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_boolean_tube_body

Function: Booleans

Action: Does a selective Boolean operation on two bodies.

Prototype:

```
outcome api_boolean_tube_body (
    BODY* blank,                  // blank body
    BODY* tube,                   // tube body
    ENTITY_LIST& start_faces,     // start face
    ENTITY_LIST& end_faces,       // end face
    SPAposition* start_pos,       // array of start
                                  // positions
    int number_of_starts,         // number of start
                                  // positions
    SPAposition* end_pos,         // array of end
                                  // positions
    int number_of_ends,           // number of end
                                  // positions
    SPAvector start_dir,          // start direction
                                  // Should be (0,0,0)
                                  // if not cyclic
    tube_options** opts,          // options for tube
    int number_of_options,        // number of options
    ENTITY_LIST& bodies,          // output bodies
    AcisOptions* ao = NULL        // acis options
    );
```

Includes:

```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/vector.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:

The API boolean_tube_body takes two bodies (blank and tube) as input and calls the selective Booleans stage 1 once and stage 2 the number_of_options times. Each time these selective Booleans are called, this function automatically sets up the subgraph from a set of options and start and end conditions.

If you pass multiple options into api_boolean_tube_body, the results will be found many times faster than calling it separately for each option.

Defining the operation (bool_type) using tube_options:

There are 3 types of cells, “only from the blank”, “only from the tool”, and “from both”.

There are 4 possible bool_types, UNITE, LIMIT, INTERSECT, and SUBTRACT:

- A UNITE operation deals with the cells ”only from the tool”, adding the specified ones to the blank. Given a variable, opts, of type tube_options, call for example: opts–>set_bool_type(UNITE);
- A LIMIT operation is the same as UNITE except that all cells from the blank are discarded instead of kept.
- An INTERSECT operation deals with ”cells from both”. Only the specified cells are kept.
- A SUBTRACT operation deals with ”cells from both”. Start with the blank, and subtract the specified cells.

Defining Laws using tube_options:

Laws are used to specify which subset of the possible cells your operation should use. Laws are string expressions consisting of the main variable that loops over all possible cell numbers (x1), some constants (X2, X3, X4, X5, X6, and possibly more), and some operators and conditionals (=, !=, >, or, etc.) For shorthand, you can use x for x1, y for x2, and z for x3.

– x1 (x) = order of this cell, or the independent variable
– x2 (y) = largest order in the component cell x belongs to
– x3 (z) = TRUE if from tool
– x4 = TRUE if from blank
– x5 = TRUE if start cell
– x6 = TRUE if end cell

Example laws:

Imagine a vertical tube cutting through the three legs of an extruded letter ’E’, where the tube also extends above the top leg and below the bottom leg. Let ’E’ be the blank, and the tube be the tool. Specify the top cell to be a start cell, and the bottom cell to be the end cell, using the ordering techniques explained below. There are now 4 possible cells that could be included in the unite operation, numbered from top to bottom 0, 1, 2, and 3.

– x1 iterates over the 4 cells, testing each one to see whether it should be kept. In this case, it takes on numbers 0, 1, 2, and 3.
– x2 is always 3, since 3 is the biggest possible number.
– x3 is always TRUE, since we are doing a unite, and only cells from the tool are considered.
– x4 is always FALSE, since we are doing a unite, and no cells from the blank are considered.
– x5 is TRUE if x1 is 0.
– x6 is TRUE if x1 is 3.
– Thus, to specify all but the last cell, use:
```
law * keep_law = NULL;
api_str_to_law(”x!=x2”,&keep_law)
opts–>set_keep_law( keep_law );
keep_law–>remove();
```
– To specify all start cells, substitute the second line with:
```
api_str_to_law(”x5”,&keep_law)
```
– To keep the second and last cells, substitute the second line with:
```
api_str_to_law(”x=1 or x6”,&keep_law)
```

Defining Cell Numbers, or Ordering:

To determine cell numbers, first pick start cells and end cells. This is done by picking start vertices and end vertices, or start faces and end faces, which are on the cells you want to pick. Start cells are all numbered zero. The shortest paths from start cells to end cells are automatically found. Depending on the operation type, non–relevant cells are discarded, and the rest are given an integral number based on the steps they are away from the start cell. Note that multiple cells could share the same number.

Keep branches:

Cells that are not on any ”shortest path” are considered ”branch cells.” You can either keep them all or discard them all with the keep_branches flag, which defaults to FALSE. To keep branch cells use: opts–>set_keep_branches(TRUE);

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_create_graph_from_cells

Function: Graph Theory

Action: Creates a graph (from cells) used in graph theory.

Prototype:
```
outcome api_create_graph_from_cells (
    ENTITY_LIST& cells,     // cells to use
    generic_graph*& graph,  // graph
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”sbool/kernapi/api/sboolapi.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/lists/lists.hxx”
#include ”kernel/kernutil/law/generic_graph.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: This API creates a graph that can be used for graph theory operations and selective Booleans.

When creating a graph from cells, the list must contain cells from Cellular Topology. The ACIS topology is analyzed to determine which cells are connected. The connections between cells become edges (or dashes) of the graph.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_create_graph_from_edges

Function: Graph Theory

Action: Creates a graph used in graph theory.

Prototype:
```
outcome api_create_graph_from_edges (
    ENTITY_LIST& edges,     // edges to use
    generic_graph*& graph,  // graph
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API creates a graph used for graph theory operations and selective Booleans.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_create_graph_from_faces

Function: Graph Theory

Action: Creates a graph (from edges) used in graph theory.

Prototype:
```
outcome api_create_graph_from_faces (
    ENTITY_LIST& faces,     // faces to use
    generic_graph*& graph,  // graph
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API creates a graph that can be used for graph theory operations and selective Booleans.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_initialize_sbooleans

Function: Booleans, Modeler Control

Action: Initializes the Selective Booleans Component library.

Prototype:
```
outcome api_initialize_sbooleans ();
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: System routine

api_selective_boolean_stage1

Function: Booleans

Action: Creates a graph for the first stage of selective Booleans from a tool body and a blank body.

Prototype:
```
outcome api_selective_boolean_stage1 (
    BODY* blank,            // blank body
    BODY* tool,             // tool body
    generic_graph*& graph,  // cell adjacency graph
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API creates a graph structure (e.g., graph theory) from the blank_body entity and the tool_body entity. Using Cellular Topology, distinctive cells of the blank_body and the tool_body become vertices of the graph. Overlapping portions of the cells become edges in the graph. Once the graph has been created, further graph theory operations can be performed to obtain a more desirable graph. This is then used as input to the second stage of selective Booleans, api_selective_boolean_stage2.

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

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_selective_boolean_stage2

Function: Booleans

Action: Completes the selective Boolean process for the cells selected.

Prototype:
```
outcome api_selective_boolean_stage2 (
    BODY* non_reg_unite_body,   // body to use
    ENTITY_LIST& cells_to_keep, // cells to keep
    AcisOptions* ao = NULL  // acis options
    );

outcome api_selective_boolean_stage2 (
    BODY* non_reg_unite_body,   // body to use
    generic_graph*              // graph
        graph_of_cells_to_keep, // cells to keep
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/lists/lists.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This function modifies the entity based on either the results of graph theory or a Cellular Topology cell list. When a cell list is used, only the cell entities in the list should be kept. When a graph is used, the resulting graph still maps to cells in the body and represents the cells to keep.

The mapping of cells in a graph to entities will not be the same from execution to execution.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_selective_unite

Function: Booleans

Action: Unites two bodies with the given positions.

Prototype:
```
outcome api_selective_unite (
    BODY* tool,             // first body
    BODY* blank,            // second body
    int tnum,               // number of position on
                            // tool to keep, 0 means
                            // keep all
    SPAposition* tpos,      // positions
    int bnum                // number of position on
        = 0,                // blank to keep, 0 means
                            // keep all
    SPAposition* bpos = NULL,// positions
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/body.hxx"
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This selective union operation takes two bodies and two lists of positions. The lists of positions are used to specify the portions to keep after the Boolean operation. If the number of positions is zero, the operation will retain the entire body.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Changes model

api_subgraph_2dcell

Function: Graph Theory

Action: Returns a subgraph containing only the vertices that are 2D cells.

Prototype:
```
outcome api_subgraph_2dcell (
    const generic_graph*        // input
        whole_graph,            // graph
    generic_graph*&             // output graph
        partial_graph,          // with 2d cells
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Read–only

api_subgraph_3dcell

Function: Graph Theory

Action: Returns a subgraph containing only the vertices that are 3D cells.

Prototype:
```
outcome api_subgraph_3dcell (
    const generic_graph*            // input
        whole_graph,                // graph
    generic_graph*& partial_graph,  // output graph
                                    // with 3d cells
    AcisOptions* ao = NULL  // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Read–only

api_subset_graph_with_plane

Function: Graph Theory

Action: Finds the subset of a graph on one side of a plane.

Prototype:
```
outcome api_subset_graph_with_plane (
    const generic_graph*            // input
        whole_graph,                // graph
    const SPAposition& plane_origin,// origin
    const SPAunit_vector&           // plane
        plane_normal,               // normal
    generic_graph*& partial_graph,  // output graph
    AcisOptions* ao = NULL          // acis options
    );
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kernutil/law/generic_graph.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: Finds the subset of a graph on one side of a plane. The graph must first be split at that plane. It is assumed that the graph is made of entities which are either faces or cells.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: Read–only

api_terminate_sbooleans

Function: Booleans, Modeler Control

Action: Terminates the Selective Booleans Component library.

Prototype:
```
outcome api_terminate_sbooleans ();
```

Includes:
```
#include "sbool/kernapi/api/sboolapi.hxx"
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: sbool

Filename: sbool/sbool/kernapi/api/sboolapi.hxx

Effect: System routine