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_edge_helix

Function: Model Geometry

Action: Creates an edge that is a helix.

Prototype:

```
outcome api_edge_helix (
    SPAposition axis_start, // axis start position
    SPAposition axis_end,   // axis end position
    SPAvector start_dir,    // vector from axis_start
                            // to helix start
    double radius,          // radius of helix
    double thread_distance, // distance between
                            // threads along axis
    logical handiness,      // TRUE is right
                            // FALSE is left
    EDGE*& new_edge,        // helix created
    AcisOptions* ao = NULL  // ACIS options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/top/edge.hxx”
#include ”offset/kernapi/api/ofstapi.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/vector.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description:	Creates a helical edge along an axis specified by a start position and an end position. The start_dir and radius arguments specify where the edge starts in relation to the axis_start position. The thread_distance argument specifies how close together the turns of the helix are spaced. The handiness argument specifies either a right–hand or left–hand twist.
Errors:	None
Limitations:	None
Library:	offset
Filename:	ofst/offset/kernapi/api/ofstapi.hxx
Effect:	Changes model

api_initialize_offsetting

Function: Modeler Control, Model Geometry

Action: Initializes the offsetting library.

Prototype:
```
outcome api_initialize_offsetting ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”offset/kernapi/api/ofstapi.hxx”
```

Description:	Refer to Action.
Errors:	None
Limitations:	None
Library:	offset
Filename:	ofst/offset/kernapi/api/ofstapi.hxx
Effect:	System routine

api_offset_face

Function: Model Geometry, Model Object, Creating Entities

Action: Creates a face that is an offset of the given face.

Prototype:
```
outcome api_offset_face (
    FACE* given_face,        // given face
    double offset_distance,  // distance to offset new
                             // face
    FACE*& offset_face,      // returns resulting face
    AcisOptions* ao = NULL   // ACIS options
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "kernel/kernapi/api/api.hxx"
#include "kernel/kerndata/top/face.hxx"
#include "offset/kernapi/api/ofstapi.hxx"
#include "kernel/kernapi/api/acis_options.hxx"
```

Description: This API creates a face whose surface is offset from the given face. The edges of the offset face are offset from the given face.

It is assumed that the user will not do offsets that result in degenerate offsets. Offset distances larger than the smallest radius of curvature (principal) will probably cause degenerate cases.

Since a face can not have a transform, it is up to the programmer to apply the transform from the body to the face after the offset, if needed, or fix the transform on the body before offsetting the face.

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

Limitations: None

Library: offset

Filename: ofst/offset/kernapi/api/ofstapi.hxx

Effect: Changes model

api_offset_planar_wire

Function: Model Geometry, Creating Entities, Offsetting

Action: Modifies a planar wire body or a single WIRE.

Prototype:

```
outcome api_offset_planar_wire (
    BODY* given_wire,       // planar wire to be
                            // offset
    law* offset_law,        // distance to offset
    law* twist_law,         // twist in radians
    const SPAunit_vector&   // resulting
        wire_normal,        // normal vector
    BODY*& offset_wire,     // returns offset wire
    int gap_type            // gap type
        = 2,
    logical trim            // trim method
        = TRUE,
    logical overlap         // overlap method
        = FALSE,
    AcisOptions* ao = NULL  // ACIS options
    );
```

```
outcome api_offset_planar_wire (
    BODY* given_wire,        // planar wire to be
                             // offset
    double offset_distance,  // distance to offset
    const SPAunit_vector&    // resulting
        wire_normal,         // normal vector
    BODY*& offset_wire,      // returns offset wire
    AcisOptions* ao = NULL   // ACIS options
    );
```

```
outcome api_offset_planar_wire (
    WIRE* given_wire,        // planar wire to be
                             // offset
    TRANSFORM const* trans,  // transformation
    double offset_distance,  // given distance
    const SPAunit_vector&    // resulting
        wire_normal,         // normal vector
    BODY*& offset_wire,      // returns offset wire
    AcisOptions* ao = NULL   // ACIS options
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”kernel/kerndata/geom/transfrm.hxx”
#include ”kernel/kerndata/top/body.hxx”
#include ”kernel/kerndata/top/wire.hxx”
#include ”lawutil/law_base.hxx”
#include ”offset/kernapi/api/ofstapi.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/unitvec.hxx”
#include ”kernel/kernapi/api/acis_options.hxx”
```

Description: The wire is offset so that no portion of the offset wire is closer to the original wire than the offset distance. Trimming the offset overlap caused by C0 edges, filling in the gaps caused by the offsets of C0 edges, and trimming areas of interference between offset segments are operations that get carried out.

Of the three overloaded APIs, the one using laws is newer and more powerful. An example of how to use this API can be found in the source code for the wire–body:offset Scheme extension. It also shows how to create the required laws and the meaning of the gap types.

The law version of api_offset_planar_wire can always be used and can obtain the same results as the non–law versions by passing it simple laws. A law can be created using api_str_to_law, which converts a law function string into the appropriate law classes and returns a pointer to the top level class which can be passed into this API.

Law Version

In the law version of api_offset_planar_wire, the wire is offset by the given law and trimmed for self–intersections. If a non–zero twist law is given, then the resulting wire will twist around the given wire and trimming will not be done.

The offset direction is given by the cross product of the wires tangent and the planar normal. If a twist law is given then the twist starts in the offset direction and rotates around the wire by the given radians of the twist law using the right–hand rule. A negative offset twists in the opposite direction.

The function expects a body containing at least one independent wire (not connected to any faces). If the body contains multiple independent wires, each independent wire in the body will be offset and united with the offsets of the others.

To offset a single wire of a multi–wire body, call the version of api_offset_planar_wire which takes a WIRE instead of a body.

The gap type is as follows; 0 = rounded like arcs, 1 = extended like lines, and 2 = natural like curve extensions. Lines are extend with lines and circles are extended with circles.

TRUE trims the self–intersections FALSE does not trim the intersections. Use FALSE only if you know that the result will not self–intersect.

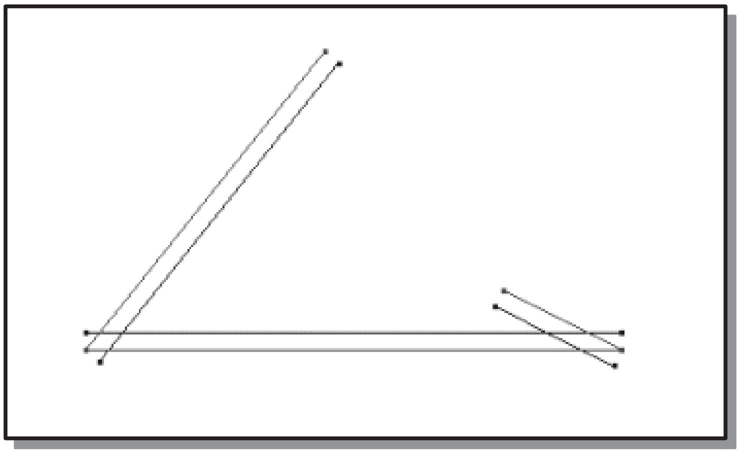
Non–Law Versions

A positive offset distance is in the direction of the cross product (wire_tangent *x* normal). A negative offset is in the opposite direction.

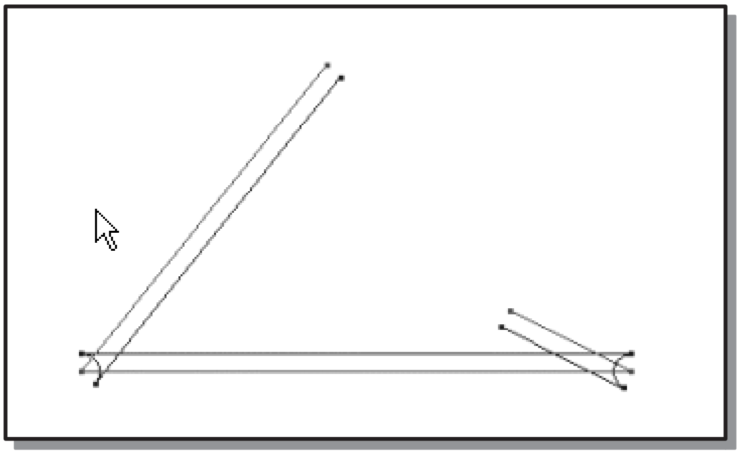
The function expects a body containing a single wire. If the body contains a list of wires, only the first wire in the list will be offset. If all wires must be offset, first split the body into several bodies (using api_separate_body), each containing a single wire, and call api_offset_planar_wire against each body.

Known Case

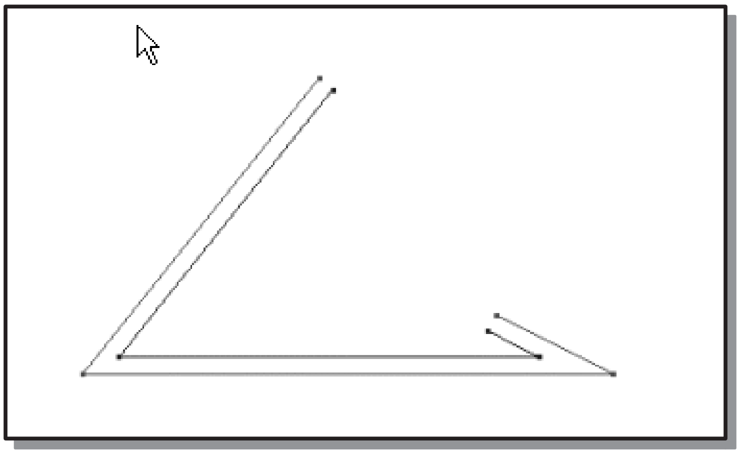
Performing a wire body offset with open wires can encounter behavior that is potentially different from what is expected. Offsetting of closed wires does not have the potential for this issue. When offsetting an open wire body, all of the edges are offset first. In the following picture red indicates the original edges and blue indicates the offset edges.



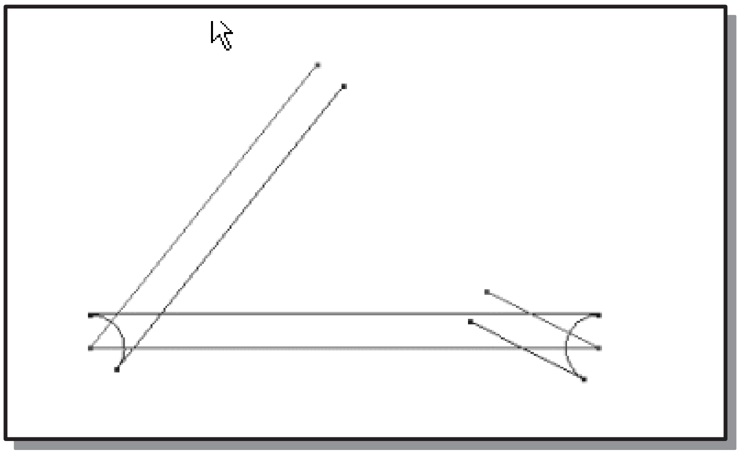
The corners of the wire are then connected with edges creating segments that intersect in the wire.



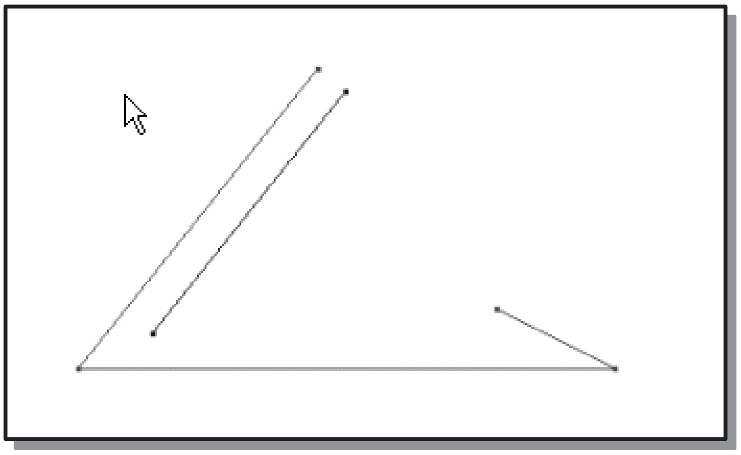
The segments connected to the inserted gap edges are removed leaving the resulting body.



The issue arrives when offsetting a greater amount leaves segments that do not intersect outside of the inserted gap edges.



The subsequent removing of segments connected to the gap edges results in the following body.



Errors: Pointer to wire is NULL or not to a wire body. Zero length normal vector specified.

Limitations: Performing a wire body offset with open wires can encounter behavior that is potentially different from what is expected (refer to the Known Case section in the description).

Library: offset

Filename: ofst/offset/kernapi/api/ofstapi.hxx

Effect: Changes model

api_terminate_offsetting

Function: Modeler Control, Model Geometry

Action: Terminates the offsetting library.

Prototype:
```
outcome api_terminate_offsetting ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”kernel/kernapi/api/api.hxx”
#include ”offset/kernapi/api/ofstapi.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library:	offset
Filename:	ofst/offset/kernapi/api/ofstapi.hxx
Effect:	System routine