# *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.

## af_delete_facets

Function: Faceting

Action: Deletes all facets attached to an entity.

Prototype:
```
logical af_delete_facets (
    ENTITY* entity          // given entity
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/attribs/meshat.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″baseutil/logical.h″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/meshat.hxx

Effect: Changes model

# af_delete_mesh

Function: Faceting

Action: Deletes all meshes attached to an entity.

Prototype:
```
void af_delete_mesh (
    ENTITY* entity          // given entity
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/attribs/meshat.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Deletes all meshes attached to an entity.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/meshat.hxx

Effect: Changes model

# api_create_refinement

Function: Faceting

Action: Creates a refinement.

Prototype:
```
outcome api_create_refinement (
    REFINEMENT*& ref,       // new refinement
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API creates a refinement. The REFINEMENT is initialized with the following values:

flatness . . . . . = 0.5
aspect ratio . . . . . = 4
others . . . . . = 0

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_create_vertex_template

Function: Faceting, Model Object

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_create_vertex_template (
    int n_tokens,           // number of tokens
    int tokens[],           // list of tokens
    VERTEX_TEMPLATE*& tplate,// new vertex template
                            // returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/vtplate.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API creates a vertex-template. Valid token types (defined in the nodedata.hxx file) and their values are:

| | |
|---|---|
| POSITION_TOKEN ..... | 3D points (always required) |
| NORMAL_TOKEN ..... | normals |
| COLOR_TOKEN* ..... | vertex-color |
| TRANSPARENCY_TOKEN* ..... | degree of transparency at a vertex |
| TEXTURE_COORDS_TOKEN* ... | *s,t* coordinates |
| UV_TOKEN ..... | surface parametric coordinates |
| UV_DERIVS_TOKEN ..... | surface parametric derivatives |
| UV_CHANGE_TOKEN ..... | magnitudes of *uv* derivatives |

An asterisk, *, means tokens cannot be generated by the faceter and should be added by the application.

Errors: None

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_delete_body_facets

Function: Faceting

Action: Deletes facets from all faces within a body.

Prototype:
```
outcome api_delete_body_facets (
    BODY* body,             // body to delete facets
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Refer to Action.

Errors: Pointer to body is NULL or not to a BODY.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_delete_entity_facets

Function: Faceting

Action: Deletes the facets attached to the entity only and not attached to its descendents.

Prototype:
```
outcome api_delete_entity_facets (
    ENTITY* entity,         // faceted entity
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: It deletes the facets attached to this entity only and does not delete the facets attached to the entities descendents. To do that, use api_delete_body_facets, api_delete_lump_facets, or api_delete_face_facets.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_delete_face_facets

Function: Faceting

Action: Deletes the facets of a face.

Prototype:
```
outcome api_delete_face_facets (
    FACE* face,             // face to delete facets
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/face.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Refer to Action.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_delete_lump_facets

Function: Faceting

Action: Deletes the facets from all the faces within a lump.

Prototype:
```
outcome api_delete_lump_facets (
    LUMP* lump,             // lump to delete facets
    AcisOptions* ao = NULL  // acis options
    );
```

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

Description: Refer to Action.

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

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_delete_shell_facets

Function: Faceting

Action: Deletes all facets from the faces within a shell.

Prototype:
```
outcome api_delete_shell_facets (
    SHELL* shell,           // shell to delete facets
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/shell.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Refer to Action.

Errors: Pointer to shell is NULL or not to a SHELL.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_faceted_face

Function: Faceting

Action: Determines if a face has been faceted or not.

Prototype:
```
outcome api_faceted_face (
    FACE* face,              // face to be examined
    logical& faceted,        // returns TRUE if
                             // faceted
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/face.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: The face is checked to see if a mesh has been attached as an attribute. Thus, if the mesh manager does not store its polygons as meshes on the face, it will always return a negative answer.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_facet_area

Function: Faceting

Action: Returns the area of the facets of entity.

Prototype:
```
outcome api_facet_area (
    ENTITY* entity,          // Entity for which area
                             // needs to be calculated
    double& area,            // Area of the entity
    AcisOptions* ao          // acis options
        = NULL               //
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Gets the area of the facets currently attached to the entity. The entity must be already faceted for this function to be effective. Any parts of the entity not faceted will be ignored.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_facet_body

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting; use api_facet_entity instead.

Prototype:
```
outcome api_facet_body (
    BODY* body,              // body to be faceted
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API facets each face of a body. If the default POLYGON_POINT_MESH mesh manager is used (set as default), the facets are attached to the face they represent. Refer to api_facet_entity for details of how REFINEMENTs are used, and to api_facet_face for details of how VERTEX TEMPLATEs are used.

Errors: Pointer to body is NULL or not to a BODY.

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_entities

Function: Faceting, Viewing

Action: Creates facets for a list of entities.

Prototype:
```
outcome api_facet_entities (
    ENTITY* owner,          // owning entity
    ENTITY_LIST* entity_list,   // entities to facet
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.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 function provides a way to facet more than one entity at a time. The entities in the list should be owned by the owner. The typical use of this API is for incrementally faceting the faces of a body, when a list of modified or new faces are passed to be faceted. It is more efficient and avoids edge consistency problems when AF_GRID_TO_EDGES is used for grid handling in a REFINEMENT.

Errors: Pointer to owner entity is NULL or is not a BODY, a LUMP, or a SHELL. Other entities is NULL or not to a BODY, a LUMP, a SHELL, or a FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_entity

Function: Faceting, Viewing

Action: Creates facets for an entity.

Prototype:
```
outcome api_facet_entity (
    ENTITY* entity,          // entity to facet
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API facets the given entity. The accuracy of the facets are controlled with REFINEMENTs. If the entity is a body, a lump, or a shell, all the contained faces are faceted. Based on the surface type of a face, the refinement is first searched from the face, then the owning shell, then the owning lump, then the owning body, then finally from the defaults of the faceter.

The facets are passed on to the current MESH_MANAGER which appropriately directs the data. For example, it may display the facets, output them to a file, or attach them to a face.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_face

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting; use api_facet_entity instead.

Prototype:
```
outcome api_facet_face (
    FACE* face,             // face to facet
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/face.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: The facets to represent the face are generated according to the REFINEMENT associated with this face. If the POLYGON_POINT_MESH mesh manager is used (set as default), the stored data is determined with a VERTEX TEMPLATE. The VERTEX TEMPLATE most closely associated with the face is used.

If the data is being stored in a POLYGON_POINT_MESH, any current body transformation will be applied to the facets.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_lump

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting; use api_facet_entity instead.

Prototype:
```
outcome api_facet_lump (
    LUMP* lump,             // lump to facet
    AcisOptions* ao = NULL  // acis options
    );
```

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

Description: This API facets each face of a lump. If the default POLYGON_POINT_MESH mesh manager is used (set as default), the facets are attached to the face they represent. Refer to api_facet_entity for details of how REFINEMENTs are used, and to api_facet_face for details of how VERTEX TEMPLATEs are used.

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

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_shell

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting; use api_facet_entity instead.

Prototype:
```
outcome api_facet_shell (
    SHELL* shell,           // shell to facet
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/shell.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API facets each face of a shell. If the default POLYGON_POINT_MESH mesh manager is used (set as default), the facets are attached to the face they represent. Refer to api_facet_entity for details of how REFINEMENTs are used, and to api_facet_face for details of how VERTEX TEMPLATEs are used.

Errors: Pointer to shell is NULL or not to a SHELL.

Limitations: Pre-1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_unfaceted_entities

Function: Faceting, Viewing

Action: Facets unfaceted faces of a list of entities.

Prototype:
```
outcome api_facet_unfaceted_entities (
    ENTITY* owner,          // owning entity
    ENTITY_LIST* entity_list,   // entities to facet
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.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 function facets unfaceted faces of the given entity list to the meet the refinement criteria specified by the user exactly like api_facet_entities. The only difference is that faceted; i.e., marked, faces are ignored. Those marks are attached to each faceted face if the user has called api_mark_faceted_faces previously. If data was attached to each face, faceted faces will keep their attached data, otherwise the user should have kept that data.

Errors: Pointer to owner entity is NULL or is not a BODY, a LUMP, or a SHELL. Other entities is NULL or not to a BODY, a LUMP, a SHELL, or a FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_facet_unfaceted_entity

Function: Faceting, Viewing

Action: Facets unfaceted; i.e., unmarked, faces of an entity.

Prototype:
```
outcome api_facet_unfaceted_entity (
    ENTITY* entity,          // entity to facet
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/data/entity.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API function facets the given entity to the refinement criteria specified by the user, exactly like api_facet_entity. The only difference is that this API facets faces that were not faceted previously; i.e., not marked with a faceted attribute. Marked faces are assumed to be faceted previously and their data is stored somewhere. If data is attached to the face it will remain there.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_fast_find_face

Function: Faceting, Viewing, Picking

Action: Fires a ray through a body and returns the hits.

Prototype:
```
outcome api_fast_find_face (
    SPAposition const& ray_pos,     // ray start
point
    SPAunit_vector const& ray_dir,// ray direction
    BODY* in_body,              // target body
    int& in_count,              // number of hits
    ENTITY**& in_faces          // entities hit
        =*(ENTITY***)NULL_REF,  // default is none
    SPAposition*& in_hits       // positions hit
        =*(SPAposition**)NULL_REF,// default is none
    double*& in_params          // ray parameters of
        =*(double**)NULL_REF,       // the hit
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: ray_pos defines the start point of the ray, and ray_dir defines the direction of the ray.

The number of hits is retuned in in_count. For each hit, the API returns the entity hit in in_faces, the position of the hit in in_hits, and the distance along the ray of the hit in in_params.

The body must be faceted prior to calling this function.

The results of this function are accurate only to within the facet tolerance. If more accurate results are needed api_raytest_body or api_raytest_ents should be used; these two functions may also be faster for purely analytic surfaces.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_body_facets

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_body_facets (
    BODY* body,                  // body to examine
    POLYGON_POINT_MESH*& pmesh, // returns mesh
                                 // collected
    logical share_edge_vertices // returns share
        = TRUE,                  // vertices
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/meshmgr/ppm.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″baseutil/logical.h″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API gets the facets from all faceted faces within a body and return them as a single POLYGON_POINT_MESH. The faceted faces must have the same vertex template.

Errors: Pointer to body is NULL or not to a BODY.
Inconsistent vertex-templates.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_body_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_get_entity_refinement instead.

Prototype:
```
outcome api_get_body_refinement (
    BODY* body,              // body to examine
    REFINEMENT*& ref,        // refinement attached to
                             // body returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Gets the REFINEMENT attached to a body. If None is attached, a NULL value is returned. If none is attached, a NULL value is returned.

Errors: Pointer to body is NULL or not to a BODY.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_body_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_body_vertex_template (
    BODY* body,              // body to examine
    VERTEX_TEMPLATE*& tplate,   // attached vertex
                             // template returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/body.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Gets the VERTEX_TEMPLATE attached to a body. If none is attached, a NULL value is returned.

Errors: Pointer to body is NULL or not to a BODY.

Limitations: Pre–1.7 compatibility only

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_default_refinement

Function: Faceting, Viewing

Action: Gets the default REFINEMENT associated with a type of surface.

Prototype:
```
outcome api_get_default_refinement (
    REFINEMENT*& ref,        // default refinement
    AF_SURF_MODE surftype    // type of surface
        = AF_SURF_ALL,       // returned (optional)
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/af_enum.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: A REFINEMENT is always returned. If a REFINEMENT specific to the surface type is not available, one that is associated with a more general type is returned. The surface type may be one of:

AF_SURF_ALL ..... = all surfaces
AF_SURF_REGULAR ..... = plane, cone, sphere, torus
AF_SURF_IRREGULAR ..... = spline
AF_SURF_PLANE ..... = plane
AF_SURF_CONE ..... = cone
AF_SURF_SPHERE ..... = sphere
AF_SURF_TORUS ..... = torus
AF_SURF_SPLINE ..... = spline

If surface type is not passed, it is set to AF_SURF_ALL.

Errors: Pointer to body is NULL or not to a BODY. Surface type is invalid.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_default_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_default_vertex_template (
    VERTEX_TEMPLATE*& tplate,   // default vertex
                                // template returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/vtplate.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Gets the vertex template. If none exists, a NULL value is returned.

Errors: None

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_entity_refinement

Function: Faceting, Viewing

Action: Gets the refinement attached to the entity.

Prototype:
```
outcome api_get_entity_refinement (
    ENTITY* entity,          // entity to examine
    REFINEMENT*& ref,        // refinement attached
    AF_SURF_MODE surftype    // type of surface
        = AF_SURF_ALL,       // returned (optional)
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/af_enum.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: If none with the given type is found, a NULL value is returned. The surface type may be one of:

AF_SURF_ALL ..... = all surfaces
AF_SURF_REGULAR ..... = plane, cone, sphere, torus
AF_SURF_IRREGULAR ..... = spline
AF_SURF_PLANE ..... = plane
AF_SURF_CONE ..... = cone
AF_SURF_SPHERE ..... = sphere
AF_SURF_TORUS ..... = torus
AF_SURF_SPLINE ..... = spline

If surface type is not passed, it is set to AF_SURF_ALL.

Errors: Pointer to entity is NULL or not to an BODY, LUMP, SHELL, or FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_facet_edge_points

Function: Faceting

Action: Gets points off the edge generated by the faceter.

Prototype:
```
outcome api_get_facet_edge_points (
    EDGE* edge,              // edge entity
    SPAposition*& polyline,  // array of positions
    int& num_pts,            // number of elements
                             // in array
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/edge.hxx"
#include "baseutil/vector/position.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This api gets the points off the edge if they are generated by the faceter. If the corresponding faces are not faceted then it returns null polyline.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_face_facets

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_face_facets (
    FACE* face,                  // face to examine
    POLYGON_POINT_MESH*& pmesh, // mesh collected
    logical share_edge_vertices // share edge
        = TRUE,                  // vertices returned
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/meshmgr/ppm.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API makes a copy of the facets from a face and returns them as a POLYGON_POINT_MESH. If the face does not have facets attached, the mesh is empty.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_face_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_get_entity_refinement instead.

Prototype:
```
outcome api_get_face_refinement (
    FACE* face,              // face to be examined
    REFINEMENT*& ref,        // refinement attached to
                             // face returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/attribs/refine.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Gets the REFINEMENT attached to a face. If None is attached, a NULL value is returned.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_face_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_face_vertex_template (
    FACE* face,              // face to examine
    VERTEX_TEMPLATE*& tplate,   // attached vertex
                             // template returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/attribs/vtplate.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Gets the VERTEX_TEMPLATE attached to a face. If None is attached, a NULL value is returned.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_indexed_mesh

Function: Faceting

Action: Gets the indexed mesh attached to the entity.

Prototype:
```
outcome api_get_indexed_mesh (
    ENTITY* entity,          // entity to examine
    INDEXED_MESH*& mesh,     // indexed mesh attached
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/meshmgr/idx_mesh.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: If none with the given type is found, a NULL value is returned.

Errors: Pointer to entity is NULL or not to a BODY, LUMP., SHELL., or FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_lump_facets

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_lump_facets (
    LUMP* lump,                 // lump to examine
    POLYGON_POINT_MESH*& pmesh, // mesh collected
    logical share_edge_vertices // share edge
        = TRUE,                 // vertices returned
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/meshmgr/ppm.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/lump.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API gets the facets from all faceted faces within a lump and returns them as a single POLYGON_POINT_MESH. The faceted faces must have the same vertex template.

Errors: Pointer to lump is NULL or not to a LUMP.
Inconsistent vertex-templates.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_lump_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_get_entity_refinement instead.

Prototype:
```
outcome api_get_lump_refinement (
    LUMP* lump,              // lump to examine
    REFINEMENT*& ref,        // refinement attached to
                             // lump returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/lump.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Gets the REFINEMENT attached to a lump. If None is attached, a NULL value is returned.

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

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_lump_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_lump_vertex_template (
    LUMP* lump,              // lump to examine
    VERTEX_TEMPLATE*& tplate,   // attached vertex
                             // template returned
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/lump.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Gets the VERTEX_TEMPLATE attached to a lump. If None is attached, a NULL value is returned. If None is attached, a NULL value is returned.

Errors: Pointer to lump is NULL or not to a LUMP. ref refers to the vertex template that is attached to the lump.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_mesh_manager

Function: Faceting

Action: Gets the current mesh manager of faceter.

Prototype:
```
outcome api_get_mesh_manager (
    MESH_MANAGER*& mesh_manager,// current mesh
                                // returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/meshmgr/meshmg.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: If the mesh manager has been set to NULL, a NULL value will be returned.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_shell_facets

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_shell_facets (
    SHELL* shell,               // shell to examine
    POLYGON_POINT_MESH*& pmesh, // mesh collected
    logical share_edge_vertices // share edge
        = TRUE,                 // vertices returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/meshmgr/ppm.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/shell.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API gets the facets from all faceted faces within a shell and returns them as a single POLYGON_POINT_MESH. The faceted faces must have the same vertex template.

Errors: Pointer to shell is NULL or not to a SHELL. Inconsistent vertex-templates.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_get_shell_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_get_entity_refinement instead.

Prototype:
```
outcome api_get_shell_refinement (
    SHELL* shell,           // shell to examine
    REFINEMENT*& ref,       // refinement attached to
                            // body returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/shell.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Gets the REFINEMENT attached to a shell. If none is attached, a NULL value is returned.

Errors: Pointer to shell is NULL or not to a SHELL.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_get_shell_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_get_shell_vertex_template (
    SHELL* shell,           // shell to examine
    VERTEX_TEMPLATE*& tplate,   // attached vertex
                            // template returned
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/shell.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Gets the VERTEX_TEMPLATE attached to a shell. If None is attached, a NULL value is returned.

Errors: Pointer to shell is NULL or not to a SHELL.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_initialise_faceter

Function: Faceting, Viewing, Modeler Control

Action: Obsolete: used only in pre-1.7 Faceting, so use api_initialize_faceter instead.

Prototype:
```
outcome api_initialise_faceter (
    MESH_MANAGER* mesh_manager  // mesh manager
        =(MESH_MANAGER*)-1      // (optional)
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/meshmgr/meshmg.hxx″
#include ″kernel/kernapi/api/api.hxx″
```

Description: This API initializes the faceter with some default refinements that produces only triangles and quadrilaterals that are planar. A mesh manager may be specified.

A mesh manager is required to direct the facet data generated in the faceter. If no argument is passed in the call, as it was in pre–1.7 versions, a mesh manager that handles POLYGON_POINT_MESH is created automatically.

Errors: Pointer to mesh manager is not to a MESH_MANAGER or a derived class.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read-only

# api_initialize_faceter

Function: Faceting, Viewing, Modeler Control

Action: Initializes the faceter library.

Prototype:
```
outcome api_initialize_faceter ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”kernel/kernapi/api/api.hxx”
```

Description: This API initializes the faceter with some default refinements that produces only triangles and quadrilaterals that are planar. A mesh manager is required to direct the facet data generated in the faceter. A mesh manager that handles POLYGON_POINT_MESH is created automatically.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: System routine

# api_mark_faceted_faces

Function: Faceting

Action: Marks faceted faces so that application may want to skip re-faceting them later.

Prototype:
```
outcome api_mark_faceted_faces (
    logical mark,           // flag for marking
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″baseutil/logical.h″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: When this API is called with a TRUE argument all consequent faceting operations will mark faceted faces with a mark showing that the face is already faceted. That is done by attaching an attribute to the faceted face. That attribute is lost when the face is engaged in merging, splitting or transformation operation.

A call to this function affects all following faceting operations. To ignore marking after initiating it, the same API function must be called with FALSE argument.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_modify_refinement

Function: Faceting, Viewing

Action: Obsolete: used only in pre-1.7 Faceting, so use REFINEMENT::modify_refinement method instead.

Prototype:

```
outcome api_modify_refinement (
    REFINEMENT* ref,          // refinement to modify
    double flatness           // flatness
        = 0.0,
    double aspect_ratio       // aspect ratio of grid
        = 0.0,                // cell
    double surtol             // surface deviation
        = 0.0,
    double nortol             // normal deviation
        = 15.0,
    double siltol             // sil tolerance
        = 0.0,
    double pixtol             // pixel tolerance
        = 0.0,
    int min_level             // min grid subdivision
        = 0,
    int max_level             // max grid subdivision
        = 0,
    int mode = 0,             // polygon mode
    AcisOptions* ao = NULL    // acis options
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API is provided purely for backward compatibility. The recommended way to change a REFINEMENT is through its methods. In the above example, the parameters with view-dependent faceting are not supplied, because they are ignored in this faceter.

The polygon modes are retrofitted as follows:

| *Pre–1.7 Definition* | *Setting to Emulate* |
|---|---|
| 0 = use surface default . . . . . | AF_GRID_INTERIOR, AF_TRIANG_FRINGE_2 |
| 1 = use 3–sided polygons . . . . . | AF_GRID_INTERIOR, AF_TRIANG_ALL |
| 2 = use 4–sided polygons . . . . . | AF_GRID_INTERIOR, AF_TRIANG_FRINGE_1 |
| 3 = use n–sided polygons . . . . . | AF_GRID_TO_EDGES, AF_TRIANG_NONE |

Mode 3 can lead to nonplanar polygons on any nonplanar faces. Modes 0 and 2 can lead to nonplanar polygons on spline surfaces.

Errors: Pointer to refinement is NULL or not to a REFINEMENT.
Invalid values specified for parameters.

Limitations: Pre–1.7 compatibility only

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_modify_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_modify_vertex_template (
    int n_tokens,           // number of tokens in
                            // the array
    int* tokens[],          // array of tokens
    VERTEX_TEMPLATE* tplate,// vertex template
                            // modified
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/vtplate.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Modifies a vertex template. The valid token types are:

POSITION_TOKEN (0) . . . . . = 3D points (always required)
NORMAL_TOKEN (1) . . . . . = normals
COLOR_TOKEN (2*) . . . . . = vertex color

Errors: Pointer to vertex template NULL or not to a VERTEX_TEMPLATE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_body_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_set_entity_refinement instead.

Prototype:
```
outcome api_set_body_refinement (
    BODY* body,                  // body to attach
                                 // refinement
    REFINEMENT* ref              // refinement to
        =(REFINEMENT*)NULL,      // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                 // to descendents
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”faceter/api/af_api.hxx”
#include ”faceter/attribs/refine.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API attaches a REFINEMENT to a body and (if apply_to_descend is TRUE) all of its lumps, shells, and faces. If the REFINEMENT is NULL, all the REFINEMENTs are removed from the body and (if apply_to_descend is TRUE) all of its lumps, shells, and faces.

Errors: Pointer to body is NULL or not to a BODY.
Pointer to refinement is not to a REFINEMENT.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_body_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_set_body_vertex_template (
    BODY* body,                 // body to apply
                                // vertex template
    VERTEX_TEMPLATE* tplate,    // vertex template to
                                // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                // to descendents
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/vtplate.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API attaches a VERTEX_TEMPLATE to a body and (if apply_to_descendents is TRUE) all of its lumps, shells and faces.

Refer to api_set_face_vertex_template for a complete description of vertex-templates.

If the VERTEX_TEMPLATE is NULL, all the VERTEX_TEMPLATEs are removed from the body and (if apply_to_descend is TRUE) all of its lumps, shells, and faces.

Errors: Pointer to body is NULL or not to a BODY.
Pointer to vertex template is not to a VERTEX_TEMPLATE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_default_refinement

Function: Faceting, Viewing

Action: Sets the default REFINEMENT in the faceter.

Prototype:
```
outcome api_set_default_refinement (
    REFINEMENT* ref,        // default refinement to
                            // set
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: If the refinement is NULL, it will default to the refinement created by api_initialise_faceter.

Errors: Pointer to refinement is not to a REFINEMENT.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_default_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_set_default_vertex_template (
    VERTEX_TEMPLATE* tplate, // default vertex
                             // template
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Sets the default vertex template in the faceter. If the vertex template is NULL, it will use the default settings in the faceter.

Errors: Pointer to vertex template is not to a VERTEX_TEMPLATE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_entity_refinement

Function: Faceting, Viewing

Action: Attaches a REFINEMENT to an entity (BODY, LUMP, SHELL, FACE).

Prototype:
```
outcome api_set_entity_refinement (
    ENTITY* entity,              // entity to attach
                                 // refinement
    REFINEMENT* ref              // refinement to
        =(REFINEMENT*)NULL,      // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                 // to descendents
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API attaches a REFINEMENT to an ENTITY and (if apply_to_descend is TRUE) all of its descendents (possibly lumps, shells, and faces). If the REFINEMENT is NULL, all the REFINEMENTs are removed from the entity and (if apply_to_descend is TRUE) all of its descendents.

Errors: Pointer to entity is NULL or not to an BODY, LUMP, SHELL, or FACE.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_face_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_set_entity_refinement instead.

Prototype:
```
outcome api_set_face_refinement (
    FACE* face,             // face to attach
                            // refinement
    REFINEMENT* ref         // refinement
        =(REFINEMENT*)NULL, // attached
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/face.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Attaches a REFINEMENT to a face. If the REFINEMENT is NULL, all the REFINEMENTs are removed from the face.

Errors: Pointer to face is NULL or not to a FACE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_face_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_set_face_vertex_template (
    FACE* face,             // face to attach vertex
                            // template
    VERTEX_TEMPLATE* tplate,// vertex template to
                            // attach
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/face.hxx″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: Attaches a VERTEX_TEMPLATE to a face. If the VERTEX_TEMPLATE is NULL, any VERTEX_TEMPLATE on the face is removed.

Errors: Pointer to face is NULL or not to a FACE.
Pointer to vertex template is not to a VERTEX_TEMPLATE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_lump_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_set_entity_refinement instead.

Prototype:
```
outcome api_set_lump_refinement (
    LUMP* lump,                 // lump to attach
                                // refinement
    REFINEMENT* ref             // refinement to
        =(REFINEMENT*)NULL,     // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                // to descendents
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/refine.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/lump.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API attaches a REFINEMENT to a lump and (if apply_to_descend is TRUE) all of its shells and faces. If the REFINEMENT is NULL, all the REFINEMENTs are removed from the lump and (if apply_to_descend is TRUE) all of its shells and faces.

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

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_lump_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_set_lump_vertex_template (
    LUMP* lump,                  // lump to attach
                                 // vertex template
    VERTEX_TEMPLATE* tplate,     // vertex template to
                                 // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                 // to descendents
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/attribs/vtplate.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/lump.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API attaches a VERTEX_TEMPLATE to a lump and (if apply_to_descend is TRUE) all of its shells and faces. If the VERTEX_TEMPLATE is NULL, any VERTEX_TEMPLATE are removed from the lump and (if apply_to_descend is TRUE) all of its shells and faces.

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

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_mesh_manager

Function: Faceting

Action: Sets the current mesh manager of faceter.

Prototype:
```
outcome api_set_mesh_manager (
    MESH_MANAGER* mesh_manager, // mesh manager
                                // to set
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "faceter/api/af_api.hxx"
#include "faceter/meshmgr/meshmg.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Set the current mesh manager in the faceter. A mesh manager is required to direct the facet data generated in the faceter. If it is set to NULL, no output will be provided. See the MESH_MANAGER class and examples of derived class.

Errors: Pointer to mesh manager is not to a MESH_MANAGER or a derived class.

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Read–only

# api_set_shell_refinement

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting, so use api_set_entity_refinement instead.

Prototype:
```
outcome api_set_shell_refinement (
    SHELL* shell,               // shell to attach
                                // refinement
    REFINEMENT* ref             // refinement to
        =(REFINEMENT*)NULL,     // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                // to descendents
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/refine.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/shell.hxx″
#include ″baseutil/logical.h″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API attaches a REFINEMENT to a shell and (if apply_to_descend is TRUE) all of its faces. If the REFINEMENT is NULL, all the REFINEMENTs are removed from the body and (if apply_to_descend is TRUE) all of its faces.

Errors: Pointer to shell is NULL or not to a SHELL.
Pointer to refinement is not to a REFINEMENT.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_set_shell_vertex_template

Function: Faceting

Action: Obsolete: used only in pre-1.7 Faceting.

Prototype:
```
outcome api_set_shell_vertex_template (
    SHELL* shell,                // shell to attach
                                 // vertex template
    VERTEX_TEMPLATE* tplate,     // vertex template to
                                 // attach
    logical apply_to_descendents// TRUE also applies
        = FALSE,                 // to descendents
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kernapi/api/api.hxx″
#include ″kernel/kerndata/top/shell.hxx″
#include ″baseutil/logical.h″
#include ″kernel/kernapi/api/acis_options.hxx″
```

Description: This API attaches a VERTEX_TEMPLATE to a shell and (if apply_to_descend is TRUE) all of its faces. If the VERTEX_TEMPLATE is NULL, any VERTEX_TEMPLATE are removed from the shell and (if apply_to_descend is TRUE) all of its faces.

Errors: Pointer to shell is NULL or not to a SHELL.
Pointer to vertex template is not to a VERTEX_TEMPLATE.

Limitations: Pre–1.7 compatibility only.

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: Changes model

# api_terminate_faceter

Function: Faceting, Viewing, Modeler Control

Action: Terminates the faceter library.

Prototype:
```
outcome api_terminate_faceter ();
```

Includes:
```
#include ″kernel/acis.hxx″
#include ″faceter/api/af_api.hxx″
#include ″kernel/kernapi/api/api.hxx″
```

Description: This API terminates the faceter and free its internal data. Further calls made to the faceter may cause unpredictable problems.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/api/af_api.hxx

Effect: System Routine

# is_ATTRIB_EYE

Function: Faceting

Action: Determines if an ENTITY is an ATTRIB_EYE.

Prototype:
```
int is_ATTRIB_EYE (
    const ENTITY* e          // entity to test
    );
```

```
logical is_ATTRIB_EYE (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”faceter/attribs/atteye3d.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/atteye3d.hxx

Effect: Read–only

# is_ATTRIB_EYE_ATTACHED_MESH

Function: Faceting

Action: Determines if an ENTITY is an ATTRIB_EYE_ATTACHED_MESH .

Prototype:
```
int is_ATTRIB_EYE_ATTACHED_MESH (
    const ENTITY* e          // entity to test
    );

logical is_ATTRIB_EYE_ATTACHED_MESH (
    const ENTITY* e          // entity to test
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”faceter/attribs/meshat.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/meshat.hxx

Effect: Read–only

# is_REFINEMENT

Function: Faceting

Action: Determines if an ENTITY is a REFINEMENT.

Prototype:

```
int is_REFINEMENT (
    const ENTITY* e          // entity to test
    );

logical is_REFINEMENT (
    const ENTITY* e          // entity to test
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/logical.h"
#include "faceter/attribs/refine.hxx"
#include "kernel/kerndata/data/entity.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/refine.hxx

Effect: Read–only

# is_VERTEX_TEMPLATE

Function: Faceting

Action: Determines if an ENTITY is a VERTEX_TEMPLATE.

Prototype:

```
int is_VERTEX_TEMPLATE (
    const ENTITY* e          // entity to test
    );

logical is_VERTEX_TEMPLATE (
    const ENTITY* e          // entity to test
    );
```

Includes:

```
#include ″kernel/acis.hxx″
#include ″baseutil/logical.h″
#include ″faceter/attribs/vtplate.hxx″
#include ″kernel/kerndata/data/entity.hxx″
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: faceter

Filename: fct/faceter/attribs/vtplate.hxx

Effect: Read–only