UF_WAVE_accept_link_broken (view source)

Defined in: uf_wave.h

Overview

Accept that a linked feature is broken. This routine cannot be called on unbroken linked features, and returns an error if this is attempted. Once this routine has been called, the linked feature continues to be accepted as broken until it is no longer broken at some later time; when the feature becomes unbroken, its accepted-as-broken status is automatically removed.

Environment

Internal and External

See Also

UF_WAVE_set_link_data
UF_WAVE_is_link_broken
UF_WAVE_ask_link_accept_broken

Required License(s)

gateway

```
int UF_WAVE_accept_link_broken
(
    const tag_t linked_feature
)
```

const tag_t	**linked_feature**	Input	The linked feature to be accepted as broken.

UF_WAVE_ask_broken_link_source_part (view source)

Defined in: uf_wave.h

Overview

This routine gets the part file name for the broken link's source part, if it is available.

A link may be broken either because the relation to the source geometry is severed (e.g. the source geometry
is deleted or the link becomes non-associative to the source geometry by hitting the "break link" icon") or
the link's context has changed. In the latter case, the source geometry information still exists. Therefore,
it can be returned provided that the source part is either fully or partially loaded in the session.

This function will return the source part for a broken link provided that (1) the break occurred only because
the link's context changed and (2) the source part is at least partially loaded. In all other other cases, no
source part is returned. The absence of this information is not an error.

Environment

Internal and External

Required License(s)
gateway

int UF_WAVE_ask_broken_link_source_part
(
tag_t broken_link,
char * * part_name,
char * * source_object_handle
)

tag_t	**broken_link**	Input	
char * *	**part_name**	Output to UF_*free*	The part name for the source of the link
char * *	**source_object_handle**	Output to UF_*free*	The handle for the source part

---

UF_WAVE_ask_delay_status (view source)

Defined in: uf_wave.h

Overview
Determines the delay status assigned to a given part.

Environment
Internal and External

See Also
UF_WAVE_delay_status_t

Required License(s)
gateway

int UF_WAVE_ask_delay_status
(
tag_t part,
UF_WAVE_delay_status_t * delay_status
)

tag_t	**part**	Input	Tag of part for which to get delay status.
UF_WAVE_delay_status_t *	**delay_status**	Output	Delay status of the part.

UF_WAVE_ask_link_accept_broken (view source)

Defined in: uf_wave.h

Overview
Ask if a linked feature is broken and has been accepted as such, either by UF_WAVE_accept_link_broken or from the WAVE user interface.

Returns false for any unbroken linked feature. Also returns false for a feature that had been accepted, but has since become unbroken and then rebroken again.

Environment
Internal and External

See Also
UF_WAVE_set_link_data
UF_WAVE_is_link_broken
UF_WAVE_accept_link_broken

Required License(s)
gateway

int UF_WAVE_ask_link_accept_broken
(
const tag_t linked_feature,
logical * is_accepted
)

const tag_t	**linked_feature**	Input	The linked feature whose accepted-as-broken status is wanted.
logical *	**is_accepted**	Output	The accepted status of the given linked feature.

UF_WAVE_ask_link_mirror_data (view source)

Defined in: uf_wave.h

Overview
Ask the current source body and mirror datum plane of a linked mirror feature, and the associated xform of each. If the source body's part is not already fully loaded, it is fully loaded if allow_load is true; otherwise, NULL_TAG is returned for all arguments and an appropriate error is given. The same applies to the mirror datum plane's part. If the link is currently broken, NULL_TAG may be returned in one or more of the arguments, and the body_xform and datum_xform arguments may be dumb xforms.

Environment
Internal and External

See Also
UF_WAVE_create_linked_mirror
UF_WAVE_set_link_mirror_data

Required License(s)
gateway

int UF_WAVE_ask_link_mirror_data
(
const tag_t linked_feature,
const logical allow_load,
tag_t * source_body,

tag_t * body_xform,
tag_t * datum_plane,
tag_t * datum_xform
)

const tag_t	**linked_feature**	Input	The linked mirror feature whose source geometry and xforms are wanted.
const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.
tag_t *	**source_body**	Output	The body which is the source geometry of the linked mirror body feature.
tag_t *	**body_xform**	Output	The transform which determines the position of the linked feature relative to its source body.
tag_t *	**datum_plane**	Output	The datum plane which provides the plane of reflection of the linked mirror body feature.
tag_t *	**datum_xform**	Output	The transform which determines the position of the plane of reflection of the linked feature relative to its source datum plane.

UF_WAVE_ask_link_region_sources (view source)

Defined in: uf_wave.h

Overview

Ask the current seed faces and boundary faces of a linked region feature, and the current values of the traverse_interior_edges and delete_openings logicals. seed_faces and boundary_faces are allocated arrays, and should both be freed afterwards with UF_free. If the part containing the source faces is not already fully loaded, it is fully loaded if allow_load is true; otherwise n_seed_faces and n_boundary_faces are set to 0, seed_faces and boundary_faces are empty, the logicals are set to false, and an appropriate error is given. If the link is currently broken and the source data is not available, the same default argument values are returned, though with an return value of 0.

Environment

Internal and External

See Also

UF_WAVE_create_linked_region
UF_WAVE_set_link_region_data

Required License(s)

gateway

int UF_WAVE_ask_link_region_sources
(

```
    const tag_t linked_feature,
    const logical allow_load,
    int * n_seed_faces,
    tag_t * * seed_faces,
    int * n_boundary_faces,
    tag_t * * boundary_faces,
    logical * traverse_interior_edges,
    logical * delete_openings
)
```

const tag_t	**linked_feature**	Input	The linked region feature whose source geometry is wanted.
const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.
int *	**n_seed_faces**	Output	The number of seed faces.
tag_t * *	**seed_faces**	Output to UF_*free*	The array of tags of the seed faces. This must be freed by calling UF_free.
int *	**n_boundary_faces**	Output	The number of boundary faces.
tag_t * *	**boundary_faces**	Output to UF_*free*	The array of tags of the boundary faces. This must be freed by calling UF_free.
logical *	**traverse_interior_edges**	Output	True if the traversal algorithm passes through holes in the body; false if it only passes over the outside surface of the body.
logical *	**delete_openings**	Output	True if any holes in the resulting sheet are closed up; false if they are left.

UF_WAVE_ask_link_source (view source)

Defined in: uf_wave.h

Overview

Ask the current source entity (parent) of a linked feature. If the source entity's part is not already fully loaded, it is fully loaded if allow_load is true; otherwise, NULL_TAG is returned and an appropriate error code is given. If the linked feature is broken, and the source cannot be identified, NULL_TAG is returned. This routine cannot be called for region or mirror linked features, which have multiple source entities. It works for curves, sketches, strings, datums, faces, bodies, and points; for points, the relevant point, curve, conic or arc is returned.

Environment

Internal and External

See Also

UF_WAVE_set_link_data

Required License(s)

gateway

```
int UF_WAVE_ask_link_source
(
    const tag_t linked_feature,
    const logical allow_load,
    tag_t * source_entity
)
```

const tag_t	**linked_feature**	Input	The linked feature whose source geometry is wanted.
const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.
tag_t *	**source_entity**	Output	The tag of the source geometry entity.

UF_WAVE_ask_link_update_time (view source)

Defined in: uf_wave.h

Overview

Queries when a linked feature updates. For linked features which update at a particular timestamp (update_at_timestamp set to true at creation or reparenting), the tag of the feature after which the linked feature updates is returned. For linked features which update after their source has updated completely (update_at_timestamp set to false at creation or reparenting), and for linked features whose type has no specified update behavior (sketches, strings, datums and points), NULL_TAG is returned. If the timestamp feature is not loaded, the behavior is controlled by the allow_load argument. If it is true, the timestamp feature's part is fully loaded; if it is false, an appropriate error code is returned.

Environment

Internal and External

See Also

UF_WAVE_set_link_update_time

Required License(s)

gateway

```
int UF_WAVE_ask_link_update_time
(
    const tag_t linked_feature,
    const logical allow_load,
    tag_t * timestamp_feature
)
```

const tag_t	**linked_feature**	Input	The linked feature whose timestamp feature is wanted.
const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.

tag_t *	**timestamp_feature**	Output	The timestamp feature of the linked feature.

UF_WAVE_ask_link_xform (view source)

Defined in: uf_wave.h

Overview

Given a linked feature, ask the xform associated with it. This can be NULL_TAG, a dumb xform (only possible for broken linked features), or an assy_ctxt_xform. On a linked mirror feature, this returns the xform of the reflected solid: use UF_WAVE_ask_link_mirror_data if the mirror datum plane's xform is wanted. The xform is in the same part as the linked feature, and is always loaded if the feature is.

Environment

Internal and External

See Also

UF_WAVE_set_link_data
UF_WAVE_ask_link_mirror_data

Required License(s)

gateway

```
int UF_WAVE_ask_link_xform
(
    const tag_t linked_feature,
    tag_t * xform
)
```

const tag_t	**linked_feature**	Input	The linked feature whose xform is wanted.
tag_t *	**xform**	Output	The transform of the linked feature relative to its source entity.

UF_WAVE_ask_linked_feature_geom (view source)

Defined in: uf_wave.h

Overview

Returns the tag of the linked entity created by the given linked feature. For linked bodies, faces, mirrors, and regions the linked sheet or solid body is returned. For linked sketches, the linked sketch entity is returned. For linked points, curves and datums the linked entity is returned. For linked strings, which do not create a single linked entity, NULL_TAG is returned. This routine works even if the link is currently broken.

Environment

Internal and External

See Also
UF_WAVE_ask_link_source

History
This function was originally released in V15.0

Required License(s)
gateway

int UF_WAVE_ask_linked_feature_geom
(
const tag_t linked_feature,
tag_t * linked_geom
)

const tag_t	**linked_feature**	Input	The linked feature whose geometry is wanted.
tag_t *	**linked_geom**	Output	The tag of the entity created by the linked feature.

UF_WAVE_ask_linked_feature_info (view source)

Defined in: uf_wave.h

Overview
Given the tag of a linked geometry, ask the name of its associated feature, the name of its owning part, and the name of its source part if the link is not broken. If the link is broken then the source_part_name field in the UF_WAVE_linked_feature_info_s structure will be NULL. Also note that for linked mirrors the source_part_name consists of both the owning part name of the parent feature and the owning part name of the mirror plane.

Environment
Internal and External

See Also
UF_WAVE_init_linked_feature_info
UF_WAVE_free_linked_feature_info
UF_WAVE_ask_link_source

History
New in NX3.0.2 after the Release

Required License(s)
gateway

int UF_WAVE_ask_linked_feature_info
(
const tag_t linked_geom,
UF_WAVE_linked_feature_info_p_t name_store
)

const tag_t	**linked_geom**	Input	Tag of the linked geometry.

UF_WAVE_linked_feature_info_p_t	**name_store**	Output to UF_*free*	The names of the associated feature, owning part, and the source part. Use UF_WAVE_init_linked_feature_info to initailize before calling this function. Use UF_WAVE_free_linked_feature_info to free the memory.

UF_WAVE_ask_linked_feature_map (view source)

Defined in: uf_wave.h

Overview

Returns arrays of the source geometry used by the linked feature and the linked geometry created by it when given a linked feature. The arrays are in the same order, so can be used to find the linked geometry corresponding to specific source geometry, or vice versa. n_map_items gives the total length of each array.
The arrays should both be freed afterwards with UF_free(). For linked bodies, faces, mirrors and regions all the linked faces and edges are mapped, except for any that no longer have associated geometry in the linked feature. For features that create a sheet body, the linked faces and all the edges joining or bounding them are mapped. The arrays list all the faces first and then all the edges. For linked sketches and strings, all the curves in the source sketch or string are mapped. This routine returns an appropriate error code if passed a linked point, datum, or curve. If the link is currently broken and the source data is not available, n_map_items is set to 0. If the source data's part is not already fully loaded, it is fully loaded if allow_load is true; otherwise, 0 or NULL is returned for all arguments and an appropriate error code is given.

Instead of making repeated calls to UF_WAVE_map_source_to_link_geom or UF_WAVE_map_link_geom_to_source, it is more efficient to use this routine and use qsort and bsearch to look through the returned arrays.

Environment

Internal and External

See Also

UF_WAVE_map_source_to_link_geom
UF_WAVE_map_link_geom_to_source

History

This function was originally released in V15.0

Required License(s)

gateway

```
int UF_WAVE_ask_linked_feature_map
(
    const tag_t linked_feature,
    const logical allow_load,
    int * n_map_items,
    tag_t * * source_geom,
    tag_t * * linked_geom
```

)

const tag_t	**linked_feature**	Input	The linked feature whose geometry map is wanted.
const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.
int *	**n_map_items**	Output	The number of source and linked entities in the arrays
tag_t * *	**source_geom**	Output to UF_*free*	The tags of the entities composing the source geometry. This array must be freed by calling UF_free.
tag_t * *	**linked_geom**	Output to UF_*free*	The corresponding tags of the entities created by the linked feature. This array must be freed by calling UF_free.

UF_WAVE_ask_linked_pt_angle (view source)

Defined in: uf_wave.h

Overview
Returns the angle of a linked point feature that is positioned with an angle on an arc. Any other linked point, including a broken linked point, will return NULL_TAG as the angle.

Environment
Internal and External

See Also
UF_WAVE_create_linked_pt_angle
UF_WAVE_set_linked_pt_angle

History
This function was originally released in V15.0

Required License(s)
gateway

```
int UF_WAVE_ask_linked_pt_angle
(
    const tag_t linked_feature,
    tag_t * angle
)
```

const tag_t	**linked_feature**	Input	The linked point feature whose angle is wanted.
tag_t *	**angle**	Output	The tag of the angle (smart scalar) which defines the position of the linked point feature around its defining arc.

UF_WAVE_ask_linked_pt_curve_prm (view source)

Defined in: uf_wave.h

Overview
Ask the parameter of a linked point feature that is positioned with a parameter on a curve. Any other linked point, including a broken linked point, returns NULL_TAG as the parameter.

Environment
Internal and External

See Also
UF_WAVE_create_linked_pt_curve
UF_WAVE_set_linked_pt_curve

History
This function was originally released in V15.0

Required License(s)
gateway

```
int UF_WAVE_ask_linked_pt_curve_prm
(
    const tag_t linked_feature,
    tag_t * prm
)
```

const tag_t	**linked_feature**	Input	The linked point feature whose parameter is wanted.
tag_t *	**prm**	Output	The tag of the parameter (smart scalar) which defines the position of the linked point feature along its defining curve.

UF_WAVE_ask_out_of_date_objects (view source)

Defined in: uf_wave.h

Overview
Gets an array of objects that are out of date in the given part.

Environment
Internal and External

Required License(s)
gateway

```
int UF_WAVE_ask_out_of_date_objects
(
    tag_t part,
    int * n_objects,
    tag_t * * objects
```

)

tag_t	**part**	Input	The part that may contain out of date objects
int *	**n_objects**	Output	Number of out of date objects in the part.
tag_t * *	**objects**	Output to UF_*free*	Array of tags of objects that are out of date. Use UF_free to deallocate memory when done.

UF_WAVE_ask_out_of_date_parts (view source)

Defined in: uf_wave.h

Overview
Returns the parts currently loaded in the session that are out of date.

Environment
Internal and External

Required License(s)
gateway

int UF_WAVE_ask_out_of_date_parts
(
int * n_parts,
tag_t * * parts
)

int *	**n_parts**	Output	Number of out of date parts in session.
tag_t * *	**parts**	Output to UF_*free*	Array of tags of parts that are out of date. Use UF_free to deallocate memory.

UF_WAVE_ask_session_delay (view source)

Defined in: uf_wave.h

Overview
Determines if the session has interpart update delayed, and sets the delayed parameter to true if so, false otherwise.

Environment
Internal and External

Required License(s)
gateway

int UF_WAVE_ask_session_delay
(

```
    logical * delayed
)
```

logical *	**delayed**	Output	True if the session is currently delayed.

UF_WAVE_convert_links_to_use_product_interface (view source)

Defined in: uf_wave.h

Overview

Convert legacy (before NX9) linked features and linked expressions to use new functionality of Product Interface object.

This function will also convert the legacy Product Interface objects in the assembly or used by the linked objects if the source part is not a component of the assembly.

User needs to fully load all their assembly parts and the source parts and turn off interpart update delay before calling this function to convert.

This is temporary and will be retired in NX11.

Environment

Internal

History

NX9.0.3

Required License(s)

gateway

```
int UF_WAVE_convert_links_to_use_product_interface
(

)
```

UF_WAVE_copy_component_as (view source)

Defined in: uf_wave.h

Overview

This routine makes a copy of the parent part as the new part name in the same assembly as the parent part. Any linked features that are in the parent part will still be maintained in the new part.

This routine is similar to performing a save-as except that the linked objects will not become dumb to assembly position.

An update is required after the call to UF_WAVE_copy_component_as to position the new component and resolve the links.

Environment
Internal and External

Required License(s)
gateway

```
int UF_WAVE_copy_component_as
(
    const tag_t source_part_occurrence,
    const char * source_part_name,
    const char * new_part_name,
    const char * reference_set_name,
    const char * instance_name,
    double transform [ 4 ] [ 4 ] ,
    tag_t * new_instance,
    tag_t * new_part
)
```

const tag_t	**source_part_occurrence**	Input	The part occurrence that is being copied
const char *	**source_part_name**	Input	The component part name that is to be copied
const char *	**new_part_name**	Input	The new component part name that will exist in the current assembly.
const char *	**reference_set_name**	Input	Name of reference set to use from component parts. The refset_name cannot exceed 30 characters, cannot include a directory path, and should not have a file extension.
const char *	**instance_name**	Input	Name of new instance The instance_name cannot exceed 30 characters, cannot include a directory path, and should not have a file extension.
double	**transform [4] [4]**	Input	The transform to apply to the new component
tag_t *	**new_instance**	Output	The new instance tag created in the assembly
tag_t *	**new_part**	Output	The new part tag that is a copy of parent part.

UF_WAVE_create_linked_body (view source)

Defined in: uf_wave.h

Overview
Create a linked body feature, and its associated body. The source body must not be an occurrence, but can be in any part. It can be either a solid body or a sheet body. Its part is fully loaded if necessary, as is the destination part of the feature.

In order to avoid the potential performance issue when multiple linked features are being created, it is required that the application call UF_MODL_update() once after the linked features have been created using UF_WAVE_xxx function calls.

Environment
Internal and External

Required License(s)
assemblies

int UF_WAVE_create_linked_body
(
const tag_t body,
const tag_t xform,
const tag_t object_in_part,
const logical update_at_timestamp,
tag_t * linked_feature
)

const tag_t	**body**	Input	The body which is the source geometry of the new linked body feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains body.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
const logical	**update_at_timestamp**	Input	If update_at_timestamp is true, the linked body feature updates at a time determined by its creation timestamp. If it is false, it updates after its source body is completely generated.
tag_t *	**linked_feature**	Output	The tag of the new linked body feature.

UF_WAVE_create_linked_curve (view source)

Defined in: uf_wave.h

Overview
Create a linked curve feature, and its associated curve in the same part as object_in_part. The source curve must not be an occurrence, but can be in any part. It can be a solid edge. Its part is fully loaded if necessary, as is the destination part of the feature.

It is required that the application call UF_MODL_update() once after the linked curve features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment
Internal and External

Required License(s)
assemblies

```
int UF_WAVE_create_linked_curve
(
    const tag_t curve,
    const tag_t xform,
    const tag_t object_in_part,
    const logical update_at_timestamp,
    tag_t * linked_feature
)
```

const tag_t	**curve**	Input	The curve which is the source geometry of the new linked curve feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains curve.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
const logical	**update_at_timestamp**	Input	If true, the linked curve feature updates at a time determined by its creation timestamp. If it is false, it updates after its source curve is completely generated.
tag_t *	**linked_feature**	Output	The tag of the new linked curve feature.

UF_WAVE_create_linked_datum (view source)

Defined in: uf_wave.h

Overview

Create a linked datum feature, and its associated datum in the same part as object_in_part. The source datum must not be an occurrence, but can be in any part, except that linked datum features must be interpart. It can be a datum axis or a datum plane. Its part is fully loaded if necessary, as is the destination part of the feature.
Note that a datum is not affected by the update of later features, so an update_at_timestamp argument would not affect the behavior of a linked datum.

It is required that the application call UF_MODL_update() once after the linked datum features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment

Internal and External

Required License(s)

assemblies

```
int UF_WAVE_create_linked_datum
(
    const tag_t datum,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**datum**	Input	The datum plane or axis which is the source geometry of the new linked datum feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains datum. Linked datums must be interpart.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked datum feature.

UF_WAVE_create_linked_face (view source)

Defined in: uf_wave.h

Overview

Create a linked face feature, and its associated sheet body in the same part as object_in_part. The source face must not be an occurrence, but can be in any part. Its part is fully loaded if necessary, as is the destination part of the feature.

Environment

Internal and External

Required License(s)

assemblies

```
int UF_WAVE_create_linked_face
(
    const tag_t face,
    const tag_t xform,
    const tag_t object_in_part,
    const logical update_at_timestamp,
    tag_t * linked_feature
)
```

const tag_t	**face**	Input	The face which is the source geometry of the new linked face feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as

			object_in_part whose to_part_occ contains face.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
const logical	**update_at_timestamp**	Input	If update_at_timestamp is true, the linked face feature updates at a time determined by its creation timestamp. If it is false, it updates after its source face's body is completely generated.
tag_t *	**linked_feature**	Output	The tag of the new linked face feature.

UF_WAVE_create_linked_mirror (view source)

Defined in: uf_wave.h

Overview

Creates a linked mirror body feature, and its associated body, in the same part as object_in_body. The source body and mirror datum plane must not be occurrences, but can be in any part or parts. Their parts are fully loaded if necessary, as is the destination part of the feature. The source body can be either a solid body or a sheet body.

It is required that the application call UF_MODL_update() once after the linked mirror body features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment

Internal and External

See Also

UF_WAVE_ask_link_mirror_data
UF_WAVE_set_link_mirror_data

Required License(s)

assemblies

```
int UF_WAVE_create_linked_mirror
(
    const tag_t body,
    const tag_t body_xform,
    const tag_t datum_plane,
    const tag_t datum_xform,
    const tag_t object_in_part,
    const logical update_at_timestamp,
    tag_t * linked_feature
)
```

const tag_t	**body**	Input	The body which is the source geometry of the new linked mirror body feature.
const tag_t	**body_xform**	Input	The transform which determines the position of the linked feature relative to its source body, before the reflection is applied. Must be NULL_TAG (apply no

			transformation) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains body.
const tag_t	**datum_plane**	Input	The datum plane which provides the plane of reflection of the new linked mirror body feature.
const tag_t	**datum_xform**	Input	The transform which determines the position of the plane of reflection of the linked feature relative to its source datum plane. Must be NULL_TAG (apply no transformation) or an assy_ctxt_xform whose to_part_occ contains datum_plane.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
const logical	**update_at_timestamp**	Input	If true, the feature updates at its creation timestamp. If false, the feature updates after its source geometry is completely generated.
tag_t *	**linked_feature**	Output	The tag of the new linked mirror feature.

UF_WAVE_create_linked_part (view source)

Defined in: uf_wave.h

Overview

Creates a linked part of the specified name whose contents are all the entities in the given smart part contained in ref_set. The linked part has the same units as the source part. ref_set can be NULL_TAG, in which case the entire contents of start_part are copied into the linked part. ref_set cannot be the empty reference set. start_part is fully loaded if necessary. This routine only runs if a WAVE licence is available, but it does not use up that license.

Environment

Internal and External

History

This function was originally released in V15.0.

Required License(s)

assemblies

```
int UF_WAVE_create_linked_part
(
    const tag_t start_part,
    const tag_t ref_set,
    char * linked_part_name,
    tag_t * linked_part
)
```

const tag_t	**start_part**	Input	Start part from which the linked part is to be made.
const tag_t	**ref_set**	Input	The reference set defining the entities to be linked into the new linked part.
char *	**linked_part_name**	Input	The name of the new linked part.
tag_t *	**linked_part**	Output	The tag of the new linked part.

UF_WAVE_create_linked_pt_angle (view source)

Defined in: uf_wave.h

Overview

Create a linked point feature, and its associated point, in the same part as object_in_part, from a source arc and an angle (smart scalar) along it. The source arc must not be an occurrence, but can be in any part, except that linked point features must be interpart. Its part is fully loaded if necessary.

The linked point updates after the source arc (and its underlying body, if any) has completely updated.

Environment

Internal and External

See Also

UF_WAVE_set_linked_pt_angle

Required License(s)

assemblies

```
int UF_WAVE_create_linked_pt_angle
(
    const tag_t arc,
    const tag_t angle,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**arc**	Input	The arc which defines the source geometry of the new linked point feature.
const tag_t	**angle**	Input	The angle (smart scalar) which defines the position of the new linked point feature around arc. It must be in the same part as object_in_part. Its value is in radians.

const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains arc.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked point feature.

UF_WAVE_create_linked_pt_center (view source)

Defined in: uf_wave.h

Overview

Create a linked point feature, and its associated point, in the same part as object_in_part, at the center of a source conic. The source conic must not be an occurrence, but can be in any part, except that linked point features must be interpart. Its part is fully loaded if necessary, as is the destination part.

The linked point updates after the source conic (and its underlying solid, if any) has completely updated.

Environment

Internal and External

See Also

UF_WAVE_set_linked_pt_center

Required License(s)

assemblies

```
int UF_WAVE_create_linked_pt_center
(
    const tag_t conic,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**conic**	Input	The conic whose center is the source geometry of the new linked point feature.
const tag_t	**xform**	Input	Gives the transformation that is applied to the center of the source curve to generate the linked point. It must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains conic.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.

tag_t *	**linked_feature**	Output	The tag of the new linked point feature.

UF_WAVE_create_linked_pt_curve (view source)

Defined in: uf_wave.h

Overview

Creates a linked point feature, and its associated point, in the same part as object_in_part, in a position corresponding to the point on curve indicated by the parameter (prm). The source curve must not be an occurrence, but can be in any part, except that linked point features must be interpart. Its part is fully loaded if necessary. The linked point updates after the source curve (and its underlying body, if any) has completely updated.

Environment

Internal and External

See Also

UF_WAVE_set_linked_pt_curve

Required License(s)

assemblies

```
int UF_WAVE_create_linked_pt_curve
(
    const tag_t curve,
    const tag_t prm,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**curve**	Input	The curve which defines the source geometry of the new linked point feature.
const tag_t	**prm**	Input	The parameter (smart scalar) which defines the position of the new linked point feature along curve. It must be in the same part as object_in_part.
const tag_t	**xform**	Input	Gives the transformation that is applied to the point on the source curve to generate the linked point. It must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains curve.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked point feature.

UF_WAVE_create_linked_pt_point (view source)

Defined in: uf_wave.h

Overview

Create a linked point feature, and its associated point, in the same part as object_in_part, in a position corresponding to the source point. The source point must not be an occurrence, but can be in any part, except that linked point features must be interpart. Its part is fully loaded if necessary.

The source point can either be an ordinary "dumb" point, or a smart point, such as one created with an UF_SO_create_point_xxx routine. The linked point updates after the source point has completely updated.

Environment

Internal and External

See Also

UF_WAVE_set_linked_pt_point

Required License(s)

assemblies

```
int UF_WAVE_create_linked_pt_point
(
    const tag_t point,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**point**	Input	The point which is the source geometry of the new linked point feature.
const tag_t	**xform**	Input	Gives the transformation that is applied to the source point to generate the linked point. It must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains point.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked point feature.

UF_WAVE_create_linked_region (view source)

Defined in: uf_wave.h

Overview

Create a linked region feature, and its associated sheet body, in the same part as object_in_part. The region includes all faces in the

seed_faces array, and all faces on the same solid that can be reached from them without crossing any face in the boundary_faces array. All faces in both arrays must be on the same solid, which can be in any part. They cannot be occurrences. The source solid's part is fully loaded if necessary, as is the destination part of the feature. Boundary faces are not normally part of the created region, but if a boundary face is a seed face as well, it is added to the region, though it does not cause any of its adjacent faces to be added to the region. If n_boundary_faces is 0, then all faces on the seed faces' solid are part of the region. At present, there must only be one face in the seed_faces array.

It is required that the application call UF_MODL_update() once after the linked region features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment

Internal and External

See Also

UF_WAVE_ask_link_region_sources
UF_WAVE_set_link_region_data

Required License(s)

assemblies

```
int UF_WAVE_create_linked_region
(
    const int n_seed_faces,
    const tag_t * seed_faces,
    const int n_boundary_faces,
    const tag_t * boundary_faces,
    const tag_t xform,
    const tag_t object_in_part,
    const logical update_at_timestamp,
    const logical traverse_interior_edges,
    const logical delete_openings,
    tag_t * linked_feature
)
```

const int	**n_seed_faces**	Input	The number of seed faces.
const tag_t *	**seed_faces**	Input	The array of seed faces.
const int	**n_boundary_faces**	Input	The number of boundary faces.
const tag_t *	**boundary_faces**	Input	The array of boundary faces.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (apply no transformation) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains the body.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.

const logical	**update_at_timestamp**	Input	If true, the linked region feature updates at a time determined by its creation timestamp. If false, the feature updates after its source faces body is completely generated.
const logical	**traverse_interior_edges**	Input	If true, the traversal algorithm passes through holes in the body. if false, it only passes over the outside surface of the body.
const logical	**delete_openings**	Input	If true, any holes in the resulting sheet are closed up; if false they are left.
tag_t *	**linked_feature**	Output	The tag of the new linked region feature.

UF_WAVE_create_linked_route_port (view source)

Defined in: uf_wave.h

Overview

Create a linked routing port with its associated characteristics in the same part as object_in_part. The parent/source port must not be an occurrence, but can be in any part. The port to be linked can be of fitting, fixture, multi or fabrication stock port type. The linked port will show up as a LINKED_PORT feature and will always be positioned using the provided assembly context xform object with respect to its parent. The source part is fully loaded as necessary, as is the destination part.

Return

Return code :
= 0 : sucessful
!= 0 : linking/extracting error

Environment

Internal and External

History

New in V18.

Required License(s)

assemblies

```
int UF_WAVE_create_linked_route_port
(
    const tag_t port,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_port_feature
)
```

const tag_t	**port**	Input	The port which is the source geometry of the new linked port.
const tag_t	**xform**	Input	The transform which determines the position of the linked object relative to its source geometry. Must be NULL_TAG

			(apply no transformation) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains curve.
const tag_t	**object_in_part**	Input	Determines the part within which the new linked port copy is made.
tag_t *	**linked_port_feature**	Output	The tag of the new linked port feature.

UF_WAVE_create_linked_route_segment (view source)

Defined in: uf_wave.h

Overview

Create a linked routing segment, and its associated end control points in the same part as object_in_part. The parent/source segment must not be an occurrence, but can be in any part. This function extracts a LINKED_CURVE feature in the work part from the parent segment and then creates a routing segment as following the extracted LINKED_CURVE feature. The end control points of the linked segment will be derived from the end points of this extracted LINKED_CURVE feature. The "linked" segment will always be positioned using the provided assembly context xform object with respect to its parent. The source part is fully loaded as necessary, as is the destination part.

NOTE: The actual linked geometry for a "linked" segment is the LINKED_CURVE. The linked segment is just a smart segment in the work part that follows the extracted LINKED_CURVE.

Return

Return code :
= 0 : sucessful
!= 0 : linking/extracting error

Environment

Internal and External

History

New in V18.

Required License(s)

assemblies

```
int UF_WAVE_create_linked_route_segment
(
    const tag_t segment,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_curve_feature,
    tag_t * linked_segment
)
```

const tag_t	**segment**	Input	The segment which is the source geometry of the new linked segment.

const tag_t	**xform**	Input	The transform which determines the position of the linked object relative to its source geometry. Must be NULL_TAG (apply no transformation) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains curve.
const tag_t	**object_in_part**	Input	Determines the part within which the new linked segment copy is made.
tag_t *	**linked_curve_feature**	Output	The tag of the LINKED_CURVE feature which the linked segment follows.
tag_t *	**linked_segment**	Output	The tag of the segment that follows the LINKED_CURVE

UF_WAVE_create_linked_sketch (view source)

Defined in: uf_wave.h

Overview

Create a linked sketch feature, and its associated sketch, in the same part as object_in_part. The source sketch must not be an occurrence, and must not have been created using the old (pre-v13) sketcher, but can be in any part, except that linked sketch features must be interpart. Its part is fully loaded if necessary, as is the destination part of the feature.

Note that a sketch is not affected by the update of later features, so an update_at_timestamp argument would not affect the behavior of a linked sketch.

It is required that the application call UF_MODL_update() once after the linked sketch features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment

Internal and External

Required License(s)

assemblies

```
int UF_WAVE_create_linked_sketch
(
    const tag_t sketch,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**sketch**	Input	The sketch which is the source geometry of the new linked sketch feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG

			(create position independent link) or an assy_ctxt_xform in the same part as object_in_part whose to_part_occ contains sketch. Linked sketches must be interpart.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked sketch feature.

UF_WAVE_create_linked_string (view source)

Defined in: uf_wave.h

Overview

Create a linked string feature, and its associated curves, in the same part as object_in_part. The source must either be an intersection curve feature, a projection curve feature, an offset curve feature, or a solid face. It can be in any part, except that linked string features must be interpart. Its part is fully loaded if necessary, as is the destination part of the feature.

A string feature's definition is not affected by the update of later features, so an update_at_timestamp argument would not affect the behavior of a linked string.

It is required that the application call UF_MODL_update() once after the linked string features have been created. See UF_WAVE_create_linked_body for more clarification.

Environment

Internal and External

History

In V15.0, this function was enhanced to accept a solid face for the source string.

Required License(s)

assemblies

```
int UF_WAVE_create_linked_string
(
    const tag_t string,
    const tag_t xform,
    const tag_t object_in_part,
    tag_t * linked_feature
)
```

const tag_t	**string**	Input	The string feature whose curves are the source geometry of the new linked string feature.
const tag_t	**xform**	Input	The transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (create position independent link) or an assy_ctxt_xform in the same part as

			object_in_part whose to_part_occ contains string. Linked strings must be interpart.
const tag_t	**object_in_part**	Input	Determines the part within which the new feature is made.
tag_t *	**linked_feature**	Output	The tag of the new linked string feature.

UF_WAVE_free_linked_feature_info (view source)

Defined in: uf_wave.h

Overview
Free memory allocated in UF_WAVE_linked_feature_info_s data structure.

Environment
Internal and External

See Also
UF_WAVE_free_linked_feature_info

History
New in NX3.0.2 after the Release

Required License(s)
gateway

int UF_WAVE_free_linked_feature_info
(
UF_WAVE_linked_feature_info_p_t name_store
)

UF_WAVE_linked_feature_info_p_t	**name_store**	Input	Pointer to the data structure.

UF_WAVE_freeze (view source)

Defined in: uf_wave.h

Overview
Freezes a set of parts from interpart update in the current session.

Environment
Internal and External

Required License(s)
wave

int UF_WAVE_freeze
(

```
    int n_parts,
    tag_t * parts
)
```

int	**n_parts**	Input	Number of parts to freeze
tag_t *	**parts**	Input	Array of tags of parts to freeze.

UF_WAVE_freeze_persistently (view source)

Defined in: uf_wave.h

Overview
Freezes a set of parts from interpart update persistently. Persistently frozen parts are marked frozen so that they do not update (once saved in this state) in any session they are subsequently loaded into.

Environment
Internal and External

Required License(s)
wave

```
int UF_WAVE_freeze_persistently
(
    int n_parts,
    tag_t * parts
)
```

int	**n_parts**	Input	Number of parts to freeze persistently.
tag_t *	**parts**	Input	Array of tags of parts to freeze persistently.

UF_WAVE_init_linked_feature_info (view source)

Defined in: uf_wave.h

Overview
Initialize UF_WAVE_linked_feature_info_s data structure.

Environment
Internal and External

See Also
UF_WAVE_free_linked_feature_info

History
New in NX3.0.2 after the Release

Required License(s)
gateway

```
int UF_WAVE_init_linked_feature_info
(
    UF_WAVE_linked_feature_info_p_t name_store
)
```

UF_WAVE_linked_feature_info_p_t	**name_store**	Input	Pointer to the data structure.

UF_WAVE_is_link_broken (view source)

Defined in: uf_wave.h

Overview

Given a linked feature, ask if the link is broken (i.e. does not have access to all its source geometry, and hence cannot update). For linked mirror features, lack of access to either the source solid or the mirror datum plane means the link is broken.

A link can lack access to its source geometry because the source geometry has been deleted, because the connecting xform's assembly has been altered, or because the link has been explicitly broken, including by reparenting it using UF_WAVE_set_link_data. A link is NOT broken just because its source geometry is currently unloaded.

Environment

Internal and External

See Also

UF_WAVE_set_link_data
UF_WAVE_accept_link_broken
UF_WAVE_ask_link_accept_broken

Required License(s)

gateway

```
int UF_WAVE_is_link_broken
(
    const tag_t linked_feature,
    logical * is_broken
)
```

const tag_t	**linked_feature**	Input	The linked feature whose broken status is to be checked.
logical *	**is_broken**	Output	True if the feature is broken, false otherwise.

UF_WAVE_is_pilo_xform (view source)

Defined in: uf_wave.h

Overview
Given the tag of a xform, this routine will return whether the xform is PILO(Position Independent) or not.

Environment
Internal and External

History
New in NX5.0

Required License(s)
gateway

```
logical UF_WAVE_is_pilo_xform
(
    tag_t xform
)
```

tag_t	**xform**	Input	Tag of the xform.

UF_WAVE_load_parents (view source)

Defined in: uf_wave.h

Overview
Load find and load the WAVE parent parts of a given part file. This will also load parent parts for any Mating Conditions of Interpart Expressions

Environment
Internal and External

Required License(s)
wave

```
int UF_WAVE_load_parents
(
    tag_t part,
    int parent_option,
    int * n_failures,
    char * * * failing_parts,
    int * * failing_statuses
)
```

tag_t	**part**	Input	Tag of the part for which to load the parts of
int	**parent_option**	Input	This is the WAVE load option as described in uf_assem_types.h UF_ASSEM_none Don't load any non-loaded parents, but fully load partially loaded parents. UF_ASSEM_immediate Only load the immediate parents (First-level parents) UF_ASSEM_all Recursively load parents, and parents of

			parents until there are no more parents left to load.
int *	**n_failures**	Output	Number of parent parts that failed to load
char * * *	**failing_parts**	Output to UF_*free*	Names of the parts that failed to load Use UF_free_string_array to deallocate memory when done.
int * *	**failing_statuses**	Output to UF_*free*	Error codes for the failures in corresponding order Use UF_free to deallocate memory when done.

UF_WAVE_map_link_geom_to_source (view source)

Defined in: uf_wave.h

Overview

Given a linked feature and geometry created by it, return the corresponding geometry in its source. For linked bodies, faces, mirrors and regions any of the faces or edges can be mapped. For linked sketches and strings, any of the source curves can be mapped. This routine returns an appropriate error code if passed a linked point, datum, or curve. If geometry that does not form part of the linked feature is passed in, or if the link is currently broken and the source data is not available, NULL_TAG is returned. This routine is likely to be slower than UF_WAVE_map_source_to_link_geom, and so should be used with caution. If the source data's part is not already fully loaded, it is fully loaded if allow_load is true; otherwise, NULL_TAG is returned and an appropriate error code is given.

Environment

Internal and External

See Also

UF_WAVE_ask_linked_feature_map
UF_WAVE_map_source_to_link_geom

History

This function was originally released in V15.0

Required License(s)

gateway

```
int UF_WAVE_map_link_geom_to_source
(
    const tag_t linked_feature,
    const tag_t linked_geom,
    const logical allow_load,
    tag_t * source_geom
)
```

const tag_t	**linked_feature**	Input	The linked feature whose source geometry is wanted.
const tag_t	**linked_geom**	Input	The tag of the entity created by the linked feature.

const logical	**allow_load**	Input	True if the source geometry's part can be loaded, false otherwise.
tag_t *	**source_geom**	Output	The tag of the corresponding entity contained in the source geometry.

UF_WAVE_map_source_to_link_geom (view source)

Defined in: uf_wave.h

Overview

Given a linked feature and geometry belonging to its source, return the corresponding linked geometry created by the linked feature. For linked bodies, faces, mirrors and regions any of the faces or edges can be mapped. For linked sketches and strings, any of the source curves can be mapped. This routine returns an appropriate error code if passed a linked point, datum, or curve. If geometry that does not form part of the source of the linked feature is passed in, NULL_TAG is returned. NULL_TAG is also returned if the geometry is part of the source, but its associated linked geometry no longer exists in the linked feature (for instance, it has been removed by a later feature applied to the linked geometry).

Environment

Internal and External

See Also

UF_WAVE_ask_linked_feature_map
UF_WAVE_map_link_geom_to_source

History

This function was originally released in V15.0

Required License(s)

gateway

```
int UF_WAVE_map_source_to_link_geom
(
    const tag_t linked_feature,
    const tag_t source_geom,
    tag_t * linked_geom
)
```

const tag_t	**linked_feature**	Input	The linked feature whose linked geometry is wanted.
const tag_t	**source_geom**	Input	The tag of the entity contained in the source geometry
tag_t *	**linked_geom**	Output	The tag of the corresponding entity created by the linked feature.

UF_WAVE_set_link_data (view source)

Defined in: uf_wave.h

Overview

Set the source entity, xform and update behavior of a linked feature. This reparents it if the new source entity is different. The new source entity must be of the same type as the previous source entity of the linked feature, but it can be in a different part. It cannot be an occurrence. It can be NULL_TAG, in which case the feature's link is broken (and xform must also be NULL_TAG); this also accepts the link as broken. The source entity's part is fully loaded if it is not already fully loaded. The xform can be set to NULL_TAG (for an identity transform) or any assy_ctxt_xform in the same part as the linked feature. update_at_timestamp sets the update behavior of the feature, in the same way as it does when a linked feature is created. It has no effect if the link is being broken, or if the link is one of the link types (sketches, strings, datums) with no update_at_timestamp option at creation.

Linked datum axis features cannot be reparented to datum planes, nor vice versa. This routine cannot be called for region or mirror linked features, which have multiple source entities, nor for point linked features, which have multiple definitions. It works for curves, sketches, strings, datums, faces and bodies. Intersection curve strings, projection curve strings, offset curve and face edge(s) strings can be reparented to one another.

Environment

Internal and External

See Also

UF_WAVE_ask_link_source
UF_WAVE_ask_link_xform
UF_WAVE_ask_link_update_time

History

In V15.0, this function was enhanced to allow reparenting for face edge strings.

Required License(s)

gateway

```
int UF_WAVE_set_link_data
(
    const tag_t linked_feature,
    const tag_t source_entity,
    const tag_t xform,
    const logical update_at_timestamp
)
```

const tag_t	**linked_feature**	Input	The linked feature whose properties are to be set.
const tag_t	**source_entity**	Input	The tag of the new source geometry; can be NULL_TAG, in which case the link is broken.
const tag_t	**xform**	Input	The new transform which determines the position of the linked feature relative to its source geometry. Must be NULL_TAG (identity transform) or an assy_ctxt_xform

			in the same part as linked_feature whose to_part_occ contains source_entity.
const logical	**update_at_timestamp**	Input	If true, the linked feature now updates at its timestamp. If false, the feature now updates after its source geometry has updated.

UF_WAVE_set_link_mirror_data (view source)

Defined in: uf_wave.h

Overview

Set the source body, mirror datum plane and associated xforms of a linked mirror feature. This reparents it if the body or plane are different. The new source entities can be in different parts from the old ones, and need not be in the same part as one another. Their parts are fully loaded if they are not already fully loaded. body_xform and datum_xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked_feature. If source_body and datum_plane are both NULL_TAG, the links to the previous source body and mirror datum plane are broken (and body_xform and datum_xform must both also be NULL_TAG). This also accepts the link as broken. It is not possible to set only one of source_body and datum_plane to NULL_TAG. The update_at_timestamp argument sets the update behavior of the feature, in the same way as it does when a linked feature is created. It is ignored if the link is being broken.

Environment

Internal and External

See Also

UF_WAVE_ask_link_mirror_data
UF_WAVE_create_linked_mirror

Required License(s)

solid_modeling

```
int UF_WAVE_set_link_mirror_data
(
    const tag_t linked_feature,
    const tag_t source_body,
    const tag_t body_xform,
    const tag_t datum_plane,
    const tag_t datum_xform,
    const logical update_at_timestamp
)
```

const tag_t	**linked_feature**	Input	The linked mirror feature whose properties are to be set.
const tag_t	**source_body**	Input	The body which is the new source geometry of the linked mirror body feature; can be NULL_TAG, in which case the link is broken.

const tag_t	**body_xform**	Input	The new transform which determines the position of the linked feature relative to its source body, before the reflection is applied. Must be NULL_TAG (identity transform) or an assy_ctxt_xform in the same part as linked_feature whose to_part_occ contains source_body.
const tag_t	**datum_plane**	Input	The datum plane which provides the new plane of reflection of the linked mirror body feature; can be NULL_TAG, in which case the link is broken.
const tag_t	**datum_xform**	Input	The new transform which determines the position of the plane of reflection of the linked feature relative to its source datum plane. Must be NULL_TAG (identity transform) or an assy_ctxt_xform in the same part as linked_feature whose to_part_occ contains datum_plane.
const logical	**update_at_timestamp**	Input	If true, the feature updates at its timestamp. If false, the feature updates after its source geometry has updated.

UF_WAVE_set_link_region_data (view source)

Defined in: uf_wave.h

Overview

Sets the seed faces, boundary faces, xform and traverse_interior_edges and delete_openings logicals of a linked region feature. This reparents it if the faces are changed. n_seed_faces is the length of the array seed_faces, which gives the new seed faces for the region feature. n_boundary_faces is the length of the array boundary_faces, which gives the new boundary faces for the region feature. All faces in the input arrays have to be in the same body, which need not be in the same part as the previous sources of the feature. The part containing the body is fully loaded if it is not already fully loaded. The xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked_feature whose to_part_occ contains the body. If n_seed_faces and n_boundary_faces are both 0, the feature's link is broken (and xform must also be NULL_TAG). This also accepts the link as broken. The update_at_timestamp argument sets the update behavior of the feature, in the same way as it does when a linked feature is created. It is ignored if the link is being broken. At present, there must only be one face in the seed_faces array, unless the link is being broken.

Environment

Internal and External

See Also

UF_WAVE_create_linked_region
UF_WAVE_ask_link_region_sources

Required License(s)

solid_modeling

```
int UF_WAVE_set_link_region_data
(
    const tag_t linked_feature,
    const int n_seed_faces,
    const tag_t * seed_faces,
    const int n_boundary_faces,
    const tag_t * boundary_faces,
    const tag_t xform,
    const logical update_at_timestamp,
    const logical traverse_interior_edges,
    const logical delete_openings
)
```

const tag_t	**linked_feature**	Input	The linked region feature whose properties are to be set.
const int	**n_seed_faces**	Input	The number of seed faces; can be 0, in which case the link is broken.
const tag_t *	**seed_faces**	Input	The array of seed faces.
const int	**n_boundary_faces**	Input	The number of boundary faces.
const tag_t *	**boundary_faces**	Input	The array of boundary faces.
const tag_t	**xform**	Input	The new transform which determines the position of the linked region feature relative to its source geometry. Must be NULL_TAG (identity transform) or an assy_ctxt_xform in the same part as linked_feature.
const logical	**update_at_timestamp**	Input	If true, the linked region feature now updates at its timestamp. If false, the feature now updates after its source geometry has updated.
const logical	**traverse_interior_edges**	Input	If true, the traversal algorithm now pass es through holes in the body; if false, it now only passes over the outside surface of the body.
const logical	**delete_openings**	Input	If true, any holes in the resulting sheet are now closed up, if false they are now left.

UF_WAVE_set_link_update_time (view source)

Defined in: uf_wave.h

Overview

Given a linked feature, sets when it is to update. Pass in NULL_TAG as timestamp_feature to specify that the linked feature update after its source has updated completely. Pass in a feature tag as timestamp_feature to specify the feature after which the linked feature is to update. For those linked features whose type has no specified update behavior (sketches, strings, datums and points), this routine can be called, but has no effect. For solid, face, region, mirror and edge-based curve linked features, the timestamp feature must be

part of the same solid as the source geometry, and must have a timestamp later than the feature that created that geometry. This routine cannot be called for extractions (linked features linked within a single part), nor for linked features whose source is unavailable. If the linked feature is later reparented, its timestamp feature is overridden.

Environment
Internal and External

See Also
UF_WAVE_ask_link_update_time

Required License(s)
solid_modeling

```
int UF_WAVE_set_link_update_time
(
    const tag_t linked_feature,
    const tag_t timestamp_feature
)
```

const tag_t	**linked_feature**	Input	The linked feature whose timestamp feature is to be set.
const tag_t	**timestamp_feature**	Input	The new timestamp feature of the linked feature.

UF_WAVE_set_linked_pt_angle (view source)

Defined in: uf_wave.h

Overview
Sets a linked point feature to be based on an angle on an arc and an xform. This reparents it, if the arc is different. The arc can be in a different part from the previous parent. The arc's part is fully loaded if it is not already fully loaded. The angle is given (in radians) by a smart scalar, which must be in the same part as the linked point feature. xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked point feature whose to_part_occ contains arc. If arc is NULL_TAG, the link to the previous parent is broken (and xform and angle must also be NULL_TAG). This will also accept the link as broken. Note that any linked point can be reparented with this routine, even if its previous parent was not an angle on an arc.

Environment
Internal and External

See Also
UF_WAVE_create_linked_pt_angle

Required License(s)
solid_modeling

int UF_WAVE_set_linked_pt_angle
(
const tag_t linked_feature,
const tag_t arc,
const tag_t angle,
const tag_t xform
)

const tag_t	**linked_feature**	Input	The linked point feature whose properties are to be set.
const tag_t	**arc**	Input	The arc which defines the new source geometry of the linked point feature.
const tag_t	**angle**	Input	The angle (smart scalar) which defines the position of the linked point feature around arc. It must be in the same part as linked_feature. Its value is in radians.
const tag_t	**xform**	Input	The new transform which determines the position of the linked point feature relative to its source geometry. Must be NULL_TAG (identity transform) or an assy_ctxt_xform in the same part as linked_feature whose to_part_occ contains arc.

UF_WAVE_set_linked_pt_center (view source)

Defined in: uf_wave.h

Overview

Sets a linked point feature to be based on the center of a conic and an xform. This reparents it if the conic is different. The conic can be in a different part from the previous parent. The conic's part will be fully loaded if it is not already fully loaded. xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked point feature whose to_part_occ contains conic. If conic is NULL_TAG, the link to the previous parent is broken (and xform must also be NULL_TAG). This also accepts the link as broken. Note that any linked point can be reparented with this routine, even if its previous parent was not the center of a conic.

Environment

Internal and External

See Also

UF_WAVE_create_linked_pt_center

Required License(s)

solid_modeling

int UF_WAVE_set_linked_pt_center
(
const tag_t linked_feature,
const tag_t conic,
const tag_t xform

)

const tag_t	**linked_feature**	Input	The linked point feature whose properties are to be set.
const tag_t	**conic**	Input	The conic whose center is the new source geometry of the linked point feature.
const tag_t	**xform**	Input	The new transform which determines the position of the linked point feature relative to its source geometry. Must be NULL_TAG or an assy_ctxt_xform in the same part as linked_feature whose to_part_occ contains conic.

UF_WAVE_set_linked_pt_curve (view source)

Defined in: uf_wave.h

Overview

Sets a linked point feature to be based on a parameterized point on a curve and an xform. This reparents it if the curve is different. The curve can be in a different part from the previous parent. The curve's part will be fully loaded if it is not already fully loaded. The parameter is given by a smart scalar, which must be in the same part as the linked point feature. xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked point feature whose to_part_occ contains curve. If curve is NULL_TAG, the link to the previous parent is broken (and xform and t must also be NULL_TAG). This also accepts the link as broken. Note that any linked point can be reparented with this routine, even if its previous parent was not a point on a curve.

Environment

Internal and External

See Also

UF_WAVE_create_linked_pt_curve

Required License(s)

solid_modeling

```
int UF_WAVE_set_linked_pt_curve
(
    const tag_t linked_feature,
    const tag_t curve,
    const tag_t prm,
    const tag_t xform
)
```

const tag_t	**linked_feature**	Input	The linked point feature whose properties are to be set.
const tag_t	**curve**	Input	The curve which defines the new source geometry of the linked point feature.

const tag_t	**prm**	Input	The parameter (smar scalar) which defines the position of the linked point feature along curve. It must be in the same part as linked_feature.
const tag_t	**xform**	Input	The new transform which determines the position of the linked point feature relative to its source geometry. Must be NULL_TAG or an assy_ctxt_xform in the same part as linked_feature whose to_part_occ contains curve.

UF_WAVE_set_linked_pt_point (view source)

Defined in: uf_wave.h

Overview

Sets a linked point feature to be based on an existing point and an xform. This reparents it if the point is different. The existing point can be in a different part from the previous parent. The existing point's part is fully loaded if it is not already fully loaded. xform can be set to NULL_TAG or any assy_ctxt_xform in the same part as the linked point feature. If point is NULL_TAG, the link to the previous parent is broken (and xform must also be NULL_TAG). This also accepts the link as broken. Note that any linked point can be reparented with this routine, even if its previous parent was not an existing point.

Environment

Internal and External

See Also

UF_WAVE_create_linked_pt_point

Required License(s)

solid_modeling

```
int UF_WAVE_set_linked_pt_point
(
    const tag_t linked_feature,
    const tag_t point,
    const tag_t xform
)
```

const tag_t	**linked_feature**	Input	The linked point feature whose properties are to be set.
const tag_t	**point**	Input	The point which is the new source geometry of the linked point feature; can be NULL_TAG, in which case the link is broken.
const tag_t	**xform**	Input	The new transform which determines the position of the linked point feature relative to its source geometry. Must be NULL_TAG or an assy_ctxt_xform in the

			same part as linked_feature whose to_part_occ contains point.

UF_WAVE_set_session_delay (view source)

Defined in: uf_wave.h

Overview

Assigns the delay status to the current NX session. Setting session delay to true means that interpart objects do not update in the current session. Setting session delay to false updates any non-frozen, out of date parts in the session. This may require loading of partially loaded out of date parts.

Environment

Internal and External

Required License(s)

gateway

```
int UF_WAVE_set_session_delay
(
    logical delayed
)
```

logical	**delayed**	Input	True if you want the session to be delayed false if you want it to be undelayed.

UF_WAVE_unfreeze (view source)

Defined in: uf_wave.h

Overview

Unfreezes a set of frozen parts. If parts in the array are not frozen, they are not changed. This function works on both frozen and persistently frozen parts.

Environment

Internal and External

Required License(s)

wave

```
int UF_WAVE_unfreeze
(
    int n_parts,
    tag_t * parts
)
```

int	**n_parts**	Input	Number of parts to unfreeze.

tag_t *	**parts**	Input	Array of tags of parts to unfreeze.

UF_WAVE_update_parts (view source)

Defined in: uf_wave.h

Overview

Updates a set of out of date parts. This updates frozen parts. If a part in the array is partially loaded (and it's out of date) it is fully loaded in order to allow for the update.

In order to run, this routine requires the existence of at least one WAVE license. The function does not use a license, but requires that one is installed. If a WAVE license is not installed, this function returns an error.

Environment

Internal and External

Required License(s)

wave

```
int UF_WAVE_update_parts
(
    int n_parts,
    tag_t * parts
)
```

int	**n_parts**	Input	Number of parts to update.
tag_t *	**parts**	Input	Array of tags of parts to update.

UF_WAVE_update_session (view source)

Defined in: uf_wave.h

Overview

Updates non-frozen, out of date parts in the session. If an out of date part is partially loaded, it is fully loaded and updated.

Environment

Internal and External

Required License(s)

gateway

```
int UF_WAVE_update_session
(
    void
)
```