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_bool_make_intersection_graph

Function: Booleans

Action: Computes all the steps to return the intersection graph between two bodies. Do not remove the attributes attached to the entities.

Prototype:

```
outcome api_bool_make_intersection_graph (
    BODY* tool,                      // slicing body
    BODY* blank,                     // body to be
                                     // sliced
    BODY*& graph,                    // returned graph
    BOOL_TYPE type                   // Boolean
        = UNION,                     // operation (for
                                     // glue only)
    const glue_options* glue_opts    // glue info
        = NULL,                      // and options
    AcisOptions* ao                  // ACIS options
        = NULL                       // like version
                                     // or journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”boolean/kernbool/boolean/boolean.hxx”
#include ”boolean/kernbool/boolean/glue_opts.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
```

Description: Returns the intersection graph of stage one of the boolean. Keeps all the intersection attributes.

Errors: Pointer to tool or blank body NULL or not to a BODY.

Limitations: Not applicable

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_boolean

Function: Booleans

Action: Executes a general Boolean operation.

Prototype:

```
outcome api_boolean(
    BODY* tool,              // first body
    BODY* blank,             // second body
    BOOL_TYPE op,            // type of Boolean
    NDBOOL_KEEP ndbool_keep  // (optional) flag
        =  NDBOOL_KEEP_NEITHER, // for
                             // non-destructive
                             // Booleans
    BODY*& result_body       // (optional) resulting
        =*(BODY**)NULL_REF,  // body for non-
                             // destructive Booleans
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”boolean/kernbool/boolean/boolean.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description:	The two bodies are combined. If the API is successful, it returns the blank body, it deletes the tool body, and it returns a successful outcome. If the BOOL_TYPE is INTERSECTION or NONREG_INTERSECTION and the bodies do not overlap, the intersection is performed, returning an empty blank body. BOOL_TYPE can be UNION, INTERSECTION, SUBTRACTION, NONREG_UNION, NONREG_INTERSECTION, or NONREG_SUBTRACTION. NDBOOL_KEEP can be NDBOOL_KEEP_BLANK, NDBOOL_KEEP_TOOL, NDBOOL_KEEP_BOTH, or NDBOOL_KEEP_NEITHER.
Errors:	The pointer to a tool or blank body is NULL or does not point to a BODY.
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_boolean_chop_body

Function: Booleans

Action: Executes Boolean intersect and subtract operations on two bodies.

Prototype:

```
outcome api_boolean_chop_body (
    BODY* tool,                 // consumed by the
                                // operation
    BODY* blank,                // reused to return
                                // intersection of tool
                                // with blank
    logical nonreg,             // TRUE when
                                // nonregularized results
                                // are required
    BODY*& outside,             // created to return
                                // subtraction of tool
                                // from blank
    BODY*& leftovers            // (optional) returns any
    =*(BODY**) NULL_REF,        // unclassified lumps
                                // from the blank, or
                                // NULL if none
    NDBOOL_KEEP ndbool_keep     // (optional) flag for
        = NDBOOL_KEEP_NEITHER,  // non-destructive
                                // Booleans
    BODY*& result_body          // (optional) resulting
        =*(BODY**)NULL_REF,     // body,necessary for
                                // nondestructive
                                // Booleans
    AcisOptions* ao = NULL      // ACIS options such as
                                // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/logical.h"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "boolean/kernbool/boolean/boolean.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Chops the blank with the tool, returning the body formed by subtracting the tool from the blank, and the body formed by intersecting the tool with the blank, simultaneously.

The logical nonreg argument controls whether nonregularized Boolean results are required.

If the tool body is an incomplete solid, any lumps of the blank which are not intersected by the faces of the tool, and which therefore cannot be classified as either *inside* or *outside*, will be returned in leftovers, if supplied. If leftovers is not supplied, any unclassified lumps will be deleted. The operation will fail if the tool body does not extend far enough to cut completely through any lump of the blank body with which its faces do intersect.

NDBOOL_KEEP can be NDBOOL_KEEP_BLANK, NDBOOL_KEEP_TOOL, NDBOOL_KEEP_BOTH, or NDBOOL_KEEP_NEITHER.

Errors:	The pointer to a tool or blank body is NULL or does not point to a BODY.
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_boolean_chop_complete

Function: Booleans

Action: Completes the last steps Boolean intersect and subtract operations on two bodies.

Prototype:

```
outcome api_boolean_chop_complete (
    logical nonreg,          // TRUE when
                             // nonregularized results
                             // are required
    BODY*& outside,          // created to return
                             // subtraction of tool
                             // from blank
    BODY*& leftovers         // (optional) returns any
        =*(BODY**)NULL_REF,  // unclassified lumps
                             // from the blank, or
                             // NULL if none
    NDBOOL_KEEP ndbool_keep  // (optional) flag for
        = NDBOOL_KEEP_NEITHER,// non-destructive
                             // Booleans
    BODY*& result_body       // (optional) resulting
        =*(BODY**)NULL_REF,  // body, necessary for
                             // non-destructive
                             // Booleans
    AcisOptions* ao          // ACIS options such as
        = NULL               // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/logical.h"
#include "boolean/kernapi/api/boolapi.hxx"
#include "boolean/kernbool/boolean/boolean.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
```

Description: Completes the steps 2, 3 and 4 of the chop operation.

The logical nonreg argument controls whether nonregularized Boolean results are required.

If the tool body is an incomplete solid, any lumps of the blank which are not intersected by the faces of the tool, and which therefore cannot be classified as either inside or outside, will be returned in leftovers, if supplied. If leftovers is not supplied, any unclassified lumps will be deleted. The operation will fail if the tool body does not extend far enough to cut completely through any lump of the blank body with which its faces do intersect.

NDBOOL_KEEP can be NDBOOL_KEEP_BLANK, NDBOOL_KEEP_TOOL, NDBOOL_KEEP_BOTH, or NDBOOL_KEEP_NEITHER.

Errors: The pointer to a tool or blank body is NULL or does not point to a BODY.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_boolean_complete

Function: Booleans

Action: Finishes a Boolean operation.

Prototype:

```
outcome api_boolean_complete (
    BOOL_TYPE op,                  // type of Boolean
                                   // operation
    NDBOOL_KEEP ndbool_keep        // destructive or
        = NDBOOL_KEEP_NEITHER,     // non-destructive
                                   // booleans
    BODY*& res                     // the body to be
        =*(BODY**) NULL_REF, // returned
    AcisOptions* ao = NULL         // ACIS options such
                                   // as version and
                                   // journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "boolean/kernbool/boolean/boolean.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
#include "kernel/kerndata/top/body.hxx"
```

Description:	This API completes stages two, three and four of the Boolean operation. The type of operation is specified to enable both regularized and nonregularized unites, subtracts, and intersects to use the information in the current intersection graph. BOOL_TYPE can be UNION, INTERSECTION, SUBTRACTION, NONREG_UNION, NONREG_INTERSECTION, or NONREG_SUBTRACTION. NDBOOL_KEEP can be NDBOOL_KEEP_NEITHER, NDBOOL_KEEP_BOTH, NDBOOL_KEEP_BLANK, or NDBOOL_KEEP_TOOL.
Errors:	None
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_boolean_glue

Function: Booleans

Action: Executes a specialized Boolean operation, where the intersection graph is known to lie along a set of coincident faces.

Prototype:

```
outcome api_boolean_glue (
    BODY* tool,              // 1st argument body
                             // to be discarded
    BODY* blank,             // 2nd argument body
                             // returns result
    BOOL_TYPE op,            // UNION or SUBTRACTION
    const glue_options*      // glue info
        glue_opts,           // and options
    NDBOOL_KEEP ndbool_keep  // (optional) enum for
        = NDBOOL_KEEP_NEITHER,   // non-destructive
                             // Booleans
    BODY*& result_body       // (optional) resulting
        =*(BODY**)NULL_REF,  // body - necessary for
                             // non-destructive
                             // Booleans
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”boolean/kernbool/boolean/boolean.hxx”
#include ”boolean/kernbool/boolean/glue_opts.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Performs a Boolean unite or subtract operation on two bodies which do not penetrate each other; i.e., the intersection of the bodies lies precisely along a set of (overlapping) coincident faces.

Two faces are coincident if the intersection of their interior point sets is non-empty and bounded by the edges of either face, and on this overlap their surface geometries are coincident.

The glue operation will perform only those face–face intersections deemed necessary by the lists of pairwise coincident faces, faces1 and faces2, of body1 and body2 respectively. There will be no verification that each pair of faces is indeed coincident and it is therefore essential that these lists are accurate and complete.

See documentation on glue_options for information on how to set up information and options for glue. The options consist of flags which can be set to improve performance. It is important that the information provided is accurate, as the glue operation will rely heavily on this information.

Errors: The pointer to a tool or blank body is NULL or does not point to a BODY.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_boolean_start

Function: Booleans

Action: Starts a Boolean operation.

Prototype:
```
outcome api_boolean_start (
    BODY* tool,              // first body
    BODY* blank,             // second body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API performs the first stage of a Boolean operation, initializing the operation to the point where face/face intersections are performed to construct an intersection graph.

Errors: NULL pointer to tool or blank body given.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_check_entity_ff_ints

Function: Object Relationships, Booleans, Debugging

Action: Checks all faces for improper intersections.

Prototype:

```
outcome api_check_entity_ff_ints (
    const ENTITY* given_entity, // entity to check
    ENTITY_LIST* insane_ents,   // error list
    logical& bad_ints,          // TRUE if
                                // errors found
    FILE* file_ptr              // output file
        = NULL,                 //
    insanity_list*& list        // insanity list
        =*(insanity_list**)NULL_REF, //
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kerndata/data/entity.hxx"
#include "intersct/sg_husk/sanity/insanity_list.hxx"
#include "baseutil/logical.h"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Intersects all pairs of faces contained in or by given_entity, looking for intersections between faces that do not belong (i.e., faces that are not adjacent but still intersect).

Also tests valid shells and lumps for improper containments. Containment tests are only performed when insane_ents is non-NULL. A shell is valid if it contains no bad faces (i.e., does not contain and is not contained by any entity in insane_ents) and does not contain intersecting faces. Similarly, lumps are valid when they contain no bad faces, intersection faces, or shells with improper containment. Two shells in the same lump have bad containment if either does not contain the other. Two lumps have bad containment when one contains the other.

If errors are found, ERROR_ENTITYs are added to the list insane_ents, if non-NULL, and bad_ints is set to TRUE. Further, any entities in the list insane_ents are considered bad and are removed from consideration along with any entities containing these bad entities.

Errors: Pointer to given_entity is NULL or not an ENTITY.

Improper face/face intersection:
    CHECK_BAD_FF_INT
Improper face/face coincidence:
    CHECK_BAD_FF_COIN
Improper shell/shell containment:
    CHECK_BAD_SHELL_CONT
Improper lump/lump containment:
    CHECK_BAD_LUMP_CONT
Boolean between a face pair failed (signals unspecified problems with the faces):
    CHECK_FAILED_FF_INT

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_check_list_ff_ints

Function: Booleans, Debugging, Object Relationships

Action: Checks all faces for improper intersections.

Prototype:

```
outcome api_check_list_ff_ints (
    int num_faces1,            // number of faces in
                               // list
    FACE* face_list1[],        // face list to check
    ENTITY_LIST* insane_ents,    // error list
    logical& bad_ints,         // TRUE if errors found
    FILE* file_ptr             // output file
        = NULL,
    int num_faces2             // number of faces
        = 0,                   // in second list
    FACE* face_list2[]         // second face list
        = NULL,                //
    AcisOptions* ao = NULL     // ACIS options such as
                               // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”baseutil/logical.h”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description:

Intersects all pairs of faces, where one face is from face_list1 and the second face is from face_list2, looking for intersections between faces that do not belong (i.e., faces that are not adjacent but still intersect). When face_list2 is not supplied, all faces in the body are used for the second list.

Also tests valid shells and lumps for improper containments. Containment tests are only performed when insane_ents is non-NULL. Note, a shell is valid if it contains no bad faces (i.e., does not contain and is not contained by any entity in insane_ents) and does not contain intersecting faces. Similarly, lumps are valid when they contain no bad faces, intersection faces, or shells with improper containment. Two shells in the same lump have bad containment if either does not contain the other. Two lumps have bad containment when one contains the other.

If errors are found, ERROR_ENTITYs are added to the list insane_ents, if non-NULL, and bad_ints is set to TRUE. Further, any entities in the list insane_ents are considered bad and are removed from consideration along with any entities containing these bad entities.

Errors: Pointer to given_entity is NULL or not an ENTITY.
Improper face/face intersection: CHECK_BAD_FF_INT
Improper face/face coincidence: CHECK_BAD_FF_COIN
Improper shell/shell containment: CHECK_BAD_SHELL_CONT
Improper lump/lump containment: CHECK_BAD_LUMP_CONT
Boolean between a face pair failed (signals unspecified problems with the faces): CHECK_FAILED_FF_INT

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_clean_body

Function: Model Topology

Action: Removes all edges (faces and associated data) that are not necessary to support the topology of the body.

Prototype:

```
outcome api_clean_body (
    BODY* body,              // body to be cleaned
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes all unnecessary edges (faces and associated data) and vertices from the entity. An edge is not needed if the surface defining the the two faces of the edge are the same geometrically. For other entity types no action takes place.

Errors: A NULL pointer to an entity is given.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_clean_entity

Function: Model Topology

Action: Removes all edges and associated data that are not needed to support the topology of the entity.

Prototype:
```
outcome api_clean_entity (
    ENTITY* ent,             // entity whose
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes all unnecessary edges (faces and associated data) and vertices from the entity. An edge is not needed if the surface defining the the two faces of the edge are the same geometrically. This API handles only edges and vertices associated with a BODY, LUMP, SHELL, FACE, EDGE, or VERTEX. For other entity types no action takes place.

Errors: NULL pointer to entity given.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_clean_wire

Function: Booleans, Attributes

Action: Removes the attributes and extra coedges present on a wire body generated by the section or slice operation.

Prototype:
```
outcome api_clean_wire (
    BODY* wire,              // wire body to clean
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: The API deletes the partner coedges and the attributes for all coedges on all wires in the wire body. The result is a wire body that is suitable for any wireframe operation (cover, etc.).

Errors: The pointer to a wire is NULL or does not point to a wire body.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_complete_intersection_graph

Function: Booleans

Action: Determines the intersection graph between two bodies. Do not remove the attributes attached to the entities.

Prototype:

```
outcome api_complete_intersection_graph (
    BODY* tbody,            // slicing body
    BODY* blank,            // body to be sliced
    BODY*& graph,           // the intersection graph
    AcisOptions* ao         // ACIS options such as
        = NULL              // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Performs the last step of bool1 to return the intersection graph. Keeps all the intersection attributes.

Errors: Pointer to tool or blank body NULL or not to a BODY.

Limitations: Not applicable

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_convert_to_spline

Function: Spline Interface

Action: Converts an entity from analytic to spline.

Prototype:

```
outcome api_convert_to_spline (
    ENTITY* given_entity,    // entity to be converted
                             // to a spline
    ENTITY*& return_entity,  // resulting spline
                             // entity
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API converts an entity with analytical faces to splines through the following stages:

First, it creates a copy of the entity.

Second, it splits faces on periodic surfaces along the seams.

Third, it splits edges at poles on surfaces with point singularities.

Fourth, it converts the underlying geometry for the faces using bs3_surface_make_sur, among a number of other functions.

Fifth, it uses set_geometry to add this geometry to the face.

Sixth, it calls sg_add_PCURVEs_to_face to convert loop geometry to pcurves.

And finally, it uses trim face to get rid of unwanted portion of the surface.

Errors: The pointer to an entity is NULL.

Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_detect_short_edges

Function: Model Topology, Tolerant Modeling, Booleans

Action: Detects edges whose lengths are less than the tolerance given and replaces the edges with TVERTEXes.

Prototype:

```
outcome api_detect_short_edges (
    ENTITY* entity,              // Entity with edges
                                 // to be checked
    ENTITY_LIST& returned_list,  // list of short
                                 // edges or TVERTEXes
    const double tolerance       // Maximum length of
        = SPAresfit,             // short edges
    logical replace              // Specifies if
        = FALSE,                 // detected short
                                 // edges should be
                                 // replaced
    AcisOptions* ao = NULL       // acis options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#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”
#include ”boolean/kernapi/api/boolapi.hxx”
```

Description: Detects edges whose lengths are less than the specified tolerance and replaces them with TVERTEXes if the Boolean replace is set TRUE. If replace is FALSE, a list of short edges is returned and no edges are replaced. If replace is TRUE, a list of TVERTEXes is returned.

Errors:	None
Limitations:	None
Library:	boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_detect_sliver_faces

Function: Model Topology, Tolerant Modeling, Booleans

Action: Returns all 2-edge and 3-edge sliver faces from a body whose maximum distance among the edges is smaller than the given tolerance.

Prototype:

```
outcome api_detect_sliver_faces (
    ENTITY* entity,             // Entity with edges
                                // to be checked
    ENTITY_LIST& returned_list, // list of short
                                // edges or tvertices
    const double tolerance      // Maximum length of
        = –1,                   // short edges
    logical replace             // Specifies if
        = FALSE,                // detected sliver
                                // faces should be
                                // replaced
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#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”
#include ”boolean/kernapi/api/boolapi.hxx”
```

Description: Returns all sliver faces from the entity passed in whose maximum distance among the edges are smaller then the specified tolerance. If tolerance is set to –1, the lesser of (10*SPAresfit) and (minimum side of bounding box 1250) is used by default. If the third argument ”replace” is set TRUE, the detected sliver faces will be automatically replaced with tolerant edges.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_fafa_int

Function: Object Relationships, Booleans

Action: Determines the intersection between two faces.

Prototype:
```
outcome api_fafa_int (
    FACE* tool,              // first face
    FACE* blank,             // second face
    BODY*& graph,            // returned graph
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API calculates a proper wire body representing the intersection between the two bodies. (Prior to Release 7.0 this function returned an intersection graph form of a wire body. Beginning with Release 7.0, this function returns a “cleaned” wire body.)

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_fixup_intersection

Function: Booleans

Action: Fix up intersection entities created by api_update_intersection().

Prototype:
```
outcome api_fixup_intersection (
    int edge_knt,            // Index of the intersect
    EDGE** edge_array,       // Array of intersect
    FACE** tfaces,           // Array of tool face
    AcisOptions* ao          // ACIS options such as
        = NULL               // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/edge.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This function has to be used after api_update_intersection() to get rid of one of its side effects. It appears that sg_update_intersection() may reverse the sense of the intersection curve, so the direction of the curve in the ATTRIB_FACEINT may not relate to the direction of the curve under the edge. By using positions obtained from the ATTRIB_FACEINT, we should avoid this problem.

Errors: Pointer to tool or blank body NULL or not to a BODY.

Limitations: Not applicable

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_imprint

Function: Booleans, Stitching

Action: Intersects two bodies and imprints their intersection graph on both without otherwise changing them.

Prototype:
```
outcome api_imprint (
    BODY* tool,              // first body
    BODY* blank,             // second body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

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

Description: This API computes the intersection graph of the tool body and the blank body and imprints the intersection on both bodies. If a closed loop of edges is created, a new face is made. An open loop of edges can be added as a spur to an existing loop on a face or as a slit in the face.

Spline/spline intersection goes directly to the spline package.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_imprint_complete

Function: Booleans

Action: Finishes an imprint operation.

Prototype:

```
outcome api_imprint_complete (
    BODY* tool,              // first body
    BODY* blank,             // second body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: The current intersection graph is imprinted on both bodies. Faces are split.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_imprint_stitch

Function: Booleans, Stitching

Action: Combines bodies along their face-face intersection curves and at coincident vertices.

Prototype:
```
outcome api_imprint_stitch (
    BODY* b1,                // result body
    BODY* b2,                // body to stitch
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

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

Description: Face normals and coedge senses must be compatible for attempted stitching of sheet edges. If not, the stitching of those edges will be ignored.

When vertices at the same location (within tolerance) are merged, they become nonmanifold and contain all surrounding face groups.

Unlike api_stitch, bodies that do not intersect or touch are grouped into one body.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_imprint_stitch_complete

Function: Booleans, Stitching

Action: Imprints bodies and then stitches them along the face-face intersection curves.

Prototype:
```
outcome api_imprint_stitch_complete (
    BODY* b1,                // first body
    BODY* b2,                // second body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

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

Description: This routine imprints the bodies and then stitches them along the face/face intersection curves (imprint edges). It gives the same result as imprinting the two bodies and then stitching them but is faster because the imprint edges are saved and used directly, instead of detecting compatible edges anew in stitch.

Face normals and coedge senses need not be compatible for attempted stitching of open solid-bounding shells along edges.

Unlike api_stitch, bodies that do not intersect or touch are grouped into one body anyway.

Errors: Body 1 and body 2 are the same.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_initialize_booleans

Function: Booleans, Modeler Control

Action: Initializes the Boolean library.

Prototype:

```
outcome api_initialize_booleans ();
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_intersect

Function: Booleans

Action: Executes a Boolean intersect operation on two bodies.

Prototype:
```
outcome api_intersect (
    BODY* tool,              // first body
    BODY* blank,             // second body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

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

Description: This API intersects two bodies. If the outcome is successful, the result is the blank body, and the tool body is deleted. The intersection is performed even if the bodies don’t overlap. In this case, a NULL body is returned.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_join_edges

Function: Model Topology

Action: Joins a list of edges into one single edge.

Prototype:
```
outcome api_join_edges (
    ENTITY_LIST& edge_list,  // edges to be joined
    EDGE*& resulting_edge,   // resulting edge
    logical join_c1          // whether or not to
    = TRUE,                  // join as C1
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kerndata/top/edge.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API joins all edges in the supplied entity list into one single ACIS edge (the resulting edge). If join_c1 is TRUE, the resulting edge is C1 continuous at the joint(s). The resulting edge is one of the input edges. This API can be used to join together edges that do not necessarily share the same geometry.

Errors: If the edges supplied are not end to end, then no merging takes place. If the edges supplied are not G1 continuous at their common vertices, the API will not merge the edges. If there are more than two edges that meet at the interior vertices of the edge list then this API will not merge.

Limitations: Does not handle cases where the edges are branched at the “interior” vertices.

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_merge_faces

Function: Model Topology

Action: Removes all faces of a specified geometry type if they are not necessary to define the body.

Prototype:

```
outcome api_merge_faces (
    BODY* tool,              // body containing faces
                             // to merge
    int& geom_type,          // type of face to merge
                             // PLANE_TYPE,CONE_TYPE,
                             // etc.
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes all unnecessary faces of the specified geometry type from the body. Also removes unnecessary edges and vertices. If the second argument is a NULL reference, then the surface geometry type is not checked and surfaces of all types are processed.

Merges only manifold edges (edges with two attached faces).

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_planar_slice

Function: Booleans

Action: Slices a BODY with a plane.

Prototype:

```
outcome api_planar_slice (
    BODY* ent,                  // body to slice
    const SPAposition& pt,      // position on plane
    const SPAunit_vector& normal,   // plane normal
                                // vector
    BODY*& slice,               // returned wire body
                                // from slice
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.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: This API creates a wire body corresponding to the intersection of the specified body with the plane defined by the specified position and normal vector.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_refresh_entity_pattern

Function: Patterns

Action: Refreshes the elements of a pattern to incorporate changes made to one of them.

Prototype:
```
outcome api_refresh_entity_pattern (
    ENTITY* in_ent,               // seed entity
    ENTITY_LIST& refresh_list,    // pattern entities
                                  // to refresh
    pattern* in_pat,              // pattern to apply
    logical copy_pat              // copy the pattern
        = TRUE,                   // and apply
                                  // the copy instead
                                  // of in_pat
    int seed_index                // zero-based
        = 0,                      // pattern index
    ENTITY_LIST& no_cross_faces // list of faces
        =*(ENTITY_LIST*)NULL_REF,   // that bound seed
    PAT_CHECK_TYPE check          // checking option
        = PAT_DONT_CHECK,         // Default value
    AcisOptions* ao = NULL    // ACIS options such as
                              // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernapi/api/ref_pat.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kernutil/law/pattern.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
#include ”kernel/kernutil/law/pattern_enum.hxx”
```

Description:	This function refreshes the elements of the pattern in_pat to to incorporate changes made to one of them. The entity in_ent should be taken from the modified element, while refresh_list should contain all pattern entities to be refreshed. It should not include anything from the modified element. By default, a copy of the pattern is made, and it is the copy that is actually applied to the entity. This behavior can be overridden by setting copy_pat to FALSE. However, when copying is overridden and in_pat is shared by multiple bodies, a transform placed upon the bodies will be transferred to the pattern multiple times, which is clearly undesirable. Also by default, in_ent is associated with the first pattern element (index 0), but may be associated with another element by furnishing the associated zero–based seed_index.

For cases in which the pattern is applied to a "bump" on a substrate rather than to an autonomous entity, the limits of the bump are automatically computed, but the user may choose to override the default limits by furnishing a list of no_cross_faces.

For performance reasons, the function does not check the generated pattern of entities for intersection, containment, or compatibility unless the user changes the checking option check from its default value. This argument takes the following values:
PAT_DONT_CHECK —– do no checking
PAT_CHECK_DONT_FIX —– check the patterned entities and roll back in the case of failure
PAT_CHECK_AND_FIX —– check the patterned entities. If only the containment check fails, drop the problem elements from the pattern and re–apply it to the seed. If either the intersection or compatibility check fails, roll back.

Errors:	An entity type not supporting patterns is specified, or (if check is TRUE) the pattern has problems with intersection, containment, or compatibility.
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/ref_pat.hxx
Effect:	Changes model

api_regularise_entity

Function: Booleans, Model Topology

Action: Removes all faces, edges and vertices (and associated data) that are not necessary to support the topology of the entity.

Prototype:

```
outcome api_regularise_entity (
    ENTITY* ent,             // whose associated faces
                             // edges and vertices are
                             // to be removed
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: All unnecessary faces, edges and vertices (and associated data) are removed from the entity. A face is *unnecessary* if it is double-sided. An edge is not needed if the surface defining the the two faces of the edge are the same (geometrically speaking). Then does the same with vertices and edges. Handles faces edges and vertices associated with a BODY, LUMP, SHELL or FACE—for other entity type no action takes place, but the function succeeds.

Errors: NULL pointer to entity given

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_remove_face

Function: Model Topology, Booleans

Action: Removes a face from a body.

Prototype:

```
outcome api_remove_face (
    FACE* given_face,        // face to be removed
                             // from owning body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description:	This API removes a face from its owning body. When edges and vertices are no longer needed to support faces, they are removed. This differs from the API api_uncover_face.
Errors:	Pointer to face is NULL or not to a FACE.
Limitations:	Works only on faces that belong to a body.
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_remove_no_merge_attrib

Function: Booleans, Model Topology

Action: Removes a NO_MERGE_ATTRIB to each edge in the input list of edges.

Prototype:

```
outcome api_remove_no_merge_attrib (
    ENTITY_LIST& list,      // entity
    AcisOptions* ao = NULL  // ACIS options such as
                            // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.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 non–merge attribute (NO_MERGE_ATTRIB) from each of the input EDGES.
Errors:	None
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_remove_wire_edge

Function: Model Topology, Booleans

Action: Removes a wire edge from a body and creates a new wire body from it.

Prototype:
```
outcome api_remove_wire_edge (
    EDGE* given_edge,        // wire edge to be
                             // removed from
                             // owning body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/edge.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes a wire edge from its owning body and creates a new wire consisting of this edge.

Errors: Pointer to edge is NULL or not to a wire edge.

Limitations: Works only on edges that belong to a wire.

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_replace_edge_with_tvertex

Function: Model Topology, Tolerant Modeling, Booleans

Action: Replaces an edge or list of edges with a tolerant vertex.

Prototype:
```
outcome api_replace_edge_with_tvertex (
    ENTITY_LIST& edgeList,      // List of edges to
                                // replace
    ENTITY_LIST& tVertex_list,  // tvertex list
                                // replacing edges
    AcisOptions* ao = NULL      // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
```

Description: Replaces an edge (EDGE) with a tolerant vertex (TVERTEX). When using this function, all edges passed in will be replaced regardless of tolerance. The only case where edges will not be replaced is if one or more of them are closed. To replace and detect short edges less than a specified tolerance, use api_detect_short_edges.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_replace_face_with_tedge

Function: Model Topology, Tolerant Modeling, Booleans

Action: Replaces a 2 or 3-edge face with a tolerant edge.

Prototype:

```
outcome api_replace_face_with_tedge (
    FACE* face,              // FACE to be replaced
    ENTITY_LIST& tedges,     // tedges replacing face
    AcisOptions* ao = NULL   // acis options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
```

Description: This replaces a face having a single loop of two or three edges with a tolerant edge. The face is removed and an edge of the face is converted into a tolerant edge and other subsequent topological manipulations are performed.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_selectively_imprint

Function: Booleans

Action: Imprints a subset of the faces of the tool body with a subset of the faces of the blank body.

Prototype:
```
outcome api_selectively_imprint (
    BODY* tool,                  // tool body
    ENTITY_LIST& tool_faces,     // tool faces to be
                                 // split, NULL
                                 // reference implies
                                 // all faces
    BODY* blank,                 // blank body
    ENTITY_LIST& blank_faces,    // list of blank
                                 // faces to be split
                                 // NULL reference
                                 // implies all faces
    logical split_checking       // check if all edges
        = TRUE,                  // and vertices
                                 // created by the
                                 // imprint contribute
                                 // to face splitting
    ENTITY_LIST& intgraph_edges // returned list of
        =*(ENTITY_LIST*)NULL_REF,    // intersection
                                 // edges. May be a
                                 // NULL reference.
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API function imprints a subset of the faces of the tool body with a subset of the faces of the blank body. If the split_checking argument is TRUE, checking will be performed to assure that all edges and vertices imprinted on the blank body contribute to the splitting of blank body faces. If they do not (e.g., there are dangling edges imprinted on the faces) then an exception will be thrown. If this argument is FALSE, then all edges and vertices of the intersection graph will be imprinted, regardless of their contribution to face splitting. The default for this argument is TRUE.

The function will return a list of the edges of the edges imprinted on the blank body, if requested.

If annotations are turned on, SPLIT_ANNOTATIONs and IMPRINT_ANNOTATIONs will be added to the entities of the blank and tool bodies during the operation.

Errors: NO_INTSCT:
No intersection was found between the specified subset of faces of the tool body and the specified subset of faces of the blank body.

IMPROPER_SPLIT: Checking discovered that improper face splitting had occurred.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_selectively_intersect

Function: Booleans

Action: Intersects an array of faces of one body with an array of faces of another body.

Prototype:

```
outcome api_selectively_intersect (
    const int number_faces, // size of faces arrays
    FACE* tool_faces[],     // array of tool body
                            // faces
    FACE* blank_faces[],    // array of blank body
                            // faces
    AcisOptions* ao = NULL  // ACIS options such as
                            // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This routine is given an array of faces on the tool body which will be intersected with the corresponding face in an array of faces on the blank body. The resulting intersections are appended to the intersection graph.

Function api_boolean_start **must** be called before using this API.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_set_no_merge_attrib

Function: Booleans, Model Topology

Action: Sets a NO_MERGE_ATTRIB to each edge in the input list of edges.

Prototype:

```
outcome api_set_no_merge_attrib (
    ENTITY_LIST& list,       // entities
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API applies the nonmerge attribute (NO_MERGE_ATTRIB) to each of the input edges. The action of this attribute is to mark each edge as nonmergeable. That is, a marked edge will not be merged out of its associated body during any subsequent operations on the body which would result in the edge being remove through a merge.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_slice

Function: Booleans

Action: Determines the intersection graph between two bodies.

Prototype:
```
outcome api_slice (
    BODY* tool,                  // slicing body
    BODY* blank,                 // body to be sliced
    SPAunit_vector const& normal,   // normal about
which
                                 // wire edges at a
                                 // vertex are to be
                                 // ordered.
    BODY*& graph,                // returned
                                 // intersection graph
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API returns a special form of wire object where each edge has two coedges (rather than one as in the case of wires made using api_make_wire or api_make_kwire). A normal vector, given in the special case when the tool body is a plane, causes slice to sequence the resultant wires into non overlapping loops of conventional sense; otherwise the normal should be given as NULL; i.e., as:

```
*(SPAunit_vector*) NULL
```

The intersection graph is not designed to be used as a general wire body. To generate a wire body suitable for subsequent general use, it is necessary to call api_clean_wire after api_slice.

Errors: Pointer to tool or blank body NULL or not to a BODY.
Normal vector given has zero length.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_slice_complete

Function: Booleans

Action: Finishes a slice operation.

Prototype:

```
outcome api_slice_complete (
    BODY* tool,                  // first body
    BODY* blank,                 // second body
    SPAunit_vector const& normal,   // normal about
which
                                 // wire edges at a
                                 // vertex are to be
                                 // ordered.
    BODY*& graph,                // returned
                                 // intersection graph
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: Returns a special form of wire object where each edge has two coedges (rather than one as in the case of wires made using api_make_wire or api_make_kwire). A normal vector, given in the special case when the tool body is a plane, causes slice to sequence the resultant wires into nonoverlapping loops of conventional sense; otherwise the normal should be given as NULL; i.e., as *(SPAunit_vector*) NULL.

Errors: Tool or blank body given NULL pointer

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_slice_of_model

Function: Booleans

Action: Creates a new model by a slice based on a clipped copy of the model.

Prototype:
```
outcome api_slice_of_model (
    const ENTITY_LIST& model,    // given model to be
                                 // clipped
    ENTITY_LIST& clipped_copy,   // clipped model
    SPAposition eye,             // eye position
    SPAvector targ_eye,          // target vector
    double hither,               // perp distance from
                                 // hither plane to
                                 // eye
    double yon,                  // perp distance from
                                 // yon plane to eye
    logical& MADE_COPY,          // flag saying a copy
                                 // was made
    AcisOptions* ao              // ACIS options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/vector.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API takes an entity list, a position (eye), a (nonzero) vector (target eye) as well as two doubles (hither and yon). It copies and then trims those parts of the copied entity list that are outside the hither and yon planes (these are orthogonal to the passed vector and the passed point is at a distance hither along the passed vector from the hither plane and a distance yon along the passed vector from the yon plane. It passes back the trimmed copy. If the model itself does not need to be trimmed; i.e., it already is completely contained within the hither-yon slice, then the logical MADE_COPY (TRUE by default) determines if a copy of the model is to be made (TRUE) or if the pointer to the original model is passed back as the pointer to the copy (FALSE, or make no copy). This is a bit dangerous, as a user could delete the model thinking it is a copy (thus the default TRUE setting to protect the unwary or unknowing). On the other hand, TRUE means a copy is made where none might be needed, thus increasing the memory usage unnecessarily.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_split_edges_at_poles

Function: Booleans

Action: Splits the edges of an entity at the poles.

Prototype:

```
outcome api_split_edges_at_poles (
    ENTITY* blank,           // entity on which edges
                             // are to be split
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: If an edge passes through a pole of a surface, the edge is split at the pole. Poles must lie on the boundary of the face. The input entity should be one of the following: BODY, LUMP, SHELL, or FACE.

Errors:	Pointer to entity is NULL.
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_split_face

Function: Booleans

Action: Splits a face along a specified *u* or *v* isoparameter curve.

Prototype:
```
outcome api_split_face (
    FACE* face,                 // face to be split
    logical split_u,            // split along a constant
                                // u parameter curve
    logical use_percent,        // use percentage vs.
                                // explicit param value
    double p,                   // parameter value or
                                // percentage at which
                                // to split
    AcisOptions* ao = NULL      // ACIS options such as
                                // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.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 function splits a face along a *u* or *v* isoparameter curve. The curve can be specified by an explicit parameter value, which must be within the range of the surface, or by a percentage value, which must be between 0 and 1.

Errors: Pointer to face is NULL or not to a FACE.
Parameter value or percentage is inappropriate.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_split_face_at_g_disc

Function: Booleans

Action: Splits a face along ”u” or ”v” isoparametric lines at G1 or G2 discontinuities.

Prototype:
```
outcome api_split_face_at_g_disc (
    FACE* face,                     //face to be split
    ENTITY_LIST& split_faces,       //resulting faces
    int cont_order                  // G1 or G2
        = 1,                        //
    AcisOptions* ao                 //ACIS options such
as
        = NULL                  // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kerndata/top/face.hxx”
```

Description: This extension splits a face along the ”u” or ”v” isoparametric lines at G1 or G2 discontinuity parameter. The result is the list of new faces. The optional acis_options contains parameters for versioning and journaling.

Errors: Face does not contain discontinuity information.

Limitations: If the supplied face is an independent face (i.e. no body, lump or shell), it will return a list of faces that share edges and do not belong to a shell, lump or body. It will also split at discontinuities in the range of the face.

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_split_periodic_faces

Function: Booleans

Action: Splits all periodic faces (along *u*, *v*, or both) to ensure that they are well formed.

Prototype:
```
outcome api_split_periodic_faces (
    ENTITY* blank,            // entity to be split
    AcisOptions* ao = NULL    // ACIS options such as
                              // version and journal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/data/entity.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This function splits all periodic faces of the given entity along *u* or *v* isoparameter curves. The input entity should be one of the following: BODY, LUMP, SHELL, or FACE.

Because a face on a spline surface requires a prop edge along its seam, it is assumed to be well formed already; therefore, this algorithm does not process faces on spline surfaces. If one wants to split a face on a spline surface, the recommended function is api_split_face.

Periodic face splitting is affected by the new_periodic_splitting option. One may use this option to affect the number of splits and the location of the splits.

Errors: Pointer to entity is NULL.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_stitch

Function: Stitching

Action: Stitches faces along edges and vertices of identical geometry.

Prototype:
```
outcome api_stitch (
    BODY* b1,                 // result body
    BODY* b2,                 // body to stitch
    logical split             // match coincident
        = FALSE,              // edges if TRUE
    AcisOptions* ao = NULL    // ACIS options such as
                              // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.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 joins two face bodies along edges and vertices that are identical. Stitching only operates on faces, not on wires, and only stitches faces to faces. If wires exist in one of the bodies being stitched, but do not participate in the stitch (i.e., they do not coincide with edges in the other body), they will transfer to the resulting body.

The argument b1, the first input body, is returned as the resulting body. The second body, b2, is deleted unless it is the same as b1. For example, a body might need internal stitching. One can stitch b1 to b1, and b1 is not deleted.

When creating two-manifold edges on single sided faces, stitching merges geometry on coedges that have opposite edge sense and identical edge geometry (within tolerance). api_stitch will fail if coedges of incompatible orientation (i.e. same edge sense) are encountered. If the faces are double sided, the coedges need not be of opposite sense.

If the split argument is FALSE, the edges must be identical along their entire length. If split is TRUE, the API splits edges in order to match coincident coedges. Coincident edges on single-sided faces and of incompatible orientation (opposite coedge sense) are not split.

When creating nonmanifold edges, the coedges are sorted and a “union” is performed around the coedge, marking faces that are now BOTH_INSIDE as such. The BOTH_INSIDE containment is then propagated to all faces not connected through a stitched edge.

When vertices at the same location (within tolerance) are merged, they become nonmanifold and they contain all surrounding face groups. If the attempt to make a two-manifold edge stitch fails, the vertices are not merged.

Unlike functions such as api_unite, api_stitch is *not* a Boolean operation. Stitching is simpler than a Boolean because it avoids face-face intersections and the evaluation of lump and shell containments.

Errors: Pointer to first or second body is NULL or not to a BODY.
None of the coedges or vertices contain identical geometry.
Incompatible coedges encountered.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_subtract

Function: Booleans

Action: Executes a Boolean subtract operation.

Prototype:

```
outcome api_subtract (
    BODY* tool,              // body to be subtracted
    BODY* blank,             // body to be subtracted
                             // from
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API subtracts one body from another. If the outcome is successful, the result is the blank body and the tool body is deleted.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_terminate_booleans

Function: Booleans, Modeler Control

Action: Terminates the Boolean library.

Prototype:

```
outcome api_terminate_booleans ();
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: System routine

api_uncover_face

Function: Model Topology

Action: Removes the surface of a face, leaving its edges.

Prototype:

```
outcome api_uncover_face (
    FACE* face,             // face to be uncovered
    AcisOptions* ao = NULL  // ACIS options such as
                            // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API uncovers a face to change a closed set of faces into an open set or diminish an open set. The function may leave a partly covered body or a wire body.

Removes a face together with its loops and coedges; however, if removing a coedge from an edge would leave the edge with no coedge; i.e., the edge is to become a wire edge, the coedge is kept and its owner is changed to the owning shell. Previous pointers of some coedges in remaining faces are altered to maintain a connecting path of coedge pointers that link all coedges at a vertex.

Errors:	Pointer to face is NULL or not to a FACE.
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_unhook_face

Function: Model Topology

Action: Removes a face from a body.

Prototype:

```
outcome api_unhook_face (
    FACE* given_face,            // face to be
                                 // unhooked from body
    BODY*& unhooked_face_body,   // returned new body
                                 // containing face
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kerndata/top/face.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API removes the face from its owning body and puts a copy into a new body. When edges and vertices are no longer needed to support faces, they are also removed. This differs from the API api_uncover_face.

Errors:	Pointer to face is NULL or not to a FACE.
Limitations:	Works only on faces that belong to a body.
Library:	boolean
Filename:	bool/boolean/kernapi/api/boolapi.hxx
Effect:	Changes model

api_unhook_wire_edge

Function: Model Topology

Action: Removes a wire edge from a body, placing wire in returned body.

Prototype:

```
outcome api_unhook_wire_edge (
    EDGE* given_edge,           // edge to unhook
    BODY*& unhooked_wire_body,  // body containing
                                // the unhooked edge
    AcisOptions* ao = NULL    // ACIS options such as
                              // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kerndata/top/edge.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_unite

Function: Booleans

Action: Executes a Boolean unite operation.

Prototype:

```
outcome api_unite (
    BODY* tool,               // first body (deleted)
    BODY* blank,              // second body (returned)
    AcisOptions* ao = NULL    // ACIS options such as
                              // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API unites two bodies. If the outcome is successful, the result is the blank body. The tool body is deleted.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_unite_wires

Function: Booleans

Action: Unites the wires of the tool body with the wires of the blank.

Prototype:

```
outcome api_unite_wires (
    BODY* tool,              // first body (deleted)
    BODY* blank,             // second body (returned)
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API unites the wires of the tool body with the wires of the blank where they meet at common vertices. The result is the blank body. The tool body is deleted.

Returns an error outcome if there is no interference.

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

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_unstitch_nonmani

Function: Stitching, Model Topology

Action: Decomposes an input body along its nonmanifold vertices and edges.

Prototype:

```
outcome api_unstitch_nonmani (
    BODY* input_body,        // body to decompose into
                             // manifold parts
    BODY*& lumps,            // returned body with
                             // each lump,
                             // a manifold lump
    BODY*& sheet,            // returned body with
                             // each lump,
                             // a manifold sheet
    BODY*& lamina,           // returned body with
                             // each lump,
                             // a lamina face
    BODY*& wires,            // returned body
                             // containing wires
                             // from input body
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”boolean/kernapi/api/boolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API decomposes the input body into four bodies, consisting of the input body’s manifold lumps, sheets, lamina, and wires. The input body is destroyed.

Definition of Nonmanifold

A nonmanifold edge has more than two faces around it, and a nonmanifold vertex has elements that can be connected only topologically through that vertex. For example, two cones meeting at their apexes or a vertex of a block with a dangling edge. Though api_manifold_class also reports three or more wire edges at a vertex as nonmanifold, this API does not unstitch them.

Information Returned

All faces and wire edges from the input body are contained somewhere in the four returned bodies. The four bodies returned contain the manifold lumps, the maximal manifold sheets, lamina (doubly covered) faces, and the wires found in the input body.

The first body returned by api_unstitch_nonmani (lumps) contains the manifold lumps from the input body, each with one peripheral shell and any void shells remaining.

The second body (sheet) contains each sheet in a separate lump, with exactly one shell in each lump.

The third body (lamina) contains one lamina face (two back-to-back faces) in each lump.

The last body (wires) has all wires from the input body in its wire pointer, and has no lumps or shells. Each wire is maximal in that it contains all wire edges that are topologically connected (through a coedge next or previous pointer) to the first edge referenced by the wire entity.

Any bodies that would be returned empty, such as no sheets found, are returned as NULL.

All nonmanifold vertices and edges are unstitched so that they are manifold except self-nonmanifolds as described below.

All shared geometry is duplicated.

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

Limitations: This API assumes that the input body is more or less sane other than nonmanifold regions, including dangling sheets or wires. The API does not unstitch shell self-nonmanifold edges, such as a shell with two projecting horns that meet at an edge at the tips. It does unstitch self-nonmanifold vertices.

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model

api_update_intersection

Function: Booleans

Action: Creates a surf_surf_int intersection structure to be used in place of an actual intersection.

Prototype:

```
outcome api_update_intersection (
    FACE* tool_face,         // tool face to get
                             // attribute
    const SPAtransf& ttrans, // tool body transform
    FACE* blank_face,        // blank face
    const SPAtransf& btrans, // blank body transform
    const int number_edges,  // number of edges in
                             // ssi_eges
    EDGE* ssi_edges[],       // surf_surf_int
                             // structure
    logical check_rels       // flag whether to avoid
        = TRUE,              // double checking the
                             // edge face relations
    AcisOptions* ao = NULL   // ACIS options such as
                             // version and journal
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "boolean/kernapi/api/boolapi.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/edge.hxx"
#include "kernel/kerndata/top/face.hxx"
#include "baseutil/logical.h"
#include "baseutil/vector/transf.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This routine is passed a tool face and blank face together with an array of edges which represent the intersection of the two faces. A surf_surf_int structure is built using the geometry of these edges as the intersection of the two surfaces, thereby eliminating the need to intersect the faces themselves. To record this, an ATTRIB_FACEINT is attached to the tool face and no intersection will be performed when these faces are to be intersected.

The final logical argument indicates whether to check if the intersection edges really lie in the faces. This can be relatively expensive so may be avoided by passing FALSE, in which case the edge–face relationships are being guaranteed by the caller.

Errors: NULL pointer to tool or blank face.

Limitations: None

Library: boolean

Filename: bool/boolean/kernapi/api/boolapi.hxx

Effect: Changes model.

is_ATTRIB_EFINT

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_EFINT.

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

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernbool/bool1/at_bool1.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernbool/bool1/at_bool1.hxx

Effect: Read–only

is_ATTRIB_FACEINT

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_FACEINT.

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

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernbool/bool1/at_bool1.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernbool/bool1/at_bool1.hxx
Effect:	Read–only

is_ATTRIB_INTCOED

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_INTCOED.

Prototype:

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

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernbool/boolean/at_bool.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernbool/boolean/at_bool.hxx
Effect:	Read–only

is_ATTRIB_INTEDGE

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_INTEDGE.

Prototype:

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

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernbool/boolean/at_bool.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernbool/boolean/at_bool.hxx

Effect: Read–only

is_ATTRIB_INTGRAPH

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_INTGRAPH.

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

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”boolean/kernbool/boolean/at_bool.hxx”
#include ”kernel/kerndata/data/entity.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: boolean

Filename: bool/boolean/kernbool/boolean/at_bool.hxx

Effect: Read–only

is_ATTRIB_INTVERT

Function: Booleans

Action: Determines if an ENTITY is an ATTRIB_INTVERT.

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

Includes:	`#include ”kernel/acis.hxx”`
	`#include ”baseutil/logical.h”`
	`#include ”boolean/kernbool/boolean/at_bool.hxx”`
	`#include ”kernel/kerndata/data/entity.hxx”`

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	boolean
Filename:	bool/boolean/kernbool/boolean/at_bool.hxx
Effect:	Read–only

is_NO_MERGE_ATTRIB

Function: Booleans

Action:	Determines if an ENTITY is a NO_MERGE_ATTRIB.

Prototype:

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

Includes:	`#include ”kernel/acis.hxx”`
	`#include ”baseutil/logical.h”`
	`#include ”boolean/sg_husk/merge/mer_attr.hxx”`
	`#include ”kernel/kerndata/data/entity.hxx”`

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	boolean
Filename:	bool/boolean/sg_husk/merge/mer_attr.hxx
Effect:	Read–only