Chapter 4.

Save Identifiers and Classes

This chapter describes the save file information for the **ACIS Kernel** and standard ACIS husks, such as the Faceter Husk, Generic Attribute Husk, Faceted Hidden Line Husk, Cellular Topology Husk, Geometry Husk, Graphic Interaction Husk, and Part Management Husk.

It also describes the subtype format for the save file. Subtypes and references are frequently used within any given save file, because they reduce the amount of information that has to be stored in the file. Instead of defining information two or more times for items which may already be specified in the file, references are made to the instance of data.

Note: *Subtypes may be nested within other subtypes. References to subtypes may be nested within other subtypes.*

The **ACIS Kernel** is divided into topology, geometry, and attributes. All classes in the **ACIS Kernel** are derived in some manner from ENTITY. The classes given in this chapter are used in class derivation. Subtypes are derived from one of the three main types: int_cur, par_cur, and spl_sur.

Class Format

The structure of a class definition in the save file is created starting from the inside and then working out. In other words, the save identifier name is saved at the left-most portion of a record, while its data is appended to the end of all other current sv id data elements. The next element in the record in turn saves its save identifier name at the left-most portion of the record while appending its data to the end.

Only save identifier (sv id) names in quotation marks (“ “) are actually written to the save file.

Subtype Format

The structure of a subtype definition in the save file is not created in the same way as a class format. Instead, at the point in the record where a subtype is defined, what is written to the save file is.

subtype_start	left curly braces, "{" or Tag 15
save identifier	subtype name
save data	subtype data
subtype_end	right curly braces, "}" or Tag 16

The information is seqential between the subtype_start and subtype_end and does not obey the inside-out encapsulation of normal save file records.

(Save-Format-Template)

<NAME> . . . . . . The *Name* title is the save identifier used either in the save file itself, or in this manual to specify a unique class of information. If the identifier could appear in a save file, then it is enclosed within quotation marks (" ") in the topic title. Identifiers or identifier strings not in quotation marks do not appear in save files, but are used in this manual to trace the data actually written to a save file.

Save identifier are generally written with all other save identifiers in a record, unless the save data section indicates otherwise (which is the case for subtypes). This means an inside-out methodology is employed: the quoted save identifier appears with other save identifiers at the left in a record, while its data is to the right (or below). The save identifiers within a group are generally separated with dashes (–).

Subtypes, however, write the save identifier immediately following the subtype_start. This is followed immediately by the subtype's data and subtype_end.

Purpose: The *Purpose* field summarizes the intended purpose of the class save information.

Derivation: The *Derivation* field specifies the derivation of the class. The class described by the template is always listed on the left, followed by its parent, grandparent, etc. The last class in the list is always a base class, indicated by a trailing hyphen (–). Classes with no parents (base classes) show only the class name and a trailing hyphen (–).

Data Element: The *Data Element* field describes the information that is saved for each class. It is divided into three columns.

The left-hand column conveys the type of information that the data is. Types are "prim" for primitives, "ctrl" for controls, "enum" for enumerations, and "sv id" for references to other save items.

The middle column contains the data written by this class to the save file, in the order the data is written.

The right-hand column provides a short description.

Description: The *Description* field describes the class save format, its use, and its surrounding conditions.

Component: The *Component* field specifies where the information comes from in the ACIS architecture.

"1d"

Purpose: Implements the ELEM1D.

Derivation: ELEM1D : ELEM : ENTITY : –

Data Elements:

sv id	"elem"	parent information
ctrl	if_cond	if type is a straight_type
prim	char	writes an "s" for "straight"
ctrl	else	if type is not a straight_type
sv id	curve type	saves the information for that specific type of curve.

Description: Represents a piece of a composite curve. It contains a curve which forms the piece of the composite one.

Component: **Mesh Surface Husk**

"2d"

Purpose: Implements the ELEM2D.

Derivation: ELEM2D : ELEM : ENTITY : –

Data Elements:

sv id	"elem"	save parent information
ctrl	repeat	perform for node count
prim	$rec_num	Pointer to records in save file for NODE(s)
ctrl	repeat	perform for element count
prim	$rec_num	Pointer to records in save file for ELEMENT(s)

Description: Represents an elemental patch on a mesh surface. It contains the patch boundary information.

Component: **Mesh Surface Husk**

“adjedge”

Purpose: Records blank body edges that are in contact with the sheet boundary, which are called *adjacent edges*.

Derivation: ATTRIB_ADJEDGE : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class records blank body edges that are in contact with the sheet boundary, which are called *adjacent edges.* The edge and the parameter value on the edge at the point of contact are recorded. The attribute is attached to the vertex of the sheet in contact with the adjacent edge.

Component: ACIS Kernel

“adjface”

Purpose: Records blank body faces that are in contact with the sheet boundary, which are called *adjacent faces*.

Derivation: ATTRIB_ADJFACE : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class records blank body faces that are in contact with the sheet boundary, which are called *adjacent faces.* The face and the parameter value on the face at the point of contact are recorded. The attribute is attached to the vertex of the sheet in contact with the adjacent face.

Component: ACIS Kernel

af_node_mapping

Purpose: Lookup table interface for token types in mesh nodes.

Derivation: af_node_mapping :–

Data Elements:	prim	integer	Number of tokens
	ctrl	repeat	Repeat for the number of tokens
	enum	token_name	Token indicating data type to be stored

Description: A node_mapping is a mapping from a set of token types to positions in an array of cells.

Component: Faceter Husk

“agc”

Purpose: Identifies that the attribute came from the ATTRIB_AGC class.

Derivation: ATTRIB_AGC : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: Identifier used externally to identify a particular entity type. This is only used within the save/restore system for translating to/from external file format, but must be unique amongst attributes derived directly from ATTRIB, across all application developers.

Component: Deformable Surfaces

“attrib”

Purpose: Represents common data and functionality for all attributes.

Derivation: ATTRIB : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for next attribute owned by entity
	prim	$rec_num	Pointer to record in save file for previous attribute owned by entity
	prim	$rec_num	Pointer to record in save file for owning entity

Description: ATTRIB is the generic class for user and system attributes. It provides housekeeping to maintain ENTITY attribute lists. Every entity may have one or more attributes. These hang off the ENTITY’s attribute pointer, and are arranged as a doubly-linked list.

Component: **ACIS Kernel** Attributes

“ATTRIB_ADV_VAR_BLEND”

Purpose: Defines the blend attribute for edge sequence-following blends.

Derivation: ATTRIB_ADV_VAR_BLEND : ATTRIB_VAR_BLEND : ATTRIB_FFBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

sv id	var_radius	refer to var_radius data type
ctrl	if_cond	if there are different radii
sv id	var_radius	refer to var_radius data type
sv id	var_cross_section	refer to var_cross_section data type
prim	$rec_num	Pointer to record in save file for left face
prim	$rec_num	Pointer to record in save file for left edge
prim	logical	either “sharp” or “smooth”
prim	$rec_num	Pointer to record in save file for right face
prim	$rec_num	Pointer to record in save file for right edge
prim	logical	“sharp” or “smooth”
prim	$rec_num	Pointer to record in save file for loft edge

Description: This class defines the attribute for edge sequence-following blends in the **Advanced Blending Husk**. It is derived from the ATTRIB_VAR_BLEND class in basic blending.

Component: **Advanced Blending Husk**

4

“attrib_fhl”

Purpose: Faceted Hidden Line Husk organizational class.

Derivation: ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: ATTRIB_FHL is the organizational class from which the other FHL attribute classes are derived. Its sole purpose is to identify those child classes as belonging to the Faceted Hidden Line Husk.

Component: Faceted Hidden Line Husk

"attrib_fhlgeom"

Purpose: Attaches FHL geometry data to model entities as attributes.

Derivation: ATTRIB_FHL_GEOM : ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

ctrl	if_cond	If the save_version_number is less than the CONSISTENT_VERSION
prim	integer	header for this geometry
prim	integer	next geometry for header
ctrl	else	
prim	$rec_num	Pointer to record in save file for Header for this geometry ATTRIB_FHL_HEAD
prim	$rec_num	Pointer to record in save file for Next geometry for header ATTRIB_FHL_GEOM

Description: All geometric data computed by the Faceted Hidden Line Husk is attached to entities in the form of attributes.

The first attribute in the list is an instance of the ATTRIB_FHL_HEAD class. The remaining attributes in the list are instances of the ATTRIB_FHL_GEOM class, and are referred to as geometry attributes. The geometry attributes store the geometry data.

Component: Faceted Hidden Line Husk

"attrib_fhlhead"

Purpose: Heads the list of FHL attributes attaching hidden line data to the model.

Derivation: ATTRIB_FHL_HEAD : ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

prim	logical	Either "invalid" or "valid"
prim	integer	View token
ctrl	if_cond	If the save_version_number is less than the CONSISTENT_VERSION
prim	integer	Next geometry for this header
ctrl	else	
prim	$rec_num	Pointer to record in save file for Next geometry for this header ATTRIB_FHL_GEOM

4

Description: All geometric data computed by the Faceted Hidden Line Husk is attached to entities in the form of attributes.

The first attribute in the list is an instance of the ATTRIB_FHL_HEAD class and is called the header attribute. The header attribute stores the view-token that identifies which view the attribute set corresponds to.

It also stores a flag that signifies whether or not the attribute set is valid. When an entity undergoes geometric transformations or participates in a Boolean operation, it turns off the validity flag of the header attribute attached to the body. The application program may delete invalid FHL attributes at a later stage.

Component: Faceted Hidden Line Husk

“attrib_fhlmark”

Purpose: Detects changes to underlying geometry.

Derivation: ATTRIB_FHL_MARK : ATTRIB_FHL_GEOM : ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: ATTRIB_FHL_MARK is an abstraction of null geometry. Instances of this class are attached to SHELLs or LUMPs and are useful to detect modifications to the entity (such as Boolean operations) that invalidate the FHL attribute set.

Component: ACIS Kernel

“attrib_fhlplist”

Purpose: Attaches FHL polyline data to model ENTITYs.

Derivation: ATTRIB_FHL_PLIST : ATTRIB_FHL_GEOM : ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

prim	logical	Visiblity: either “invisible” or “visible”
prim	integer	Number of points
ctrl	repeat	Repeat for the Number of points
prim	real	*x*-coordinate
prim	real	*y*-Coordinate

Description: Line segment data stored in ATTRIB_FHL_SLIST objects is unordered, and so inefficient for plotters. A function converts the unordered segments into polylines and stores them in ATTRIB_FHL_PLIST objects.

ATTRIB_FHL_PLIST objects store the polyline as a list of 2D points. The points in the point list are accessed via an index by some methods as though they are stored in an array.

Visibility information is stored for the polyline, and applies to the entire polyline.

Component: Faceted Hidden Line Husk

“attrib_fhl_slist”

Purpose: Attaches FHL line segment data to model entities.

Derivation: ATTRIB_FHL_SLIST : ATTRIB_FHL_GEOM : ATTRIB_FHL : ATTRIB : ENTITY : –

Data Elements:

prim	integer	Number of segments
ctrl	repeat	Repeat for the number of segments
prim	real	Start *x*-coordinate
prim	real	Start *y*-coordinate
prim	real	End *x*-coordinate
prim	real	End *y*-coordinate
prim	logical	Either “invisible” or “visible”

Description: ATTRIB_FHL_SLIST stores unordered line segment data in a list of segments. Each segment can be visible or not visible.

Component: Faceted Hidden Line Husk

4

ATTRIB_NODE

Purpose: Represents a grazing touch of an intersection curve with the side (boundary) of an element.

Derivation: ATTRIB_NODE : ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:

sv id	“ms”	ATTRIB_MESH parent
prim	integer	side
prim	position	node location

Description: Attribute to represent a grazing touch of an intersection curve with the side (boundary) of an element. Used in mesh surface remeshing. We make this attribute ”savable” since remeshing is optionally on demand.

Component: **Mesh Surface Husk**

ATTRIB_PHL

Purpose: Defines the phl tag identifier.

Derivation: ATTRIB_PHL : ENTITY_PHL : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: The phl tag is used by two different classes.

Component: **Precise Hidden Line Husk**

“attrib_var_blend”

Purpose: Defines the blend attribute for variable radius blends.

Derivation: ATTRIB_VAR_BLEND : ATTRIB_FFBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for defining curve
	prim	real	*v*-range start
	prim	real	*v*-range end
	prim	logical	Calibrated two ends: “uncalibrated” or “calibrated”
	ctrl	if_cond	If calibrated two ends
	prim	real	Parameter of start radius
	prim	real	Parameter of end radius
	ctrl	if_cond	If the save_version_number is less than the CONSISTENT_VERSION
	prim	integer	radius form
	ctrl	else	
	enum	rad_form_ents	Radius specification format enumeration
	ctrl	if_cond	if the radius form is set for two ends
	prim	real	Start radius
	prim	real	End radius
	ctrl	else if_cond	else if the radius form is set for functional
	prim	logical	Two radii: “one_radius” or “two radii”
	sv id	bs2_curve_def	Radius function
	ctrl	if_cond	if there are two radii
	sv id	bs2_curve_def	Right radius function
	ctrl	else if_cond	else if the radius form is set for fixed width
	prim	real	Width
	ctrl	if_cond	if the save_version_number is less than the CONSISTENT_VERSION
	prim	integer	section form
	ctrl	else	
	enum	sec_form_ents	Cross section form enumeration
	ctrl	if_cond	if the section form is set for thumbweights
	prim	real	Left thumbweight
	prim	real	Right thumbweight
	ctrl	if_cond	if the save_version_number greater than or equal to SAFERANGE_VERSION)
	prim	logical	sense data, direction of blend with respect to the defining curve: “forward” or “reversed”
	prim	$rec_num	Pointer to record in save file for first edge of blend sequence
	prim	$rec_num	Pointer to record in save file for last edge of blend sequence

Description: Refer to the Purpose.

Component: **ACIS Kernel** Attributes

4

BDY_GEOM

Purpose: The base class for different representations of the boundary.

Derivation: BDY_GEOM : –

Data Elements:

ctrl	if_cond	if save_version is set to 105
prim	real	
ctrl	else if_cond	if save_version is greater than 105
prim	logical	“non_cross” or “cross”
prim	vector	magic
prim	logical	“non_smooth” or “smooth”
prim	logical	“non_smooth” or “smooth”
prim	real	Fullness

Description: This is all the VBL_SURF has to know about and provides a variety of different functions to do with computing the cross boundary derivative, setting up cached data, default relaxation methods etc.

Component: **ACIS Kernel**

BDY_GEOM_CIRCLE

Purpose: Implements the boundary geometry as an ellipse (actually a circle).

Derivation: BDY_GEOM_CIRCLE : BDY_GEOM : –

Data Elements:

sv id	curve type	Curve
prim	position	Center
enum	underlying_sf_type	Surface type enumeration
ctrl	if_cond	if the surface type is set to a torus
prim	position	Center
ctrl	if_cond	if the surface type is set to a pipe
sv id	surface type	Pipe surface
sv id	bs2_curve_def	Parameter space curve
prim	boolean	Reverse pcurve
prim	real	Pcurve fit tolerance
ctrl	if_cond	if the surface type is set to a given twist
prim	vector	Twist 0
prim	vector	Twist 1
prim	real	Start parameter
prim	real	End parameter
prim	boolean	Reverse

Description: This class implements the boundary geometry as an ellipse (actually a circle) which is a section of a cylindrical, toroidal or pipe surface, corresponding to the types CYLINDER, TORUS or PIPE. The only difference between these cases is how the twist vectors are calculated. This gives rise to the fourth case, GIVEN_TWIST, where it is no longer known what surface the circle really lies on but it is known what the relevant twist values are. As soon as the twists have been calculated any of the first three types could mutate themselves into a GIVEN_TWIST type.

Component: **ACIS Kernel**

BDY_GEOM_DEG

Purpose: This class implements a boundary section as a degenerate point.

Derivation: BDY_GEOM_DEG : BDY_GEOM : –

Data Elements:

prim	position	Position
prim	vector	Normal 0
prim	vector	Normal 1

Description: This class implements a boundary section as a degenerate point. For example, the “zero radius circle” which is imagined across an unblended edge. It will still represent an entire polygon edge in parameter space.

Component: **ACIS Kernel**

BDY_GEOM_PCURVE

Purpose: Defines all the functions the underlying geometry of the boundary curves must provide.

Derivation: BDY_GEOM_PCURVE : BDY_GEOM : –

Data Elements:

sv id	surface type	Surface on which pcurve lies
sv id	bs2_curve_def	Parameter space curve on surface
prim	boolean	Reverse pcurve
prim	real	Fit tolerance

Description: This implements the boundary geometry as a pcurve on the boundary surface – we can’t compute Nll for these.

Component: **ACIS Kernel**

BDY_GEOM_PLANE

Purpose: Implements the boundary geometry as a curve on a surface which must be a plane.

Derivation: BDY_GEOM_PLANE : BDY_GEOM : –

Data Elements:

prim	vector	Plane normal
prim	real	Start parameter for curve
prim	real	End parameter for curve
sv id	curve type	Curve on plane

Description: This implements the boundary geometry as a curve on a surface which must be a plane: we only need keep the unit normal (and a non-const pointer to the curve) in addition to the bounded curve. It is also used to represent a curve for which we have no normal data to interpolate.

Component: **ACIS Kernel**

"bl_support"

Purpose: Derived class for the blend support point.

Derivation: ATTRIB_BLEND_SUPPORT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

prim	$rec_num	pointer to
prim	$rec_num	pointer to previous support pointer
prim	$rec_num	pointer to next support pointer
prim	integer	support index data

Description: The blend support class is a pure base class. Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

"bldcur"

Purpose: This is one long edge of a general blend surface – a spring curve.

Derivation: blend_int_cur : int_cur : subtrans_object : subtype_object :

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	int_cur	Generic int_cur data
	ctrl	if_cond	if save_version_number is less than INTCURVE_VERSION
	sv id	bs3_curve_def	cur_data for curve
	prim	real	fit tolerance
	prim	newline	
	sv id	surface type	surface 1
	prim	newline	
	sv id	surface type	surface 2
	prim	newline	
	sv id	bs2_curve_def	pcurve 1
	prim	newline	
	sv id	bs2_curve_def	pcurve 2
	ctrl	else if_cond	else if save_version_number is less than PARCUR_VERSION
	sv id	bs3_curve_save	cur_data for curve
	prim	real	fit tolerance data
	prim	newline	
	sv id	surface type	surface 2
	prim	newline	
	sv id	surface type	surface 1
	prim	newline	
	sv id	bs2_curve_def	pcurve 2
	prim	newline	
	sv id	bs2_curve_def	pcurve 1
	ctrl	else	
	sv id	int_cur	
	ctrl	if_cond	if save_version_number is greater than or equal to PARCUR_VERSION
	prim	logical	“surf2” or “surf1”
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Defines an interpolated curve subtype which is a parameter line on a blend spline surface, for a parametrisation where the true parameter line is everywhere within fit tolerance of the spline approximation’s parameter line.

This is a copy of the par_int_cur class, just with certain functions propped up to watch out for the zero length first derivatives which can occur at the ends the curve.

All the par_int_cur members are private rather than protected, so we could not just derive the new class from par_int_cur. surf1_data (in the parent) is used for the blend surface, and surf2_data for the other surface containing this curve.

4

Component: **ACIS Kernel**

blend_spl_sur

Purpose: This class is used as a base class for all blend surfaces.

Derivation: blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, "{" or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	blend_support	Left support
sv id	blend_support	Right support
sv id	curve type	Definition curve
prim	real	Left offset
prim	real	Right offset
prim	newline	
prim	logical	"single_radius" or "two_radii"
ctrl	left_rad–>save()	
ctrl	if_cond	if there are two_radii
ctrl	right_rad–>save();	
ctrl	section	
prim	newline	
prim	interval	*u*-param range
prim	interval	support *u*-param range
prim	interval	*v*-param range
prim	integer	Closed in *u*
prim	integer	Closed in *v*
prim	newline	
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: This is an abstract class that tries to predict some of the fields that derived classes will need; for example, it contains pointers for a left surface, a left curve and a left point although in practice only one of these will be needed in a particular derived class. The reason for doing this is that the base class can completely handle the administrative functions such as Save and Restore, making these trivial for the derived classes. This class name does not appear in the save file, but is a base class for other subtype identifiers that do appear in the save file.

Component: **ACIS Kernel**

"blendsupcos"

Purpose: Derived class for the curve–on–surface case.

Derivation: blend_support_curve_on_surface : blend_support : –

Data Elements:	prim	No data	This class does not save any data

Description: Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

"blendsupcur"

Purpose: Derived class for the blend support curve.

Derivation: blend_support_curve : blend_support : –

Data Elements:	prim	No data	This class does not save any data

Description: Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

"blendsuppnt"

Purpose: Derived class for the blend support point.

Derivation: blend_support_surface : blend_support : –

Data Elements:	prim	No data	This class does not save any data

Description: The blend support class is a pure base class. Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

“blendsupsur”

Purpose: Derived class for the blend support surface.

Derivation: blend_support_surface : blend_support : –

Data Elements: prim No data This class does not save any data

Description: The blend support class is a pure base class. Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

“blendsupzro”

Purpose: Derived class for the blend support zero curve.

Derivation: blend_support_zero_curve : blend_support : –

Data Elements: prim No data This class does not save any data

Description: The blend support class is a pure base class. Extends this blend support. This is for use by constant round blend surfaces. They pass the newly extended spine and whether this is the left or right support.

Component: **ACIS Kernel**

“blinfo”

Purpose: Defines attributes used internally by the blending algorithm to record intermediate results.

Derivation: ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class defines attributes used internally by the blending algorithm to record intermediate results. It derive from a common base attribute to simplify clean up.

Component: ACIS Kernel

"blndsprngcur"

Purpose: This is one long edge of a general blend surface – a spring curve.

Derivation: spring_int_cur : int_cur : subtrans_object : subtype_object :

Data Elements:

prim	subtype_start	Left curly braces, "{" or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	int_cur	Generic int_cur data
prim	logical	Curve is left edge of blend
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: surf1_data (in the parent) is used for the blend surface, and surf2_data for the other surface containing this curve.

Component: **ACIS Kernel**

"bk"

Purpose: Links surface elements to curve elements.

Derivation: ATTRIB_SURFBACK : ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:

sv id	"ms"	parent information
prim	$rec_num	Pointer to record in save file for most of the relevant link info ATTRIB_CURSURF
prim	$rec_num	Pointer to record in save file for remaining link info COMPCURV

Description: The back pointer attribute is used to link surface elements to curve elements via the curve element's ATTRIB_CURSURF, which contains most of the relevant link information.

Component: **Mesh Surface Husk**

blend_support

Purpose: Represents the geometry on which a spring curve of a blend lies.

Derivation: blend_support : –

Data Elements:	prim	ident	Type name
	sv id	surface type	Surface
	sv id	curve type	Curve
	sv id	bs2_curve_def	Parameter space curve
	prim	position	Point

Description: It will be either a surface, a curve or a point. In general, only one of these will be stored in the class, although there is room for all three to be stored. This is to allow the administrative functions such as Save and Restore to be implemented just once, in the base class.

Component: **ACIS Kernel**

“blend”

Purpose: Defines a base class for basic blend attributes from which specific blend attributes are derived.

Derivation: ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	integer	Dummy form for compatibility with old versions.

Description: Implicit blend attributes. These are attached to entities on the body to be blended, to record the information necessary for the blending operation to take place.

Component: **ACIS Kernel** Attributes

4

“bl_cr”

Purpose: Implements several trivial constant round geometries for edge-face and some face-face cases.

Derivation: ATT_BL_CR : ATT_BL_ENT_ENT : ATT_BL_ENT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	real	Blend radius

Description: The geometry-making routines are provided for development only.

Component: **Advanced Blending Husk**

“bl_ent”

Purpose: A base class for a blend on some unspecfied entities.

Derivation: ATT_BL_ENT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

prim	integer	Number of support entities
ctrl	repeat	Repeat for the number of support entities
prim	$rec_num	Pointer to record in save file for support ENTITY

Description: This base class represents a blend of some unspecified entities.

Component: **Advanced Blending Husk**

“bl_ent_ent”

Purpose: A base class for entity-entity blends.

Derivation: ATT_BL_ENT_ENT : ATT_BL_ENT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

enum	bl_convexity	text from blend convexity enumeration table
prim	logical	either “unset” or “set”
prim	position	help_pos

Description: Entity-entity blends process two spring curves in parallel to produce a chain of segments. All entity-entity blend classes are derived from this class.

Component: **Advanced Blending Husk**

“bl_inst”

Purpose: Defines the instruction attributes that hang onto the edges and vertices of a blend sequence, that instruct the blend algorithm.

Derivation: ATT_BL_INST : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

enum	transition_ents	text from blend transition enumeration table
prim	logical	either “unset” or “set” for position
prim	position	vertex position

Description:	This class defines the attributes that provide special processing instructions to the blend algorithm.
Component:	**Advanced Blending Husk**

“bl_taned”

Purpose:	Defines a variable radius entity-entity blend.
Derivation:	ATT_BL_TAN_ED : ATT_BL_VR : ATT_BL_ENT_ENT : ATT_BL_ENT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –
Data Elements:	prim No data This class does not save any data
Description:	Attribute for defining a variable radius entity-entity blend which is tangent ot one surface along an edge and either goes through the other surface or is tangent to it.
Component:	**Advanced Blending Husk**

“bl_vr”

Purpose: Defines a variable radius entity-entity blend.

Derivation: ATT_BL_VR : ATT_BL_ENT_ENT : ATT_BL_ENT : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

sv id	curve (2) class	refer to curve object
sv id	var_radius	refer to var_radius object, left object
prim	newline	
prim	boolean	if same_radii
ctrl	if_cond	if the radii are not the same
sv id	var_radius	refer to var_radius, right object
prim	newline	
sv id	var_cross_section	refer to var_cross_section object
prim	newline	

Description: Attribute for defining a variable radius entity-entity blend. Very similar to the constant radius version, but we have a defining curve and a radius function instead of a single radius value.

Component: **Advanced Blending Husk**

“body”

Purpose: Represents a wire, sheet, or solid body.

Derivation: BODY : ENTITY : –

Data Elements:

ctrl	if_cond	if the save_version_number less than the LUMP_VERSION
ctrl	if_cond	if the lump is NULL
prim	$NULL	Pointer to NULL
ctrl	else	if the lump is not NULL
prim	$rec_num	Pointer to record in save file for first lump’s shell in body
ctrl	else	if the save_version_number greater than or equal to the LUMP_VERSION
prim	$rec_num	Pointer to record in save file for first lump’s shell in body
prim	$rec_num	Pointer to record in save file for first wire in body
prim	$rec_num	Pointer to record in save file for body transform

Description: A BODY models a wire, sheet, or solid body (possibly several disjoint bodies treated as one). The body contains zero or more wires, zero or more lumps, and a transform.

A pure *wire body* contains only wires, but no lumps, shells, or faces. Wires can represent isolated points, open or closed profiles, and also general wireframe models that are unpurified; i.e., have no faces.

A *lump body* represents partially-surfaced wireframes, sheets and solids. In a partially-surfaced wireframe, edges will border on zero, one or two faces, whereas edges in a sheet will meet one or two (or more) faces. In a manifold solid, every edge is adjacent to two faces and every vertex one group of adjacent faces (a nonmanifold solid has edges adjacent to more than two faces or more than one set of edge-adjacent faces at a vertex).

In principle, a body can contain both wire and lump components. Wire and lump components should not be connected to one another (in other words, there should be only one path from the body to any edge – via a wire or lump, but not both).

A solid is represented by the boundaries of the regions of space that it encloses. Each solid is made up of one lump or a group of disjoint lumps.

Lumps and wires are described in a local coordinate system. The local coordinate system is related to the global coordinate system by a transform stored with the body.

Component: **ACIS Kernel** Topology

bounded_curve

Purpose: Defines a bounded curve.

Derivation: bounded_curve :–

Data Elements: prim No data This class does not save any data

Description: This class defines bounded curves. A bounded curve is a curve with a start and end parameters that specify the bounds of the curve. This class makes it easy to extract data from wireframe geometry. This class supports most of the functions, such as evaluation, curve length, etc., that are provided in the curve class.

Component: ACIS Kernel

"bulletin"

Purpose: Holds the changes to a single ACIS entity.

Derivation: BULLETIN : –

Data Elements:

prim	$rec_num	old entity model
prim	$rec_num	new entity model

Description: Maintains information relating to creation, modification, or deletion of an entity.

Component: History and Roll Back

"bulletin_board"

Purpose: Holds all bulletins resulting from an API call.

Derivation: BULLETIN_BOARD : –

Data Elements:	prim	$rec_num	owner pointer
	prim	integer	status number
	ctrl	repeat	Repeat for all bulletins in bulletin board
	prim	integer	1: there is a next bulletin
	prim	newline	
	sv id	"bulletin"	save the individual bulletin
	prim	integer	0: there is no next bulletin

Description: When an entity in the ACIS model is created, altered, or deleted, a bulletin recording the data structure change is added to a bulletin board.

Component: History and Roll Back

bs2_curve_def

Purpose: Defines a 2D B-spline curve.

Derivation: bs2_curve_def : –

Data Elements:	prim	ident	Type ("nullbs" "nurbs" or "nubs")
	ctrl	if_cond	if the type is set to "nullbs" No curve saved
	ctrl	if_cond	if the type is not set to "nullbs"
	prim	integer	Degree
	prim	ident	Closure ("open" "closed" or "periodic")
	prim	integer	Number of unique knot values
	ctrl	repeat	Repeat for the number of unique knot values
	prim	real	Knot value
	prim	integer	Knot multiplicity
	ctrl	repeat	Repeat for the (sum(Knot multiplicity)+1–degree)
	prim	real	*x*-coordinate of control point
	prim	real	*y*-coordinate of control point
	prim	if_cond	if the type is set to "nullbs"
	prim	real	Weight

Description: Definition of a 2D B-spline curve, used to approximate to true curves in the parameter space of a surface. This provides insulation between the modeler and the underlying spline library.

Component: **ACIS Kernel**

bs3_curve_def

Purpose: Provides an interface between ACIS and the underlying spline library.

Derivation: bs3_curve_def : –

Data Elements:

prim	ident	Type (“nullbs”, “nurbs”, or “nubs”)
ctrl	if_cond	if the type is set to “nullbs” No curve saved
ctrl	if_cond	if the type is not set to “nullbs”
prim	integer	Degree
prim	ident	Closure (“open” “closed” or “periodic”)
prim	integer	Number of unique knot values
ctrl	repeat	Repeat for the number of unique knot values
prim	real	Knot value
prim	integer	Knot multiplicity
ctrl	repeat	Repeat for (sum(Knot multiplicity)+1–degree)
prim	real	*x*-coordinate of control point
prim	real	*y*-coordinate of control point
prim	real	*z*-coordinate of control point
prim	if_cond	if the type is set to “nullbs”
prim	real	Weight

Description: This simply provides a definition for a spline curve type for the major part of ACIS to use.

Component: **ACIS Kernel**

4

bs3_surface_def

Purpose: Provides an interface between ACIS and the underlying spline library.

Derivation: bs3_surf_def : –

Data Elements:	prim	ident	Type ("nullbs" "nurbs" or "nubs")
	ctrl	if_cond	if the type is set to "nullbs" No surface saved
	ctrl	if_cond	if the type is not set to "nullbs"
	prim	integer	*u*-degree
	prim	integer	*v*-degree
	ctrl	if_cond	if the type is set to "nurbs"
	prim	ident	Rational ("u" "v" or "both")
	prim	ident	*u* Closure ("open" "closed" "periodic" or "unknown")
	prim	ident	*v* Closure ("open" "closed" "periodic" or "unknown")
	prim	ident	*u* Singularities ("none" "low" "high" or "both")
	prim	ident	*v* Singularities ("none" "low" "high" or "both")
	prim	integer	Number unique *u*-knots
	prim	integer	Number unique *v*-knots
	ctrl	repeat	Repeat for the number unique *u*-knots
	prim	real	*u*-knot value
	prim	integer	Knot multiplicity
	ctrl	repeat	Repeat for the number unique *v*-knots
	prim	real	*v*-knot value
	prim	integer	Knot multiplicity
	ctrl	repeat	Repeat for the number (sum(*v*-Knot multiplicity)+1–*v* degree)
	ctrl	repeat	Repeat for the number (sum(*u*-Knot multiplicity)+ 1–*u* degree)
	prim	real	*x*-coordinate of control point
	prim	real	*y*-coordinate of control point
	prim	real	*z*-coordinate of control point
	prim	if_cond	if the type is set to "nurbs"
	prim	real	Weight

Description: This simply provides a definition for a spline surface type for ACIS to use.

Component: **ACIS Kernel**

4

"camera"

Purpose: Used to save camera data in an ACIS part file.

Derivation: ACIS_CAMERA : ENTITY : –

Data Elements:	prim	position	camera position
	prim	position	target
	prim	vector	up vector
	prim	real	view width
	prim	real	view height
	prim	real	near clipping plane
	prim	real	far clipping plane
	prim	real	center offset *x*
	prim	real	center offset *y*
	prim	real	image scale
	prim	boolean	perspective

Description: An ACIS_CAMERA ENTITY is used to save camera data in an ACIS part file.

Component: Graphic Interaction Husk

“cell”

Purpose: Attaches the subportion of a lump.

Derivation: CELL : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for next cell
	prim	$rec_num	Pointer to record in save file for lump containing cell
	prim	$rec_num	Pointer to record in save file for supercell containing cell
	prim	boolean	Validity data

Description: The CELL is an abstract class for CELL2D and CELL3D. It provides the data and member functions for the list pointers, the lump back pointer, the supercell pointer, and the bounding box. It should never be instanced.

The cell is a minimal connected subportion of a lump. Usually a lump consists of a single cell,which is bounded by an outer cshell and zero, one, or more inner cshells representing voids within the cell. However, when an operation that leaves internal faces in a lump returns a lump with fully enclosed internal regions, each region is represented by a cell.

Component: Cellular Topology Husk

“cell2d”

Purpose:	Connects the faces of a sheet.		
Derivation:	CELL2D : CELL : ENTITY : –		
Data Elements:	prim	$rec_num	Pointer to record in save file for first CFACE in cell
Description:	The cell2d is a maximal connected sheet. The CELL2D is a set of faces that are DOUBLE-SIDED and BOTH-OUTSIDE. It is directly descended from CELL.		
Component:	Cellular Topology Husk		

“cell3d”

Purpose:	Connects subportion of a lump.		
Derivation:	CELL3D : CELL : ENTITY : –		
Data Elements:	prim	$rec_num	Pointer to record in save file for first CSHELL in cell
Description:	The CELL3D is a three-dimensional sub-portion of a lump bounded by SINGLE-SIDED or DOUBLE-SIDED BOTH-INSIDE faces. Usually a lump consists of a single cell that is bounded by an outer cshell and zero, one or more inner cshells representing voids within the cell. However, when an operation that leaves internal faces in a lump returns a lump with fully enclosed internal regions, each region is a three-dimensional cell.		
Component:	Cellular Topology Husk		

4

“cell_ptr”

Purpose:	Implements an attribute describing a cell in the Cellular Topology Husk.		
Derivation:	ATTRIB_CELL : ATTRIB_CT : ATTRIB : ENTITY : –		
Data Elements:	prim	$rec_num	Pointer to record in save file for cell list
	prim	$rec_num	Pointer to record in save file for supercell list
	prim	boolean	Automatic update

Description:	The ATTRIB_CELL attribute hangs from a lump and contains a pointer to the list of cells for that lump. It can optionally contain a list of supercells (a cell box hierarchy). An auto_update flag indicates to the automatic cell re-computation mechanism whether or not to update the cell data of the attribute. Each lump may have at most one ATTRIB_CELL.
Component:	Cellular Topology Husk

"cface"

Purpose: A reference to one *side* of a FACE.

Derivation: CFACE : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for next CFACE in CELL2D or CSHELL
prim	$rec_num	Pointer to record in save file for owning CELL2D of CSHELL
prim	$rec_num	Pointer to record in save file for FACE that defines CFACE
prim	integer	Orientation of CFACE to face normal; REVBIT portion of integer

Description: A CFACE is a reference to one *side* of a face. If a face is double-sided, it should be pointed to by two cfaces, if it is single-sided, one cface uses it in the forward direction.

Component: Cellular Topology Husk

4

"cface_ptr"

Purpose: Defines an attribute that records two CFACE entities that refer to a given face.

Derivation: ATTRIB_FACECFACE : ATTRIB_CT : ATTRIB : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for front face
prim	$rec_num	Pointer to record in save file for back face

Description: Defines an attribute that is attached to a face and records at most two cface entities that refer to its front and back sides.

Component: Cellular Topology Husk

“cface_col_att”

Purpose: Assigns a color to a volume.

Derivation: ATTRIB_VOL_COL : ATTRIB_CFACE_VOL : ATTRIB_CT : ATTRIB : ENTITY : –

Data Elements:

enum	color	Color enumeration

Description: The ATTRIB_VOL_COL attribute is derived from ATTRIB_CFACE_VOL. This attribute assigns a color to a volume, and resides on all CFACEs of a three-dimensional cell. It also participates in CFACE volume attribute propagation.

Component: Cellular Topology Husk

“cface_vol_att”

Purpose: Used for volume attribute propagation member functions.

Derivation: ATTRIB_CFACE_VOL : ATTRIB_CT : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: Serves as a base class for attributes which should be propagated with volume modifications

Component: Cellular Topology Husk

“chamfer”

Purpose: Defines a flat chamfer blend.

Derivation: ATTRIB_CHAMFER : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

prim	real	Left range
prim	real	Right range

Description: This class defines a flat chamfer blend. An edge chamfer is a ruled surface between two spring curves, one on each lateral surface. Each spring curve is the perpendicular projection of a spine curve on to the appropriate lateral surface. The spine curve is the intersection of the two lateral surfaces offset by constant distances, which may be different for the two surfaces.

Component: **ACIS Kernel** Attributes

“coedge”

Purpose: Relates EDGEs with adjacent EDGEs and owning ENTITYs.

Derivation: COEDGE : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for next coedge in loop or wire
prim	$rec_num	Pointer to record in save file for previous coedge in loop or wire
prim	$rec_num	Pointer to record in save file for next coedge on edge
prim	$rec_num	Pointer to record in save file for edge on which coedge lies
prim	integer	Direction of coedge with respect to the edge enumeration
ctrl	if_cond	if the save_version_number is less than the WIREBOOL_VERSION
prim	$rec_num	Pointer to record in save file for shell owned by a wire. If shell also contains faces, then old_style data structures had the coedge owned by the shell, and no wire.
ctrl	else	if the save_version_number greater than or equal to the WIREBOOL_VERSION
prim	$rec_num	Pointer to record in save file for loop or wire to which coedge belongs
prim	$rec_num	Pointer to record in save file for parameter space curve or geometry

Description: A COEDGE stores the relationships of the EDGE with adjacent EDGEs and superior owning ENTITYs. The data structures formed by the COEDGE pointers and their interpretation depends upon the nature of the owning ENTITY.

When the EDGE is part of a well-formed (manifold) solid body SHELL, it is adjacent to exactly two FACES. There are two COEDGEs, each associated with a LOOP in one FACE. In principle, the two FACEs could be the same, and even the LOOPs could be the same. All COEDGEs in a LOOP are linked into a doubly-linked list through next COEDGE and previous COEDGE pointers. The two COEDGEs for each EDGE are linked through their partner pointers (next coedge on edge). Several extensions to this arrangement are possible:

– A LOOP that is not closed has either a partially-defined or infinite FACE boundary. In this case, the next and previous lists are not circular, but terminate with NULL pointers.

– A SHELL that is not closed has free EDGEs at its boundary. For such EDGEs, there is only one COEDGE, with a NULL partner pointer.

– Nonmanifold SHELLs in which more than two FACEs meet in an EDGE result in the partner pointers for the COEDGEs (one for each FACE) being linked in a circular list.

– A WIRE body can be an object that may be a directed or undirected graph, made up of one or more disjoint WIREs, each of which is a collection of connected EDGEs. In this case, each EDGE has only one COEDGE, and COEDGEs are linked in circular lists around each VERTEX, using next and previous pointers according to which end of the COEDGE lies at the VERTEX.

– A SHELL can be of mixed dimensionality, containing both FACEs and unembedded EDGEs. The unembedded EDGEs are connected together as in WIREs, and are connected to any FACEs sharing their VERTEXes in a similar way. An additional rule is that in each FACE, the next list always yields the LOOP of EDGEs embedded in the FACE – any unembedded EDGE is connected via the previous pointer of one of the face COEDGEs. The unembedded EDGE can be reached from the FACEs through the VERTEX’s EDGE list of which the unembedded EDGE is a member.

Component: **ACIS Kernel** Topology

“coinvert”

Purpose: Creates a coincident vertex attribute for stitching.

Derivation: ATTRIB_COINVERT : ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: For internal use only.

Component: **ACIS Kernel** Attributes

“colour”

Purpose: User attribute class definition for a color.

Derivation: ATTRIB_COL : ATTRIB_TSL : ATTRIB : ENTITY : –

Data Elements:	enum	color	Color enumeration for entity display

Description: Implementation of color attribute. This is attached to body, face or edge objects, and is used to demonstrate attribute migration, as well as to produce prettier pictures in the ACIS test harness.

Component: **ACIS Kernel** Attributes, Graphic Interaction Husk, and ACIS Test Harness

“comment”

Purpose: Constructs a PTC attribute.

Derivation: comment_attr_c : ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements:	prim	integer	Data value
	ctrl	if_cond	if the save_version_number is less than the CONSISTENT_VERSION
	prim	integer	Number of characters
	ctrl	repeat	Repeat for the number of characters
	prim	integer	One character of comment (one char per value)
	ctrl	else	if the save_version_number is greater than or equal to the CONSISTENT_VERSION
	prim	ident	Whole comment string

Description: Include file for ACIS version of Pro Engineer™ text attributes.

Component: **ACIS Kernel** Attributes

4

com_cur

Purpose: Represents of a patchwork curve used to represent boundaries of a meshsurf, which is a patchwork surface.

Derivation: com_cur :–

Data Elements:	prim	logical	“forward” or “reversed”
	prim	newline	
	prim	integer	count of elements
	prim	integer	count of attributes
	prim	logical	“open” or “closed”
	prim	newline	
	sv id	“p1”	start node for that element, P1NODE
	ctrl	repeat	repeat until finished with elements
	sv id	“1d”	current element for class ELEM1D
	prim	newline	
	sv id	“p1”	end node for that element, P1NODE
	prim	newline	
	ctrl	repeat	repeat until finished with attributes
	sv id	“ms”	current attribute for class
	sv id	“cs”	current attribute for class
	sv id	curve (2) class	curve element

Description: This class defines the header for compcurv, which is the general representation of a patchwork curve used to represent boundaries of a meshsurf, which is a patchwork surface. Separate edges and vertices could have been used to represent this data, but the compcurv class is implemented to save memory, increase execution speed, represent parameter lines, and store attributes that refer to corresponding nodes and elements on the meshsurfs that they lie in.

Component: ACIS Kernel

“compcurv”

Purpose: Records a composite curve as a (lowercase) compcurv.

Derivation: COMPCURV : CURVE : –

Data Elements: sv id compcurv (2) class start node for that element, P1NODE

Description: The COMPCURV class records a composite curve as a (lowercase) compcurv. The compcurv holds the arrays of coordinate nodes and elements (segments) that define a piecework curve.

Like other surface and curve types, COMPCURV contains a lowercase compcurv class object. Aside from the usual surface and curve entity class members, it has member functions to provide access to the nodes and elements that make up the surface and a binary tree of curve or surface bounding boxes.

The COMPCURV class contains a few specialized functions to help the Boolean operations process new composite curve intersections on mesh surfaces. The constructor is also specialized to process the CURVE back pointers that surface elements can contain when, for example, a curve is split.

Component: ACIS Kernel

compcurv (2) class

Purpose: Records a composite curve as a (lowercase) compcurv.

Derivation: COMPCURV : CURVE : –

Data Elements:

ctrl	if_cond	if save_version_number is less than the MESH_VERSION, there is an error
prim	logical	"forward" or "reverse"
sv id	com_cur	save the com_cur
sv id	curve (2) class	save the curve data

Description: The COMPCURV class records a composite curve as a (lowercase) compcurv. The compcurv holds the arrays of coordinate nodes and elements (segments) that define a piecework curve.

Like other surface and curve types, COMPCURV contains a lowercase compcurv class object. Aside from the usual surface and curve entity class members, it has member functions to provide access to the nodes and elements that make up the surface and a binary tree of curve or surface bounding boxes.

The COMPCURV class contains a few specialized functions to help the Boolean operations process new composite curve intersections on mesh surfaces. The constructor is also specialized to process the CURVE back pointers that surface elements can contain when, for example, a curve is split.

Component: **Mesh Surface Husk**

4

"cone"

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
sv id	CONE (1) class	derived from CONE class
ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
sv id	cone (2) class	derived from cone class

Description: Used to determine which class specified the cone. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

CONE (1) class

Purpose: Defines a cone as an object in the model.

Derivation: CONE : SURFACE : ENTITY : –

Data Elements: sv id cone (2) class Cone data definition given in another section of this manual.

Description: A CONE is defined by the base ellipse and the sine and cosine of the major half-angle.

Component: **ACIS Kernel** Geometry

cone (2) class

Purpose: Defines the elliptical single cone.

Derivation: cone : surface : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
prim	integer	cone_type; integer for type of cone
prim	integer	ellipse_type; integer for type of ellipse
ctrl	else	if save_version_number is greater than the SURFACE_VERSION
prim	string	save identifier; "cone".
sv id	ellipse (2) class	Base ellipse
prim	real	Sine of cone angle
prim	real	Cosine of cone angle
ctrl	if_cond	if the save_version_number is greater than or equal to the SURFACE_VERSION
prim	logical	*u* parameter reversed, either "forward" or "reversed"
sv id	surface (2) class	Generic surface data given in another section of this manual
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: The cone class defines an elliptical single cone. It is defined by a base ellipse and the sine and cosine of the major half-angle of the core. The normal of the base ellipse represents the axis of the cone.

As special cases, the cross-section may be circular, the *cone* may be a cylinder, or both.

The polarity (sign) of the trigonometric functions define the slant of the surface of the cone and the sense of the surface.

- If sine_angle has different polarity than cosine_angle, the cross-section decreases in the direction of the axis of the cone (ellipse surface normal).
- If sine_angle has the polarity as cosine_angle, the cross-section increases in the direction of the axis of the cone (ellipse surface normal).
- If cosine_angle is positive (+), the sense of the surface is away from the axis of the cone (surface is convex).
- If cosine_angle is negative (–), the sense of the surface is toward the axis of the cone (surface is concave).
- If sine_angle is identically zero (sine_angle == 0), the cone is a cylinder.
- If cosine_angle is identically zero (cosine_angle ==0), the cone is planar.

4

The surface stops at the apex, if any; i.e., this surface type does not represent a double cone.

The *u*-parameter direction is along the generators of the cone, with zero representing the intersection of the generator with the base ellipse, and parameter increasing in the direction of the cone axis; i.e., the normal of the base ellipse, if reverse_u is FALSE, and in the opposite direction if reverse_u is TRUE. The *v*-parameter direction is along a cross-sectional ellipse clockwise around the cone axis, parameterized as for the base ellipse.

To evaluate the position corresponding to a given *uv* pair, first evaluate the base ellipse at parameter *v*, and subtract the center point to give vector *V*. Let *s* and *c* be sine_angle and cosine_angle if cosine_angle is positive, or –sine_angle and –cosine_angle if not. Let *R* be the length of the major axis of the base ellipse, negated if reverse_u is TRUE. Then:

```
pos = base.center + (1 + s*u)* V + c*u*R*base.normal
```

This parameterization is left-handed for a convex cone (cosine_angle > 0) with reverse_u FALSE or for a concave cone with reverse_u TRUE, and right-handed otherwise.

When the cone is transformed, the sense of reverse_u is inverted if the transform includes a reflection. A negation requires no special action.

In summary, cones are:

– Not true parametric surfaces.
– Are closed in *v* but not in *u*.
– Periodic in *v* (–pi to pi with period 2pi) but not in *u*.
– Singular in *u* at the apex; nonsingular for all other *u* and *v* values.

4

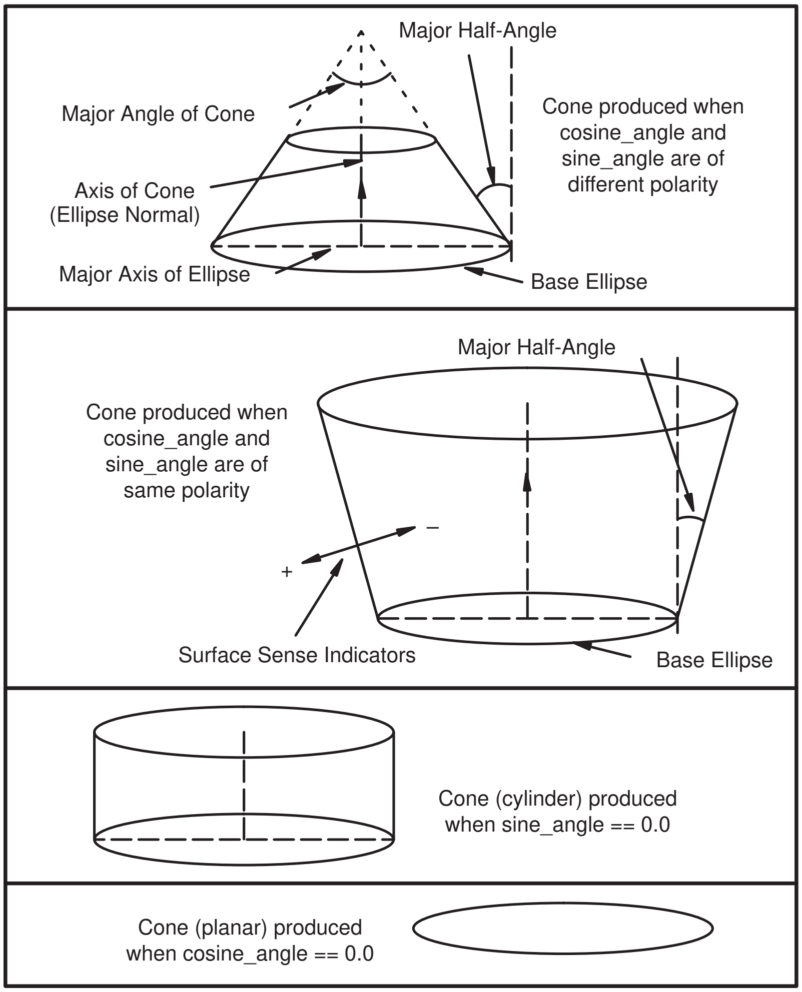

Figure 4-1. cone (2) class Definition

Component: **ACIS Kernel** Geometry

“const_chamfer”

Purpose: Defines a flat chamfer blend.

Derivation: ATTRIB_CONST_CHAMFER : ATTRIB_FFBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	real	Left range
	prim	real	Right range

Description: This class defines a flat chamfer blend. An edge chamfer is a ruled surface between two spring curves, one on each lateral surface. A spring curve is found by projecting the spine curve to each lateral face. The spine curve is the intersection of the two lateral surfaces offset by constant distances chosen to give the desired ranges which may be different for the two surfaces.

Component: **ACIS Kernel** Attributes

"const_round"

Purpose: Defines a circular rolling-ball blend of constant radius.

Derivation: ATTRIB_CONST_ROUND : ATTRIB_FFBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	real	Radius

Description: ATTRIB_CONST_ROUND is the blend attribute for constant radius. This attribute records a circular, rolling ball blend of constant radius.

Component: **ACIS Kernel** Attributes

"crvcrvblndsur"

Purpose: Implements the variable-radius edge-edge blend surface.

Derivation: crv_crv_v_bl_spl_sur : var_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	No data	This class does not save any data

Description: This class implements the surface geometry of a variable-radius blend between the curves. The blend interpolates both curves, with no tangency constraints.

Component: ACIS Kernel

"crv_cstrn"

Purpose: Curve constraints.

Derivation: ATTRIB_CRV_CSTRN : ATTRIB_DSCSTRN : ATTRIB : ENTITY : –

Data Elements:	prim	long		numerical accuracy of integration along the curve.
	ctrl	if_cond		if (acc_domain_crv_save)
	prim		long	acc_domain_crv_save–>Type_id( )
	ctrl		acc_domain_crv_save–>Save_def( )	
	ctrl	else		
	prim		long	–2: special flag

Description: For internal use only.

Component: **Deformable Surface Husk**

“crv_load”

Purpose: Curve loads.

Derivation: ATTRIB_CRV_LOAD : ATTRIB_DSLOAD : ATTRIB : ENTITY : –

Data Elements:	ctrl	if_cond		if (acl_domain_crv_save)
	prim		long	acl_domain_crv_save–>Type_id( )
	ctrl		acl_domain_crv_save–>Save_def( )	
	ctrl	else		
	prim		long	
	ctrl	if_cond		if (acl_src_pfunc_save)
	prim		long	acl_src_pfunc_save–>Type_id( )
	ctrl		acl_src_pfunc_save–>Save_def( )	
	ctrl	else		
	prim		long	

Description: For internal use only.

Component: **Deformable Surface Husk**

4

“crvsrfblndsur”

Purpose: Implements the variable-radius edge-face blend surface.

Derivation: crv_srf_v_bl_spl_sur : var_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements: prim No data This class does not save any data

Description: This class implements the surface geometry of a variable radius blend between a curve and a surface. The blend will be tangent to the surface, but have no tangency constraints where it interpolates the curve.

Component: ACIS Kernel

“cs”

Purpose: Implements the mesh curve and surface link attributes.

Derivation: ATTRIB_CURSURF : ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:

sv id	“ms”	parent information
prim	$rec_num	Pointer to record in save file for surface MESHSURF
prim	$rec_num	Pointer to record in save file for partner attribute on surface ATTRIB_SURFBACK
prim	$rec_num	Pointer to record in save file for starting node
prim	$rec_num	Pointer to record in save file for ending node
prim	integer	start side data
prim	integer	end side data
prim	integer	coincident side data

Description: The mesh curve and surface link attributes.These attributes interconnect mesh surface and curve elements, which are required to maintain their compatibility at all times. (Each surface node must map into a curve node and each curve element must fully span one element [or two if on their shared boundary]). The MESHSURF pointer in the attribute is used to obtain access to the higher level topology, which will be necessary during save, restore or copy.

Component: **Mesh Surface Husk**

“cshell”

Purpose: One connected portion of a cell’s boundary.

Derivation: CSHELL : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for next CSHELL in CELL3D
prim	$rec_num	Pointer to record in save file for first CFACE in CSHELL
prim	$rec_num	Pointer to record in save file for owning CELL3D

4

Description: The CSHELL is one portion of a cell’s boundary; it is in a different shell from any other CSHELL in that cell. If a cell has no voids, then exactly one CSHELL gives its overall extent. Any other CSHELLs bound voids wholly within the cell. There is no distinction made in the data structure between peripheral and void CSHELLs, indeed, some CSHELLs may be open; e.g., a single face floating within a cell.

A CSHELL is constructed from a collection of CFACEs, each of which is one side of an existing face.

Component: Cellular Topology Husk

“ct”

Purpose: Defines a base attribute for the Cellular Topology Husk.

Derivation: ATTRIB_CT : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: Defines a base attribute for the Cellular Topology Husk. All Cellular Topology Husk attributes are derived from this class.

Component: Cellular Topology Husk

“curve”

Purpose: Represents a variety of curve geometries as an object in the model.

Derivation: CURVE : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: A CURVE provides the basic framework for the range of curve geometries implemented in the modeler. CURVEs may be used by more than one ENTITY.

Component: **ACIS Kernel** Geometry

curve (2) class

Purpose: Derives specific data used for the curve classes.

Derivation: curve : –

Data Elements:	ctrl	if_cond	if the save_version_number is greater than or equal to the BNDCUR_VERSION
	prim	interval	subset range

Description: The curve class is a base class from which all specific geometry classes (straight, ellipse, and intcurve) are derived. Considered as a function of its parameter, a curve is assumed to have a continuous first derivative whose length is bounded above and below by nonzero constants. There is no hard and fast rule about the values of these bounds or about the rate of change of the length of the derivative.

Component: **ACIS Kernel** Geometry

curve type

Purpose: Defines more specifically the type of curves.

Derivation: curve : –

Data Elements:	prim	ident	Curve type
	ctrl	if_cond	if Curve type is set to “null_curve” No curve saved
	ctrl	if_cond	if Curve type is set to “ellipse”
	sv id	ellipse (2) class	Ellipse definition
	ctrl	if_cond	if Curve type is set to “intcurve”
	sv id	intcurve (2) class	Intcurve definition
	ctrl	if_cond	if Curve type is set to “straight”
	sv id	straight (2) class	Straight definition
	ctrl	if_cond	if Curve type is set to “undefc”
	sv id	undefc (2) class	Undefc definition

Description: Refer to purpose.

Component: **ACIS Kernel** Geometry

4

“d5c2_cur”

Purpose: Defines an interpolated curve subtype which is a subtype of an int_cur.

Derivation: skin_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	int_cur	save the interpolated curve data
	prim	integer	Number of parameters
	prim	newline	New line for readablility in sat file
	ctrl	repeat	repeat for the number of parameters
	prim	real	specific parameter
	prim	newline	New line every fifth parameter
	ctrl	repeat	repeat for the number of parameters
	prim	real	specific parameter point *x*
	prim	real	specific parameter point *y*
	prim	real	specific parameter point *z*
	prim	newline	New line for readablility in sat file
	ctrl	repeat	repeat for the number of parameters
	prim	real	specific parameter first derivative at *x*
	prim	real	specific parameter first derivative at *y*
	prim	real	specific parameter first derivative at *z*
	prim	newline	New line for readablility in sat file
	ctrl	repeat	repeat for the number of parameters
	prim	real	specific parameter second derivative at *x*
	prim	real	specific parameter second derivative at *y*
	prim	real	specific parameter second derivative at *z*
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Defines an interpolated curve subtype which is a subtype of an int_cur. This subtype is constrained to be a degree 5 spline with C2 continuity. It is defined in terms of positions, first and second derivatives at series of points. This curve is used exclusively as a temporary (working) geometry object, and should not appear in a BODY.

Component: **ACIS Kernel**

“delta_state”

Purpose: Holds all bulletin boards resulting from modification to a model.

Derivation: DELTA_STATE : –

Data Elements:	prim	integer	Number for this state
	prim	integer	Number of states that can be rolled back
	prim	integer	hidden
	prim	$rec_num	previous delta state with respect roll back
	prim	$rec_num	next delta state with respect to roll back
	prim	$rec_num	partner delta state; if no branches, points to itself.
	prim	$rec_num	delta state this is merged with
	prim	$rec_num	owner stream
	prim	ident	writes a name string or NULL
	ctrl	repeat	repeat for the number of bulletin boards
	prim	integer	1: a next bulletin board follows
	prim	newline	
	sv id	"bulletin_board"	save bulletin board information
	prim	integer	0: no more bulletin boards follow
	prim	newline	
	ctrl	if_cond	if there are merged_states
	prim	integer	number of iteration counts
	ctrl	repeat	Repeat for all merged states
	prim	$rec_num	merged state
	ctrl	else	
	prim	integer	0: no merged states
	prim	terminator	no more delta states

Description: A modification to a model may involve multiple API calls, which means multiple bulletin boards.

Component: History and Roll Back

"dict_entry"

Purpose: Stores names and their corresponding entities in an unordered linked list.

Derivation: DICT_ENTRY : ENTITY_TSL : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: Simple name dictionary for the ACIS testbed. Simply stores names and their corresponding entities in an unordered linked list. Dictionary entries get logged and rolled back or forth with model. The name is placed in free store and a pointer to it recorded in the dictionary entry. Whenever the entry is copied for backup, the name is copied too, so is only used by one entity at a time.

Component: ACIS Test Harness

“display_attribute”

Purpose: Defines DISPLAY_ATTRIB attribute to link ENTITYs with their display.

Derivation: DISPLAY_ATTRIB : ATTRIB_ST : ATTRIB : ENTITY : –

Data Elements:	prim	integer	display revision

Description: The class rendering_context_data is used in a DISPLAY_ATTRIB to link an ENTITY with its data in a specific rendering_context. An ENTITY can have more than one DISPLAY_ATTRIB attached to it. In fact, it will usually have one DISPLAY_ATTRIB for each rendering_context in which the ENTITY is displayed. The DISPLAY_ATTRIB serves to keep the display of an ENTITY up to date with its definition.

Component: Graphic Interaction Husk

“dispose”

Purpose: Makes a face disposal attribute.

Derivation: ATTRIB_DISPOSE : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: For internal use only.

Component: **ACIS Kernel** Attributes

4

“dist_press”

Purpose: Distributed pressure.

Derivation: ATTRIB_DIST_PRESS : ATTRIB_DSLOAD : ATTRIB : ENTITY : –

Data Elements:	prim	long	apr_domain_dim
	ctrl	repeat	Repeat for all apr_domain_dim
	prim	real	apr_domain_min[ii]
	prim	real	apr_domain_max[ii]

Description: For internal use only.

Component: **Deformable Surface Husk**

“ds_2acis”

Purpose: Connects the deformable surface functions to the ACIS modeler.

Derivation: ATTRIB_DS2ACIS : ATTRIB_AGC : ATTRIB : ENTITY : –

Data Elements:

prim	long	d2a_draw_state
prim	real	d2a_surfC.red( )
prim	real	d2a_surfC.green( )
prim	real	d2a_surfC.blue( )
prim	real	d2a_cptC.red( )
prim	real	d2a_cptC.green( )
prim	real	d2a_cptC.blue( )
prim	real	d2a_cstrn_onC.red( )
prim	real	d2a_cstrn_onC.green( )
prim	real	d2a_cstrn_onC.blue( )
prim	real	d2a_cstrn_offC.red( )
prim	real	d2a_cstrn_offC.green( )
prim	real	d2a_cstrn_offC.blue( )
prim	real	d2a_loadC.red( )
prim	real	d2a_loadC.green( )
prim	real	d2a_loadC.blue( )
prim	long	d2a_nu
prim	long	d2a_nv
prim	long	d2a_tag_count
prim	real	d2a_comb_gain
prim	long	d2a_comb_pt_count

Description: The ATTRIB_DS2ACIS class encapsulates the programmable interface to deformable modeling. Using its methods, application programs can create, sculpt, and save deformable models. In addition, its methods support the API functions for managing the movement of data between ACIS FACE objects and deformable modeling objects.

This does not save any derived data – make sure those pointers are NULL because all the pointers in ATTRIB_DS2ACIS save all the graphics data

Component: **Deformable Surface Husk**

4

“ds_cstrn”

Purpose: Deformable surface constraint.

Derivation: ATTRIB_DSCSTRN : ATTRIB_AGC : ATTRIB : ENTITY : –

Data Elements:

prim	long	acs_row_count
prim	long	acs_tag

Description: For internal use only.

Component: **Deformable Surface Husk**

“ds_load”

Purpose: Deformable surface load.

Derivation: ATTRIB_DSLOAD : ATTRIB_AGC : ATTRIB : ENTITY : –

Data Elements:

prim	real	ald_gain
prim	long	ald_tag

Description: For internal use only.

Component: **Deformable Surface Husk**

“ds_model”

Purpose: Deformable surface model.

Derivation: ATTRIB_DSMODEL : ATTRIB_AGC : ATTRIB : ENTITY : –

Data Elements:

prim	real	adm_a11
prim	real	adm_a12
prim	real	adm_a22
prim	real	adm_b11
prim	real	adm_b12
prim	real	adm_b22
prim	real	adm_d11
prim	real	adm_dt
prim	real	adm_mass
prim	real	adm_damp
prim	long	adm_ntgrl_degree

Description: For internal use only.

Component: **Deformable Surface Husk**

“ecoin”

Purpose: Defines the attribute to be attached to edges of the sheet, where they are coincident with edges of the blended body.

Derivation: ATTRIB_ECOIN : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: ATTRIB_ECOIN attaches to edges of the sheet, where they are coincident with edges of the blended body.

Component: ACIS Kernel

“edge”

Purpose: Represents a physical edge.

Derivation: EDGE : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for start vertex
ctrl	if_cond	if save_version_number is less than PARAM_VERSION
prim	real	Start parameter
prim	$rec_num	Pointer to record in save file for end vertex
ctrl	if_cond	if save_version_number is less than PARAM_VERSION)
prim	real	End parameter
prim	$rec_num	Pointer to record in save file for one of the coedges lying on the edge
prim	$rec_num	Pointer to record in save file for curve on which the edge lies
prim	logical	Direction of edge with respect to curve, “forward” or “reversed”

Description: An EDGE represents a physical edge as recognized by the user. It consists of a bounded portion of a space curve, the bounds being given as object-space VERTEXes.

The VERTEX pointer at either or both ends can be NULL, in which case the EDGE is taken to be unbounded in that direction. If the underlying curve is infinite, so is the unbounded EDGE. If the curve is closed, the VERTEX pointers must both be present and represent the same data structure, or both NULL. In the latter case the EDGE is the whole curve.

As a special case, the geometry pointer can be NULL while both VERTEX pointers point to the same VERTEX. This means that the EDGE is an isolated POINT (for example the apex of a CONE).

Component: **ACIS Kernel** Topology

"efint"

Purpose: Attached to bodies during the first stage of Booleans.

Derivation: ATTRIB_EFINT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: For internal use only. EFINT attributes are attached on the bodies during the first stage of Booleans (and similar algorithms).

Component: **ACIS Kernel** Attributes

"elem"

Purpose: Implements the ELEM class.

Derivation: ELEM : ENTITY : –

Data Elements: sv id ATTRIB_MESH Pointer to attribute associated with ELEM

Description: Linked list of elements which make up a mesh surface on a compcurv.

Component: **Mesh Surface Husk**

"ellipse"

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	ELLIPSE (1) class	derived from ELLIPSE class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	ellipse (2) class	derived from ellipse class

Description: Used to determine which class specified the ellipse. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

ELLIPSE (1) class

Purpose: Defines an ellipse as an object in the model.

Derivation: ELLIPSE : CURVE : ENTITY : –

Data Elements: sv id ellipse (2) class Ellipse definition

Description: ELLIPSE class defines an ellipse by its center, a unit normal vector, major-axis vector, and a real giving the eccentricity of the ellipse; i.e., length minor-axis/length major-axis. The inherent direction of the ellipse is given by the normal using right-hand rule.

Component: **ACIS Kernel** Geometry

ellipse (2) class

Purpose: Defines an ellipse.

Derivation: ellipse : curve : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the CURVE_VERSION
prim	integer	ellipse_type; integer for type of ellipse
ctrl	else	if save_version_number is greater than the SURFACE_VERSION
prim	string	save identifier; "ellipse".
prim	position	Center
prim	vector	Normal
prim	vector	Major axis
prim	real	Radius ratio
sv id	curve (2) class	Generic curve data, given in another section of this manual
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: The ellipse class represents circles and ellipses on any plane. An ellipse is defined by a center point, a unit vector normal to the plane of the ellipse, a vector defining the major axis of the ellipse (including the magnitude of the major axis), and the radius ratio of the minor axis length to the major axis length. Currently, the length of the major axis should be greater than resfit. In a circle, the radius ratio is exactly 1.0. The direction of the ellipse is defined by the right hand rule using the normal vector. Direction is important when defining an edge supported by this geometry. An ellipse is a closed curve that has a period 2*pi and an interval range of [–pi, pi]. It is parameterized by:

```
point = center + M cos(t - offset) + N sin(t - offset);
```

where *M* and *N* are the major and minor axes respectively. The offset is stored explicitly, and is present to allow future addition of an isometric scaling that does not transform the axes into those of the transformed ellipse.

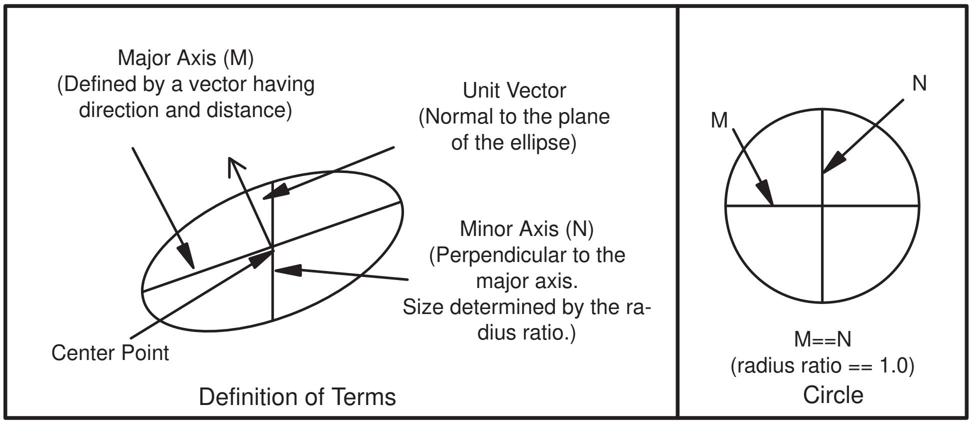

Figure 4-2. Ellipse Examples

Component: **ACIS Kernel** Geometry

ENTITY

Purpose: Represents common data and functionality that is mandatory in all classes that are permanent objects in the model.

Derivation: ENTITY : –

Data Elements: prim $rec_num Pointer to record in save file for first attribute owned by entity.

Description: All classes representing permanent objects in ACIS are derived from the ENTITY class. ENTITY does not represent any object within the modeler. Instead, ENTITY represents common data and functionality that is mandatory in all classes that represent permanent objects.

Component: **ACIS Kernel**

"entity_attrib"

Purpose: Defines a generic attribute that owns an entity.

Derivation: ATTRIB_GEN_ENTITY : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –

Data Elements: prim $rec_num Pointer to record in save file for owned entity

Description: Defines a generic attribute that owns an entity. The owned entity is copied, transformed, and lost along with the attribute's owner.

Component: Generic Attribute Husk

ENTITY_PHL

Purpose: Defines the phl tag identifier.

Derivation: ENTITY_PHL : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: The phl tag is used by two different classes.

Component: **Precise Hidden Line Husk**

"exactcur"

Purpose: Represents an exact spline curve.

Derivation: exact_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, "{" or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	int_cur	Generic int_cur data
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: The 3D spline curve representing the given curve is considered to be exact.

Component: **ACIS Kernel**

"exactsur"

Purpose: Represents an exact spline surface.

Derivation: exact_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	sv id	spl_sur	Generic spl_sur data
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: The 3D spline surface is considered to be exact.

Component: **ACIS Kernel**

“expblend”

Purpose: Implements a derived blend attribute for marking blend sheet faces to indicate their provenance.

Derivation: ATTRIB_EXPBLEND : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: ATTRIB_EXPBLEND defines an attribute to be attached to a sheet face, to record the implicit blend that gave rise to it.

Component: ACIS Kernel

“exppc”

Purpose: Defines an explicit parameter-space curve.

Derivation: exp_par_cur : par_cur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	bs2_curve_def	Parameter space curve
	prim	real	Fit tolerance
	sv id	surface type	Surface
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class, derived from the abstract base class, par_cur, handles certain operations and provides virtual functions for the rest.

Component: **ACIS Kernel**

“eye”

Purpose: A base class for classes used in the Faceter Husk.

Derivation: ATTRIB_EYE : ATTRIB : –

Data Elements:	prim	No data	This class does not save any data

Description: Refer to purpose.

Component: Faceter Husk

“eye_refinement”

Purpose: Controls the accuracy and types of polygons generated in the Faceter Husk.

Derivation: REFINEMENT : ENTITY : –

Data Elements:	prim	string	id grid mode
	enum	AF_GRID_MODE	grid mode enumeration
	prim	string	id triangular mode
	enum	AF_TRIANG_MODE	triangular mode enumeration
	prim	string	id surface mode
	enum	AF_SURF_MODE	surface mode enumeration
	prim	string	id adjust mode
	enum	AF_ADJUST_MODE	adjust mode enumeration
	prim	string	id grading mode
	prim	integer	grading mode
	prim	string	id postcheck mode
	prim	integer	postcheck mode
	prim	string	id surface tolerance
	prim	real	surface tolerance
	prim	string	id normal tolerance
	prim	integer	normal tolerance
	prim	string	id silhouette tolerance
	prim	real	silhouette tolerance
	prim	string	id flatness tolerance
	prim	real	flatness tolerance
	prim	string	id pixel area tolerance
	prim	real	pixel area tolerance
	prim	string	id max edge length
	prim	real	maximum edge length
	prim	string	id grid aspect ratio
	prim	real	grid aspect ratio
	prim	string	id max grid lines
	prim	integer	maximum grid lines
	prim	string	id end fields
	ctrl	#ifdef SAVE_OLD_FIELDS	
	prim	integer	minimum level
	prim	integer	maximum level
	prim	real	flatness tolerance
	prim	real	silhouette tolerance
	prim	real	surface tolerance
	prim	real	normal tolerance
	prim	real	pixel area tolerance
	prim	real	maximum aspect ratio
	prim	integer	maximum sides mode
	prim	real	maximum edge length
	ctrl	#endif	

Description: Each refinement can be specific to a type of surface. Admissible types are given by the surface mode enumerations.

Component: Faceter Husk

4

"face"

Purpose: Represents a bounded portion of a SURFACE.

Derivation: FACE : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for next face in shell or subshell
prim	$rec_num	Pointer to record in save file for first loop bounding face
prim	$rec_num	Pointer to record in save file for shell containing face
prim	$rec_num	Pointer to record in save file for subshell containing face
prim	$rec_num	Pointer to record in save file for surface on which face lies
prim	logical	("forward", "reversed") Direction of face normal with respect to the surface
prim	logical	("single", "double") Double sided face
ctrl	if_cond	if double sided face
prim	logical	("out", "in"), Double sided face containment.

Description: A FACE is a bounded portion of a single geometric surface in space, the two-dimensional analogue of a BODY. The boundary is represented by one or more LOOPs of EDGEs.

Although it is possible to construct a FACE with disjoint parts, (regions on the FACE surface that have no points in common) it is not recommended because the modeling algorithms in ACIS are designed to deal only with connected FACEs. Also, the LOOPs of a FACE are not distinguished as internal or external at the FACE level, because the distinction is not relevant for some surface types (e.g., a cylindrical FACE with two circular LOOPs bounding the ends).

FACE LOOPs need not be closed. If not, either open end can be finite or infinite. If either end is infinite, the FACE is infinite. If either end is finite, the FACE is incomplete.

Component: **ACIS Kernel** Topology

4

“faceint”

Purpose: Defines an attribute to record the intersection of a face of one body with a face of the other body during a Boolean operation.

Derivation: ATTRIB_FACEINT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This attribute is attached to a face of the tool body, and refers to a face of the blank body. This attribute may be attached by code that precedes a Boolean operation, such as blending.

Most Boolean operations do not bother to construct these attributes, as each face-to-face combination is looked at only once, but they may be attached before entering the Boolean code, if the intersections have already been determined by preliminary code (for example in blending).

If one of these attributes is stored, it is attached to the tool body face, and refers to the blank face.

In the special case of intersecting a body with itself (with no transformation), as may occur in testing for self-intersection, each face pair will in the normal course of events be inspected twice, and so it may be worth saving the results of the first operation. When it is seen a second time, the original tool face will be the blank, and vice versa, so a search on the blank face is needed also, and if found, the intersections must be flipped to reflect the inverted face order.

Component: ACIS Kernel

“fctd_mark”

Purpose: Marks faceted faces.

Derivation: ATTRIB_EYE_FCTD_MARK : ATTRIB_EYE : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: The ATTRIB_EYE_FCTD_MARK class is an ACIS attribute used to mark faceted faces by attaching one instance of it to each faceted face.

Component: Faceter Husk

"ffblend"

Purpose: Defines a general face-face blend.

Derivation: ATTRIB_FFBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
prim	integer	convexity
ctrl	else	
enum	bl_ed_convexity	Convexity enumeration
prim	$rec_num	Pointer to record in save file for left face
prim	$rec_num	Pointer to record in save file for right face
prim	real	Setback at start
prim	real	Setback at end
ctrl	if_cond	if save_version_number is greater than or equal to ANG_XCUR_VERSION
prim	real	Setback difference at start
prim	real	Setback difference at end
prim	boolean	Setback difference at start set
prim	boolean	Setback difference at end set
ctrl	if_cond	if save_version_number is greater than or equal to CONSISTENT_VERSION
enum	blend_how	how to blend enumeration
ctrl	else	
prim	integer	how to blend
sv id	surface type	plane: Starting plane

Description: ATTRIB_FFBLEND is a face-to-face blend, and records the blend convexity, the faces that the blend runs between, the setback at either end of the owning entity (when an edge), and the defining plane so an approximate starting position for a rolling ball can be specified.

Usually local geometry supplies these details, but if not, the user must set one or more of them directly. A blend can be of zero size but have nonzero setbacks.

Specific types of face-face blends are derived from this class.

Component: **ACIS Kernel** Attributes

“fmesh”

Purpose:	Identifies that a mesh is attached to the entity.		
Derivation:	ATTRIB_EYE_ATTACHED_MESH : ATTRIB_EYE : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	A mesh attaches to an entity through this entity. MESHes are not saved.		
Component:	Faceter Husk		

“gen”

Purpose:	Defines the organization attribute class for the Generic Attribute Husk.		
Derivation:	ATTRIB_GENERIC : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	Refer to the Purpose.		
Component:	Generic Attribute Husk		

“group”

Purpose:	Entity class that points to a list of entities.		
Derivation:	GROUP : ENTITY : –		
Data Elements:	prim	integer	Number of members
	ctrl	repeat	Repeat for the number of members
	prim	$rec_num	Pointer to record in save file for group member
Description:	A GROUP is a special entity class that points to a list of entities. It may point to entities on more than one body. Each entity it points to is required to have a back pointer through an ATTRIB_GROUP. GROUPs are useful for representing special relationships between entities.		
Component:	Cellular Topology Husk		

"group_ptr"

Purpose:	Implements an attribute that points to a list of entities.		
Derivation:	ATTRIB_GROUP : ATTRIB_CT : ATTRIB : ENTITY : –		
Data Elements:	prim	$rec_num	Pointer to record in save file for group
Description:	The ATTRIB_GROUP attribute contains a pointer to a group entity that in turn contains a list of entities in the group. Any entity containing an ATTRIB_GROUP is in the group, and the group entity list must contain it.		
Component:	Cellular Topology Husk		

"history"

Purpose:	Defines an attribute for the history stream mechanism.		
Derivation:	ATTRIB_HISTORY : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	No data is saved for ATTRIB_HISTORY, but HISTORY_STREAM remembers what attribute it was attached to and patches things up in it's fix_pointers function. This means a stream can be attached to at most one attribute.		
Component:	**ACIS Kernel** Attributes		

"history_stream"

Purpose:	Contains pointers to the initial delta state, to the last noted active delta state, and to the delta state under construction.		
Derivation:	ATTRIB : ENTITY : –		
Data Elements:	prim	integer	current state
	prim	integer	next state with respect to roll back
	prim	integer	maximum number of states to keep
	prim	$rec_num	current delta state, or state "under construction"
	prim	$rec_num	active delta state
	prim	$rec_num	root delta state
	prim	$rec_num	attributes associated with this state
	prim	terminator	indication that history information finished.

Description:	The History Manager handles the history streams.The History Stream maintains three pointers. The first pointer is to the root delta state. The second pointer is to the last noted active delta state. The third pointer is to the delta state currently being created.
Component:	History and Roll Back

“id_attribute”

Purpose:	Defines class ID_ATTRIB to allow identification of ENTITYs in a table.		
Derivation:	ID_ATTRIB : ATTRIB_ST : ENTITY : –		
Data Elements:	prim	long	entity ID
Description:	Implements class ID_ATTRIB to allow identification of ENTITYs in a table.		
Component:	Part Management Husk		

“imppc”

Purpose:	Defines an implicit parameter-space curve.		
Derivation:	imp_par_cur : par_cur : subtrans_object : subtype_object : –		
Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	curve type	Curve (intcurve) containing surface and parameter curve
	prim	boolean	Use first surface
	prim	subtype_end	Right curly braces, “}” or Tag 16
Description:	This class refers to one of the parameter-space curves of an intcurve and the corresponding surface.		
Component:	**ACIS Kernel**		

“int”

Purpose:	Used to translate mesh surfaces to and from external file formats.
Derivation:	ATTRIB_INT : ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: Refer to purpose.

Component: **Mesh Surface Husk**

“intcoed”

Purpose: Defines an attribute for linking edges and faces.

Derivation: ATTRIB_INTCOED : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
	prim	integer	face relation data integer
	ctrl	else	
	enum	face_body_rel_ents	Relationship of face to intersecting body enumeration

Description: For internal use only. Defines an attribute for linking intersection coedges with body faces to which they will attach. It is private to the Boolean operator code, but is required by more than one phase. ATTRIB_INTCOED is an intersection graph attribute that links graph entities with the relevant body entities.

The wires of the graph consist of coedges, edges and vertices, together with their geometries. Each of these entities carries exactly one attribute, recording information relevant to its role in the later stages of Booleans. All these attributes are cleaned out during the latter stages of Booleans, as they cease to be useful.

Component: **ACIS Kernel** Attributes

4

“intcurve”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	INTCURVE (1) class	derived from INTCURVE class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	intcurve (2) class	derived from intcurve class

Description: Used to determine which class specified the intcurve. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

INTCURVE (1) class

Purpose: Defines an intersection curve as an object in the model.

Derivation: INTCURVE : CURVE : ENTITY : –

Data Elements:

sv id	intcurve (2) class	intcurve data definition is given later in this manual

Description: An INTCURVE may represent an intersection between two surfaces, the projection of a curve onto a surface, or any other general curve.

An INTCURVE, itself a derived class of CURVE, records an interpolated approximation to a general curve as an intcurve.

Component: **ACIS Kernel** Geometry

intcurve (2) class

Purpose: Represents parametric object-space curves that map an interval of the real line into a 3D real vector space (object-space).

Derivation: intcurve : curve : –

Data Elements:	ctrl	if_cond		if used as a subtype reference
	prim	subtype_start		Left curly braces, "{" or Tag 15
	ctrl	if_cond		if save_version_number is less than the SURFACE_VERSION
	prim	integer		intcurve type
	ctrl	else		
	prim	string		save identifier; "intcurve"
	prim	logical		Reverse int_cur direction, "forward" or "reversed"
	ctrl	if_cond		if save_version_number is less than INTCURVE_VERSION
	prim	int_cur		save just the data associated with that type of curve. In earlier versions, there was only one type of int_cur, which covered what is now "exact", "surf", and "int". There was no ID.
	ctrl	else		
	ctrl	case_cond		save just the data associated with that type of curve. In earlier versions, there was only one type of int_cur, which covered what is now "exact", "surf", and "int". There was no ID.
	sv id	"blndsprngcur"		if int_cur is spring_int_cur or "blndsprngcur"
	sv id	"exactcur"		if int_cur is exact_int_cur or "exactcur"
	sv id	"offintcur"		if int_cur is off_int_cur or "offintcur"
	sv id	"offsetintcur"		if int_cur is offset_int_cur or "offsetintcur"
	sv id	"bldcur"		if int_cur is blend_int_cur or par_cur
	sv id	"parcur"		if int_cur is par_int_cur or par_cur
	sv id	"projcur"		if int_cur is proj_int_cur or "projcur"
	sv id	"subsetintcur"		if int_cur is subset_int_cur or "subsetintcur"
	sv id	"surfintcur"		if int_cur is int_int_cur or "surfint_cur"
	sv id	"surfcur"		if int_cur is surf_int_cur or "surfcurr"
	sv id	curve (2) class		Generic curve data. Refer to another section of this manual
	ctrl	if_cond		if used as a subtype reference
	prim	subtype_end		Right curly braces, "}" or Tag 16

Description: The intcurve class represents parametric object-space curves that map an interval of the real line into a 3D real vector space (object-space). This mapping is continuous, and one-to-one except possibly at the ends of the interval whose images may coincide. It is differentiable twice, and the direction of the first derivative with respect to the parameter must be continuous. This direction is the positive sense of the curve.

If the two ends of the curve are different in object space, the curve is open. If they are the same, it is closed. If the curve joins itself with g^2 continuity, the curve is periodic, and its period is the length of the interval that it is primarily defined. A periodic curve is defined for all parameter values by adding a multiple of the period to the parameter value so that the result is within the definition interval, and evaluating the curve at that resultant parameter. The point at the ends of the primary interval is known as the seam.

The intcurve class provides an abstraction of the concept of a parametric representation of an interpolated curve. This interpolated curve can be either an "exact" curve or an "approximate" curve that is a fit to a true curve within some fit tolerance.

The intcurve contains a "reversed" bit together with a pointer to another structure, an int_cur or something derived from it that contains the bulk of the information about the curve.

Component: **ACIS Kernel** Geometry

int_cur

Purpose: Defines spline curves.

Derivation: int_cur : subtrans_object : subtype_object : –

Data Elements:

sv id	bs3_curve_def	BSpline approximation of curve
prim	real	Fit tolerance of approximation to true
prim	newline	
sv id	surface type	First surface for curve definition
prim	newline	
sv id	surface type	Second surface for curve definition
prim	newline	
sv id	bs2_curve_def	Parameter space curve on first surface
prim	newline	
sv id	bs2_curve_def	Parameter space curve on second
prim	newline	
ctrl	if_cond	if save_version_number is greater than or equal to SAFERANGE_VERSION
prim	interval	Safe range for curve evaluation

Description: This class defines spline curves, which are defined to allow the use of use-counts to avoid copying and to allow derivation to construct curves only approximated by the intcurve. This class name does not appear in the save file, but is a base class for other subtype identifiers that do appear in the save file.

This class is supported by virtual functions that depend on the true definition of the curve. The virtual functions allow the derived curves to implement the functionality on their own. For curves with an exact bs3_curve, there is no need to implement the functionality because the methods written for the base class are sufficient.

Component: **ACIS Kernel**

“intedge”

Purpose: Defines an attribute for linking intersection edges with the intersecting entities.

Derivation: ATTRIB_INTEDGE : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class defines an attribute for linking intersection edges with the intersecting entities. This is a private class to the Boolean operator code, but is required by more than one phase.

ATTRIB_INTEDGE is attached to each intersection edge, and contains the following information for each body (this_body and other_body, meaning at this stage blank body and tool body respectively):

- A pointer to the entity on the body that this edge corresponds; an EDGE if it lies coincident with some part of that edge, otherwise the face in which it lies.
- The sense relating the intersection edge and the body edge if the body entity is an EDGE. This sense is FORWARD if the two edges are in the same direction, REVERSED if they are in opposite directions.
- A pointer to one of the coedges belonging to this edge in the wire corresponding to the tool body.
- A pointer to a *partner* INTEDGE attribute, which at this stage must be NULL.

The following functions are defined for ATTRIB_INTEDGE.

Component: ACIS Kernel

"integer_attrib"

Purpose: Defines a generic attribute that contains an integer value.

Derivation: ATTRIB_GEN_INTEGER : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –

Data Elements: prim integer Value

Description: Refer to the Purpose.

Component: Generic Attribute Husk

"intgraph"

Purpose: Defines an attribute for classifying shells and lumps of two bodies participating in a Boolean operation.

Derivation: ATTRIB_INTGRAPH : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class maintains a linked list of shell-lump objects. It is private to the Boolean operator code, but is required by more than one phase.

Component: ACIS Kernel

"intvert"

Purpose: Defines an attribute for linking graph vertices with the intersection record(s) giving rise to them.

Derivation: ATTRIB_INTVERT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class defines an attribute for linking graph vertices with the intersection record(s) giving rise to them. Where there are several, one from each body is chosen, as the necessary information is recorded in all.

ATTRIB_INTVERT is attached to each intersection vertex, and contains the following information for each body (as for INTEDGE attributes):

- A pointer to the entity on the body that the vertex corresponds - a VERTEX if it lies coincident with that body vertex, an EDGE if it lies on that edge and not at either end, or NULL if it lies properly within a face.
- The parameter value along the edge, if the entity pointed to is an EDGE, undefined otherwise.
- A pointer to a partner VERTEX (note - not an INTVERT attribute), which at this stage must be NULL.

Component: ACIS Kernel

"link"

Purpose: Used to translate mesh surfaces to and from external file formats.

Derivation: ATTRIB_LINK : ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: Refer to purpose.

Component: **Mesh Surface Husk**

"loop"

Purpose: Bounds a FACE.

Derivation: LOOP : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for next loop in boundary of face
prim	$rec_num	Pointer to record in save file for first coedge in loop
prim	$rec_num	Pointer to record in save file for face which loop bounds

Description: A LOOP represents a connected portion of the boundary of a FACE. LOOPs may be open or closed. Refer to the description of FACE for details.

Component: **ACIS Kernel** Topology

“lop_coedge”

Purpose: Creates a coedge specific to local operations.

Derivation: LOP_COEDGE : COEDGE : ENTITY : –

Data Elements: prim No data Doesn’t save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

“lop_curve_ext_att”

Purpose: Creates an attribute specific to local operations.

Derivation: ATTRIB_LOP_CURVE_EXT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data Doesn’t save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

“lop_edge”

Purpose: Creates an edge specific to local operations.

Derivation: LOP_EDGE : EDGE : ENTITY : –

Data Elements: prim No data Doesn’t save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

“lop_edge_att”

Purpose: Creates a local operations attribute for an edge.

Derivation: ATTRIB_LOP_EDGE : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	Doesn't save any data
Description:	Refer to Purpose.		
Component:	**Local Operations Husk**		

"lop_loop_attr"

Purpose: Creates a coedge specific to local operations.

Derivation: ATTRIB_LOP_LOOP : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	Doesn't save any data

Description: Contains all the solutions for a loop to be solved, by holding a pointer to the solution tree for the loop.

Component: **Local Operations Husk**

"lop_surface_ext_att"

Purpose: Creates an attribute specific to local operations.

Derivation: ATTRIB_LOP_SURFACE_EXT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	Doesn't save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

4

"lop_vertex"

Purpose: Creates a vertex specific to local operations.

Derivation: LOP_VERTEX : VERTEX : ENTITY : –

Data Elements:	prim	No data	Doesn't save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

“lop_vert_att”

Purpose: Creates an attribute specific to local operations.

Derivation: ATTRIB_LOP_VERTEX : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	Doesn’t save any data

Description: Refer to Purpose.

Component: **Local Operations Husk**

“lump”

Purpose: Represents a bounded, connected portion of space.

Derivation: LUMP : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for next lump in body
	prim	$rec_num	Pointer to record in save file for first shell in lump
	prim	$rec_num	Pointer to record in save file for body containing lump

Description: A LUMP represents a connected portion of space, bounded by one or more SHELLs, one of which is an external skin.

Component: **ACIS Kernel** Topology

“meshsurf”

Purpose: Implements the MESHSURF subclass of the SURFACE class.

Derivation: MESHSURF : SURFACE : ENTITY : –

Data Elements:	prim	logical	"forward" and "reversed"
	prim	write sv id	save identifier for this particular subtype id meshsurf id type
	prim	newline	
	prim	integer	number of nodes
	prim	integer	number of elements
	prim	integer	number of attributes
	ctrl	repeat	repeat for the number of nodes
	sv id	"p2"	save node information
	ctrl	repeat	repeat for the number of elements
	sv id	"t3"	save element information
	ctrl	repeat	repeat for the number of attributes
	sv id	"bk"	save attribute information
	sv id	surface (2) class	generic surface data

Description: This surface type is for any surface which can be represented by nodes and elements. The surface is therefore a patchwork of interconnected elements and may have C1 or even C0 discontinuities. The geometry of each element is arbitrary with the restriction that it contain its nodes.

Component: **Mesh Surface Husk**

"ms"

Purpose: Represents attribute base class for the **Mesh Surface Husk**.

Derivation: ATTRIB_MESH : ATTRIB : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for current ATTRIB_MESH in list
	prim	$rec_num	Pointer to record in save file for next ATTRIB_MESH in list
	prim	$rec_num	Pointer to record in save file for previous ATTRIB_MESH in list
	prim	$rec_num	Pointer to record in save file for owner ENTITY

Description: Represents the base attribute for the **Mesh Surface Husk**. All other **Mesh Surface Husk** attributes are derived from this base class.

Component: **Mesh Surface Husk**

msh_sur

Purpose:	Records a composite mesh surface.		
Derivation:	msh_sur :–		
Data Elements:	prim	No data	This class does not save any data

Description: A meshsurf holds a pointer to a msh_sur, an internal class defined so that we can use use-counts to avoid too much copying, and a logical denoting reversal of the sense of the stored surface. The msh_sur, which is always an object of a class derived from msh_sur, contains the nodes/elements definition of the surface. All access to the msh_sur is done via virtual functions, which may be adapted to different mesh types.

This class is the actual container class for mesh data. It is defined separately with a use count so that copying of enormous quantity of data can be deferred and so that new mesh class and so that new mesh class definitions can be derived from this class.

There are virtual functions to support all the functions defined for the meshsurf class that depend on the true definition of the surface.

The base class is pure: classes for real mesh surface types must be derived from it.

Component: ACIS Kernel

“name_attrib”

Purpose: Defines a named attribute for the Generic Attribute Husk.

Derivation: ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –

Data Elements:

ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
prim	integer	Split action
prim	integer	Merge action
prim	integer	Transform action
ctrl	else	
enum	split_action	Action when owner split
enum	merge_action	Action when owner merged
enum	trans_action	Action when owner transformed
ctrl	if_cond	if there is a name
prim	string	Name

Description:	Refer to the Purpose.
Component:	Generic Attribute Husk

“named_attribute”

Purpose:	Defines class NAMED_ATTRIB to allow generic named attributes.		
Derivation:	NAMED_ATTRIB : ATTRIB_ST : ENTITY : –		
Data Elements:	ctrl	if_cond	if there is an attribute_name
	prim	string	attribute name
	ctrl	else	
	prim	integer	default value of 0.
Description:	Defines class NAMED_ATTRIB to allow generic named attributes.		
Component:	Part Management Husk		

“named_logical_attribute”

Purpose:	Defines class NAMED_LOGICAL_ATTRIB to provide named attributes with logical values.		
Derivation:	NAMED_LOGICAL_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –		
Data Elements:	prim	integer	value of logical attribute
Description:	Implements class NAMED_LOGICAL_ATTRIB to allow named attributes with logical values.		
Component:	Part Management Husk		

4

“named_pos_attribute”

Purpose:	Defines class NAMED_POS_ATTRIB to provide named attributes with position values.		
Derivation:	NAMED_POS_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –		
Data Elements:	prim	position	position

Description: Implements class NAMED_POS_ATTRIB to allow named attributes with positions.

Component: Part Management Husk

“named_real_attribute”

Purpose: Defines class NAMED_REAL_ATTRIB to provide named attributes with real values.

Derivation: NAMED_REAL_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –

Data Elements:	prim	real	the real value

Description: Implements class NAMED_REAL_ATTRIB to allow named attributes with real values.

Component: Part Management Husk

“named_string_attribute”

Purpose: Defines class NAMED_STRING_ATTRIB to provide named attributes with string values.

Derivation: NAMED_STRING_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –

Data Elements:	ctrl	if_cond	
			if there is an attribute_value
	prim	string	the name
	ctrl	else	
	prim	integer	default value 0

Description: Implements class NAMED_STRING_ATTRIB to allow named attributes with string values.

Component: Part Management Husk

“named_vec_attribute”

Purpose: Implements class NAMED_VEC_ATTRIB to allow named vector attributes.

Derivation: NAMED_VEC_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –

Data Elements: prim vector the vector

Description: Implements class NAMED_VEC_ATTRIB to allow named attributes with vectors.

Component: Part Management Husk

“named_int_attribute”

Purpose: Defines class NAMED_INT_ATTRIB to provide named attributes with integer values.

Derivation: NAMED_INT_ATTRIB : NAMED_ATTRIB : ATTRIB_ST : ENTITY : –

Data Elements: prim integer Value for attribute

Description: Implements class NAMED_INT_ATTRIB to allow named attributes with integer values.

Component: Part Management Husk

“node”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond		if used in the meshing routine and derived from attribute
sv id		ATTRIB_NODE	derived from ATTRIB_NODE class
ctrl	else		if used elsewhere and derived from entity
sv id		NODE	derived from NODE class

Description: Refer to Purpose.

Component: **Mesh Surface Husk**

NODE

Purpose: Represents a grazing touch of an intersection curve with the side (boundary) of an element.

Derivation: NODE : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for owning mesh
	prim	$rec_num	Pointer to record in save file for owning element
	prim	position	node location

Description: Entity to represent a grazing touch of an intersection curve with the side (boundary) of an element.

Component: **Mesh Surface Husk**

“norender_attribute”

Purpose: Attribute to mark items not to be rendered.

Derivation: NORENDER_ATTRIB : ATTRIB_ST : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: Refer to purpose.

Component: **ACIS RenderWare Husk**

“offintcur”

Purpose: Represents a spline curve obtained by the intersection of two surfaces that are offsets of the given surfaces.

Derivation: off_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
		sv id save identifier for this particular subtype	
	sv id	int_cur	Generic int_cur data
	prim	real	Offset from first surface
	prim	real	Offset from second surface
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Refer to the Purpose.

Component: **ACIS Kernel**

4

“offrel”

Purpose: Implementation of the offset relation attribute.

Derivation: ATTRIB_OFFREL : ATTRIB_SG : ATTRIB : ENTITY

Data Elements:

prim	$rec_num	Pointer to record in save file for coedge from which offset was generated
prim	$rec_num	Pointer to record in save file for vertex from which offset was generated

Description: Attribute declaration for attribute class relating a trimmed offset wire to the original wire. The attribute relates the coedges in the offset wire to the COEDGEs and VERTEXes of the original wire. This attribute currently stores the COEDGE or VERTEX that generated the offset COEDGE. The offset COEDGE can only come from one or the other (a VERTEX or a COEDGE), not both.

Component: **ACIS Kernel** Attributes

“offsetintcur”

Purpose: Represents an offset curve, which is offset in any plane.

Derivation: offset_int_cur : int_cur : subtrans_object : subtype_object : –

4

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	ctrl	if_cond	
			if save_version_number is less than INTCURVE_VERSION
	sv id	curve (2) class	bs3 curve save
	prim	newline	
	ctrl	else	
	sv id	int_cur	Generic int_cur data
	sv id	curve type	Original curve
	prim	real	Start parameter
	prim	real	End parameter
	prim	newline	
	prim	vector	Offset plane normal
	prim	newline	
	prim	real	Offset distance
	prim	newline	
	prim	real	Draft distance
	prim	newline	
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents either the offset of a planar curve, in the plane of the curve, or the offset of a 3D curve in any plane. The transformation and the true offset distance are calculated internally.

Component: **ACIS Kernel**

“offsetvbsur”

Purpose: Represents an offset vertex blend surface.

Derivation: VBL_OFFSURF : VBL_SURF : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	prim	real	offset
	prim	newline	Start parameter
	sv id	“vertexblendsur”	VBL_SURF data
	prim	subtype_end	Right curly braces, “}” or Tag 16

4

Description: This class represents an offset vertex blend surface. Derived from a VBL_SURF. We store all the data about the UNoffset vertex blend as usual (boundaries, cached positions and so forth) so that _evaluate (and _eval) return positions on the unoffset surface as normal. The offset appears ONLY in evaluate. Finally, there are just a few other functions that need to be specialised.

Component: **ACIS Kernel**

“offsur”

Purpose: Represents the offset of a surface.

Derivation: off_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	surface type	Original surface
prim	real	Offset distance
prim	real	Start *u*-parameter
prim	real	End *u*-parameter
prim	real	Start *v*-parameter
prim	real	End *v*-parameter
prim	boolean	Original surface reversed
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents the offset of a surface. This is derived from the base class, spl_sur, which is used by the spline surface class to contain the surface description proper. If $S(u,v)$ is the original surface, the offset surface $O(u,v)$ is:

```
O(u,v) = S(u,v) + d* N(u,v)
```

where $N(u,v)$ is the surface normal and d is the offset distance. Along with the benefits of having evaluators based off the offset definition, the off_spl_sur class simplifies offsets of offset surfaces. So, if the user offsets the progenitor surface by some distance (d), and then offsets that surface by the distance ($-d$), the result is the progenitor surface again. Also, if the progenitor is offset by the distance ($d1$) and the offset surface is then offset by the distance ($d2$), the resulting surface is an offset of the progenitor by the distance ($d1 + d2$).

Component: **ACIS Kernel**

"p1"

Purpose:	Implements one dimensional parameter nodes.		
Derivation:	P1NODE : NODE : ENTITY : –		
Data Elements:	sv id	NODE	parent information
	prim	real	parameter value
Description:	Represents vertex nodes of the compcurv entity.		
Component:	**Mesh Surface Husk**		

"p2"

Purpose:	Implements parameter nodes.		
Derivation:	P2NODE : NODE : ENTITY : –		
Data Elements:	sv id	NODE	Parent node
	prim	float	*u* value
	prim	float	*v* value
Description:	Represents vertex nodes of the mesh surface.		
Component:	**Mesh Surface Husk**		

"parasil"

Purpose:	Creates an interpolated curve subtype which can precisely represent a parallel–view silhouette curve.		
Derivation:	para_silh_int_cur : int_cur : subtrans_object : subtype_object : –		
Data Elements:	prim	subtype_start	Left curly braces, "{" or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	int_cur	Save interpolated curve data
	prim	vector	Save view direction vector
	prim	subtype_end	Right curly braces, "}" or Tag 16
Description:	Defines an interpolated curve subtype which can precisely represent a parallel–view silhouette curve.		

Component: **ACIS Kernel**

“parcur”

Purpose: Represents a spline curve as a parameter curve on a spline surface for a parameterization.

Derivation: par_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
ctrl	if_cond	if save_version_number is less than INTCURVE_VERSION
sv id	curve type	bs3 curve
prim	real	fit tolerance
prim	newline	
sv id	surface type	surface 1 data
prim	newline	
sv id	surface type	surface 2 data
prim	newline	
sv id	curve type	bs2 curve pcurve 1
prim	newline	
sv id	curve type	bs2 curve pcurve 2
ctrl	else if_cond	if save_version_number is less than PARCUR_VERSION
sv id	curve type	bs3 curve
prim	real	fit tolerance
prim	newline	
sv id	surface type	surface 2 data
prim	newline	
sv id	surface type	surface 1 data
prim	newline	
sv id	curve type	bs2 curve pcurve 2
prim	newline	
sv id	curve type	bs2 curve pcurve 1
ctrl	else	
sv id	int_cur	Generic int_cur data
prim	logical	Consistent version: “surf2” or “surf1”
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents a 3D spline curve as a 2D parameter curve on a spline surface. The spline surface is used to map the 2D parameter curve from (u,v) parameter space into (x,y,z) euclidean space. The approximate parameter curve is everywhere within the fit tolerance of the exact parameter curve.

Component: **ACIS Kernel**

"pcurcur"

Purpose: Defines an interpolated curve subtype that is the 3D extension of the parameter curve representing a curve on a surface.

Derivation: pcur_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	No data	This class does not save any data

Description: This class defines an interpolated curve subtype that is the 3D extension of the parameter curve representing a curve on a surface. This is used internally by ACIS during point-in-face testing on a parametric surface, and certain member functions that are not required by ACIS are disabled, to simplify the implementation. It should not be used by an application.

Component: ACIS Kernel

"pcurve"

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
sv id	PCURVE (1) class	derived from PCURVE class
ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
sv id	pcurve (2) class	derived from pcurve class

Description: Used to determine which class specified the pcurve. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

PCURVE (1) class

Purpose: Defines a 2D parameter-space approximation to a CURVE as an object in the model.

Derivation: PCURVE : ENTITY : –

Data Elements: sv id pcurve (2) class save identifier

Description: The PCURVE class represents a 2D parameter-space approximation to a CURVE lying on a parameterized SURFACE. The approximation may be a private copy, or it may be the first or second PCURVE defined for an INTCURVE. In either case, it may be negated from the underlying PCURVE.

Component: **ACIS Kernel** Geometry

pcurve (2) class

Purpose: Defines a 2D curve defined in the parameter space of a parametric surface.

Derivation: pcurve : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, “{” or Tag 15
prim	string	save identifier; “pcurve”.
prim	logical	“forward” or “reverse” with respect to underlying definition
ctrl	if_cond	if the save_version_number is less than PCURVE_VERSION
sv id	bs2_curve_def	save the curve data
prim	real	fit tolerance
prim	newline	
sv id	surface type	save specific surface type
prim	newline	
ctrl	else	if the save_version_number is greater than PCURVE_VERSION
ctrl	if_cond	if par_cur subtype is exp_par_cur or “exppc”
sv id	“exppc”	then refer to its data
ctrl	if_cond	if par_cur subtype is imp_par_cur or “imppc”
sv id	“imppc”	then refer to its data
prim	real	*u* offset
prim	real	*v* offset
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, “}” or Tag 16

Description:

The pcurve class represents parameter-space curves that map an interval of the real line into a 2D real vector space (parameter space). This mapping is continuous, and one-to-one except possibly at the ends of the interval whose images may coincide. It is differentiable twice, and the direction of the first derivative with respect to the parameter is continuous. This direction is the positive sense of the curve.

A parameter-space curve is always associated with a surface, that maps the parameter-space image into 3D real space (object space); therefore, the two mappings together can be considered to be a single mapping from a real interval into object space. Most of the properties of a parameter-space curve relate in fact to this combined mapping.

If the two ends of the curve are different in object space, the curve is open. If they are the same, it is closed. If the curve joins itself with G^2 continuity, the curve is periodic, with its period being the length of the interval that it is primarily defined. A periodic curve is defined for all parameter values, by adding a multiple of the period to the parameter value so the result is within the definition interval, and evaluating the curve at that resultant parameter. The point at the ends of the primary interval is known as the seam. If the surface is periodic, a closed or periodic parameter-space curve cannot in fact be closed in parameter space, but its end values can differ by the surface parameter period in one or both directions.

Also, a parameter-space curve is always associated with an object-space curve lying in (or fitted to) the surface. This curve is used to assist in the determination of the surface parameter values corresponding to object-space points on the 3D curve, by using the parameter value on the 3D curve to evaluate the 2D curve for an approximation to the surface parameter values for iterative refinement. For this reason, a parameter-space curve must always have the same parameter range as its associated object-space curve, and its internal parameterization must be similar, though not necessarily identical, to that of the object-space curve. A parameter-space curve can have the same sense as its associated object-space curve, or be opposite. In the latter case, the parameterization is negated one with respect to the other.

A pcurve consists of pointer to a par_cur that holds the data defining the 2D parameter space curve and a logical flag indicating reversal of the pcurve from the underlying spline curve. In addition, a parameter space vector is stored that represents the displacement of this pcurve in the parameter space of the surface the pcurve lies in. By having a nonzero vector for a periodic surface, a continuous sequence of object space curves (3D curves) can have a continuous sequence of parameter space curves (2D curves).

Component: **ACIS Kernel** Geometry

pcurve type

Purpose: Defines a 2D curve defined in the parameter space of a parametric surface.

Derivation: pcurve : –

Data Elements:

prim	ident	Pcurve type
ctrl	if_cond	if pcurve type is set to “null_pcurve” No pcurve saved
ctrl	if_cond	if pcurve type is set to “pcurve”
sv id	pcurve (2) class	Specific pcurve data

Description: Used to more specifically define pcurves.

Component: **ACIS Kernel** Geometry

“perspsil”

Purpose: Creates an interpolated curve subtype which can precisely represent a perspective–view silhouette curve.

Derivation: persp_silh_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	logical	Consistent version: “surf2” or “surf1”
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Defines an interpolated curve subtype which can precisely represent a perspective–view silhouette curve.

Component: **ACIS Kernel**

“phl”

Purpose: Defines the phl tag identifier.

Derivation: None

Data Elements:

ctrl	if_cond	if derived from entity
sv id	ENTITY_PHL	class
ctrl	if_cond	if derived from attrib
sv id	ATTRIB_PHL	class

Description: The phl tag is used by two different classes.

Component: **Precise Hidden Line Husk**

“phl_camera”

Purpose: Defines a camera viewpoint against which hidden lines are calculated.

Derivation: PHL_CAMERA : ENTITY_PHL : ENTITY : –

Data Elements:	prim	position	eyepos
	prim	position	target
	prim	integer	perspective flag

Description: Each camera is defined by its position (eye position), its aim (target position), both in global model space, and by a perspective flag. The flag is TRUE for a simple perspective projection, where two positions define the view direction and the distance between eye and target. The flag is FALSE for a parallel projection, where the eye and target positions define the view direction, and the distance is not relevant.

Component: **Precise Hidden Line Husk**

“phl_edge”

Purpose: Defines a regular or silhouette edge.

Derivation: PHL_EDGE : ENTITY_PHL : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for EDGE referenced
	prim	$rec_num	Pointer to record in save file for BODY reference
	prim	$rec_num	Pointer to record in save file for PHL_SEGMENT
	prim	$rec_num	Pointer to record in save file for owning FACE of PHL_EDGE
	prim	$rec_num	Pointer to record in save file for next PHL_EDGE
	prim	$rec_num	Pointer to record in save file for previous PHL_EDGE

Description: A PHL_EDGE represents one edge (regular edge or silhouette edge).

A PHL_EDGE points to the edge, to the owning body, to the owning face for silhouette edges, and to a list of segments.

Component: **Precise Hidden Line Husk**

“phl_segment”

Purpose: Defines a line segment with visibility information.

Derivation: PHL_SEGMENT : ENTITY_PHL : ENTITY : –

Data Elements:	prim	real	start of segment
	prim	real	end of segment
	prim	integer	inner or outer segment
	prim	integer	visibility of segment
	prim	$rec_num	Pointer to record in save file for previous PHL_SEGMENT
	prim	$rec_num	Pointer to record in save file for next PHL_SEGMENT

Description: A PHL_SEGMENT object defines the visibility of an edge.

An interval (parameter range) indicates which piece of the edge is covered by the segment.

Status information records whether the segment is an inner or outer segment.

Visibility information records whether the segment is *visible*, *hidden* by a FACE, or *occluded* by an EDGE.

Segments can be hooked together in doubly-linked lists. One list of segments, showing the outlook of the edge, occurs everywhere an edge’s owning body occurs.

Component: **Precise Hidden Line Husk**

“phl_vw”

Purpose: Attaches hidden line data and viewing parameters to BODYs.

Derivation: ATTRIB_PHL_VW : ATTRIB_PHL : ATTRIB : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for camera
	prim	$rec_num	Pointer to record in save file for linked list of PHL_EDGE’s
	prim	integer	view token to identify attribute

Description: The ATTRIB_PHL_VW class attaches to BODYs. The purpose of this class is to maintain hidden line data along with the viewing parameters in effect at the time of hidden line calculation.

The hidden line data is stored as a doubly-linked list of PHL_EDGEs and a camera definition. A view token is also stored to distinguish attributes from others of the same type.

Component: **Precise Hidden Line Husk**

“pid_name”

Purpose: Persistent identifier data attribute.

Derivation: ATTRIB_PID : ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements:

ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
prim	integer	Number of characters used in the the user name
prim	repeat	Repeat for the number of characters
prim	integer	One character in base name
ctrl	else	
prim	string	Base name
prim	long	Time value in seconds from January 1, 1970
prim	integer	Entity index
prim	integer	Copy number

Description: The Persistent Identification Husk generates an identifier that is attached to an entity and is retained by the entity from session to session. The identifier is designed to be unique over all sessions of ACIS. The identifier is stored in an ATTRIB_PID object.

Component: **ACIS Kernel** Attributes

“pipesur”

Purpose: A surface that is the envelope of a fixed-radius circle.

Derivation: pipe_spl_sur : tube_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	ctrl	if_cond	if save_version_number is less than SPLINE_VERSION
	sv id	spl_sur	spline surface
	ctrl	else	
	prim	real	Radius
	prim	newline	
	sv id	curve type	spine curve
	prim	newline	
	sv id	curve type	*u* = 0 curve
	prim	newline	
	prim	interval	bs3 surface *u*-parameter range
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents a surface that is the envelope of a fixed-radius circle centered on a point on a given curve, and normal to the curve at each point.

Component: **ACIS Kernel**

“plane”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	PLANE (1) class	derived from PLANE class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	plane (2) class	derived from plane class

Description: Used to determine which class specified the plane. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

4

PLANE (1) class

Purpose: Defines a plane as an object in the model.

Derivation: PLANE : SURFACE : ENTITY : –

Data Elements:

sv id	plane (2) class	plane data given in another section of this manual.

Description: A PLANE is defined by a plane that is in turn given by a point on the plane and a unit normal. The direction of the normal is regarded as the inherent direction of the surface.

Component: **ACIS Kernel** Geometry

plane (2) class

Purpose: Defines a planar surface.

Derivation: plane : surface : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
prim	integer	plane_type; integer for type of plane
ctrl	else	if save_version_number is greater than the SURFACE_VERSION
prim	string	save identifier; "plane".
prim	position	root point of plane
prim	vector	normal to plane
ctrl	if_cond	if save_version_number is greater than or equal to SURFACE_VERSION
prim	vector	u derivative
prim	logical	*v*-pararameter, either "forward_v" or "reverse_v" with respect to right hand rule.
sv id	surface (2) class	Generic surface data. Refer to another section of this manual.
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: A plane class defines a plane with a point and a unit vector normal to the plane. Usually, the point chosen to define the plane is near the center of interest. The normal represents the outside of the surface. This is important when a plane is used to define a FACE of a shell or solid.

Four data members describe the parameterization of the plane. To find the object-space point corresponding to a given (*u,v*) pair, first find the cross product of the plane normal with *u*-derivitive vector, negate it if reverse_v is TRUE, and call it *v*-derivitive vector. Then the evaluated position is:

```
pos = root_point + u* u_deriv + v* v_deriv
```

When the plane is transformed, u_deriv is transformed in the usual way, along with the root point and normal, and reverse_v is inverted if the transform includes a reflection. When the plane is negated, the direction of the normal is reversed, and reverse_v is inverted.

When a plane is constructed, u_deriv is automatically generated to be a fairly arbitrary unit vector perpendicular to the normal, and reverse_v is set FALSE. If the normal is of zero length, or if the plane is constructed using the raw constructor with no normal, u_deriv is set to be a zero vector, and the arbitrary direction is generated whenever a parameter-based function is called. Whenever an application changes the normal directly, it should also ensure that u_deriv is perpendicular to it.

In summary, planes are:

– Not true parametric surfaces.
– Open in *u* and *v*.
– Not periodic in either *u* or *v*.
– Not singular at any *u* or *v*.

Component: **ACIS Kernel** Geometry

“point”

Purpose: Represents the position of a VERTEX as an object in the model.

Derivation: APOINT : ENTITY : –

Data Elements: prim position Point coordinates

Description: The APOINT class is written to the .sat file as a “point”. This class records the object space position of a VERTEX. Unlike the other geometric classes, there is no need for derived types. Cartesian coordinates are assumed.

Component: **ACIS Kernel** Geometry

“pointer_attrib”

Purpose:	Defines a generic attribute that contains a reference to an entity.		
Derivation:	ATTRIB_GEN_POINTER : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –		
Data Elements:	prim	$rec_num	Pointer to record in save file for referenced entity
Description:	The referenced entity is *not* copied, transformed, or lost along with the attribute’s owner.		
Component:	Generic Attribute Husk		

“position_attrib”

Purpose:	Defines a generic attribute that contains a position.		
Derivation:	ATTRIB_GEN_POSITION : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –		
Data Elements:	prim	position	*x*,*y* *z* values
Description:	Refer to the Purpose.		
Component:	Generic Attribute Husk		

4

“projcur”

Purpose:	Implements an interpolated curve subtype to represent the perpendicular projection of a curve to a surface.
Derivation:	proj_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start		Left curly braces, “{” or Tag 15
prim	write sv id		save identifier for this particular subtype
ctrl	if_cond		if save_version_number is less than INTCURVE_VERSION
sv id		curve type	bs3 curve
prim		real	fit tolerance
prim		newline	
sv id		surface (2) class	surface 1
prim		newline	
sv id		surface (2) class	surface 2
prim		newline	
sv id		curve (2) class	pcurve 1
prim		newline	
sv id		curve (2) class	pcurve 2
ctrl	else		
sv id		int_cur	Generic int_cur data
sv id		curve type	Curve being projected
prim		interval	Range of curve to be projected
prim		logical	consistent version: “surf2” or “surf1”
prim	subtype_end		Right curly braces, “}” or Tag 16

Description: This class implements an interpolated curve subtype to represent the perpendicular projection of a curve to a surface. This also includes a second surface, for the sake of its pcurve, as one significant use of this curve type is blending, where it is used to bound a (parametric) blend surface.

Component: **ACIS Kernel**

“pt_cstrn”

Purpose: Point constraint.

Derivation: ATTRIB_PT_CSTRN : ATTRIB_DSCSTRN : ATTRIB : ENTITY : –

Data Elements:

prim	long		apc_image_dim
prim	long		apc_domain_dim
ctrl	repeat		Repeat for all apc_image_dim
prim		real	apc_image_pt[ii]
ctrl	repeat		Repeat for all apc_domain_dim
prim		real	apc_domain_pt[ii]

Description: For internal use only.

Component: **Deformable Surface Husk**

4

“ptlist”

Purpose:	Associates a cyclic doubly linked list with the entity.		
Derivation:	ATTRIB_EYE_POINTLIST_HEADER : ATTRIB_EYE : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	The ATTRIB_EYE_POINTLIST_HEADER class is an ACIS attribute that can be attached to any entity to associate a cyclic doubly linked list of AF_POINT instances with the entity.		
Component:	Faceter Husk		

“pt_press”

Purpose:	Point pressure.		
Derivation:	ATTRIB_PT_PRESS : ATTRIB_DSLOAD : ATTRIB : ENTITY : –		
Data Elements:	prim	long	app_domain_dim
	ctrl	repeat	Repeat for all app_domain_dim
	prim	real	app_domain_pt[ii]
Description:	For internal use only.		
Component:	**Deformable Surface Husk**		

4

“rbase”

Purpose:	Defines a rendering attribute that holds information about surface visual attributes, sidedness, and texture space.		
Derivation:	ATTRIB_RH : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	This class defines a rendering attribute that holds information about surface visual attributes (RH_MATERIAL*), surface-sidedness for rendering (int), and surface texture space for wrapping (RH_TEXTURE_SPACE*) textures onto the surface.		

Component: ACIS Kernel

“rbblnsur”

Purpose: This class implements the constant radius rolling ball blend surface.

Derivation: rb_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	blend_spl_sur	Blend data
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This is a straightforward derivation of blend_spl_sur. The ball rolls on two support entities, which may be either curves, surfaces or points. The point-point case is not included because this is always a sphere. The surface-surface case is equivalent to the pipe surface.

Component: **ACIS Kernel**

“real_attrib”

Purpose: Defines a generic attribute that contains a real value.

Derivation: ATTRIB_GEN_REAL : ATTRIB_GEN_NAME : ATTRIB_GENERIC : : ENTITY : – ATTRIB

Data Elements:

prim	real	Value

Description: Refer to the Purpose.

Component: Generic Attribute Husk

4

“render”

Purpose: Defines a generic attribute that contains a real value.

Derivation: ATTRIB_RENDER : ATTRIB_RH : ATTRIB : ENTITY : –

Data Elements:	prim	$rec_num	entity material
	prim	$rec_num	entity tspace
	prim	integer	number of sides
	ctrl	if_cond	if ent_tran is set
	ctrl	if_cond	if tran_mod is set
	prim	integer	write the number 2
	ctrl	else_cond	if not tran_mod
	prim	integer	write the number 1
	prim	transf	write the transform for ent_tran
	ctrl	else_cond	if not ent_tran
	prim	integer	write integer 0

Description: Refer to the Purpose.

Component: Basic Rendering Husk

“rgb_color”

Purpose: Defines a ATTRIB_RGB color attribute.

Derivation: ATTRIB_RGB : ATTRIB_ST : ATTRIB : ENTITY : –

Data Elements:	prim	real	red color component
	prim	real	green color component
	prim	real	blue color component

Description: Defines ATTRIB_RGB to store RGB color information for an ENTITY. This attribute takes precedence over ATTRIB_COL when displaying ENTITYs.

Component: Graphic Interaction Husk

4

“ref”

Purpose: This references a previously defined subtype in the save file.

Derivation: None

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	prim	integer	Subtype index number being referenced.
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: A subtype that has been previously defined can be referenced using this. Thus duplicate information does not have to be stored in the save file. Subtypes receive an index number starting with zero at the beginning of the file. This integer is then later used by a reference to obtain the correct information.

Component: **ACIS Kernel**

"ref_vt"

Purpose: Attaches REFINEMENT and VERTEX_TEMPLATE instances to other entities.

Derivation: ATTRIB_EYE_REF_VT : ATTRIB_EYE : ATTRIB : –

Data Elements:			
	prim	$rec_num	Pointer to record in save file for refinement
	prim	$rec_num	Pointer to record in save file for vertex template

Description: The ATTRIB_EYE_REF_VT class is an ACIS attribute used to attach REFINEMENT and VERTEX_TEMPLATE instances to other entities. The class can hold a pointer to one of each class. However, this implementation assumes that exactly one of those pointers is non-NULL. This allows independent replacement, and use of multiple refinements or vertex templates, without undue complication.

Component: Faceter Husk

"rh_background"

Purpose: Defines a background.

Derivation: RH_BACKGROUND : RH_ENTITY : ENTITY : –

Data Elements:			
	prim	No data	This class does not save any data

Description: This class defines the color of a pixels at any point in the image which is not covered by an entity surface. A background can comprise a single uniform color or pattern, or can be composed of a previously-generated image or an image scanned from a photograph. Only one background can be active at any one time.

Component: ACIS Kernel

"rh_entity"

Purpose: Provides common methods and data for other rendering classes.

Derivation: RH_ENTITY : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: Rendering entities provide the basis for manipulating the appearance of ACIS geometric entities, image backgrounds and lighting conditions. Child classes include RH_BACKGROUND, RH_FOREGROUND, RH_ENVIRONMENT_MAP, RH_LIGHT, RH_MATERIAL, and RH_TEXTURE_SPACE.

Component: ACIS Kernel

"rh_env_map"

Purpose: Defines an environment map.

Derivation: RH_ENVIRONMENT_MAP : RH_ENTITY : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: Environment maps simulate interobject reflections, both between bodies in a scene and between a body and the external environment. RH_ENVIRONMENT_MAP objects are used with one of the component shaders of the other rendering entities specified by an RH_MATERIAL.

Component: ACIS Kernel

"rh_foreground"

Purpose: Defines a foreground.

Derivation: RH_FOREGROUND : RH_ENTITY : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: A foreground is the counterpart to a background. Foreground shaders provide an extra level of image processing during the shading process. It can be thought of as a filter and may be used to support atmospheric effects, such as fog or depth cueing. Only one foreground can be active at any given time.

Component: ACIS Kernel

"rh_light"

Purpose: Defines a light source.

Derivation: RH_LIGHT : RH_ENTITY : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: RH_LIGHTs define light sources within the Renderer. Supported light source types are “ambient,” “distant,” “eye,” “point,” and “spot.”

An enumerated type, Fall_Off_Type, is a parameter to some light types which selects how the intensity of the light varies with the distance from the light source, and has possible values of FALL_OFF_CONSTANT, FALL_OFF_INVERSE, or FALL_OFF_INVERSE_SQUARE.

Shadowing is supported for distant, point, and spot in all rendering modes except flat and simple. If an image is to be rendered with shadows, a shadow map must be computed before rendering, using api_rh_create_light_shadow for each light for which shadows are required. A shadow map is view-independent and can be reused for any number of images provided there is no change in the light source geometry or the entities it illuminates.

Component: ACIS Kernel

“rh_material”

Purpose: Defines a material consisting of color, displacement, reflectance, and transparency.

Derivation: RH_MATERIAL : RH_ENTITY : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: A material defines the appearance of the surface of an ACIS topological entity in terms of four components: color, reflectance, transparency, and displacement.

A color defines the color for any point on the surface of an entity to which applies and can be a simple single color or a complex pattern, such as a procedurally-defined marble effect.

The reflectance governs how the surface behaves visually in the presence of light. The reflectance defines the surface finish of an entity and models effects, such as matte, metal, or mirrored surfaces. Reflectance is not supported in the flat or gouraud rendering modes.

Component: ACIS Kernel

4

“rh_texture_space”

Purpose: Defines a texture space.

Derivation: RH_TEXTURE_SPACE : RH_ENTITY : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: RH_TEXTURE_SPACE entities assist in the production of a shading effect known as a wrapped texture. A wrapped texture produces the effect of a sheet of paper shrink wrapped onto the surface of a solid object. A texture space uses one of several texture space shaders to map between the coordinate system of the sheet and the coordinate system of the surface of the solid object. Texture space arguments are treated in a similar fashion to those of material components.

Component: ACIS Kernel

“round”

Purpose: Defines a circular rolling-ball blend.

Derivation: ATTRIB_ROUND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	real	Radius
	ctrl	if_cond	if save_version_number is greater than or equal to BLEND_VERSION
	prim	real	Setback at start
	prim	real	Setback at end
	prim	real	Bulge

Description: Refer to the Purpose.

Component: **ACIS Kernel** Attributes

“rotsur”

Purpose: Represents a surface of rotation.

Derivation: rot_spl_sur : spl_sur : subtrans_object : subtype_object : –

4

Data Elements:	prim	subtype_start	Left curly braces, "{" or Tag 15
	prim	write sv id	save identifier for this particular subtype
	ctrl	if_cond	if save_version_number is less than SPLINE_VERSION
	sv id	spl_sur	save data
	ctrl	else	
	prim	newline	
	sv id	curve type	Curve being rotated
	prim	newline	
	prim	position	Root of axis
	prim	vector	Direction of axis
	prim	interval	*u* range
	prim	interval	*v* range
	prim	subtype_end	Right curly braces, "}" or Tag 16

Description: This class represents a surface of rotation. The surface is defined by an axis of rotation and a curve. The curve must not intersect with the axis, except possibly at its ends, and must not be tangential to a circle centered on the axis and perpendicular to it (i.e., at no point on the curve can the tangent direction be the same as that of a circle that is centered on the axis of revolution, perpendicular to it, and through the point). The parameter ranges defining the surface are the *u*-direction is along the curve, and follows its parametrization, while the *v*-direction is clockwise around the axis, with the given curve as the *v*=0 parameter line.

Component: **ACIS Kernel**

"sfcvfreeblndsur"

Purpose: Implements the variable-radius surface-curve/free blend surface.

Derivation: sfcv_free_bl_spl_sur : var_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	No data	This class does not save any data

Description: This class implements the surface geometry of a variable-radius blend between a curve in a surface and another surface. THe blend is tangent to the surface containing the surface curve, and is not tangent to the other surface.

Component: ACIS Kernel

“sg”

Purpose: Organization base attribute class for the SG Husk.

Derivation: ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements:

prim	No data		This class does not save any data

Description: ATTRIB_SG is the organizational class from which the other SG Husk attribute classes are derived. Its sole purpose is to identify those child classes as belonging to the SG Husk, and so adds no new data or methods to those of ATTRIB.

Component: ACIS Kernel

“shell”

Purpose: Bounds a LUMP peripherally, or as an internal void.

Derivation: SHELL : ENTITY : –

Data Elements:

prim	$rec_num		Pointer to record in save file for next shell in lump
prim	$rec_num		Pointer to record in save file for first subshell in shell
prim	$rec_num		Pointer to record in save file for first face in shell
ctrl	if_cond		if save_version_number is greater than or equal to WIREBOOL_VERSION
prim		$rec_num	Pointer to record in save file for first wire in shell
ctrl	if_cond		if save_version_number is less than LUMP_VERSION
prim		$rec_num	Pointer to record in save file for body owning the LUMP containing shell
ctrl	else		
prim		$rec_num	Pointer to record in save file for lump containing shell

Description: The SHELL is one portion of a LUMP’s boundary, and has no internal connection with any other SHELL. If a LUMP has no voids, exactly one *peripheral* SHELL gives it its overall extent; any other SHELLs bound *voids* wholly within the LUMP. In the data structure, no distinction is made between peripheral and void SHELLs. In this context a SHELL is closed and bounded.

It is technically possible for a SHELL to be open and bounded or unbounded. If bounded, the containing LUMP (and BODY) is considered *incomplete*, more accurately, it is incompletely bounded. It interacts with other BODYs only so far as the defined portions of their SHELLs interact, and there are configurations of that interaction that are disallowed. If the SHELL is unbounded, it can be semi-infinite (e.g., a plane bounded by a single infinite straight line) or infinite (two half-infinite planes joined at their boundaries). If the SHELL is semi-infinite, the BODY is incomplete, while an infinite SHELL is completely defined, though of infinite extent.

The concepts of *peripheral* and *void* SHELLs, and of *connected* and *disjoint* BODYs have no meaning when applied to incomplete LUMPs or BODYs.

A SHELL is constructed from a collection of WIREs or FACEs. If this is a large collection, it may be subdivided into a hierarchy of SUBSHELLs, each containing a proper subcollection. A SHELL subdivided into SUBSHELLs may also contain WIREs and FACEs directly, if these entities do not fit naturally into any SUBSHELL.

Component: **ACIS Kernel** Topology

"skinsur"

Purpose: Defines a skin surface between a list of curves.

Derivation: skin_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, "{" or Tag 15
prim	write sv id	save identifier for this particular subtype
prim	integer	Number of curves
ctrl	repeat	Repeat for the number of curves
prim	real	Tangent length at start of curve or –1
prim	real	Tangent length at end of curve or –1
prim	real	Matching tangent length at start of curve or –1
prim	real	Matching tangent length at end of curve or –1
prim	real	*v* knot
sv id	curve type	Curve to be skinned/lofted
prim	vector	*v* derivative or zero vector
sv id	surface type	Surface
prim	real	Tangent factor or 0
sv id	spl_sur	Generic spl_sur data
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: This class defines a skin surface between a list of curves.

Surface Parameterization
The surface parameterization is the *u*-direction corresponds to the parameterization of the curves to be skinned and the *v*-direction corresponds to the cubic Bezier between the skin-curves.

The input to this surface class are the curves to be skinned (all the curves are reparameterized to lie in [0.0 – 1.0] range), optional tangents (the magnitude of the curves’ tangents have to match on the ends) in *u*-direction, and the optional surfaces on which the curves lie. If surfaces containing the curves are provided, these determine the tangent directions in *v*.

Evaluation Process
The evaluation process is a three-step process.

1. If any matching tangent magnitudes are given, the section curves (curves to be skinned) are reparameterized as follows: parameter *t* is the parameter on the original curve. Parameter *u* on the skin surface is determined such that the *u* partial at each end of the skin surface is equal to the matching tangent magnitude.

$$t = f(u)$$

where:

$$f(u) =$$

$$t_s * H_0^3(u) + m_0 * H_1^3 + m_1 * H_2^3(u) + t_e * H_3^3(u)$$

In the above expression, the H’s are the Hermite polynomials and the t_s and t_e are the start and end parameter values of the original curves to be skinned, which here are (0 and 1) respectively.

So, ds/du on the ends are:

$$dc/du = dc/dt * dt/du$$

where dt/du on the ends is m_0 and m_1.

So by choosing the values m_0 and m_1 such that the dc/du on left surface is equal to dc/du on the right surface (provided that the curves are G1), a C1 continuous surface is achieved even when skinning G1 continuous curves.

2. The tangent directions for the *v* are determined by fitting a circle through the points corresponding to the same *u* value on the adjacent section curves to the left and right. The scheme followed is similar to the way Bessel tangents are computed. If there are only two section curves, the circle radius is chosen to be infinity. If the surfaces are given for any section, the tangent direction in *v* when on that curve is obtained by the cross product of surface normal and the section curve tangent at that point. The direction also has an optional scalar value that can be applied. The surface is called a loft surface when such a surface is provided.

3. Now the skin/loft surface is defined using Hermite interpolants between sections that join each other C1 continuously. To evaluate the surface S(*u,v*) at a particular *v*-parameter, the first step is to find the segment to which this parameter corresponds. Then a local parameter v_i is computed, which ranges from 0 to 1. The section curves c_i and c_{i+1}, and the tangents s_i and t_{i+1} are also obtained. The surface is defined as:

$$s(u,v) = c_i(u) * H_0^3(v_i) + t_i * H_1^3(v_i) + t_{i+1} * H_2^3(v_i) + c_{i+1} * H_3^3(v_i)$$

The parametric derivatives of this surface are obtained by differentiating the above equation algebraically.

Component: **ACIS Kernel**

"skinsur2"

Purpose: Defines a skin surface between a list of curves.

Derivation: skin_spl_sur2 : spl_sur : subtrans_object : subtype_object : –

4

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	ctrl	if_cond	if the save version number is less than the SPLINE_VERSION
	sv id	spl_sur	save the spline surface data
	ctrl	else	if the save version number is greater than the SPLINE_VERSION
	prim	integer	Number of curves
	prim	newline	Newline for readablility
	ctrl	repeat	Repeat for the number of curves
	sv id	curve type	Curve to be skinned/lofted
	prim	newline	Newline for readablility
	sv id	spl_sur	Generic spl_sur data
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class defines a skin surface between a list of curves.

Component: **ACIS Kernel**

“sphere”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	SPHERE (1) class	derived from SPHERE class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	sphere (2) class	derived from sphere class

Description: Used to determine which class specified the sphere. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

SPHERE (1) class

Purpose: Defines a sphere as an object in the model.

Derivation: SPHERE : SURFACE : ENTITY : –

Data Elements:	sv id	sphere (2) class	sphere data given in another section of this manual.

Description: A SPHERE is defined by sphere data, which includes a radius and position. The sign of the radius determines the direction of the surface. A positive radius defines a convex sphere, a negative radius a concave sphere.

Component: **ACIS Kernel** Geometry

sphere (2) class

Purpose: Defines a spherical surface.

Derivation: sphere : surface : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
prim	integer	sphere_type; integer for type of sphere
ctrl	else	if save_version_number is greater than the SURFACE_VERSION
prim	string	save identifier; "sphere".
prim	position	center of sphere
prim	real	radius of sphere
ctrl	if_cond	if save_version_number is greater than or equal to SURFACE_VERSION
prim	vector	*uv* origin direction
prim	vector	pole direction
prim	logical	Either "forward_v" or "reverse_v"
sv id	surface (2) class	Generic surface data. Refer to another section of this manual.
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: A sphere is defined by a center point and radius. Normally, the radius is positive (i.e., material inside the surface), but it can be negative, indicating a "hollow" sphere (i.e., material outside the surface).

The *u*-parameter is the latitude metric, running from –pi/2 at the south pole through 0 at the equator to pi/2 at the north pole. The *v*-parameter is the longitude metric, running from –pi to pi, with 0 on the meridian containing ori_dir, and increasing in a clockwise direction around pole_dir, unless reverse_v is TRUE.

4

Let *P* be pole_dir and *Q* ori_dir, and let *R* be *P* x *Q*, negated if reverse_v is TRUE. Let *r* be the absolute value of the sphere radius. Then:

```
pos = center + r* sin(u)* P + r* cos(u)*
      (cos(v)* Q + sin(v) R)
```

This parameterization is left-handed for a convex sphere and right-handed for a hollow one, if reverse_v is FALSE, and reversed if it is TRUE.

When the sphere is transformed, the sense of reverse_v is inverted if the transform includes a reflection. No special action is required for a negation.

In summary, spheres are:

– Not true parametric surfaces.
– Periodic in *v* (–pi to pi with period 2pi) but not in *u*.
– Closed in *v* but not in *u*.
– Singular in *u* at the poles; nonsingular everywhere else.

Component: **ACIS Kernel** Geometry

“spline”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
sv id	SPLINE (1) class	derived from SPLINE class
ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
sv id	spline (2) class	derived from spline class

Description: Used to determine which class specified the spline. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

SPLINE (1) class

Purpose: Defines a parametric surface as an object in the model.

Derivation: SPLINE : SURFACE : ENTITY : –

Data Elements:	sv id	spline (2) class	Spline surface definition

Description: A SPLINE records a parametric surface as a spline. In turn, a SPLINE holds a pointer to a spl_sur and a logical denoting reversal of the sense of the stored surface.

Component: **ACIS Kernel** Geometry

spline (2) class

Purpose: Records a B-spline surface.

Derivation: spline : surface : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
prim	integer	spline type
ctrl	else	
prim	save ID	spline's save ID
ctrl	if_cond	if save_version_number is less than SPLINE_VERSION
ctrl	case_cond	specific surface definition subtype
sv id	"exactsur"	if spl_sur is exact_spl_sur or "exact_sur"
sv id	"offsur"	if spl_sur is off_spl_spl_sur or "offsur"
sv id	"pipesur"	if spl_sur is pipe_spl_sur or "pipesur"
sv id	"rbblnsur"	if spl_sur is rb_blend_spl_sur or "rbblsur"
sv id	"rotsur"	if spl_sur is rot_spl_sur or "rotsur"
sv id	"skinsur"	if spl_sur is skin_spl_sur or "skinsur"
sv id	"subsur"	if spl_sur is sub_spl_sur or "subsur"
sv id	"sumsur"	if spl_sur is sum_spl_sur or "sum_sur"
sv id	"sweepsur"	if spl_sur is sweep_spl_sur or "sweep_sur"
sv id	"tubesur"	if spl_sur is tube_spl_sur or "tube_sur"
sv id	"vertexblendsur"	if spl_sur is VBL_SURF or "vertexblendsur"
sv id	"varblendsplsur"	if spl_sur is exact_spl_sur or "exact_sur"
ctrl	else	
prim	logical	"forward" or "reversed"
ctrl	case_cond	specific surface definition subtype
sv id	"exactsur"	if spl_sur is exact_spl_sur or "exact_sur"

4

sv id	"offsur"	if spl_sur is off_spl_spl_sur or "offsur"
sv id	"pipesur"	if spl_sur is pipe_spl_sur or "pipesur"
sv id	"rbblnsur"	if spl_sur is rb_blend_spl_sur or "rbblsur"
sv id	"rotsur"	if spl_sur is rot_spl_sur or "rotsur"
sv id	"skinsur"	if spl_sur is skin_spl_sur or "skinsur"
sv id	"subsur"	if spl_sur is sub_spl_sur or "subsur"
sv id	"sumsur"	if spl_sur is sum_spl_sur or "sum_sur"
sv id	"sweepsur"	if spl_sur is sweep_spl_sur or "sweep_sur"
sv id	"tubesur"	if spl_sur is tube_spl_sur or "tube_sur"
sv id	"vertexblendsur"	if spl_sur is VBL_SURF or "vertexblendsur"
sv id	"varblendsplsur"	if spl_sur is exact_spl_sur or "exact_sur"
sv id	surface (2) class	Generic surface data
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description:

The spline class represents a parametric surface that maps a rectangle within a 2D real vector space (parameter space) into a 3D real vector space (object space). This mapping must be continuous, and one-to-one except possibly at the boundary of the rectangle in parameter space. It is differentiable twice, and the normal direction is continuous, though the derivatives need not be. The positive direction of the normal is in the sense of the cross product of the partial derivatives with respect to *u* and *v* in that order. The portion of the neighborhood of any point on the surface that the normal points to is outside the surface, and the other part is inside.

Opposite sides of the rectangle can map into identical lines in object space, in which case the surface is closed in the parameter direction normal to those boundaries. If the parameterization and derivatives also match at these boundaries, the surface is periodic in this parameter direction. The line in object space corresponding to the coincident boundaries is known as the seam of a periodic surface.

If a surface is periodic in one parameter direction, it is defined for all values of that parameter. A parameter value outside the domain rectangle is brought within the rectangle by adding a multiple of the rectangle's width in that parameter direction, and the surface evaluated at that value. If the surface is periodic in both parameters, it is defined for all parameter pairs (*u,v*), with reduction to standard range happening with both parameters.

One side of the rectangle can map into a single point in object space. This point is a parametric singularity of the surface. If the surface normal is not continuous at this point, it is a surface singularity.

The spline contains a “reversed” bit together with a pointer to another structure, a spl_sur or something derived from it, that contains the bulk of the information about the surface.

Component: **ACIS Kernel** Geometry

“split”

Purpose: Attached to each edge of each body which has a graph vertex properly within it.

Derivation: ATTRIB_SPLIT : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:

prim	No data	This class does not save any data

Description: For internal use only. This attribute is attached to each edge of each affected body which has a graph vertex properly within it. It simply contains a list of all the graph vertices, in order along the edge, so that when the edge is split at one of them, the others may be associated with the correct piece for later splitting.

Component: **ACIS Kernel** Attributes

spl_sur

Purpose: Defines an abstract base class from which spline surface definitions are derived.

Derivation: spl_sur : subtrans_object : subtype_object : –

Data Elements:

sv id	bs3_surface_def	B-spline approximation of surface
prim	real	Fit tolerance

Description: In ACIS a sculptured surface is represented by the class spline, which contains a pointer to an internal description called spl_sur. The spl_sur further contains a bs3_surface that is a pointer to a rational or nonrational, nonuniform B-spline surface in the underlying surface package.

Component: **ACIS Kernel**

4

“spring”

Purpose: Marks edges lying on spring curves so they may be specially handled later.

Derivation: ATTRIB_SPRING : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: This class marks edges lying on spring curves so they may be specially handled later. The attribute refers to the face of the blended edge from which the blend springs.

Component: ACIS Kernel

“spring_load”

Purpose: For internal use only.

Derivation: ATTRIB_DS_SPRING : ATTRIB_DSLOAD : ATTRIB : ENTITY : –

Data Elements:	prim	long	asp_domain_dim
	ctrl	repeat	Repeat for all asp_domain_dim
	prim	real	asp_domain_pt[ii]
	prim	long	asp_image_dim
	ctrl	repeat	Repeat for all asp_image_dim
	prim	real	asp_free_pt[ii]

Description: Refer to Purpose.

Component: **Deformable Surface Husk**

“spring_set_load”

Purpose: Set of spring loads.

Derivation: ATTRIB_SPRING_SET : ATTRIB_DSLOAD : ATTRIB : ENTITY : –

Data Elements:	prim	long	ass_pt_count
	prim	long	ass_domain_dim
	prim	long	ass_image_dim
	ctrl	repeat	Repeat for all ass_pt_count * ass_domain_dim
	prim	real	ass_domain_pt[ii]
	prim	repeat	Repeat for all ass_pt_count * ass_image_dim
	prim	real	ass_free_pt[ii]

Description: For internal use only.

4

Component: **Deformable Surface Husk**

“srfsrfblndsur”

Purpose: Implements the variable-radius face-face blend surface.

Derivation: srf_srf_v_bl_spl_sur : var_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements: prim No data This class does not save any data

Description: This class implements the surface geometry of a variable radius blend between two surfaces. The blend will be tangent to both surfaces.

Component: ACIS Kernel

“st”

Purpose: Organization attribute from which various color, display, id, and other attributes are derived.

Derivation: ATTRIB_ST : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class is an attribute declaration for a private container attribute. This class derives from the ACIS base class, ATTRIB. Each application developer receives a customized attribute declaration. The application developer then makes all attributes specific to the application-derived classes of this attribute, ensuring that different developers can assign identifiers independently without interference.

Component: ACIS Kernel

“sti_elat_attr”

Purpose: Creates a temporary attribute used in sweeping.

Derivation: ATTRIB_STI_ELAT_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

“sti_nor_attr”

Purpose:	Creates a temporary attribute used in sweeping.		
Derivation:	ATTRIB_STI_NOR_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

“sti_prof_attr”

Purpose:	Creates a temporary attribute used in sweeping.		
Derivation:	ATTRIB_STI_PROF_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

4

“sti_psplit_attr”

Purpose:	Creates a temporary attribute used in sweeping.		
Derivation:	ATTRIB_STI_PSPLIT_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

“sti_rel_attr”

Purpose:	Creates a temporary attribute used in sweeping.
Derivation:	ATTRIB_STI_REL_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

"sti_sect_attr"

Purpose:	Creates a temporary attribute used in sweeping.		
Derivation:	ATTRIB_STI_PSPLIT_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

"sti_vlat_attr"

Purpose:	Creates a temporary attribute used in sweeping.		
Derivation:	ATTRIB_STI_VLAT_ATTR : ATTRIB_SG : ATTRIB : ENTITY : –		
Data Elements:	prim	No data	This class does not save any data
Description:	For internal use only.		
Component:	**ACIS Kernel** Attributes		

"straight"

Purpose:	Identifier used by more than one class.		
Derivation:	None		
Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	STRAIGHT (1) class	derived from STRAIGHT class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	straight (2) class	derived from straight class

4

Description: Used to determine which class specified the straight. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

STRAIGHT (1) class

Purpose: Defines an infinite line as an object in the model.

Derivation: STRAIGHT : CURVE : ENTITY : –

Data Elements:

sv id	straight (2) class	Line definition

Description: A STRAIGHT is defined by a point (position) on an infinite line and its direction (unit_vector).

Component: **ACIS Kernel** Geometry

straight (2) class

Purpose: Defines an infinite straight line represented by a point and a unit vector specifying the direction.

Derivation: straight : curve : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the CURVE_VERSION
prim	integer	straight type
ctrl	else	
prim	string	save identifier; "straight"
prim	position	Root point
prim	vector	Direction
sv id	curve (2) class	Generic curve data
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: This class defines an infinite straight line represented by a point and a unit vector specifying the direction. A straight also has a scale factor for the parametrization, so the parameter values can be made invariant under transformation. A straight line is an open curve that is not periodic. It is parameterized as:

4

```
point = root_point + t* param_scale* direction
```

where t is the parameter.

Component: **ACIS Kernel** Geometry

“string_attrib”

Purpose: Defines a generic attribute that contains a string value.

Derivation: ATTRIB_GEN_STRING : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –

Data Elements:

ctrl	if_cond	if value is not equal to NULL
prim	string	Value
ctrl	else	
prim	NULL	indicator for empty string

Description: Refer to the Purpose.

Component: Generic Attribute Husk

“stripc”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
sv id	STRIPC (1) class	derived from STRIPC class
ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
sv id	stripc (2) class	derived from stripc class

Description: Used to determine which class specified the stripc. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

4

STRIPC (1) class

Purpose: Records a parametric surface as a STRIPC.

Derivation: STRIPC : SURFACE : ENTITY : –

Data Elements:

sv id	stripc (2) class	Strip curve definition

Description: A class derived from SURFACE, records a parametric surface as a (lowercase) stripc.

Component: **ACIS Kernel** Geometry

stripc (2) class

Purpose: The strip curve (stripc) is a surface defined in a neighborhood of and passing through a given object-space curve.

Derivation: stripc : surface : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, "{" or Tag 15
ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
prim	integer	stripc_type; integer for type of stripc type
ctrl	else	if save_version_number is greater than the SURFACE_VERSION
prim	string	save identifier; "stripc".
sv id	curve type	Object space curve
sv id	surface type	Surface on which curve lies
prim	boolean	Parameter space curve reversed
sv id	bs2_curve_def	Parameter space curve
prim	real	Fit tolerance for parameter space curve
prim	newline	
prim	logical	*v*-parameter: "forward_v" or "reversed_v" with respect to right hand rule
sv id	surface (2) class	Generic surface data
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: The strip curve (stripc) is a surface defined in a neighborhood of and passing through a given object-space curve, which is everywhere perpendicular to a given surface in which the curve lies. The surface is a strip two resabs wide, centered on the curve and normal to the surface the strip curve lies in. It is only guaranteed to be well behaved in a neighborhood of the curve. It is used for giving a sense to a curve lying in a surface, allowing portions of the surface on either side of the curve to be distinguished. It may not be used as the surface of a FACE.

The parameterization of the strip curve is determined by the object-space curve and an additional item reverse_v.

Given a surface parameter value (u,v), the underlying curve is evaluated at parameter u, to give position P and first derivative U. The underlying surface normal is then obtained, and negated if reverse_v is TRUE, giving N. Then the evaluated position is:

```
pos = P + v * |U| * N
```

This is significant only for infinitesimal values of v, but demonstrates how derivatives can be defined. In this implementation, second derivatives and curvatures are not accurate for nonzero v-parameter; i.e., off the defining curve and surface.

When a strip curve is transformed, its underlying curve and surface are transformed, and then, if the transformation includes reflection, the surface is negated and reverse_v inverted. Negate the strip curve, negate the supporting surface and invert reverse_v.

Component: **ACIS Kernel** Geometry

"subsetintcur"

Purpose: Represents a subset of a longer curve.

Derivation: subset_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, "{" or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	int_cur	Generic int_cur data
enum	CURVE_EXTENSION_TYPE	Extension type
sv id	curve type	Original curve
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: Refer to the Purpose.

Component: **ACIS Kernel**

“subshell”

Purpose: Represents a subdivision of a SHELL or SUBSHELL.

Derivation: SUBSHELL : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for parent subshell
prim	$rec_num	Pointer to record in save file for next subshell belonging to parent
prim	$rec_num	Pointer to record in save file for first child subshell
prim	$rec_num	Pointer to record in save file for first face in subshell
prim	$rec_num	Pointer to record in save file for first wire in subshell

Description: A subshell represents a subdivision of a SHELL or SUBSHELL. It allows groups of WIREs and FACEs to be excluded by a single box test and improves the efficiency of many-to-many comparisons. The subdivision is determined by the system, and may change at any time. The SUBSHELL has no significance to the end user (the application programmer may find the implied spatial subdivision useful).

Component: **ACIS Kernel** Topology

4

“subsur”

Purpose: Represents the geometry of a spline surface, which is a subset region of another spl_sur.

Derivation: sub_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
prim	interval	*u* range
prim	interval	*v* range
sv id	surface type	Spline, original surface
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents the geometry of a spline surface, which is a subset region of another spl_sur. The subset *uv* range may be smaller or larger than the range of the progenitor surface, or it may overlap it.

Component: **ACIS Kernel**

“sumsur”

Purpose: Represents a linear sum of two curves.

Derivation: sum_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
ctrl	if_cond	if save_version_number is less than SPLINE_VERSION
sv id	spl_sur	save data
ctrl	else	
prim	newline	
sv id	curve type	*u* curve
prim	newline	
sv id	curve type	*v* curve
prim	newline	
prim	position	Datum point
prim	newline	
prim	interval	*u* range
prim	interval	*v* range
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents a surface that is a linear sum of two curves. This is derived from the class spl_sur, which is used by the spline surface class to contain the surface descriptions. The surface is defined primarily by two curves that are assumed not parallel, and the parameter ranges over which the surface is defined.

Parametric Representation
If the curves are represented as:

```
x = c1(t)  and x=c2(t)
```

respectively, the surface is:

```
x=s(u,v) = c1(u) + c2(v) - p
```

where *p* is a constant position, normally initialized to be the value of *c2* at the start of the parameter range.

4

Component: **ACIS Kernel**

subtype_object

Purpose: Defines the master object from which all subtype objects must be derived.

Derivation: subtype_object : –

Data Elements:

prim	No data	This class does not save any data

Description: This class defines the master object from which all subtype objects must be derived. This object contains a use count (in case the object is shareable) and defines two virtual functions and a destructor.

Component: ACIS Kernel

“supercell”

Purpose: Identifies a grouping of cells or inferior supercells.

Derivation: SUPERCELL : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for parent SUPERCELL
prim	$rec_num	Pointer to record in save file for next SUPERCELL (sibling)with same parent
prim	$rec_num	Pointer to record in save file for first child SUPERCELL
prim	$rec_num	Pointer to record in save file for first cell contained in SUPERCELL

Description: This represents a grouping of cells or inferior supercells. It allows the system to improve the efficiency of many-to-many comparisons, by allowing quantities of cells to be excluded by a single box test. The subdivision is determined by the system, and may change at any time, so the supercell has no significance to the user (though the application program may find the spatial subdivision implied useful).

Component: Cellular Topology Husk

"surface"

Purpose: Defines a generic surface as an object in the model.

Derivation: SURFACE : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: A SURFACE provides the basic framework for the range of surface geometries implemented in the modeler. It may be referenced by more than one entity.

Component: **ACIS Kernel** Geometry

surface (2) class

Purpose: Base class for all ACIS surface types that defines the basic virtual functions that are supplied for all specific surface classes.

Derivation: surface :–

Data Elements:	ctrl	if_cond	if save_version_number is greater than or equal to BNDSUR_VERSION
	prim	interval	*u* range
	prim	interval	*v* range

Description: The surface class is the base class that all ACIS surface types (plane, cone, sphere, torus, and spline) are derived. The surface class defines the basic virtual functions that are supplied for all specific surface classes. Some of these functions are pure; i.e., the derived classes must define their own version; others have default definitions that can be used by the derived classes.

Component: **ACIS Kernel** Geometry

surface type

Purpose: More detailed definition of surface.

Derivation: surface :–

Data Elements:	prim	ident	Surface type
	ctrl	if_cond	if Surface type is set to “null_surface” No surface saved
	ctrl	if_cond	if Surface type is set to “plane”
	sv id	plane (2) class	Plane definition
	ctrl	if_cond	if Surface type is set to “cone”
	sv id	cone (2) class	Cone definition
	ctrl	if_cond	if Surface type is set to “sphere”
	sv id	sphere (2) class	Sphere definition
	ctrl	if_cond	if Surface type is set to “torus”
	sv id	torus (2) class	Torus definition
	ctrl	if_cond	if Surface type is set to “spline”
	sv id	spline (2) class	Spline definition
	ctrl	if_cond	if Surface type is set to “stripc”
	sv id	stripc (2) class	Strip curve definition

Description: Refer to purpose.

Component: **ACIS Kernel** Geometry

“surfcur”

Purpose: Represents a spline curve projected onto a surface within the given fit tolerance.

Derivation: surf_int_cur : int_cur : subtrans_object : subtype_object : –

4

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	ctrl	if_cond	if save_version_number is less than INTCURVE_VERSION
	sv id	bs3_curve_def	
	prim	real	fit tolerance
	prim	newline	
	sv id	surface (2) class	surface 1
	prim	newline	
	sv id	surface (2) class	surface 2
	prim	newline	
	sv id	bs2_curve_def	surface 1
	prim	newline	
	sv id	bs2_curve_def	surface 2
	ctrl	else if_cond	if save_version_number is less than PARCUR_VERSION
	sv id	bs3_curve_def	
	prim	real	fit tolerance
	prim	newline	
	sv id	surfase data	surface 2
	prim	newline	
	sv id	surface (2) class	surface 1
	prim	newline	
	sv id	bs2_curve_def	surface 2
	prim	newline	
	sv id	bs2_curve_def	surface 1
	ctrl	else	
	sv id	int_cur	Generic int_cur data
	prim	logical	Consistent version: “surf2” or “surf1”
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Refer to the Purpose.

Component: **ACIS Kernel**

“surfintcur”

Purpose: Represents the spline curves obtained from the intersection of two surfaces.

Derivation: int_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	int_cur	Generic int_cur data
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents the spline curves obtained from the intersection of two surfaces. The given surfaces must not be tangential, except at the ends of the parameter range.

Component: **ACIS Kernel**

“sweepsur”

Purpose: Defines the perpendicular sweep of a planar profile curve along a path curve.

Derivation: sweep_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
prim	boolean	Path normal to plane of profile
sv id	curve type	Profile curve
sv id	curve type	Path curve
prim	boolean	Path normal to plane of profile
prim	vector	Normal to plane of profile
prim	position	Start of path
prim	vector	Normal to plane of path
prim	vector	Cross product of path normal and start tangent
prim	vector	Start tangent of path
prim	real	Start *u*-parameter
prim	real	End *u*-parameter
prim	real	Start *v*-parameter
prim	real	End *v*-parameter
prim	real	Start draft distance
prim	real	Added draft distance
sv id	spl_sur	Generic spl_sur data
prim	subtype_end	Right curly braces, “}” or Tag 16

4

Description: This class defines the perpendicular sweep of a planar profile curve along a path curve. The start of the path is in the plane of the shape curve.

Curve and Path Restrictions
The profile curve must be a planar curve. The path curve must be a planar curve and must start in the plane of the profile.

Surface Equation
The surface equation s(u,v) for the sweep surface is defined as:

$$s(u, v) = p(v) + f(u) + g(u, v) + zc(u, v) + dr(u, v)$$

where:

p(v) evaluates the path curve at *v*-parameter value.

*f(u) = ((s(u) – path_start) . path_normal) * path_normal*

s(u) evaluates the shape curve at *u*-parameter value.

*g(u, v) = ((s(u) – path_start) . (path_normal * path_start_dir)) * xd(v)*

*xd(v) = path_normal * pd(v)*

$$pd(v) = \frac{p'(v)}{\| p'(v) \|}$$

pd(v) is the normalized path tangent at the *v*-parameter.

*zc(u, v) = ((s(u) – path_start) * path_start_dir) (pd(v)*

This *zc(u,v)* component is nonzero only when the shape curve that is being swept is not in the plane that normal is path_start_dir

dr(u,v) is the draft component calculated similarly as above same equations to the draft vector at (*u,v*) values. This component is 0 if the draft distance specified is 0.

Component: **ACIS Kernel**

“sys”

Purpose: Defines a base class for the **ACIS Kernel**.

Derivation: ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: The base class from which all attributes defined in the **ACIS Kernel** are derived.

Component: **ACIS Kernel** Attributes

“t3”

Purpose: Implements planar triangular elements.

Derivation: TRI3_ELEM : ELEM2D : ELEM : ENTITY : –

Data Elements:	sv id	"2d"	Save parent class ELEM2D

Description: Contains all of the information for each triangular patch on the mesh surface.

Component: **Mesh Surface Husk**

"tan_xedge"

Purpose: Attaches an attribute to the cross edges of the blend sheet.

Derivation: ATTRIB_TAN_XEDGE : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	No data	This class does not save any data

Description: Attribute that attaches to cross edges of the blend sheet, that meet the blend body tangentially. These occur when the sheet face has been made for a blended cuspate vertex. The unblended edge records the curve where the sheet edge lies. This is used internally during blend1, but should be removed from the sheet at the end of this stage.

Component: ACIS Kernel

"tapersil"

Purpose: Creates an interpolated curve subtype which can precisely represent a silhouette curve formed by applying a taper.

Derivation: taper_silh_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, "{" or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	int_cur	Save the interpolated curve data
	prim	vector	direction of curve
	prim	real	taper angle
	prim	subtype_end	Right curly braces, "}" or Tag 16

Description: Refer to Purpose.

Component: **ACIS Kernel**

“tapersur”

Purpose: Creates an edge–tapered surface.

Derivation: taper_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	spl_sur	save specific spline surface data
sv id	int_cur	save specific interpolated curve data
prim	vector	direction of taper
prim	real	sine of angle
prim	real	cosine of angle
prim	interval	*u* range
prim	interval	*v* range
prim	integer	closed in *u*
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Class to describe an edge–tapered surface, in which a surface is tapered by a constant angle, relative to a draft angle, about an edge. The surface is a ruled surface between two u parameter curves.

Component: **ACIS Kernel**

“text_ent”

Purpose: Routine to restore a TEXT_ENT entity from file.

Derivation: TEXT_ENT : ENTITY : –

Data Elements:

prim	position	Location of baseline at start of first character.
prim	string	String to be displayed.
prim	string	Font style used to display string. (NULL means to use the current font).
prim	integer	Size in points of font used to display string.

Description: These items define the font and the size of the text, as well as the text itself and its location.

Component: Geometry Husk

"torus"

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:

ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
sv id	TORUS (1) class	derived from TORUS class
ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
sv id	torus (2) class	derived from torus class

Description: Used to determine which class specified the torus. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

TORUS (1) class

Purpose: Defines a torus as an object in the model.

Derivation: TORUS : SURFACE : ENTITY : –

Data Elements: sv id torus (2) class Torus definition

Description: A TORUS is a circular thickening of a circular spine, defined by a center, normal, major radius, and minor radius.

The minor radius is the radius of the thickening circle. The major radius is the radius of the spine from its center point. The normal determines the orientation in space of the torus.

Component: **ACIS Kernel** Geometry

torus (2) class

Purpose: Represents tori.

Derivation: torus : surface : –

Data Elements:	ctrl	if_cond	if used as a subtype reference
	prim	subtype_start	Left curly braces, “{” or Tag 15
	ctrl	if_cond	if save_version_number is less than the SURFACE_VERSION
	prim	integer	torus type
	ctrl	else	
	prim	string	save identifier for torus “torus”
	prim	position	center
	prim	vector	normal
	prim	real	major radius
	prim	real	minor radius
	ctrl	if_cond	if save_version_number is greater than or equal to SURFACE_VERSION
	prim	vector	*uv* origin direction
	prim	logical	“forward_v” or “reverse_v”
	sv id	surface (2) class	information about surface; refer to another section of this manual.
	ctrl	if_cond	if used as a subtype reference
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description:

A torus is defined by a circular spine and a circular cross-section at each point on the spine. The spine of a torus is defined by a center point, normal, and major radius. The circular cross-section is defined by a minor radius.

A normal torus (donut) is defined when the major radius is larger than the minor radius. Special degenerate cases (lemon, vortex, and apple) occur if the major radius is smaller than or equal to the minor radius. Two data members define the parameterization of the torus:

unit_vector uv_oridir
Direction from the center of the torus to the origin of parameter space.

logical reverse_v . .
Constant *u*-parameter lines are circles around the torus axis, normally clockwise, but reversed if this is TRUE.

The *u*-parameter is the latitude, with zero on the circle of greatest radius about the torus axis, and the positive direction in the direction of the torus axis. The *u*-parameter range depends upon the relative values of the major and minor radii. For a doughnut, where the major radius is greater than the magnitude of the minor, it runs from –pi to pi, and is periodic. For degenerate tori, where the magnitude of the major axis is less than that of the minor, it runs from $-U$ to U, where $U = arccos(-maj / |min|)$, and the surface is singular at each end of the range. The *v*-parameter is the longitude, running from –pi to pi, with 0 on the meridian containing uv_oridir, and increasing in a clockwise direction around the torus axis, unless reverse_v is TRUE, when it increases in an counterclockwise direction.

4

Let *N* be normal and *Q* uv_oridir, and let *R* be *N* X *Q*, negated if reverse_v is true. Let *r* be the absolute value of the minor radius. Then:

```
pos = center + r* sinu* N +
(major_radius + (r* cosu)) * (cosv* Q + sinv R)
```

This parameterization is left-handed for a convex torus and right-handed for a hollow one, if reverse_v is false, and reversed if it is TRUE. When the torus is transformed, the sense of reverse_v is inverted if the transform includes a reflection. No special action is required for a negation.

In summary:

- Tori are not true parametric surfaces.
- Tori are closed in *v* but may or may not be closed in *u*.
- Degenerate tori are not periodic in *u*; nondegenerate tori are periodic in *u* (–pi to pi with period 2pi).
- All tori are periodic in *v* (with range –pi to pi, and period 2pi).
- Degenerate tori are singular in *u* at the poles (apices); all other values of *u* and *v* are nonsingular.

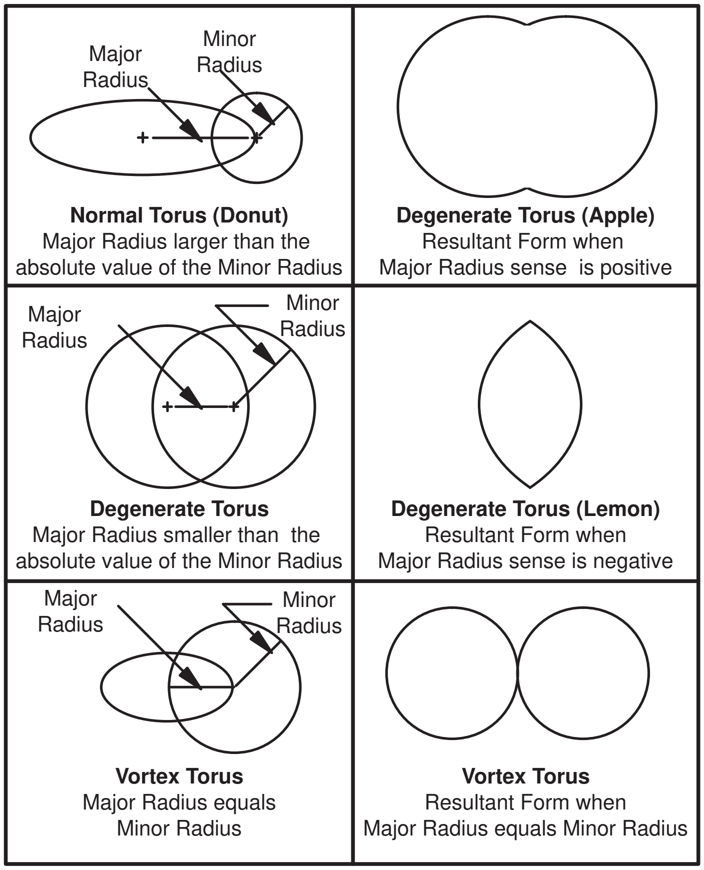

Figure 4-3. torus Class Definition

Component: **ACIS Kernel** Geometry

“transform”

Purpose: Represents an overall transformation applied to a BODY.

Derivation: TRANSFORM : ENTITY : –

Data Elements: prim transform Transformation matrix

Description: The TRANSFORM class represents an overall transformation applied to a BODY. TRANSFORM allows object-space transformations to be applied without the need to recompute the BODY geometry.

It allows a general affine transformation, but records the separate elements of the transformation (scaling, rotation, translation, etc.) to simplify the task of geometry transformation in the common case of solid-body transformations.

Component: **ACIS Kernel** Geometry

"tri3sur"

Purpose: Represents a mesh surface consisting only of planar triangular elements.

Derivation: tri3_msh_sur : msh_sur : –

Data Elements: prim No data This class does not save any data

Description: Depicts a mesh surface consisting only of planar triangular elements and is derived from the abstract msh_sur class.

Component: ACIS Kernel

"tsl"

Purpose: Defines a base class for a specific application developer.

Derivation: ATTRIB_TSL : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: Identifier used externally to identify an particular entity type. This is only used within the save/restore system for translating to/from external file format, but must be unique amongst attributes derived directly from ATTRIB, across all application developers.

Component: **ACIS Kernel** Attributes

"tubesur"

Purpose: A surface that is the envelope of a fixed-radius circle.

Derivation: tube_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	prim	subtype_start	Left curly braces, “{” or Tag 15
	prim	write sv id	save identifier for this particular subtype
	sv id	spl_sur	spline surface
	prim	real	Radius
	prim	newline	
	sv id	curve type	spine curve
	prim	newline	
	prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class represents a surface that is the envelope of a fixed-radius circle centered on a point on a given curve, and normal to the curve at each point. This has been replaced by the new pipe_spl_sur.

Component: **ACIS Kernel**

“undefc”

Purpose: Identifier used by more than one class.

Derivation: None

Data Elements:	ctrl	if_cond	if not a subtype reference; save identifier appended to beginning of record, while its data is appended to the end of the record.
	sv id	UNDEFC (1) class	derived from UNDEFC class
	ctrl	else	it is a subtype reference; save identifier is followed immediately by its data, both enclosed by subtype_start and subtype_end.
	sv id	undefc (2) class	derived from undefc class

Description: Used to determine which class specified the undefc. A subtype reference is inline with a definition and is surrounded by curly braces { }, or Tag 15 and 16.

Component: **ACIS Kernel** Geometry

UNDEFC (1) class

Purpose: Defines a curve that is undefined except for its end points.

Derivation: UNDEFC : CURVE : ENTITY : –

Data Elements:	sv id	undefc (2) class	Curve definition

Description: This class defines an undefined curve that records the start and end points, directions, and curvatures. The start point has a parameter value of 0 and the end point has a parameter value of 1. No other points are defined.

Component: **ACIS Kernel** Geometry

undefc (2) class

Purpose: Denotes a curve that is undefined except for its end points, for which there are explicit positions, directions, and curvatures.

Derivation: undefc : curve : –

Data Elements:

ctrl	if_cond	if used as a subtype reference
prim	subtype_start	Left curly braces, “{” or Tag 15
ctrl	if_cond	if save_version_number is less than the CURVE_VERSION
prim	integer	straight type
ctrl	else	
prim	string	save identifier; “undefc”
prim	position	Start point
prim	vector	Start direction
prim	vector	Start curvature
prim	position	End point
prim	vector	End direction
prim	vector	End curvature
sv id	curve (2) class	Generic curve data
ctrl	if_cond	if used as a subtype reference
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class denotes a curve that is undefined except for its end points, for which there are explicit positions, directions, and curvatures. This class is used in blending to allow the blend surface to spread at its ends. It may be useful elsewhere, as well.

The curve is parameterized so that the start point has parameter 0, and the end point has parameter 1. No other point lies on the curve so the parameter value is meaningless, but it returns as 0.5.

Component: **ACIS Kernel** Geometry

“units”

Purpose: Specifies the units a model is defined in.

Derivation: UNITS_SCALE : ENTITY : –

Data Elements:

prim	real	Model scale
prim	real	Input scale
prim	real	Output scale

Description: Implements the UNITS_SCALE class. A UNITS_SCALE ENTITY is used to specify what units a model is defined in. It contains a scale factor which specifies the conversion factor between model units and millimeters.

Component: Geometry Husk

“unknown”

Purpose: Represents common data and functionality that is mandatory in all classes that are permanent objects in the model.

Derivation: ENTITY : –

Data Elements:

sv id	ENTITY	used with the ENTITY class

Description: Refer to purpose.

Component: **ACIS Kernel**

“varblendsplsur”

Purpose: Implementation of the base class for variable radius and other nonpipe blends. Derived from blend_spl_sur.

Derivation: var_blend_spl_sur : blend_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
sv id	blend_spl_sur	Generic blend surface data
sv id	curve type	Slicing plane curve
prim	logical	Either “concave” or “convex”
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: Implementation of the base class for variable radius and other nonpipe blends. Derived from blend_spl_sur. This class name does not appear in the save file, but is a base class for other subtype identifiers that do appear in the save file.

Component: **ACIS Kernel**

var_cross_section

Purpose: Represents the cross section of a blend surface.

Derivation: var_cross_section : –

Data Elements:

ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
prim	integer	form_data
ctrl	else	
enum	cross_section_forms	Form
ctrl	if_cond	if cross section form is set to thumbweights
prim	real	Left thumbweight
prim	real	Right thumbweight
ctrl	else if_cond	if cross section form is set to round chamfer
prim	logical	“no_radius” or “radius”
ctrl	if_cond	if height saved
sv id	var_radius	Round height specification

Description: The cross section of a blend surface corresponds to the *u*-parameter of the surface. When evaluating the surface, a slice is taken at the given *v*-parameter, and then that slice is evaluated at the *u*-parameter.

During the initial construction of the blend surface geometry, the cross section is irrelevant. It first comes into play when the blend surface is intersected with other faces.

Only circular cross sections are available in standard blending. The **Advanced Blending Husk** allows other shapes. Parameterization runs from 0 to 1, and that covers the whole section.

Component: **ACIS Kernel**

var_rad_fixed_width

Purpose:	Represents data for a variable radius blend with a fixed width.		
Derivation:	var_rad_fixed_width : var_radius :–		
Data Elements:	prim	real	Width
Description:	Refer to Purpose.		
Component:	**ACIS Kernel**		

var_rad_functional

Purpose:	Represents data for a variable radius blend using a radius defined by a function.		
Derivation:	var_rad_functional : var_radius :–		
Data Elements:	sv id	bs2_curve_def	Radius function
Description:	Refer to Purpose.		
Component:	**ACIS Kernel**		

var_radius

Purpose:	Defines variable radius information for a variable radius blend.		
Derivation:	var_radius : –		
Data Elements:	enum	rad_form_ents	Form
	prim	boolean	Two ends calibrated
	prim	real	Radius start parameter
	prim	real	Radius end parameter
	ctrl	if_cond	if radius form is set to two ends
	sv id	var_rad_two_ends	Specific radius data
	ctrl	if_cond	if radius form is set to functional
	sv id	var_rad_functional	Specific radius data
	ctrl	if_cond	if radius form is set to elliptical
	sv id	var_rad_rot_ellipse	Specific radius data
	ctrl	if_cond	if radius form is set to fixed width
	sv id	var_rad_fixed_width	Specific radius data

4

Description: This class defines a variable radius. Start and end parameters must always be set to something reasonable, even if it's not calibrated. The parameter range of an edge being blended is reasonable.

Component: **ACIS Kernel**

var_rad_rot_ellipse

Purpose: Represents data for an elliptical variable radius blend.

Derivation: var_rad_rot_ellipse : var_radius :–

Data Elements:

prim	real	Start major radius
prim	real	End major radius
prim	real	Start minor radius
prim	real	End minor radius
prim	real	Start rotation angle
prim	real	End rotation angle
prim	boolean	Left face is reference

Description: Refer to Purpose.

Component: **ACIS Kernel**

var_rad_two_ends

Purpose: Represents data for a blend using a variable radius at both ends.

Derivation: var_rad_two_ends : var_radius :–

Data Elements:

prim	real	Start radius
prim	real	End radius

Description: Refer to the Purpose.

Component: **ACIS Kernel**

"vblend"

Purpose: Defines the vertex blend attribute.

Derivation: ATTRIB_VBLEND : ATTRIB_BLEND : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	real	Bulge
	ctrl	if_cond	if save_version_number is less than CONSISTENT_VERSION
	prim	integer	Continuity
	ctrl	else	
	enum	bl_continuity	Continuity

Description: Refer to purpose.

Component: **ACIS Kernel** Attributes

"vector_attrib"

Purpose: Defines a generic attribute that contains a vector.

Derivation: ATTRIB_GEN_VECTOR : ATTRIB_GEN_NAME : ATTRIB_GENERIC : ATTRIB : ENTITY : –

Data Elements:	prim	vector	Value

Description: Refer to the Purpose.

Component: Generic Attribute Husk

"vertedge"

Purpose: Contains a list of edge pointers.

Derivation: ATTRIB_VERTEDGE : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements:	prim	integer	Number of edges
	ctrl	repeat	Repeat for the number of edges
	prim	$rec_num	Pointer to record in save file for an edge which use vertex

Description: This is used to contain the edge pointer list if there should be more than one pointer. At nonmanifold vertices, there should be apointer to an edge in each separable manifold region.

Component: **ACIS Kernel** Attributes

“vertex”

Purpose: Represents an end of an EDGE.

Derivation: VERTEX : ENTITY : –

Data Elements:

prim	$rec_num	Pointer to record in save file for an edge which uses vertex
prim	$rec_num	Pointer to record in save file for point at which vertex lies

Description: A VERTEX embodies the user’s view of a corner of a FACE or the end of an EDGE. It refers to an APOINT in object space and to the groups of EDGEs that it bounds. All EDGEs in each group can be found by following pointers through the COEDGEs.

The VERTEX may contain pointers to multiple EDGEs to provide access to all the EDGEs at the VERTEX, such as when a body is nonmanifold at a VERTEX, or when an unembedded EDGE dangles from a VERTEX of an otherwise well-formed solid.

If all the EDGEs at the VERTEX are WIRE (each adjacent to no FACEs) or if all are embedded (each adjacent to two FACEs and in one manifold group), the VERTEX will contain a pointer to a single EDGE. The others can be found by following the next, previous, and partner pointers of the COEDGEs of the EDGEs as appropriate for WIREs or embedded EDGEs.

Component: **ACIS Kernel** Topology

4

“vertexblendsur”

Purpose: Defines the vertex blend surface class.

Derivation: VBL_SURF : spl_sur : subtrans_object : subtype_object : –

Data Elements:

prim	subtype_start	Left curly braces, “{” or Tag 15
prim	write sv id	save identifier for this particular subtype
prim	integer	Number of boundaries
ctrl	repeat	Repeat for the number of boundaries
sv id	BDY_GEOM	Boundary geometry
prim	integer	Grid size
prim	real	Fit tolerance
prim	subtype_end	Right curly braces, “}” or Tag 16

Description: This class defines the vertex blend surface class. It is defined entirely by the n boundaries that make it up.

Component: **ACIS Kernel**

“vertex_template”

Purpose: Represents the data to be generated at a facet node.

Derivation: VERTEX_TEMPLATE : ENTITY : –

Data Elements: sv id af_node_mapping Data types to be generated for each vertex

Description: Every node of a facet contains its coordinates. In addition, a node contains a pointer to an array of additional fields. These fields are defined by the vertex template.

Component: Faceter Husk

“wcs”

Purpose: Defines the WCS class.

Derivation: WCS : ENTITY : –

Data Elements: prim transform Transform to get to model space.

Description: A WCS is used to define a transform which maps input into the coordinate system of the model.

Component: Geometry Husk

“wire”

Purpose: Represents a collection of EDGEs.

Derivation: WIRE : ENTITY : –

Data Elements:	prim	$rec_num	Pointer to record in save file for next wire in body, shell or subshell
	prim	$rec_num	Pointer to record in save file for first coedge in wire
	prim	$rec_num	Pointer to record in save file for body or shell containing wire
	ctrl	if_cond	if save_version_number is greater than or equal to WIREBOOL_VERSION
	prim	$rec_num	Pointer to record in save file for subshell containing wire
	prim	logical	(“out” “in”) Containment of wire

Description: A WIRE represents a connected collection of EDGEs, and is owned by a BODY or a SHELL. WIREs stand for construction points and bounded, unbounded or semi-bounded curves. They can represent open or closed profiles, and also general wireframe models that are unsurfaced; i.e., have no FACEs.

Component: **ACIS Kernel** Topology

“xedge”

Purpose: Attaches to cross edges of the blend sheet, recording the blended edge giving rise to the face on one side of the cross edge.

Derivation: ATTRIB_XEDGE : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class attaches to cross edges of the blend sheet, recording the blended edge giving rise to the face on one side of the cross edge. This is used internally during blend1, but it should be removed from the sheet at the end of this stage.

Component: ACIS Kernel

“xvert”

Purpose: Implements the derived blend attribute for flagging a blend sheet pointed vertex.

Derivation: ATTRIB_XVERT : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class implements the derived blend attribute for flagging a blend sheet pointed vertex.

Component: ACIS Kernel

“xverted”

Purpose: Defines an attribute to be attached to point vertices of the vertex blend sheet.

Derivation: ATTRIB_XVERTED : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

Data Elements: prim No data This class does not save any data

Description: This class defines an attribute to be attached to point vertices of the vertex blend sheet, pointing to the associated edge that was blended with a zero radius blend.

Component: ACIS Kernel