UF_WEIGHT_ask_part_cset (view source)

Defined in: uf_weight.h

Overview
Ask the weight component set of part, which is returned as a string in cset_name. Returns "AllComponents" if no component set is in use. The weight component set need not be an actual component set in the part when this routine is called.

Environment
Internal and External

See Also
UF_WEIGHT_set_part_cset

Required License(s)
gateway

```
int UF_WEIGHT_ask_part_cset
(
    const tag_t part,
    char * * cset_name
)
```

const tag_t	**part**	Input	The part whose weight component set is to be returned.
char * *	**cset_name**	Output to UF_*free*	The name of part's current weight component set. The string must be freed with UF_free.

UF_WEIGHT_ask_part_max_weight (view source)

Defined in: uf_weight.h

Overview
Ask the maximum permitted weight limit for part, expressed in units. If the limit is not set, is_set is false, and max_weight is 0.0.

Environment
Internal and External

See Also
UF_WEIGHT_set_part_max_weight
UF_WEIGHT_unset_part_max_weight

Required License(s)
gateway

```
int UF_WEIGHT_ask_part_max_weight
(
    const tag_t part,
    const UF_WEIGHT_units_type_t units,
    double * max_weight,
    logical * is_set
)
```

const tag_t	**part**	Input	The part whose maximum weight limit is to be returned.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the maximum weight limit is to be expressed.
double *	**max_weight**	Output	The value of the maximum weight limit of part (0.0 if the limit is not set).
logical *	**is_set**	Output	True if the maximum weight limit of the part is currently set, false if not.

UF_WEIGHT_ask_part_min_weight (view source)

Defined in: uf_weight.h

Overview
Ask the minimum permitted weight limit for part, expressed in units.
If the limit is not set, is_set is false, and min_weight is 0.0.

Environment
Internal and External

See Also
UF_WEIGHT_set_part_min_weight
UF_WEIGHT_unset_part_min_weight

Required License(s)
gateway

```
int UF_WEIGHT_ask_part_min_weight
(
    const tag_t part,
    const UF_WEIGHT_units_type_t units,
    double * min_weight,
    logical * is_set
)
```

const tag_t	**part**	Input	The part whose minimum weight limit is to be returned.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the minimum weight limit is to be expressed.
double *	**min_weight**	Output	The value of the minimum weight limit of part (0.0 if the limit is not set).
logical *	**is_set**	Output	True if the minimum weight limit of the part is currently set, false if not.

UF_WEIGHT_ask_part_ref_set (view source)

Defined in: uf_weight.h

Overview

Ask the weight reference set of part, which is returned as a string in ref_set_name. Returns "Entire Part" if no reference set is in use, "Empty" if the empty reference set is in use. The weight reference set need not be an actual reference set in the part when this routine is called.

Environment

Internal and External

See Also

UF_WEIGHT_set_part_ref_set

Required License(s)

gateway

```
int UF_WEIGHT_ask_part_ref_set
(
    const tag_t part,
    char * * ref_set_name
)
```

const tag_t	**part**	Input	The part whose weight reference set is to be returned.
char * *	**ref_set_name**	Output to UF_*free*	The name of part's current weight reference set. The string must be freed with UF_free.

UF_WEIGHT_ask_part_save_option (view source)

Defined in: uf_weight.h

Overview

Ask the state of the update weight properties save option for part: update_on_save is true if the save option is on, false otherwise.

Environment

Internal and External

See Also

UF_WEIGHT_set_part_save_option

Required License(s)

gateway

```
int UF_WEIGHT_ask_part_save_option
(
    const tag_t part,
    logical * update_on_save
)
```

const tag_t	**part**	Input	The part whose update weight properties save option is to be returned.
logical *	**update_on_save**	Output	The current value of part's update weight properties save option.

UF_WEIGHT_ask_props (view source)

Defined in: uf_weight.h

Overview

Ask the current weight properties of the part, component or solid passed in. If properties are not currently present on the part, UF_WEIGHT_no_cache is returned as the cache state in properties.

This routine does not calculate the weights, this must have already been done by calling UF_WEIGHT_estab_part_props, UF_WEIGHT_estab_comp_props or UF_WEIGHT_estab_solid_props.

If object is a part, the properties returned are its own, exclusive of the properties of any components it may have. If object is a component, the properties returned do not allow for the position of the component in the assembly (use UF_WEIGHT_transform_props if this is desired), and properties stored on the component's prototype part will not be returned. If object is a promotion, its returned cache, if available, is the delta between the properties of the promotion geometry and the properties of its base solid.

Like the other UF_WEIGHT routines that can take a solid as an input argument, this routine requires that the solid is not suppressed, is not a boolean tool, is not a sheet solid, and is not view dependent. No solids in these categories can sensibly be given weight properties.

Environment

Internal and External

See Also

UF_WEIGHT_estab_part_props
UF_WEIGHT_estab_comp_props
UF_WEIGHT_estab_solid_props

Required License(s)

gateway

```
int UF_WEIGHT_ask_props
(
    const tag_t object,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties
)
```

const tag_t	**object**	Input	Part, component or solid body whose weight properties are wanted.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The current properties of object.

UF_WEIGHT_assert_comp_props (view source)

Defined in: uf_weight.h

Overview

Assert the properties of component to be those specified by properties. The cache_state of properties must be UF_WEIGHT_asserted, and the other states must be compatible as well, as described in the description of UF_WEIGHT_properties_t. The accuracy of the properties is ignored, and the errors are set to 0.0. The part containing component will be fully loaded.

Environment

Internal and External

See Also

UF_WEIGHT_delete_comp_assertion

Required License(s)

adv_assemblies

```
int UF_WEIGHT_assert_comp_props
(
    const tag_t component,
    UF_WEIGHT_properties_pc_t properties
)
```

const tag_t	**component**	Input	The component whose properties are to be asserted.
UF_WEIGHT_properties_pc_t	**properties**	Input	The properties to be asserted on the component.

UF_WEIGHT_assert_part_props (view source)

Defined in: uf_weight.h

Overview

Assert the properties of part to be those specified by properties. The cache_state of properties must be UF_WEIGHT_asserted, and the other states must be compatible as well, as described in the description of UF_WEIGHT_properties_t. The accuracy of the properties is ignored, and the errors are set to 0.0. The part will be fully loaded.

Environment
Internal and External

See Also
UF_WEIGHT_delete_part_assertion

Required License(s)
adv_assemblies

```
int UF_WEIGHT_assert_part_props
(
    const tag_t part,
    UF_WEIGHT_properties_pc_t properties
)
```

const tag_t	**part**	Input	The part whose weight properties are to be asserted.
UF_WEIGHT_properties_pc_t	**properties**	Input	The properties to be asserted on the part.

UF_WEIGHT_convert_prop_units (view source)

Defined in: uf_weight.h

Overview
Copies the old_properties struct's values to a new struct new_properties, whose units are given by units, but which is otherwise equivalent to old_properties.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_WEIGHT_convert_prop_units
(
    const UF_WEIGHT_properties_p_t old_properties,
    const UF_WEIGHT_units_type_t new_units,
    UF_WEIGHT_properties_t * new_properties
)
```

const UF_WEIGHT_properties_p_t	**old_properties**	Input	Weight properties to be converted to other units.
const UF_WEIGHT_units_type_t	**new_units**	Input	The units to which the properties are to be converted.
UF_WEIGHT_properties_t *	**new_properties**	Output	properties, converted to units.

UF_WEIGHT_copy_props (view source)

Defined in: uf_weight.h

Overview
Copy properties to a new struct new_properties with the same values.

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_WEIGHT_copy_props
(
    const UF_WEIGHT_properties_p_t properties,
    UF_WEIGHT_properties_t * new_properties
)
```

const UF_WEIGHT_properties_p_t	**properties**	Input	The specified properties.
UF_WEIGHT_properties_t *	**new_properties**	Output	A copy of properties.

UF_WEIGHT_delete_comp_assertion (view source)

Defined in: uf_weight.h

Overview
Delete the asserted weight properties on component. Will return
UF_WEIGHT_has_no_assertion if there are no asserted properties
on component. The part containing component will be fully loaded.
If component had previously had cached values on it, these will have
been overwritten by the asserted values, and will not be available even
after the assertion is deleted. However, when the assertion is deleted,
caches on solids in and children of component become available again.

Environment
Internal and External

See Also
UF_WEIGHT_assert_comp_props

Required License(s)
adv_assemblies

```
int UF_WEIGHT_delete_comp_assertion
(
    const tag_t component
)
```

const tag_t	**component**	Input	The component whose asserted properties are to be deleted.

UF_WEIGHT_delete_part_assertion (view source)

Defined in: uf_weight.h

Overview

Delete the asserted weight properties on part. Will return UF_WEIGHT_has_no_assertion if there are no asserted properties on the part. Does not affect the weight reference set or component set of the part, or any weight limits set on it. The part will be fully loaded.

If the part had previously had cached values on it, these will have been overwritten by the asserted values, and will not be available even after the assertion is deleted. However, when the assertion is deleted, caches on solids and components of the part become available again.

Environment

Internal and External

See Also

UF_WEIGHT_assert_part_props

Required License(s)

adv_assemblies

```
int UF_WEIGHT_delete_part_assertion
(
    const tag_t part
)
```

const tag_t	**part**	Input	The part whose asserted properties are to be deleted.

UF_WEIGHT_estab_comp_props (view source)

Defined in: uf_weight.h

Overview

Establish the weight properties of component, returning the result in properties, given in units, and caching the calculated data on the component. The part containing the component will be fully loaded. If recurse is true, include contributions from any of its children present in the weight component set of the part containing the component (and cache their own contributions), otherwise include only solids in the component's part itself. In either case, solids must be in the weight reference set of their part to contribute. This routine will use any assertions applicable.

Calculations of weight properties will be made to an accuracy of at least accuracy; existing caches of that accuracy or greater will be used where possible; elsewhere recalculations are made from individual solids. accuracy must be 0.9, 0.99, 0.999, 0.9999, 0.99999 or 0.999999. exceptions must first be initialised by UF_WEIGHT_init_exceptions and must be freed afterward by UF_WEIGHT_free_exceptions.

Environment

Internal and External

See Also

UF_WEIGHT_init_exceptions
UF_WEIGHT_free_exceptions

Required License(s)

adv_assemblies

```
int UF_WEIGHT_estab_comp_props
(
    const tag_t component,
    const double accuracy,
    const logical recurse,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties,
    UF_WEIGHT_exceptions_t * exceptions
)
```

const tag_t	**component**	Input	The component whose properties are to be calculated.
const double	**accuracy**	Input	The accuracy to which the component's properties are to be calculated.
const logical	**recurse**	Input	True if the properties are to include the component's child components, false otherwise.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The established properties of the component.
UF_WEIGHT_exceptions_t *	**exceptions**	Output to UF_*free*	Any exceptions encountered during the calculation. This structure must be freed by calling UF_WEIGHT_free_exceptions

UF_WEIGHT_estab_comp_props1 (view source)

Defined in: uf_weight.h

Overview

In order to provide appropriate .NET binding for UF_WEIGHT_estab_comp_props, UF_WEIGHT_estab_comp_props1 is introduced.

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

For docuementation, refer to documentation of UF_WEIGHT_estab_comp_props.

Required License(s)

gateway

```
int UF_WEIGHT_estab_comp_props1
(
    const tag_t component,
    const double accuracy,
    const logical recurse,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties,
    UF_WEIGHT_exceptions_t * * exceptions
)
```

const tag_t	**component**	Input	The component whose properties are to be calculated.
const double	**accuracy**	Input	The accuracy to which the component's properties are to be calculated.
const logical	**recurse**	Input	True if the properties are to include the component's child components, false otherwise.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The established properties of the component.
UF_WEIGHT_exceptions_t * *	**exceptions**	Output to UF_*free*	Any exceptions encountered during the calculation. This structure must be freed by calling UF_WEIGHT_free_exceptions

---

UF_WEIGHT_estab_part_props (view source)

Defined in: uf_weight.h

Overview

Establish the weight properties of part, returning the result in properties, given in units, and caching the calculated data on the part. If recurse is true, include contributions from any of its components present in the part's weight component set (and cache their own contributions), otherwise include only solids in the part itself. In either case, any solids in the part in its weight reference set will contribute. This routine will use any assertions applicable. The part will be fully loaded.

Calculations of weight properties will be made to an accuracy of at least accuracy; existing caches of that accuracy or greater will be used where possible; elsewhere recalculations are made from individual

solids. accuracy must be 0.9, 0.99, 0.999, 0.9999, 0.99999 or 0.999999.
exceptions must first be initialised by UF_WEIGHT_init_exceptions
and must be freed afterward by UF_WEIGHT_free_exceptions.

Environment
Internal and External

See Also
UF_WEIGHT_init_exceptions
UF_WEIGHT_free_exceptions

Required License(s)
adv_assemblies

int UF_WEIGHT_estab_part_props
(
    const tag_t part,
    const double accuracy,
    const logical recurse,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties,
    UF_WEIGHT_exceptions_t * exceptions
)

const tag_t	**part**	Input	The part whose weight properties are to be established.
const double	**accuracy**	Input	The accuracy to which the part's properties are to be calculated.
const logical	**recurse**	Input	True if the properties are to include components of the part, false otherwise.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The established properties of the part.
UF_WEIGHT_exceptions_t *	**exceptions**	Output to UF_*free*	Any exceptions encountered during the calculation. This must be freed by calling UF_WEIGHT_free_exceptions.

UF_WEIGHT_estab_part_props1 (view source)

Defined in: uf_weight.h

Overview
In order to provide appropriate .NET binding for UF_WEIGHT_estab_part_props, UF_WEIGHT_estab_part_props1 is introduced.

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

For docuementation, refer to documentation of UF_WEIGHT_estab_part_props.

Required License(s)

adv_assemblies

```
int UF_WEIGHT_estab_part_props1
(
    const tag_t part,
    const double accuracy,
    const logical recurse,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties,
    UF_WEIGHT_exceptions_t * * exceptions
)
```

const tag_t	**part**	Input	The part whose weight properties are to be established.
const double	**accuracy**	Input	The accuracy to which the part's properties are to be calculated.
const logical	**recurse**	Input	True if the properties are to include components of the part, false otherwise.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The established properties of the part.
UF_WEIGHT_exceptions_t * *	**exceptions**	Output to UF_*free*	Any exceptions encountered during the calculation. This must be freed by calling UF_WEIGHT_free_exceptions.

UF_WEIGHT_estab_solid_props (view source)

Defined in: uf_weight.h

Overview

Establish the weight properties of solid, returning the result in properties, given in units, and caching the calculated data on the solid. Note that exceptions are not applicable to this lower level routine.

Calculations of weight properties will be made to an accuracy of at least accuracy; existing caches of that accuracy or greater will be used where possible; elsewhere recalculations are made from individual solids. accuracy must be 0.9, 0.99, 0.999, 0.9999, 0.99999 or 0.999999.

If solid is in the work part, a cache is written on it (if an existing cache was not available). Otherwise, the relevant properties are still returned, but no cache is written. solid can be a solid occurrence instead of a solid. If so, the properties returned are those of the prototype solid with an appropriate transform applied. A cache is written on the prototype solid if it is in the work part, but never on a solid occurrence.

Like the other UF_WEIGHT routines that can take a solid as an input argument, this routine requires that the solid is not suppressed, is not a boolean tool, is not a sheet solid, and is not view dependent. No solids in these categories can sensibly be given weight properties. If solid is a promotion, its returned cache is the delta between the properties of the promotion geometry and the properties of its base solid. This ensures that any calculation including both the promotion and its base returns the correct answer (i.e. the properties of the promoted geometry).

Environment
Internal and External

Required License(s)
adv_assemblies

```
int UF_WEIGHT_estab_solid_props
(
    const tag_t solid,
    const double accuracy,
    const UF_WEIGHT_units_type_t units,
    UF_WEIGHT_properties_t * properties
)
```

const tag_t	**solid**	Input	The solid whose properties are to be calculated.
const double	**accuracy**	Input	The accuracy to which the solid's properties are to be calculated.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the properties are to be returned.
UF_WEIGHT_properties_t *	**properties**	Output	The established properties of the solid.

UF_WEIGHT_free_exceptions (view source)

Defined in: uf_weight.h

Overview
Free an exceptions structure used when establishing weight properties.

Environment
Internal and External

See Also
UF_WEIGHT_init_exceptions
UF_WEIGHT_estab_part_props
UF_WEIGHT_estab_comp_props
UF_WEIGHT_estab_solid_props

Required License(s)
adv_assemblies

```
int UF_WEIGHT_free_exceptions
(
    UF_WEIGHT_exceptions_p_t exceptions
)
```

UF_WEIGHT_exceptions_p_t	**exceptions**	Input	The exceptions structure to be freed.

UF_WEIGHT_init_exceptions (view source)

Defined in: uf_weight.h

Overview

Initialise an exceptions structure for use when establishing weight properties.
An initialised exceptions structure can be passed into more than one
establish routine, and the exceptions generated by each call will be
accumulated within the exceptions struct. However, usually an
exceptions structure should be initialised before one call, and freed
immediately after that call and code to check that the exceptions
struct was filled correctly.

Environment

Internal and External

See Also

UF_WEIGHT_free_exceptions
UF_WEIGHT_estab_part_props
UF_WEIGHT_estab_comp_props
UF_WEIGHT_estab_solid_props

Required License(s)

adv_assemblies

```
int UF_WEIGHT_init_exceptions
(
    UF_WEIGHT_exceptions_p_t exceptions
)
```

UF_WEIGHT_exceptions_p_t	**exceptions**	Input	The exceptions structure to be initialised.

UF_WEIGHT_set_part_cset (view source)

Defined in: uf_weight.h

Overview

Set the weight component set for part to cset_name: only child
components in the weight component set of a part contribute to that's
part's recursive weight properties. The part will be fully loaded. Use
the string "AllComponents" to set the weight-component set to be all
components (this is the initial value before the weight component set
is explicitly set). cset_name does not have to be the name of a
component set currently in the part (however, using such a weight
component set during a weight calculation will generate exceptions).
It is also possible to delete a component set that is in use as a weight

component set, causing the same exceptions on following weight calculations.

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_cset
UF_ASSEM_create_cset
UF_ASSEM_add_to_cset

Required License(s)

adv_assemblies

```
int UF_WEIGHT_set_part_cset
(
    const tag_t part,
    const char * cset_name
)
```

const tag_t	**part**	Input	The part whose weight component set is to be set.
const char *	**cset_name**	Input	Name of component set to be made the part's weight component set.

UF_WEIGHT_set_part_max_weight (view source)

Defined in: uf_weight.h

Overview

Set the maximum permitted weight limit on part to be max_weight, expressed in units. max_weight must be greater than zero. If this limit is exceeded in future weight property calculations, an exception is raised. The part will be fully loaded.

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_max_weight
UF_WEIGHT_unset_part_max_weight

Required License(s)

adv_assemblies

```
int UF_WEIGHT_set_part_max_weight
(
    const tag_t part,
    const double max_weight,
    const UF_WEIGHT_units_type_t units
)
```

const tag_t	**part**	Input	The part whose maximum weight limit is to be set.

const double	**max_weight**	Input	The value of the new maximum weight limit for the part.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the maximum weight limit is expressed.

UF_WEIGHT_set_part_min_weight (view source)

Defined in: uf_weight.h

Overview
Set the minimum permitted weight limit on part to be min_weight, expressed in units. min_weight must be greater than zero. If this limit is exceeded in future weight property calculations, an exception is raised. The part will be fully loaded.

Environment
Internal and External

See Also
UF_WEIGHT_ask_part_min_weight
UF_WEIGHT_unset_part_min_weight

Required License(s)
adv_assemblies

```
int UF_WEIGHT_set_part_min_weight
(
    const tag_t part,
    const double min_weight,
    const UF_WEIGHT_units_type_t units
)
```

const tag_t	**part**	Input	The part whose minimum weight limit is to be set.
const double	**min_weight**	Input	The value of the new minimum weight limit for the part.
const UF_WEIGHT_units_type_t	**units**	Input	The units in which the minimum weight limit is expressed.

UF_WEIGHT_set_part_ref_set (view source)

Defined in: uf_weight.h

Overview
Set the weight reference set for part to ref_set_name: only solids in the weight reference set of a part contribute to that part's weight properties. The part will be fully loaded. Use the strings "Entire Part" and "Empty" respectively to set the weight reference set to be

all solids or none. If the part contains a model reference set then it is designated as the initial value before the weight reference set is explicitly set. If the model reference set is not defined on the part then the weight reference set is "Entire Part".
ref_set_name does not have to be the name of a reference set currently in the part (however, using such a weight reference set during a weight calculation will generate exceptions).
It is also possible to delete a reference set that is in use
as a weight reference set, causing the same exceptions on following weight calculations.

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_ref_set
UF_ASSEM_create_ref_set

Required License(s)

adv_assemblies

```
int UF_WEIGHT_set_part_ref_set
(
    const tag_t part,
    const char * ref_set_name
)
```

const tag_t	**part**	Input	The part whose weight reference set is to be set.
const char *	**ref_set_name**	Input	Name of reference set to be made the part's weight reference set.

UF_WEIGHT_set_part_save_option (view source)

Defined in: uf_weight.h

Overview

Set the update weight properties save option for part. The part will be fully loaded. If update_on_save is true, the weight properties will be updated for the part every time it is saved thereafter. If it is false, weight properties will not be updated on save thereafter (though they can still be established in response to an explicit user request, of course).

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_save_option

Required License(s)

adv_assemblies

```
int UF_WEIGHT_set_part_save_option
(
    const tag_t part,
    const logical update_on_save
)
```

const tag_t	**part**	Input	The part whose update weight properties save option is to be set.
const logical	**update_on_save**	Input	New value of update weight properties save option for the part.

UF_WEIGHT_sum_props (view source)

Defined in: uf_weight.h

Overview

Add together the elements of properties_array and return the result in total_properties. count gives the length of properties_array. This summation will combine the properties as if they represented actual solid bodies of those properties, and returns the properties of those solids considered as a set.

All the properties must be given in the same units. If not, UF_WEIGHT_incompatible_units is returned.

Environment

Internal and External

Required License(s)

adv_assemblies

```
int UF_WEIGHT_sum_props
(
    const int count,
    UF_WEIGHT_properties_t * properties_array,
    UF_WEIGHT_properties_t * total_properties
)
```

const int	**count**	Input	Number of properties structs in properties_array.
UF_WEIGHT_properties_t *	**properties_array**	Input	The array of properties structs to be summed.
UF_WEIGHT_properties_t *	**total_properties**	Output	The summed properties.

UF_WEIGHT_transform_props (view source)

Defined in: uf_weight.h

Overview

Applies a transform to properties. This routine can be used with UF_ASSEM_ask_transform_of_occ to get the properties of an occurrence given those of its prototype (this includes getting the properties of a component given those of a part). The caller must ensure that the units of properties and transform are compatible.

The transform argument must be of the following form:
transform[0][0] to transform[2][2] must be an orthonormal (rotation) matrix.
transform[0][3] to transform[2][3] must be a translation vector.
transform[3][0] to transform[3][2] must be 0.0.
transform[3][3] (the scale factor) must be 1.0.

Environment

Internal and External

See Also

UF_ASSEM_ask_transform_of_occ

Required License(s)

adv_assemblies

```
int UF_WEIGHT_transform_props
(
    double transform [ 4 ] [ 4 ] ,
    const UF_WEIGHT_properties_p_t properties,
    UF_WEIGHT_properties_t * transformed_properties
)
```

double	**transform [4] [4]**	Input	Transform to be applied to properties.
const UF_WEIGHT_properties_p_t	**properties**	Input	The current properties.
UF_WEIGHT_properties_t *	**transformed_properties**	Output	The properties that result from the transform.

UF_WEIGHT_unset_part_max_weight (view source)

Defined in: uf_weight.h

Overview

Remove the maximum permitted weight limit on part. This restores the initial state of the part, whereby no maximum weight limit exceptions are given. The part will be fully loaded.

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_max_weight
UF_WEIGHT_set_part_max_weight

Required License(s)

adv_assemblies

```
int UF_WEIGHT_unset_part_max_weight
(
    const tag_t part
)
```

const tag_t	**part**	Input	The part whose maximum weight limit is to be removed.

UF_WEIGHT_unset_part_min_weight (view source)

Defined in: uf_weight.h

Overview

Remove the minimum permitted weight limit on part. This restores the initial state of the part, whereby no minimum weight limit exceptions are given. The part will be fully loaded.

Environment

Internal and External

See Also

UF_WEIGHT_ask_part_min_weight
UF_WEIGHT_set_part_min_weight

Required License(s)

adv_assemblies

```
int UF_WEIGHT_unset_part_min_weight
(
    const tag_t part
)
```

const tag_t	**part**	Input	The part whose minimum weight limit is to be removed.
