ask_num_solution_descriptors_nx, UF_SFL_ask_nth_solution_descriptor_nx, and then UF_SFL_solution_descriptor_ask_name_nx to find the name for each solution descriptor.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_solution_descriptor_nx
(
    tag_t language_tag,
    const char* name,
    tag_t * solution_descriptor_tag
)
```

tag_t	**language_tag**	Input	Language tag.
const char*	**name**	Input	Name of the solution descriptor.
tag_t *	**solution_descriptor_tag**	Output	The tag of the solution descriptor.

UF_SFL_ask_solver_nx (view source)

Defined in: uf_sf.h

Overview
Given a solver name, returns the tag of the solver descriptor.

Valid only when the work part is a Simulation

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_ask_solver_nx
(
    const char* solver_name,
    tag_t * solver_desc_tag
)
```

const char*	**solver_name**	Input	Name of the solver.
tag_t *	**solver_desc_tag**	Output	The tag of the solver descriptor.

UF_SFL_set_default_env_nx (view source)

Defined in: uf_sf.h

Overview
Set the default environment for the FE Model.
The default environment is used when no Solution is active.

Valid only when the work part is a FEM

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_set_default_env_nx
(
    const char* language_name
)
```

const char*	**language_name**	Input	Language name. The name of the current language. The syntax of the language name is "<solver name> - <analysis type>". For example: "NX NASTRAN - Structural"

UF_SFL_solution_ask_lbc_desc_valid_nx (view source)

Defined in: uf_sf.h

Overview
Identifies whether a lbc descriptor is valid for a given solution descriptor.

Environment

Internal and External

History
Originally released in NX3.0

int UF_SFL_solution_ask_lbc_desc_valid_nx
(
tag_t sol_desc_tag,
tag_t lbc_desc_tag,
logical * valid
)

tag_t	**sol_desc_tag**	Input	Solution descriptor tag.
tag_t	**lbc_desc_tag**	Input	LBC descriptor tag.
logical *	**valid**	Output	True if the lbc descriptor is valid for the solution, otherwise false.

UF_SFL_solution_ask_nth_allowable_step_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution descriptor tag and index into the allowable step descriptor list, returns the tag of the allowable step descriptor.

This function can be used to loop over all allowable step descriptors for a given solution descriptor.

The number of allowable step descriptors can be obtained from function UF_SFL_solution_ask_num_allowable_step_descriptors_nx.

The index is just a counter.
Valid indices are from 0 to (number of allowable step descriptors-1).

Valid only if the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

int UF_SFL_solution_ask_nth_allowable_step_descriptor_nx
(
tag_t sol_desc_tag,
int index,
tag_t * step_descriptor_tag
)

tag_t	**sol_desc_tag**	Input	Solution descriptor tag.
int	**index**	Input	Index into list of allowable step descriptors. Valid indices are from 0 to (number of allowable step descriptors-1). The number of allowable step descriptors can be obtained from function UF_SFL_solution_num_allowable_step_descriptors_nx.
tag_t *	**step_descriptor_tag**	Output	The tag of the allowable step descriptor.

UF_SFL_solution_ask_num_allowable_step_descriptors_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution descriptor tag, returns the number of allowable step descriptors. An allowable step descriptor defines a step type that is valid for a given solution type.

Valid only if the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_solution_ask_num_allowable_step_descriptors_nx
(
    tag_t sol_desc_tag,
    int* num_allow_step_descs
)
```

tag_t	**sol_desc_tag**	Input	Solution descriptor tag.
int*	**num_allow_step_descs**	Output	The number of allowable step descriptors.

UF_SFL_solution_ask_step_descriptor_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution descriptor tag and allowable step descriptor name, returns the tag of the step descriptor.

Valid only when the work part is a Simulation.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_solution_ask_step_descriptor_nx
(
    tag_t sol_desc_tag,
    const char * step_desc_name,
    tag_t * step_descriptor_tag
)
```

tag_t	**sol_desc_tag**	Input	Solution descriptor tag.
const char *	**step_desc_name**	Input	Step descriptor name.
tag_t *	**step_descriptor_tag**	Output	The tag of the allowable step descriptor.

UF_SFL_solution_descriptor_ask_name_nx (view source)

Defined in: uf_sf.h

Overview
Given a solution descriptor tag, returns the solution descriptor name.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_solution_descriptor_ask_name_nx
(
    tag_t sol_desc_tag,
    char* * name_pp
)
```

tag_t	**sol_desc_tag**	Input	Solution descriptor tag.
char* *	**name_pp**	Output to UF_*free*	The name of the solution descriptor. NOTE: Caller must UF_free the string.

UF_SFL_step_ask_lbc_desc_valid_nx (view source)

Defined in: uf_sf.h

Overview
Identifies whether a lbc descriptor is valid for a given step descriptor.

Environment
Internal and External

History
Originally released in NX3.0

```
int UF_SFL_step_ask_lbc_desc_valid_nx
(
    tag_t step_desc_tag,
    tag_t lbc_desc_tag,
    logical * valid
)
```

tag_t	**step_desc_tag**	Input	Step descriptor tag.
tag_t	**lbc_desc_tag**	Input	LBC descriptor tag.
logical *	**valid**	Output	True if the lbc descriptor is valid for the step, otherwise false.

UF_SFL_step_descriptor_ask_name_nx (view source)

Defined in: uf_sf.h

Overview
Given a step descriptor tag, returns the step descriptor name.

Environment
Internal and External

History

Originally released in NX3.0

int UF_SFL_step_descriptor_ask_name_nx
(
tag_t step_desc_tag,
char* * name_pp
)

tag_t	**step_desc_tag**	Input	Step descriptor tag.
char* *	**name_pp**	Output to UF_*free*	The name of the step descriptor. NOTE: Caller must UF_free the string.
