Chapter 6. Options

Topic: Ignore

Options may be set to modify the behavior of ACIS. An option's value may be a flag (indicating an on/off state), a number (integer or real number), or a string. Options may be set in a Scheme application (such as Scheme AIDE) using the Scheme extension option:set; in the ACIS Test Harness using the command option; or in a C++ application using one of several API functions. Refer to the *3D ACIS Online Help User's Guide* for a description of the fields in the reference template.

abl_remote_ints

Option: Blending, Modeler Control

Action: Sets the use of the global Boolean for entity-entity blending.

Name String: `abl_remote_ints`

Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: If *off*, this indicates to the entity-entity blending algorithm that it can assume there are no intersections between the blend sheet and blank body faces, except for those along the spring curves and any end or side caps. The benefit is that entity-entity blending will generally work faster with this option *off*.

Normally, blend stage two looks out for and computes intersections of the blend faces against remote blank faces (e.g., ones that it intersects without the spring curves touching any face boundary), such as isolated bosses that are preserved poking through the blend sheet at the end of the blend. If the user can guarantee that there are no such intersections (usually the case) then this option could be set to *off* to avoid these checks, sometimes with significant performance gains.

Usually, even if there are remote intersections, turning this option *off* still allows the blend to work, with the effect of “swallowing” the boss. If, however, the boss is a “handle” and intersects the blank body elsewhere, the blend will fail. Occasionally, if the boss joins the blank completely away from the blend, but part of the boss pokes through the blend from the outside, without touching the blank there, an invalid body will result.

Example:

```
; abl_remote_ints
; Turn off remote intersection option
(option:set ”abl_remote_ints” #f)
;; #f
```

abl_require_on_support

Option: Blending, Modeler Control

Action: Sets whether or not both sides of a blend must be on their supports.

Name String: **abl_require_on**_support

Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: With this set to *off*, BLND does not require both sides of a blend to be actually on their supports, and instead will attempt to cap off the side that is not on its support.

Note *Applications should not modify this option except when needed for backward compatibility.*

Example:

```
; abl_require_on_support
; Turn option off
(option:set ”abl_require_on_support” #f)
;; #t
```

add_bl_atts

Option: Blending, Modeler Control, Attributes

Action: Controls whether or not blend attributes are left on blend faces.

Name String: **add_bl_atts**

Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: When this option is on (true), blending will leave blend attributes on the blend faces created. These attributes allow later operations to recognize the faces as blends. For example, local operations will treat such blend faces as blends, preserving the tangency between the blend face and its support faces.

Example:

```
; add_bl_atts
; Create a blend and leave blend attributes so local
; op will work
(define b (solid:block (position -10 -10 -10)
    (position 10 10 10)))
;; b
(define e (pick:edge (ray (position 0 0 0)
    (gvector 1 0 1))))
;; e
(option:set "add_bl_atts" #t)
;; #f
(blend:const-rad-on-edge e 5)
;; #[entity 3 1]
(blend:network e)
;; #[entity 2 1]
(define f (pick:face (ray (position 0 0 0)
    (gvector 1 0 0))))
;; f
(define blend_face (pick:face (ray (position 0 0 0)
    (gvector 1 0 1))))
;; blend_face
; If add_bl_atts is not on, the local op will fail
(lop:taper-faces (list f blend_face)
    (position 0 0 -10) (gvector 0 0 1) 20)
;; #[entity 2 1]
```

blend_make_simple

Option: Modeler Control, Blending

Action: Sets whether or not blends are simplified, if possible.

Name String: **blend_make_s**imple

Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: If on, this option causes blending (both standard and advanced) to create the simplest blend possible that gives the same shape as the blend requested. If the option is off, no blends are simplified.

Note *Applications are strongly discouraged from changing this option.*

If the option is on, three blend types can be changed, one in standard blending and two in advanced blending.

Standard Blending

– If the requested blend is a standard two–ends blend (the only variable radius standard blend), then, if the two end values are equal, a standard constant–radius blend is created.

Advanced Blending

– If the requested blend has a round cross section and is a face–face blend *and* the radius is constant, a standard constant–radius blend is created.
– If the requested blend has a round cross section and is a face–face blend *and* the radius is two–ends, a standard two–ends blend is created.

Example:

```
; blend_make_simple
; Turn off blend simplification
(option:set ”blend_make_simple” #f)
;; #t
```

blend_mix_convexity

Option: Modeler Control, Blending

Action: Controls how blending handles mixed convexity edges.

Name String: **blend_mix_convexity**

Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: ACIS allows the blending of a concave and a convex edge when this option is on. However, in older versions of ACIS (prior to 6.1), when blending a concave and a convex edge, blending returned an error about the mixed convexity situation. An application can force blending to use the older algorithm by turning this option off.

Example:

```
; blend_mix_convexity
; Use old blending behavior; don’t allow mixed
(option:set ”blend_mix_convexity” #f)
;; #t
```

bl_cap_box_factor

Option: Modeler Control, Blending

Action: Sets the size of the box used by the new capping algorithm.

Name String: **bl_cap_box**_factor

Scheme:	real	> 0	2.0
Test Harness:	double	> 0	2.0
C++:	double	> 0	2.0

Description: Controls the size of the box used by the new capping algorithm for caps in which it is prepared to explore. Any caps that require exploration beyond this box will fail. Increasing the value increases the size of the box, but also the length of time capping may take. The default value strikes a fair balance between not exploring ludicrously far, but still successfully completing all but “unreasonable” examples.

Example:

```
; bl_cap_box_factor
; Set to 3.0
(option:set ”bl_cap_box_factor” 3.0)
;; 2
```

bl_new_capping

Option: Modeler Control, Blending

Action: Sets whether or not the new capping algorithm is used.

Name String: **bl_new_c**apping

Scheme:	boolean	#f, #t	#t

Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: By default, the new capping algorithm (introduced in Release 3.0) is used by both standard and advanced blending. The new capping algorithm includes special case analysis to determine the simplest geometry to use. If this option is turned off, the old capping algorithm is used.

Note *Applications are strongly discouraged from changing this option.*

Example:

```
; bl_new_capping
; use the old algorithm
(option:set ”bl_new_capping” #f)
;; #t
```

bl_new_technology

Option: Modeler Control, Blending

Action: This is an internal testing option that sets how the new internal blending algorithms are used.

Name String: **bl_new_t**echnology

Scheme:	integer	0, 1, 2, 3	3
Test Harness:	integer	0, 1, 2, 3	3
C++:	int	0, 1, 2, 3	3

Description: This option is not intended for general use, but may be used by customers that are participating in testing of new blending technology. Customers should only change the value of this option for testing examples with the new algorithms.

The valid values are:

0	Off. Never use the new algorithms (i.e., behave like release 4.x)
1	On. Reserved for internal use by *Spatial Corp.* only. Customers should *never set this value*, as its meaning may change from release to release.
2	Use the new algorithms *only* for entity-entity blends.
3	Use the new algorithms for entity-entity blends, and for the construction of sheet faces in standard blending.
>3	Reserved for future use.

Blending should be regression free only for values equal to or lower than the default setting (*excluding the value of 1*). As the new algorithms are completed, *Spatial Corp.* may increase the default value in future releases.

Example:
```
; bl_new_technology
; Do not use the new algorithms
(option:set ”bl_new_technology” 0)
;; 3
```

bl_preview_approx_sf

Option: Modeler Control, Blending

Action: Sets whether or not api_preview_blends should use the spline approximation of any underlying surfaces in computing the preview sheet.

Name String: **bl_preview_app**rox_sf

Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: If on, api_preview_blends uses the spline approximation of any underlying surfaces in computing the preview sheet. The default is on. In general, customers should have not reason to change this.

Example:
```
; bl_preview_approx_sf
; Don’t use approximations
(option:set ”bl_preview_approx_sf” #f)
;; #t
```

bl_preview_tightness

Option: Modeler Control, Blending

Action: Sets how tight the blend preview is to the true blend.

Name String: **bl_pre**view_tightness

Scheme:	real	1.0 ... 20.0	8.0
Test Harness:	double	1.0 ... 20.0	8.0
C++:	double	1.0 ... 20.0	8.0

Description: This option provides some control over how close (tight) the blend preview is to the real blend. The default should give good results for most blends, but if a preview sheet does not seem to agree well with the final result, then increasing this value may help.

The larger the value, the better the preview, but it can also be slower. Because of the approximations made in generating the preview, there are fundamental reasons why even a "largest tightness" preview will still differ from the true blend.

Example:

```
; bl_preview_tightness
; Make preview tighter
(option:set ”bl_preview_tightness” 10.0)
;; 8
```

bl_remote_ints

Option: Modeler Control, Blending

Action: Sets whether or not global interference checking is performed.

Name String: **bl_rem**ote_ints

Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: A critical portion of the blend algorithm is to detect the interference of the blend sheet with the geometry of the part being blended. Usually, interfering geometry can be detected by intersecting the spring curves of the sheet with geometry local to the blend. Occasionally, the surfaces of the sheet must be intersected with all of the geometry of the body to detect such global interferences.

The option bl_remote_ints directs the blending algorithm to perform global or local interference checking. This option is off by default, indicating that the algorithm is to perform local interference checking only. This speeds up blending dramatically in many cases. Although not common, cases do sometimes arise which require global interference checking. If desired, option bl_remote_ints can be turned on, indicating that global interference checking is to be performed.

Example:

```
; bl_remote_ints
; Turn on global interference checking
(option:set ”bl_remote_ints” #t)
;; #f
```

bl_tb_autoblend

Option:	Modeler Control, Blending		
Action:	Determines whether autosetback uses api_set_vblend_autoblend or api_set_vblend_auto in the ACIS Test Harness.		
Name String:	**`bl_tb_autoblend`**		
Scheme:	Not applicable		
Test Harness:	integer	0, 1	0
C++:	Not applicable		
Description:	This option only applies to the ACIS Test Harness. If this option is on, autosetback uses api_set_vblend_autoblend, which allows rolling ball vertex blends to override the autosetbacks. If this option is off, autosetback uses api_set_vblend_auto, which always produces a vertex blend *n*-sided patch in preference to a rolling ball surface.		
Example:	Not applicable		

bl_tb_preview

Option:	Modeler Control, Blending		
Action:	Determines whether blend previews are shown as blend attributes are applied in the ACIS Test Harness.		
Name String:	`bl_tb_preview`		
Scheme:	Not applicable		
Test Harness:	integer	0, 1	1
C++:	Not applicable		
Description:	This option only applies to the ACIS Test Harness. If this option is on, blend previews are shown as the blend attributes are applied.		
Example:	Not applicable		

cap_preference

Option:	Modeler Control, Blending
Action:	Determines whether capping tries to use as many or as few capping faces as possible.

Name String:	**cap_pref**erence		
Scheme:	string	"min", "max", "old"	"old"
Test Harness:	string	"min", "max", "old"	"old"
C++:	char*	"min", "max", "old"	"old"

Description: It this option is set to "min", blending tries to use as few capping faces as possible. If this option is set to "max", blending tries to use as many capping faces as possible. If this option is set to "old" (default), blending tries to emulate the old capping algorithm. Usually this gives a maximum number of faces solution, but it will give a minimum number of faces solution if the cap can be started and ended with the same face. The precise set of heuristics of the algorithm cannot be emulated in all cases.

Note *Applications are advised against changing this option in order to obtain the best blending results.*

Example:

```
; cap_preference
; Set to minimum number of cap faces
(option:set ”cap_preference” ”min”)
;; ”old”
```

force_capping

Option: Modeler Control, Blending

Action: Determines whether blending will fail or force a cap in some special cases.

Name String:	**force_cap**ping		
Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: In many cases, it is preferable for a blend to fail rather than produce a cap that cuts right across the blend face, or that deletes large portions of the original body. If this option is off (false), the capping code will fail in these cases. However, some applications need the capping code to cap whenever possible. Switching this option on (true) forces the capping code to cap whenever possible.

Example:

```
; force_capping
; Turn option on so capping is always done
(option:set ”force_capping” #t)
;; #f
```

show_vertices

Option: Modeler Control, Blending

Action:	Visually marks vertices of a part for rendering.		
Name String:	**show_vert**ices		
Scheme:	boolean	#t, #f	#f
Test Harness:	Not applicable		
C++:	Not applicable		

Description: Turning this option on will force all the vertices of a part to be marked with an ”X” (or whatever the current default is for drawing points). Vertices with blend attributes are drawn in white. The display list must be rebuilt (use render:rebuild) for it actually to take effect.

Example:

```
; show_vertices
; Visually mark vertices of a part.
(solid:block (position 0 0 0) (position 10 10 10))
;; #[entity 2 1]
(option:set ”show_vertices” #t)
;; #f
(render:rebuild)
;; ()
```

vbl_check

Option: Modeler Control, Blending

Action:	Sets whether or not further checks are done on vertex blend surfaces created by blending.		
Name String:	**vbl_check**		
Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: If off, no checks are performed and this behaves like ACIS 4.x. If turned on, blending may sometimes trap the fact that an illegal surface could result, and will issue an error instead.

Example:

```
; vbl_check
; Turn option on so checks are done
(option:set ”vbl_check” #t)
;; #f
```

vbl_quick_check

Option: Modeler Control, Blending

Action:	Sets whether or not quick checks are done on vertex blends.		
Name String:	**vbl_q**uick_check		
Scheme:	boolean	#f, #t	#f
Test Harness:	integer	0, 1	0
C++:	logical	FALSE, TRUE	FALSE

Description: If on, reasonably quick checks are performed on the legality of a vertex blend. If off, no checks are performed.

Example:

```
; vbl_quick_check
; Turn option on so checks are done
(option:set ”vbl_quick_check” #t)
;; #f
```

vtx_blnd_simple

Option: Modeler Control, Blending

Action:	Sets whether or not the new algorithm for vertex blend boundaries is used.		
Name String:	**vtx_blnd_simple**		
Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: If on, the new algorithm for calculation of the vertex blend boundaries is used. The new algorithm tries to produce a minimal vertex blend surface that is legal and well defined using canted cross curves. Rolling ball vertex blend solutions are used wherever possible.

Example:

```
; vtx_blnd_simple
; Turn option on
(option:set ”vtx_blnd_simple” #t)
;; #t
```

v_blend_nsurf

Option: Modeler Control, Blending

Action: Enables or disables construction of general vertex blend surfaces.

Name String:	**`v_blend_nsurf`**		
Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: Turning this option off suppresses construction of *n*-sided vertex blend surfaces, allowing the boundary of the face being made for the vertex blend to be inspected.

If the boundary is very convoluted or has two adjacent edges meeting tangentially or in a cusp, it is not possible to construct an *n*-sided blend surface, and the system error BL_NO_VTX_GEOM is returned.

The user can often improve the boundary by changing the blend sizes or setbacks on the blended edges meeting in the blended vertex. To do this, one needs to see what the boundary looks like. To view the boundary, this option should be turned off, and the blend should be repeated, making only the sheet. Attempting to do a fix causes the missing surface to result in the error BL_NO_SHEET_SURF in stage two of blending.

Example:

```
; v_blend_nsurf
; Suppress n–sided vertex blend surfaces
(option:set ”v_blend_nsurf” #f)
;; #t
```

v_blend_rb

Option: Modeler Control, Blending

Action: Sets whether or not a vertex blend surface can be a rolling ball surface.

Name String:	**`v_blend_rb`**		
Scheme:	boolean	#f, #t	#t
Test Harness:	integer	0, 1	1
C++:	logical	FALSE, TRUE	TRUE

Description: If on, the search for a surface to cover the *n*-sided boundary of a face to be made for a vertex blend, is broadened to include a rolling ball surface.

A rolling ball surface is considered if *all* of the following conditions are true:

– The option v_blend_rb is on.
– The number of edges (all blended) at the blended vertex is three (3).
– Each edge has a constant round with no setback at the vertex.
– The blends on two of the edges (of the same convexity) are the same size.
– The blend on the third edge is of a greater size or is of opposite convexity to the blends on the other two edges.

The search for a surface considers possible surfaces in the order:

– plane
– cone
– sphere
– torus
– rolling–ball
– *n*–sided polygon

The effect of using a rolling–ball surface is the same as if the blend on the ”third” edge above were evaluated (fixed) first, and then a blend was run along the smooth sequence of three edges formed by the other two blended edges with the edge made by the first fix interposed between them.

Example:

```
; v_blend_rb
; Turn off rolling ball vertex blend
(option:set ”v_blend_rb” #f)
;; #t
```