Chapter 3.
Functions

Topic: Ignore

The function interface is a set of Application Procedural Interface (API) and Direct Interface (DI) functions that an application can invoke to interact with ACIS. API functions, which combine modeler functionality with application support features such as argument error checking and roll back, are the main interface between applications and ACIS. The DI functions provide access to modeler functionality, but do not provide the additional application support features, and, unlike APIs, are not guaranteed to remain consistent from release to release. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

api_check_ct

Function: Cellular Topology

Action: Check the sanity of cellular topology.

Prototype:
```
outcome api_check_ct (
    const ENTITY* ent,       // entity to check
    insanity_list*& list,    // list of insane
                             // entities and info
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "intersct/sg_husk/sanity/insanity_list.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API traverses and checks the cells attached to the given entity. If any problems are found, they are added to the insanity_list.

Errors: The pointer to an ENTITY is NULL.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: System routine

api_ct_add_to_group

Function: Cellular Topology

Action: Adds one entity to an existing group.

Prototype:
```
outcome api_ct_add_to_group (
    ENTITY* ent,             // entity to add
    SPAGROUP* group,         // group to get the
                             // entity
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "ct_husk/classes/group.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: If the given entity does not already belong to the group, this API adds it to the group's list and attach an ATTRIB_SPAGROUP on the entity that points to the group.

Errors: The pointer to an entity is NULL.
The pointer to a group is NULL or does not point to a SPAGROUP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_attach

Function: Cellular Topology

Action: Attaches cellular topology to each lump within each body in the input list.

Prototype:
```
outcome api_ct_attach (
    ENTITY_LIST& body_list, // list of bodies
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/lists/lists.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API attaches cellular topology to each LUMP within each BODY in the input list.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_attach_cells

Function: Cellular Topology

Action: Attaches cell data to a lump.

Prototype:
```
outcome api_ct_attach_cells (
    LUMP* lump,             // lump to accept cell
                            // data
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/lump.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API computes and attaches cell data using ATTRIB_CELL to a lump that does not have the data. If the lump already has the cell data attribute, perform a cell update computation instead.

Errors: The pointer to a lump is NULL or does not point to a LUMP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_cell_area

Function: Cellular Topology, Physical Properties

Action: Computes the area of a cell.

Prototype:
```
outcome api_ct_cell_area (
    CELL* cell,                  // cell to be
                                 // examined
    double req_rel_accy,         // requested relative
                                 // accuracy
    double& area,                // returned cell area
                                 // computed
    double&                      // returned estimate
        est_rel_accy_achieved,   // of relative
                                 // accuracy achieved
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "ct_husk/classes/cell.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: The area of a CELL2D comprised both sides of all its faces. The area of a CELL3D does not include any faces used twice. This API computes only the cell's external area.

Errors: The pointer to a cell is NULL or does not point to a CELL.
Negative accuracy is requested.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read-only

api_ct_cell_mass_pr

Function: Cellular Topology, Physical Properties

Action: Computes mass properties of a CELL3D.

Prototype:

```
outcome api_ct_cell_mass_pr (
    CELL3D* cell,                 // cell to be
                                  // examined
    SPAposition const&            // projection
        root_proj_pl,             // plane root
    SPAunit_vector const&         // projection
        normal_proj_pl,           // plane normal
    int selector,                 // properties to
                                  // compute
    double req_rel_accy,          // requested relative
                                  // accuracy
    double& volume,               // returned volume of
                                  // cell
    SPAposition& cofg,            // returned center of
                                  // gravity
    tensor& inertia,              // returned tensor
                                  // (3 X 3) matrix
    double p_moments[],           // returned moments
                                  // of inertia
    SPAunit_vector* p_axes[],       // returned axes
of
                                  // inertia
    double&                       // returned estimate
        est_rel_accy_achieved,    // of relative
                                  // accuracy achieved
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”ct_husk/api/ctapi.hxx”
#include ”ct_husk/classes/cell3d.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kernutil/tensor/tensor.hxx”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API determines the volume, center of gravity and moments of inertia of a 3D cell.

The major diagonal of the inertia tensor returned contains values for the inertias about each axis. For the inertia about the *x*-axis (integral)((y*y + z*z)dV); the off-diagonal terms are for the products of inertia, e.g., ((integral)(x*y)dV).

Specify the projection plane by a point and normal (in global body space). For speed, choose the plane so that as many faces as possible project on to it as lines; i.e., are edge-on to it. To improve the accuracy of the result, set the plane to pass through a point within the cell; generally, the mid-point of its box.

The selector controls the properties computed. A value of zero calculates all properties; 1 negates the calculation of inertias; 2 negates the calculation of inertias, first moments, or center of gravity (only the volume is found).

Set the requested accuracy to the desired relative accuracy. For example, setting the requested accuracy to 01 requests an accuracy of 1 percent.

Although computation of mass properties does not make substantive changes to a model, boxes may be found and set into the model, creating a bulletin board. To make the process Read-only for the user, call api_ct_note_state before the call to api_ct_cell_mass_pr to roll the model back to its state before api_ct_cell_mass_pr was called, then call:

```
DELTA_STATE* ds = NULL;
api_note_state(ds);
api_change_state(ds);
api_delete_ds(ds);
```

Errors: Pointer to cell NULL or not to a CELL3D.
Zero length normal vector specified.
Negative accuracy requested.
Invalid value specified for selector.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_copy_cell

Function: Cellular Topology

Action: Copies a cell as a body.

Prototype:
```
outcome api_ct_copy_cell (
    CELL* cell,             // cell to copy as a body
    BODY*& body,            // returned resulting
                            // body
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "ct_husk/classes/cell.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API copies a cell to a single, separate body. The original body is not modified. The new body is a valid body in and of itself. The cell may be either 2D or 3D. Cell data is automatically attached to the copy body.

Errors: Pointer to cell is NULL or not to a CELL.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_expand

Function: Cellular Topology

Action: Expands the cellular topology by grouping cells within each body in the given list into supercells.

Prototype:
```
outcome api_ct_expand (
    ENTITY_LIST& body_list, // list of bodies
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API expands the cellular topology by grouping cells within each body in the given list into supercells. If cellular topology is attached to a body in the input body_list, this API expands the cellular topology by grouping the CELLs into SUPERCELLs. This may increase the speed and efficiency of other cellular topology operations.

This API does not try to expand the cell list into supercells unless 50 or more cells exist.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_expand_cells

Function: Cellular Topology

Action: Transforms the cell lists of lump into a hierarchy of supercells with spatial locality.

Prototype:
```
outcome api_ct_expand_cells (
    LUMP* lump,             // lump to be subdivided
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/lump.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API expands a lump's cells into a hierarchy of supercells with spatial locality. A successful outcome indicates the process ran to completion, even if nothing has been changed.

Errors: Pointer to lump is NULL or not to a LUMP.

Limitations: May have no effect on lumps with few cells.

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_flatten

Function: Cellular Topology

Action: Removes any supercells within each body in the given list.

Prototype:
```
outcome api_ct_flatten (
    ENTITY_LIST& body_list, // list of bodies
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/lists/lists.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API removes any supercells within each body in the given list. If cellular topology is attached to a body in the input body_list, this API checks for and removes any SUPERCELLs.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_flatten_cells

Function: Cellular Topology

Action: Flattens lump cells.

Prototype:
```
outcome api_ct_flatten_cells (
    LUMP* lump,             // lump to be flattened
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/lump.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API removes any supercell structure from the cells of the given lump and places the cells in a single list for the lump. No action if no supercells exist. A successful outcome indicates the process ran to completion, even if nothing has been changed.

Errors: Pointer to lump is NULL or not to a LUMP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_get_all_cells

Function: Cellular Topology

Action: Gets all cells attached to each LUMP within the given list of BODYs and adds them to an entity list.

Prototype:
```
outcome api_ct_get_all_cells(
    ENTITY_LIST const& body_list,   // body list
    ENTITY_LIST& cell_list,         //cell list
    AcisOptions* ao = NULL          // acis options
    );
```

Includes:
```
#include "ct_husk/api/ctapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This extension gets all CELLs attached to each LUMP within the given list of BODYs and adds them to an entity list. This procedure assumes that the elements of the given ENTITY_LIST are of type BODY.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_lose_cells

Function: Cellular Topology

Action: Removes the cell data from a lump.

Prototype:
```
outcome api_ct_lose_cells (
    LUMP* lump,             // lump to be cleaned of
                            // cells
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”ct_husk/api/ctapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/lump.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes all cell data from a lump. There must be an ATTRIB_CELL attached to the lump.

Errors: The pointer to lump is NULL or does not point to a LUMP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_lose_group

Function: Cellular Topology

Action: Removes a group entity and all ATTRIB_SPAGROUPs of those entities attached to the group.

Prototype:
```
outcome api_ct_lose_group (
    SPAGROUP* group,        // group to be removed
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”ct_husk/api/ctapi.hxx”
#include ”ct_husk/classes/group.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes a group entity by removing all references (in the form of an ATTRIB_SPAGROUP) to the group, and finally, the group entity itself.

Errors: The pointer to a group is NULL or does not point to a SPAGROUP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_make_group

Function: Cellular Topology

Action: Creates a group from a list of entities.

Prototype:
```
outcome api_ct_make_group (
    ENTITY_LIST& list,      // list of entities to
                            // group
    SPAGROUP*& group,       // returned resulting
                            // group
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”ct_husk/api/ctapi.hxx”
#include ”ct_husk/classes/group.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API combines a set of entities by creating a group entity that points to each entity in the group. Each entity in the group must point to the group through a new ATTRIB_SPAGROUP.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_point_in_cell

Function: Cellular Topology, Object Relationships

Action: Determines whether a given point lies inside, outside, or on the boundary of a 3D cell.

Prototype:
```
outcome api_ct_point_in_cell (
    SPAposition const& test_point,  // point to be
tested
    CELL3D* target_cell,        // cell to be
                                // examined
    point_containment& pc,      // returned inside,
                                // outside, boundary,
                                // unknown
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "ct_husk/classes/cell3d.hxx"
#include "intersct/kernapi/api/ptcont.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "baseutil/vector/position.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API determines the location of a point (given in global coordinate space) for a given cell. Returns a point_containment value of point_inside, point_outside, point_boundary, or point_unknown.

A call to this API may cause boxes to be computed and hence the model will change and a bulletin board will be made.

If the user wants to make the process Read-only, call api_ct_note_state before the call to api_ct_point_in_cell which rolls the modeler back to the state before api_ct_point_in_cell was called, then call:

```
DELTA_STATE* ds = NULL;
api_note_state(ds);
api_change_state(ds);
api_delete_ds(ds);
```

Errors: Pointer to cell is NULL or not to a CELL3D.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_propagate_cface_attribs

Function: Cellular Topology, Attributes

Action: Copies the cface volume attributes (ATTRIB_CFACE_VOL) on a lump.

Prototype:
```
outcome api_ct_propagate_cface_attribs (
    LUMP* lump,              // lump of interest
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/lump.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API propagates any attributes descended from ATTRIB_CFACE_VOL on the cell data on the lump. Such attributes are intended to represent cell volume data, such as material type, and must be propagated after any change to the model (such as a Boolean) to reflect volume containment and merging.

The algorithm follows:

First, copy each attribute descended from ATTRIB_CFACE_VOL found on a cface to all other cfaces on the CELL3D, unless other cfaces already have that attribute.

Then, if the cface has an attribute copied to it, it is also copied to the partner (opposing) cface, if any.

After propagation, the cell data can be scanned for attribute conflicts, such as two different material types on one cell.

Errors: Pointer to lump is NULL or not to a LUMP.
No cell data attached to lump.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_remove

Function: Cellular Topology

Action: Removes cellular topology from each lump within each body in the input list.

Prototype:
```
outcome api_ct_remove (
    ENTITY_LIST& body_list, // list of bodies
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/lists/lists.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API removes the attached cellular topology from each LUMP within each body in the body_list. If the given BODY has no cellular topology, this API has no effect.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_remove_from_group

Function: Cellular Topology

Action: Removes an entity from a group.

Prototype:
```
outcome api_ct_remove_from_group (
    ENTITY* ent,            // entity to be removed
    SPAGROUP* group,        // group containing
                            // entity
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″ct_husk/classes/group.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API removes the entity from the group’s list of entities,along with removing the ATTRIB_SPAGROUP from the entity. If the entity is not in the group, does nothing.

Errors: Pointer to entity is NULL.
Pointer to group is NULL or not to a SPAGROUP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_ct_return_ents

Function: Cellular Topology

Action: Gets a list of all entities in a group.

Prototype:
```
outcome api_ct_return_ents (
    SPAGROUP* group,         // group to be examined
    ENTITY_LIST& list,       // returned entities in
                             // the group
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "ct_husk/api/ctapi.hxx"
#include "ct_husk/classes/group.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API retrieves a list of all entities in a group.

Errors: Pointer to group is NULL or not to a SPAGROUP.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read-only

api_ct_return_groups

Function: Cellular Topology

Action: Gets a list of all groups in which an entity participates.

Prototype:
```
outcome api_ct_return_groups (
    ENTITY* ent,            // entity of interest
    ENTITY_LIST& list,      // returned groups
                            // containing the entity
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″kernel/kerndata/lists/lists.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API retrieves a list of all groups that an entity is in. The entity will have a separate ATTRIB_SPAGROUP for each SPAGROUP.

Errors: NULL entity.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read-only

api_ct_vacate_cell

Function: Cellular Topology

Action: Modifies a 3D cell from filled to void.

Prototype:
```
outcome api_ct_vacate_cell (
    CELL3D* cell,           // 3D cell to be vacated
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″ct_husk/classes/cell3d.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API modifies the face sides and containment of all the cell’s faces so that the cell is now void. Does not change the cell data, except to invalidate affected cells. The cells are updated during the cell recompute at the end of the api.

Errors: Pointer to cell is NULL or not to a CELL3D.

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Changes model

api_get_edges_from_all_entities

Function: Cellular Topology

Action: Gets all the edges that a given entity contains and lists them in an ENTITY_LIST.

Prototype:
```
outcome api_get_edges_from_all_entities(
    ENTITY* ent,                // entity to query
    ENTITY_LIST& edge_list,     // list of edges
    PAT_NEXT_TYPE include_pat   // how to treat
        = PAT_CAN_CREATE,       // patterned edges
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "ct_husk/api/ctapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernutil/law/pattern_enum.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: By default, patterned objects are included in the list of entities. In general, however, the parameter include_pat determines how this function deals with such objects. The following are valid values for this argument:

PAT_CAN_CREATE patterned objects are created if they do not already exist, and are included in the list

PAT_NO_CREATE only those patterned objects that have already been created are included in the list

PAT_IGNORE no patterned objects besides seed pattern objects are included in the list

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read–only

api_get_faces_from_all_entities

Function: Cellular Topology

Action: Gets all the faces that a given entity contains and lists them in an ENTITY_LIST.

Prototype:
```
outcome api_get_faces_from_all_entities(
    ENTITY* ent,                // entity to query
    ENTITY_LIST& face_list,     // list of faces
    PAT_NEXT_TYPE include_pat   // how to treat
        = PAT_CAN_CREATE,       // patterned faces
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "ct_husk/api/ctapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernutil/law/pattern_enum.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: By default, patterned objects are included in the list of entities. In general, however, the parameter include_pat determines how this function deals with such objects. The following are valid values for this argument:

PAT_CAN_CREATE patterned objects are created if they do not already exist, and are included in the list

PAT_NO_CREATE only those patterned objects that have already been created are included in the list

PAT_IGNORE no patterned objects besides seed pattern objects are included in the list

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read–only

api_get_vertices_from_all_entities

Function: Cellular Topology

Action: Gets all the vertices that a given entity contains and lists them in an ENTITY_LIST.

Prototype:
```
outcome api_get_vertices_from_all_entities(
    ENTITY* ent,                // entity to query
    ENTITY_LIST& vertex_list,   // list of vertices
    PAT_NEXT_TYPE include_pat   // how to treat
        = PAT_CAN_CREATE,       // patterned vertices
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "ct_husk/api/ctapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernutil/law/pattern_enum.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: By default, patterned objects are included in the list of entities. In general, however, the parameter include_pat determines how this function deals with such objects. The following are valid values for this argument:

PAT_CAN_CREATE patterned objects are created if they do not already exist, and are included in the list

PAT_NO_CREATE only those patterned objects that have already been created are included in the list

PAT_IGNORE no patterned objects besides seed pattern objects are included in the list

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: Read–only

api_initialize_cellular_topology

Function: Cellular Topology, Modeler Control

Action: Initializes the cellular topology library.

Prototype:
```
outcome api_initialize_cellular_topology ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: System routine

api_terminate_cellular_topology

Function: Cellular Topology, Modeler Control

Action: Terminates the cellular topology library.

Prototype:
```
outcome api_terminate_cellular_topology ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″ct_husk/api/ctapi.hxx″
#include ″kernel/kernapi/api/api.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/api/ctapi.hxx

Effect: System routine

is_ATTRIB_CELL

Function: Cellular Topology

Action: Determines if an ENTITY is an ATTRIB_CELL.

Prototype:
```
logical is_ATTRIB_CELL (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”ct_husk/classes/at_cell.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/at_cell.hxx

Effect: Read–only

is_ATTRIB_CFACE_VOL

Function: Cellular Topology

Action: Determines if an ENTITY is an ATTRIB_CFACE_VOL.

Prototype:
```
logical is_ATTRIB_CFACE_VOL (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”ct_husk/classes/at_cfv.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/at_cfv.hxx

Effect: Read–only

is_ATTRIB_CT

Function: Cellular Topology

Action: Determines if an ENTITY is an ATTRIB_CT.

Prototype:
```
logical is_ATTRIB_CT (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/at_ct.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/at_ct.hxx

Effect: Read–only

is_ATTRIB_FACECFACE

Function: Cellular Topology

Action: Determines if an ENTITY is an ATTRIB_FACECFACE.

Prototype:
```
logical is_ATTRIB_FACECFACE (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/at_fcf.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/at_fcf.hxx

Effect: Read–only

is_ATTRIB_VOL_COL

Function: Cellular Topology

Action: Determines if an ENTITY is an ATTRIB_VOL_COL.

Prototype:
```
logical is_ATTRIB_VOL_COL (
    const ENTITY* e         // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/at_vcol.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/at_vcol.hxx

Effect: Read–only

is_CELL

Function: Cellular Topology

Action: Determines if an ENTITY is a CELL.

Prototype:
```
logical is_CELL (
    const ENTITY* e         // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/cell.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/cell.hxx

Effect: Read–only

is_CELL2D

Function: Cellular Topology

Action: Determines if an ENTITY is a CELL2D.

Prototype:
```
logical is_CELL2D (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/cell2d.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/cell2d.hxx

Effect: Read–only

is_CELL3D

Function: Cellular Topology

Action: Determines if an ENTITY is a CELL3D.

Prototype:
```
logical is_CELL3D (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/cell3d.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/cell3d.hxx

Effect: Read–only

is_CFACE

Function: Cellular Topology

Action: Determines if an ENTITY is a CFACE.

Prototype:
```
logical is_CFACE (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″ct_husk/classes/cface.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/cface.hxx

Effect: Read–only

is_CSHELL

Function: Cellular Topology

Action: Determines if an ENTITY is a CSHELL.

Prototype:
```
logical is_CSHELL (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”ct_husk/classes/cshell.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/cshell.hxx

Effect: Read–only

is_SPAGROUP

Function: Cellular Topology

Action: Determines if an ENTITY is a SPAGROUP.

Prototype:
```
logical is_SPAGROUP (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”ct_husk/classes/group.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: ct_husk

Filename: ct/ct_husk/classes/group.hxx

Effect: Read–only