# *Chapter 2.*
# 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.

## antiparallel

Function: Mathematics

Action: Determines if two vectors are anti–parallel (within some resolution).

Prototype:

```
logical antiparallel (
    SPAunit_vector const& v1,   // first vector
    SPAunit_vector const& v2,   // second vector
    const double res        // resolution
        = SPAresnor
    );

logical antiparallel (
    SPAunit_vector const& v1,   // first vector
    SPAvector const& v2,    // second vector
    const double res        // resolution
        = SPAresnor
    );

logical antiparallel (
    SPAvector const& v1,    // first vector
    SPAvector const& v2,    // second vector
    const double res        // resolution
        = SPAresnor
    );
```

```
logical antiparallel (
    SPAvector const& v1,     // first vector
    SPAunit_vector const& v2,   // second vector
    const double res         // resolution
        = SPAresnor
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/unitvec.hxx”
#include ”baseutil/vector/vector.hxx”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: Read–only

# are_parallel

Function: Mathematics, Analyzing Models

Action: Determines if two vectors or two unit vectors are parallel within SPAresnor.

Prototype:
```
logical are_parallel (
    const SPAunit_vector& v1,   // first vector
    const SPAunit_vector& v2,   // second vector
    logical same_dir         // TRUE means same
        = FALSE              // direction
    );

logical are_parallel (
    const SPAvector& v1,     // first vector
    const SPAvector& v2,     // second vector
    logical same_dir         // TRUE means same
        = FALSE              // direction
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”baseutil/vector/acistol.hxx”
#include ”baseutil/logical.h”
```

Description: This function is overloaded, and can accept two vectors or two unit vectors as arguments.

***Note*** *These functions are antiquated and they should be replaced in applications by the appropriate* SPAvector *and* SPAunit_vector *methods declared in* vector.hxx. *These methods include* parallel, antiparallel, *and* biparallel *determination tests.*

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# are_perpendicular

Function: Mathematics, Analyzing Models

Action: Determines if two vectors or two unit vectors are perpendicular (within SPAresnor).

Prototype:

```
logical are_perpendicular (
    const SPAunit_vector& v1,   // first vector
    const SPAunit_vector& v2// second vector
    );

logical are_perpendicular (
    const SPAvector& v1,    // first vector
    const SPAvector& v2     // second vector
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/vector/vector.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "baseutil/vector/acistol.hxx"
#include "baseutil/logical.h"
```

Description: This function is overloaded, and can accept two vectors or two unit vectors as arguments.

***Note*** *These functions are antiquated and they should be replaced in applications by the appropriate* SPAvector *and* SPAunit_vector *perpendicular methods declared in* vector.hxx.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# biparallel

Function: Mathematics

Action: Determines if two vectors are bi–parallel (within some resolution).

Prototype:

```
logical biparallel (
    SPAunit_vector const& v1,   // first vector
    SPAunit_vector const& v2,   // second vector
    const double res        // resolution
        = SPAresnor
    );

logical biparallel (
    SPAunit_vector const& v1,   // first vector
    SPAvector const& v2,    // second vector
    const double res        // resolution
        = SPAresnor
    );

logical biparallel (
    SPAvector const& v1,    // first vector
    SPAvector const& v2,    // second vector
    const double res        // resolution
        = SPAresnor
    );

logical biparallel (
    SPAvector const& v1,    // first vector
    SPAunit_vector const& v2,   // second vector
    const double res        // resolution
        = SPAresnor
    );
```

Includes:

```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: Read–only

# collapse_all_free_lists

Function: Memory Management

Action: Returns all unused blocks of memory to the operating system.

Prototype:
```
void collapse_all_free_lists ();
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/mmgr/freelist.hxx"
```

Description: Unused memory blocks can be retained and reused until the application specifically collapses the free lists to return the unused memory to the operating system. For example, the application could keep all unused blocks while a model is read in and worked on, thus speeding execution, and then, when done with that model, collapse the free lists and return all now unused memory to the operating system.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/mmgr/freelist.hxx

Effect: System routine

# coordinate_transf

Function: Construction Geometry, Transforms, Modifying Models

Action: Constructs a coordinate transformation.

Prototype:
```
SPAtransf coordinate_transf (
    SPAposition const& new_origin,  // position
    SPAunit_vector const& new_x_axis,// first unit
                                     // vector
    SPAunit_vector const& new_y_axis// second unit
                                     // vector
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/transf.hxx"
#include "baseutil/vector/unitvec.hxx"
```

Description: Sets the transformation to carry origin to given position, and x–axis and y–axis to the given unit vectors. If the second unit vector is not orthogonal to the first, this method uses a unit vector in the plane of the two given vectors.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/transf.hxx

Effect: System routine

# debug_real

Function: Debugging

Action: Prints to the output file a real sometimes specified with the given title.

Prototype:
```
void debug_real (
    double value,           // real value
    FILE* fp                // file pointer
        = debug_file_ptr
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/debug/debugmsc.hxx"
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/debug/debugmsc.hxx

Effect: System routine

# distance_to_line

Function: Mathematics, Analyzing Models

Action: Determines the distance from a position to a line.

Prototype:
```
double distance_to_line (
    const SPAposition& pt,      // position to test
    const SPAposition& line_pt, // position on line
    const SPAunit_vector& line_dir// direction of
                                  // line
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector_utils.hxx”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector_utils.hxx

Effect: Read–only

# distance_to_plane

Function: Mathematics, Analyzing Models

Action: Determines the distance from a position to a plane.

Prototype:
```
double distance_to_plane (
    const SPAposition& pt,      // position to test
    const SPAposition& plane_pt,// position on plane
    const SPAunit_vector& normal// normal to plane
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector_utils.hxx”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector_utils.hxx

Effect: System routine

# distance_to_point

Function: Mathematics, Analyzing Models

Action: Determines the distance between two points.

Prototype:
```
double distance_to_point (
    const SPAposition& pt1, // first point
    const SPAposition& pt2  // second point
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector_utils.hxx”
#include ”baseutil/vector/position.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector_utils.hxx

Effect: Read–only

# find_err_ident

Function: Debugging, Error Handling

Action: Gets the identifier string for an error number. Returns “UNKNOWN” if the error number is invalid.

Prototype:
```
char const* find_err_ident (
    err_mess_type err_num   // error number
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/errorsys/errorbase.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/errorsys/errorbase.hxx

Effect: Read–only

# find_err_mess

Function: Debugging, Error Handling

Action: Gets the message string for an error number. Returns “unknown error” if the error number is invalid.

Prototype:
```
char const* find_err_mess (
    err_mess_type err_num   // error number
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/errorsys/errorbase.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/errorsys/errorbase.hxx

Effect: Read–only

# find_option

Function: Modeler Control

Action: Gets the given option in the list.

Prototype:
```
option_header* find_option (
    char const* name        // option name
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/option/option.hxx”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/option/option.hxx

Effect: Read–only

# get_major_version

Function: Software Architecture

Action: Returns the major version number of the ACIS executable.

Prototype:
```
int get_major_version ();
```

Includes:
```
#include ”baseutil/version/vers.hxx”
#include ”kernel/acis.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/version/vers.hxx

Effect: Read–only

# get_minor_version

Function: Software Architecture

Action: Returns the minor version number of the ACIS executable.

Prototype:
```
int get_minor_version ();
```

Includes:
```
#include ”baseutil/version/vers.hxx”
#include ”kernel/acis.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/version/vers.hxx

Effect: Read–only

# get_option_list

Function: Modeler Control

Action: Returns pointer to the head of the option list.

Prototype:
```
option_header* get_option_list ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/option/option.hxx”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/option/option.hxx

Effect: Read–only

# get_resabs

Function: Precision and Tolerance

Action: Gets the SPAresabs resolution.

Prototype:
```
double get_resabs ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# get_resfit

Function: Precision and Tolerance

Action: Gets the SPAresfit resolution.

Prototype:
```
double get_resfit ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# get_resmch

Function: Precision and Tolerance

Action: Gets the resmch resolution.

Prototype:
```
double get_resmch ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# get_resnor

Function: Precision and Tolerance

Action: Gets the SPAresnor resolution.

Prototype:
```
double get_resnor ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# initialize_base

Function: Modeler Control

Action: Initializes the Base library.

Prototype:
```
logical initialize_base (
    base_configuration* base_config // configuration
        = NULL                      // default
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/base.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/base.hxx

Effect: System routine

# interpolate

Function: Mathematics

Action: Interpolates between two positions.

Prototype:
```
SPAposition interpolate (
    double param,           // parameter to
                            // interpolate
    SPAposition const& p1,  // first position (p1)
    SPAposition const& p2   // second position (p2)
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/position.hxx”
```

Description: The parameter gives the portion of the segment between the first position and the second position. This function returns:

```
((1 – parameter)*p1 + (parameter*p2))
```

for the given parameter and two positions. The parameter can be less than 0 or greater than 1, in which case the function extrapolates.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/position.hxx

Effect: Read–only

# is_equal

Function: Mathematics, Analyzing Models

Action: Determines if two values are equal.

Prototype:

```
logical is_equal (
    const SPAposition& p1,  // first position
    const SPAposition& p2   // second position
    );

logical is_equal (
    const SPAunit_vector& v1,   // first unit vector
    const SPAunit_vector& v2// second unit vector
    );

logical is_equal (
    const SPAvector& v1,    // first vector
    const SPAvector& v2     // second vector
    );

logical is_equal (
    double n1,              // first double
    double n2               // second double
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/vector/acistol.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "baseutil/vector/vector.hxx"
#include "baseutil/logical.h"
```

Description: This function is overloaded, and can accept pairs of doubles, SPApositions, SPApar_pos (shown above), SPAvectors, or SPAunit_vectors as arguments. It returns TRUE if the items are equal within SPAresabs.

***Note*** *The functions that take* SPA*positions, SPAvectors, and SPAunit_vectors are antiquated and they should be replaced in applications by the appropriate* operator== *member functions.*

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: System routine

# is_greater_than

Function: Mathematics, Analyzing Models

Action: Determines if the first value is greater than the second value.

Prototype:
```
logical is_greater_than (
    double n1,              // first value
    double n2               // second value
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: System routine

# is_less_than

Function: Mathematics, Analyzing Models

Action: Determines if the first value is less than the second value.

Prototype:
```
logical is_less_than (
    double n1,              // first value
    double n2               // second value
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: System routine

# is_negative

Function: Mathematics, Analyzing Models

Action: Determines if a value is negative.

Prototype:
```
logical is_negative (
    double n                // value
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/acistol.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: System routine

# is_on_line

Function: Mathematics, Analyzing Models

Action: Determines if a SPAposition is on a line within SPAresabs.

Prototype:
```
logical is_on_line (
    const SPAposition& pt,      // position to test
    const SPAposition& line_pt,     // position on
line
    const SPAunit_vector& line_dir  // direction of
line
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector_utils.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector_utils.hxx

Effect: Read–only

# is_on_plane

Function: Mathematics, Analyzing Models

Action: Determines if a SPAposition is on a plane within SPAresabs.

Prototype:
```
logical is_on_plane (
    const SPAposition& pt,      // position to test
    const SPAposition& plane_pt,// position on
    const SPAunit_vector& normal// plane normal
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/vector_utils.hxx”
#include ”baseutil/logical.h”
#include ”baseutil/vector/position.hxx”
#include ”baseutil/vector/unitvec.hxx”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector_utils.hxx

Effect: Read–only

# is_positive

Function: Mathematics, Analyzing Models

Action: Determines if a value is positive.

Prototype:
```
logical is_positive (
    double n                // value
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/acistol.hxx"
#include "baseutil/logical.h"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: System routine

# is_zero

Function: Mathematics, Analyzing Models

Action: Determines if a value is zero relative to SPAresabs.

Prototype:
```
logical is_zero (
    const SPAposition& p    // position
    );

logical is_zero (
    const SPAunit_vector& u // unit vector
    );

logical is_zero (
    const SPAvector& v      // vector
    );

logical is_zero (
    double n                // double
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/acistol.hxx"
#include "baseutil/vector/position.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "baseutil/vector/vector.hxx"
#include "baseutil/logical.h"
```

Description: This overloaded function accepts a double, SPAposition, SPAvector, or SPAunit_vector (shown above) as inputs, and returns TRUE if that item is zero within SPAresabs, FALSE if it is nonzero.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/acistol.hxx

Effect: Read–only

# normalise

Function: Mathematics

Action: Converts a vector into a unit vector.

Prototype:
```
unit_vector normalise (
    SPAvector const& v      // vector
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”baseutil/vector/vector.hxx”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: Read–only

# parallel

Function: Mathematics

Action: Determines if two vectors are parallel (within some resolution).

Prototype:
```
logical parallel (
    SPAunit_vector const& v1,   // first vector
    SPAunit_vector const& v2,   // second vector
    const double res        // resolution
        = SPAresnor
    );
```

```
logical parallel (
    SPAunit_vector const& v1,// first vector
    SPAvector const& v2,     // second vector
    const double res         // resolution
        = SPAresnor
    );

logical parallel (
    SPAvector const& v1,     // first vector
    SPAvector const& v2,     // second vector
    const double res         // resolution
        = SPAresnor
    );

logical parallel (
    SPAvector const& v1,     // first vector
    SPAunit_vector const& v2,// second vector
    const double res         // resolution
        = SPAresnor
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/unitvec.hxx”
#include ”baseutil/vector/vector.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: Read–only

# perpendicular

Function: Mathematics

Action: Determines if two vectors are perpendicular (within some resolution).

Prototype:
```
logical perpendicular (
    SPAunit_vector const& v1,// first vector
    SPAunit_vector const& v2,// second vector
    const double res         // resolution
        = SPAresnor
    );
```

```
logical perpendicular (
    SPAunit_vector const& v1,// first vector
    SPAvector const& v2,     // second vector
    const double res         // resolution
        = SPAresnor
    );

logical perpendicular (
    SPAvector const& v1,     // first vector
    SPAvector const& v2,     // second vector
    const double res         // resolution
        = SPAresnor
    );

logical perpendicular (
    SPAvector const& v1,     // first vector
    SPAunit_vector const& v2,// second vector
    const double res         // resolution
        = SPAresnor
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/unitvec.hxx"
#include "baseutil/vector/vector.hxx"
#include "baseutil/logical.h"
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: Read–only

# reflect_transf

Function: Construction Geometry, Transforms, Modifying Models

Action: Constructs a reflection transformation for a reflection about the plane through the perpendicular to the given vector.

Prototype:
```
SPAtransf reflect_transf (
    SPAvector const& axis   // vector for reflection
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/transf.hxx”
#include ”baseutil/vector/vector.hxx”
```

Description: The vector defines a normal to the plane passing through the origin. The function generates a transformation matrix describing a reflection across this plane.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/transf.hxx

Effect: Read–only

# rotate_transf

Function: Construction Geometry, Transforms, Modifying Models

Action: Constructs a rotation transformation for a simple rotation by an angle about the given vector.

Prototype:
```
SPAtransf rotate_transf (
    double angle,           // rotation angle
    SPAvector const& axis   // vector
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/transf.hxx”
#include ”baseutil/vector/vector.hxx”
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/transf.hxx

Effect: Read–only

# same_matrix

Function: Mathematics

Action: Determines if two matrices are equal, given some tolerance.

Prototype:
```
logical same_matrix (
    SPAmatrix const& m1,     // first matrix
    SPAmatrix const& m2,     // second matrix
    const double res         // resolution
        = SPAresabs
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/matrix.hxx"
#include "baseutil/logical.h"
```

Description: This is used by the matrix class.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/matrix.hxx

Effect: System routine

# same_vector

Function: Mathematics

Action: Determines whether or not two vectors are the same.

Prototype:
```
logical same_vector (
    SPAvector const& v1,     // first vector
    SPAvector const& v2,     // second vector
    const double res         // resolution for
        = SPAresabs          // comparison
    );
```

Includes:
```
#include "kernel/acis.hxx"
#include "baseutil/vector/vector.hxx"
#include "baseutil/logical.h"
```

Description: Refer to action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/vector.hxx

Effect: System routine

# scale_transf

Function: Construction Geometry, Transforms, Modifying Models

Action: Constructs a transformation for overall scaling.

Prototype:

```
SPAtransf scale_transf (
    double scale            // scale factor
    );

SPAtransf scale_transf (
    double xs,              // x scale factor
    double ys,              // y scale factor
    double zs,              // z scale factor
    double xys,             // xy shear factor
    double xzs,             // xz shear factor
    double yzs              // yz shear factor
    );

SPAtransf scale_transf(
    double xs,              // x scale factor
    double ys,              // y scale factor
    double zs               // z scale factor
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/vector/transf.hxx"
```

Description: This function constructs a transformation for overall scaling. It is overloaded and has two forms. The first form sets the transformation to the scale factor times the identity matrix. In other words, the same scaling factor is applied in all directions.

The second and third forms of this function set the transformation to differential scaling. Such non-uniform scaling (i.e., when *x*, *y* and *z* are not equal, or when there is shearing) invokes a dependency on the Operators Component (OPER). The functions api_initialize_operators and api_terminate_operators must be used to initialize and terminate the OPER library.

Errors: If non-uniform scaling is used, the OPER library must be initialized.

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/transf.hxx

Effect: Read–only

# scaling

Function: Mathematics

Action: Creates the matrix for scaling.

Prototype:

```
SPAmatrix scaling (
    double factor           // scale factor
                            // scaling is equal
    );

SPAmatrix scaling (
    double xfactor,         // x scale factor
    double yfactor,         // y scale factor
    double zfactor          // z scale factor
                            // scaling is
                            // differential by
                            // coordinate
    );

SPAmatrix scaling (
    SPAvector factor        // xyz factors in
                            // a vector
    );

SPAmatrix scaling (
    double xfactor,         // x scale factor
    double yfactor,         // y scale factor
    double zfactor,         // z scale factor
    double xyshear,         // xy shear factor
    double xzshear,         // xz shear factor
    double yzshear          // yz shear factor
                            // scaling is
                            // differential by
                            // coordinate
    );
```

Includes:

```
#include "kernel/acis.hxx"
#include "baseutil/vector/matrix.hxx"
#include "baseutil/vector/vector.hxx"
```

Description: This is used by the matrix class.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/vector/matrix.hxx

Effect: Read–only

# terminate_base

Function: Modeler Control

Action: Terminates the Base library.

Prototype:
```
logical terminate_base ();
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/base.hxx”
#include ”baseutil/logical.h”
```

Description: Refer to Action.

Errors: None

Limitations: None

Library: baseutil

Filename: base/baseutil/base.hxx

Effect: System routine

# translate_transf

Function: Construction Geometry, Transforms, Modifying Models

Action: Constructs a translation transformation to translate along the given vector.

Prototype:
```
SPAtransf translate_transf (
    SPAvector const& disp   // vector
    );
```

Includes:
```
#include ”kernel/acis.hxx”
#include ”baseutil/vector/transf.hxx”
#include ”baseutil/vector/vector.hxx”
```

| | |
|---|---|
| Description: | Refer to action. |
| Errors: | None |
| Limitations: | None |
| Library: | baseutil |
| Filename: | base/baseutil/vector/transf.hxx |
| Effect: | Read–only |