Chapter 6.
Save Identifiers E thru Q

"E"

Purpose: Provides the representation for e to the accuracy of the system.

Derivation: e_law : constant_law : law : –

Data Elements:

prim	string	The character "E" not already part of another word appears somewhere within this double quoted string.

Description: *e* is the base of the natural system of logarithms. It is the limit of $(1+1/n)^n$ as *n* increases without limit; it is also the sum of the infinite series 1+1/1!+1/2!+1/3!+1/4!+...; its numerical value is 2.718281828459045....

"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.

"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).

“EDGE#”

Purpose: Composes a law function with a tag for an edge or bounded curve used as an input argument.

Derivation: curve_law_data : path_law_data : law_data : –

Data Elements:	prim	string	The word "EDGE" followed by an integer appears somewhere within this double quoted string.
	prim	integer	An integer greater than 0. It indicates how many law data support items there are.
	ctrl	repeat	Repeat the next steps for each law data support item.
	prim	string	This is a double quoted string with one of the words: "TRANS", "EDGE", "SURF", or "WIRE".
	ctrl	if_cond	If the string is the double quoted "EDGE"
	sv id	curve type	Save the underlying curve for this edge.
	prim	real	Beginning parameter for edge.
	prim	real	Ending parameter for edge.

Description: Some law functions, such as curc, cur, and dcur, accept curve law data as input arguments. When working with APIs, these can be anything derived from the curve class and bounded. When working in Scheme, however, these should be edges.

When a bounded curve (e.g., edge) is used as input into a law function, it is always followed by an integer *n* that specifies its index into the input argument list. The index numbering starts at 1. For any given index number *n*, the argument list has to contain at least *n* arguments.

A law expression with edge1 and law1 followed by a curve and a law is invalid, because each is requesting a different argument type as the first element of the argument list. To correct this problem, specify the ordering of the arguments in the input argument list (e.g., law and then curve) and then specify the index number (e.g., edge2 and law1).

"edgetapersur"

Purpose: Evaluator for a general edge tapered surface.

Derivation: edge_tpr_spl_sur : 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	"tapersur"	Save the information from the taper_spl_sur.
	prim	vector	Draft vector.
	prim	subtype_end	Right curly braces, "}" or Tag 16

Description: Refer to Purpose.

"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).

"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.

"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.

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.

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.

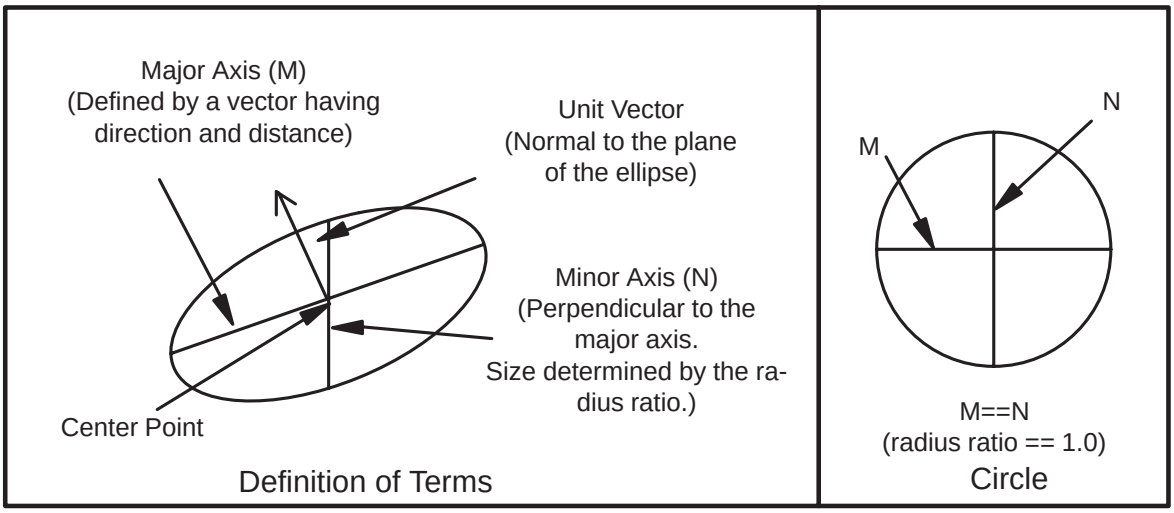

Figure 6-1. Ellipse Examples

"elliptical"

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.

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.

“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.

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.

“entity_ptr”

Purpose: Describes a pointer entity as a derived class of ENTITY so that these entities are logged and rolled back and forth together with models.

Derivation: ENTITY_PTR : ENTITY : –

Data Elements:			
	prim	No Data	No data is saved or restored.

Description: This class was needed, because static variables cannot be logged in the bulletin board. To overcome this, static variables have been changed to non–changing handles to changing ptr_entities. In this way, the static variables are logged. This class is useful, as well, for a rollable version of a linked list.

equal

Purpose: Used with PIECEWISE to create a logical = conditional.

Derivation: equal_law : binary_law : law : –

Data Elements:			
	prim	string	The character "=" appears somewhere within this double quoted string.

Description: Refer to Purpose.

"error_entity"

Purpose: For internal use only.

Derivation: ERROR_ENTITY : ENTITY : –

Data Elements:			
	prim	string	Message is "Warning: ERROR_ENTITY not suitable for save!

Description: If an attempt is made to save this entity, a message is displayed: "Warning: ERROR_ENTITY not suitable for save!"

"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
	ctrl	if_cond	if save_version_number is greater than or equal to EXT_CU_SF_VERSION
	prim	interval	The original curve range before any extensions were applied to it.
	prim	subtype_end	Right curly braces, "}" or Tag 16

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

“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.

“EXP”

Purpose: Composes a law mathematic function that takes *e* to a given power.

Derivation: exp_law : unary_law : law : –

Data Elements:			
	prim	string	The word “EXP” followed by something in parenthesis appears somewhere within this double quoted string.

Description: Refer to Purpose statement.

“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.

exponent

Purpose: Composes a law mathematic function that uses the exponentiation, or power, ("^") operator.

Derivation: exponent_law : binary_law : law : –

Data Elements:

prim	string	The character "^" appears somewhere within this double quoted string with items preceding it and following it.

Description: Parsing actually involves the "^" character. my_law1 and my_law2 can be any valid law mathematic function. If my_law is x and my_law2 is 3, this takes x to the third power. Exponents take precedence over multiplication/division and addition/subtraction. Both my_law1 and my_law2 have to be single dimension.

"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.

"eye"

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

Derivation: ATTRIB_EYE : ATTRIB : –

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

Description: Refer to purpose.

“eye_refinement”

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

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.

“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.

"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.

"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.

"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.

"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.

"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.

"FRENET"

Purpose: Composes a law mathematic function that returns the second geometric derivative of its sublaw.

Derivation: frenet_law : unary_law : law : –

Data Elements:	prim	string	The word "FRENET" followed by something in parenthesis appears somewhere within this double quoted string.

Description: Returns second geometric derivative of its sublaw, my_law. This is a vector pointing in the direction of the direction of curvature. The geometric derivative is the derivative of the curve parameterized with respect to the arc length.

This law symbol is used to specify the orientation of a surface by defining a vector fields along a curve. This defines rails which help orient a surface when performing sweep operations.

“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: When a bs2_curve is passed into a var_rad_functional constructor, the constructors will not make a copy of the curve. The bs2_curve becomes the exclusive property of the var_rad_functional object, and will be deleted by var_rad_functional destructors. If for some reason the caller wants to keep a bs2 radius function after calling our constructor, they must make a copy.

“gen”

Purpose: Defines the organization attribute class for the Generic Attributes Component.

Derivation: ATTRIB_GENERIC : ATTRIB : ENTITY : –

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

Description: Refer to the Purpose.

greater_than

Purpose: Used with PIECEWISE to create a logical > conditional.

Derivation: greater_than_law : binary_law : law : –

Data Elements: prim string The character “>” appears somewhere within this double quoted string.

Description: Refer to Purpose.

greater_than_or_equal

Purpose: Used with PIECEWISE to create a logical >= conditional.

Derivation: greater_than_or_equal_law : binary_law : law : –

Data Elements:

prim	string	The characters ">=" appears somewhere within this double quoted string.

Description: Refer to Purpose.

"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.

"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.

"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.

"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.

"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.

"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.

"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.

"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.

“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.

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.

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.

int_cur

Purpose: Defines spline curves.

Derivation: int_cur : subtrans_object : subtype_object : –

Data Elements:

sv id	bs3_curve_def		B-spline 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
ctrl	if_cond		if save_version_number is greater than or equal to DISCONTINUITY_VERSION
prim		newline	
prim		discontinuity_info	Parameter values of discontinuities

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.

“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.

“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.

"intercept"

Purpose: For internal use only.

Derivation: ATTRIB_INTERCEPT : ATTRIB_BLINFO : ATTRIB_SYS : ATTRIB : ENTITY : –

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

Description: Refer to Purpose.

"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.

"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.

law

Purpose: Creates the base class for the derived law classes.

Derivation: law : –

Data Elements:

ctrl	if_cond		Check whether the law is NULL.
prim		string	Writes out “null_law”
ctrl	else		repeat for the number of parameters
prim		string	Writes out the associated law string within a set a set of double quotes. Valid strings for within the double quotes are any valid combination of law symbols.
prim		newline	
prim		integer	Number of law data items associated with the law defined within the double quotation marks.
prim		newline	
ctrl		repeat	repeat for the number of law data items
sv id		law_data	save the associated law data items (e.g., the other associated ACIS classes needed for the law.)

Description: The law class is the base class from which all other law classes are derived. Laws are functions from *n* dimensional Euclidean space to *m* dimensional Euclidean space. The law class provides virtual methods used by all derived law class types.

Laws are parsed the same way that equations are presented in mathematics textbooks. For example, the equation "$f(x,y) = x^2+\cos(x)-\sin(y)$" becomes the law function "X^2+COS(X)–SIN(Y)", with a two dimensional domain and a one dimensional range.

The data elements given in this template assume that the law data is being restored ***from*** the SAT file. If the law data is to be written ***to*** the SAT file, the string of law symbols used in the law save ID determines the number, the order, and the type of law data elements.

For example, assume the string of law symbols has "...TRANS2...EDGE1..." as part of its definition. The integer following the law string would be at least 2. Moreover, the first law data string would be "EDGE" followed by its associated ACIS class data. The second law data string would be "TRANS" followed by its associated ACIS class data.

law_data

Purpose: Creates a wrapper for other ACIS classes for passing as arguments to laws.

Derivation: law_data : –

Data Elements:

ctrl	if_cond		If the string is “TRANS”
sv id		“transform”	the associated transform
ctrl	if_cond		If the string is “WIRE”
sv id		“curve”	the associated curve for the given wire
prim		integer	Number of wires
ctrl		repeat	repeat for the number of wires
prim			real starting parameter for given wire
prim			real scale factor for wire.
prim			interval for the wire
ctrl	if_cond		If the string is “EDGE”
sv id		“curve”	the associated curve for the given edge
prim		real	starting parameter for given edge
prim		real	ending parameter for given edge
ctrl	if_cond		If the string is “SURF”
sv id		“surface”	the associated surface
prim		interval	interval for the *u* domain
prim		interval	interval for the *v* domain
ctrl	else		An error message is displayed “Error unknown law data type” and a system error LAW_DATA_UNKNOWN is initiated.

Description: This is the base class for a series of wrappers that handle specific ACIS entities and ACIS classes.

The data elements given in this template assume that the law data is being restored ***from*** the SAT file. If the law data is to be written ***to*** the SAT file, the string of law symbols used in the law save ID determines the number, the order, and the type of law data elements.

For example, assume the string of law symbols has “...TRANS2...EDGE1...” as part of its definition. The integer following the law string would be at least 2. Moreover, the first law data string would be “EDGE” followed by its associated ACIS class data. The second law data string would be “TRANS” followed by its associated ACIS class data.

"lawintcur"

Purpose: Defines a curve from a law.

Derivation: law_int_cur : int_cur : subtrans_object : subtype_object : –

Data Elements:

ctrl	if_cond		if save_version_number is less than BNDSUR_VERSION
sv id		int_cur	Save as approximations
ctrl	else if_cond		if save_version_number is less than or equal to LAW_VERSION, reports system error using LAW_NO_SAVE.
ctrl	else if_cond		if save_version_number is less than INTCURVE_VERSION or save_version_number is less than LAW_VERSION, an exact_int_cur approximation is used. System Warning LAW_SAVE_APPROX.
sv id		bs3_curve_def	Save the exact_int_cur for the bs3 curve.
prim		newline	
ctrl	else if_cond		if save_version_number is less than or equal to LAW_SPL_VERSION, reports system error using LAW_NO_SAVE.
ctrl	else		Save the default data
sv id		int_cur	Save the bs3 curve.
prim		real	Curve range starting point
prim		real	Curve range ending point
prim		newline	
sv id		law	Save the law and law data associated with this curve.
prim		newline	
prim		integer	Number of helper laws in the array of helper laws
ctrl		repeat	For each helper law
sv id		law	Save the law and law data associated with this.
prim		newline	

Description: Refer to Purpose.

“lawsur”

Purpose: Creates a surface using a law.

Derivation: law_spl_sur : spl_sur : subtrans_object : subtype_object : –

Data Elements:	ctrl	if_cond		if (save_version_number is less than BNDSUR_VERSION and option save unknown subtype as aprrox is on), or save_version_number is less than the SPLINE_VERSION, or save_version_number is less than the LAW_VERSION
	sv id		spl_sur	Save as an approximation
	ctrl	else if_cond		if save_version_number is less than or equal to LAW_SPL_VERSION, reports system error using LAW_NO_SAVE.
	ctrl	else if_cond		if save_version_number is less than SPLINE_VERSION or save_version_number is less than LAW_VERSION, an approximation is used. System Warning LAW_SAVE_APPROX.
	sv id		spl_sur	Save as an approximation
	ctrl	else if_cond		if save_version_number is less than or equal to LAW_VERSION, reports system error using LAW_NO_SAVE.
	ctrl	else		Save the default data
	prim		real	u range starting point
	prim		real	u range ending point
	prim		real	v range starting point
	prim		real	v range ending point
	prim		newline	
	sv id		law	Save the law and law data associated with this curve.
	prim		integer	Number of helper laws in the array of helper laws
	ctrl		repeat	For each helper law
	sv id		law	Save the law and law data associated with this.
	prim		newline	
	sv id		spl_sur	Save the rest of the spline surface.

Description: Refer to Purpose.

less_than

Purpose: Used with PIECEWISE to create a logical < conditional.

Derivation: less_than_law : binary_law : law : –

Data Elements: prim string — The character "<" appears somewhere within this double quoted string.

Description: Refer to Purpose.

less_than_or_equal

Purpose: Used with PIECEWISE to create a logical <= conditional.

Derivation: less_than_or_equal_law : binary_law : law : –

Data Elements: prim string — The characters "<=" appears somewhere within this double quoted string.

Description: Refer to Purpose.

"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.

"LN"

Purpose: Composes a law mathematic function that takes the log base *e* (or the natural log) of the given value.

Derivation: natural_log_law : unary_law : law : –

Data Elements:	prim	string	The word "LN" followed by something in parenthesis appears somewhere within this double quoted string.

Description: Refer to Purpose statement.

"LOG"

Purpose: Composes a law mathematic function that takes the log of a given base of the given value.

Derivation: log_law : multiple_law : law : –

Data Elements:	prim	string	The word "LOG" followed by something in parenthesis appears somewhere within this double quoted string.

Description: This composes a law mathematic function that takes the log to a given my_base of the given my_law. If no my_base is specified, this assumes log to the base 10.

"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.

“lopt_copy_att”

Purpose: For internal use only.

Derivation: ATTRIB_LOPT_COPY_MAP : ATTRIB_SYS : ATTRIB : ENTITY : –

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

Description: Results in a system error with LOPT_TWK_BAD_OP_ON_ATT.

“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.

“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.

“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.

"lop_edge_cvxty_att"

Purpose: For internal use only.

Derivation: ATTRIB_LOP_EDGE_CVTY : ATTRIB_SYS : ATTRIB : ENTITY : –

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

Description: Refer to Purpose.

"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.

"lop_loop_class_att"

Purpose: For internal use only.

Derivation: ATTRIB_LOP_LOOP : ATTRIB_SYS : ATTRIB : ENTITY : –

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

Description: Results in a system error with LOPT_TWK_BAD_OP_ON_ATT.

"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.

“lop_protected_list_att”

Purpose: For internal use only.

Derivation: ATTRIB_LOP_PROTECTED_LIST : ATTRIB_SYS : ATTRIB : ENTITY : –

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

Description: Results in a system error with LOPT_TWK_BAD_OP_ON_ATT.

“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.

“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.

“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.

"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.

"MAX"

Purpose: Composes a law mathematic function that finds the maximum of two or more input laws.

Derivation: max_law : multiple_law : law : –

Data Elements:

prim	string	The word "MAX" followed by something in parenthesis appears somewhere within this double quoted string.

Description: For a given set of input variables, this law symbol evaluates its included law symbols, e.g., my_law1, my_law2, my_law*n*, and returns the largest value from all included functions. All sublaws must return one value.

"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.

"MIN"

Purpose: Composes a law mathematic function that finds the minimum of two or more input laws.

Derivation: min_law : multiple_law : law : –

Data Elements:	prim	string	The word "MIN" followed by something in parenthesis appears somewhere within this double quoted string.

Description: For a given set of input variables, this law symbol evaluates its included law symbols, e.g., my_law1, my_law2, my_law*n*, and returns the smallest value from all included functions. All sublaws must return one value.

"MINROT"

Purpose: Composes a law mathematic function that returns the minimum rotation.

Derivation: min_rotation_law : multiple_law : law : –

Data Elements: prim string — The word “MINROT” followed by something in parenthesis appears somewhere within this double quoted string.

Description: This law mathematic function returns a vector field. my_path is an edge or a wire. my_vector is the starting direction vector. The other vectors along my_path are determined by a minimal adjustment from the previous vector. The result is a vector field on my_path starting at my_vector, which has minimal rotation about my_path. This law mathematic function obtains as little twist as possible.

This law mathematic function is used to specify the orientation of a surface by defining a vector fields along a curve. This defines rails which help orient a surface when performing sweep operations.

minus

Purpose: Composes a law mathematic function that uses the minus, or subtraction (“–”) operator.

Derivation: minus_law : binary_law : law : –

Data Elements: prim string — The character “–” appears somewhere within this double quoted string and has elements preceding and following it.

Description: Parsing actually involves the “–” character. my_law1 and my_law2 can be any valid law mathematic function. Both my_law1 and my_law2 can be multiple dimensions; the smaller of the two is padded with zeros.

“ms”

Purpose: Represents attribute base class for the Mesh Surface Component.

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 Component. All other Mesh Surface Component attributes are derived from this base class.

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.

“name_attrib”

Purpose: Defines a named attribute for the Generic Attributes Component.

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.

"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.

"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.

“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.

“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.

“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.

“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.

“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.

negate

Purpose: Composes a law mathematic function that uses the unary minus, or negation (“–”) operator.

Derivation: negate_law : unary_law : law : –

Data Elements:

prim	string	The character “–” appears somewhere within this double quoted string and only has elements following it.

Description: Parsing actually involves the “–” character. my_law1 can be any valid law mathematic function.

“netsur”

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

Derivation: net_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 the save version is less than BND_SUR_VERSION and the option to save subtype as an approximation is on.
sv id	"spl_sur"	save an approximation of the spl_sur.
ctrl	else	
sv id	"spl_sur"	Save the subtype object.
ctrl	if_cond	if the save version is less than SPLINE_VERSION print out a warning with OLD_SAVE_REV.
sv id	"spl_sur"	Save the subtype object.
ctrl	else	
prim	integer	Save the no_crv_v.
prim	newline	
ctrl	repeat	For the number of no_crv_v.
prim	real	v knot
prim	newline	
prim	curve	Save the data for the v curve.
prim	newline	
prim	integer	Save the no_crv_u.
prim	newline	
ctrl	repeat	For the number of no_crv_u.
prim	real	u knot
prim	newline	
prim	curve	Save the data for the u curve.
prim	newline	
ctrl	repeat	For the number of no_crv_u.
ctrl	repeat	For the number of no_crv_v.
prim	real	Corner data s par.
prim	real	Corner data t par.
prim	newline	
sv id	"spl_sur"	Save the parent subtype object.
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: Defines a net surface between a list of curves. The surface parameterization is: *u* direction and *v* direction corresponds to the U and V curves to be surfaced. The input to this surface class is : The curves to be surfaced. All the curves are reparameterized to lie in (0.0 – 1.0) range.

“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.

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.

“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.

"NORM"

Purpose: Composes a law mathematic function that normalizes a law.

Derivation: norm_law : unary_law : law : –

Data Elements: prim string — The word "NORM" followed by something in parenthesis appears somewhere within this double quoted string.

Description: This law symbol normalizes the length of my_law to be of unit length. This is accomplished by dividing each dimension element by the square root of the sum of the squares of all of the return elements. This is applicable to a law that returns any dimension.

"O"

Purpose: Creates function composition, as in "f of g", where f and g are both law mathematic functions.

Derivation: composite_law : binary_law : law : –

Data Elements: prim string — The character "O" not already part of another word appears somewhere within this double quoted string and has elements preceding and following it.

Description: The composition function is useful whenever complicated input expressions, my_law2, are needed for my_law1. The output dimension of my_law2 must be the input dimension of my_law1. The input values to my_law2 are evaluated first, and the results are used as the input values to my_law1.

For example, lets assume we have the complicated expression for my_law:

*(define my_law (x^2 + 1) / (x^2 + sqrt(x) – 4*x^3))*

Now let's assume that for every value of *x* in the above expression, we need to substitute the my_law2 expression:

*(define my_law2 (x/(2*pi) + 1))*

The result, when written out by hand or typed into the computer, isn't very easy to understand.

*(define my_law3 ((x/(2*pi) + 1)^2 + 1) / ((x/(2*pi) + 1)^2 + sqrt((x/(2*pi) + 1)) – 4*(x/(2*pi) + 1)^3))*

Two equivalent function composition are:

*(define my_law5 ((x^2 + 1) / (x^2 + sqrt(x) – 4*x^3) o (x/(2*pi) + 1))*

(define my_law5 (my_law O my_law2))

"NOT"

Purpose: Used with PIECEWISE to create a logical NOT conditional.

Derivation: not_law : unary_law : law : –

Data Elements: prim string — The word "NOT" followed by something in parenthesis appears somewhere within this double quoted string.

Description: Refer to Purpose.

not_equal

Purpose: Used with PIECEWISE to create a logical != conditional.

Derivation: not_equal_law : binary_law : law : –

Data Elements: prim string — The characters "!=" appears somewhere within this double quoted string.

Description: Refer to Purpose.

"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	write 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.

"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.

"offsetintcur"

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

Derivation: offset_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 , or if save_version_number is less than LAW_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	
	ctrl		if_cond	if save_version_number is less than LAW_VERSION .
	prim			real	Offset distance
	prim			newline	
	prim			real	Draft distance
	prim			newline	
	ctrl		else	
	sv id			law	Distance law.
	prim			newline	
	sv id			law	Twist law.
	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.

"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

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, etc.) 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 specialized.

"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 ($d1 + d2$).

"offsurfintcur"

Purpose: Represents the offset of a curve lying on a surface.

Derivation: off_surf_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 data	Generic int_cur data
prim	interval	U range of base surface
prim	interval	V range of base surface
sv id	curve data	Base curve
prim	interval	Range of base curve
prim	real	Offset distance
prim	real	Shift between this and the base curve parameterization
prim	real	Scaling factor between this and the base curve parameterization
prim	subtype_end	Right curly braces, "}" or Tag 16

Description: This curve represents the offset of its base curve along the base surface normal. The curve generally has the same parameterization as its base curve, but if made to differ, the parameter shift and scaling factor are stored in the class.

"OR"

Purpose: Used with PIECEWISE to create a logical OR conditional.

Derivation: or_law : binary_law : law : –

Data Elements:	prim	string	The word "OR" followed by something in parenthesis appears somewhere within this double quoted string.

Description: Refer to Purpose.

"orthosur"

Purpose: Creates an orthogonal surface.

Derivation: ortho_spl_sur : 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	"tapersur"	Save the information from the taper_spl_sur.
	prim	logical	sense of the orthogonal.
	prim	subtype_end	Right curly braces, "}" or Tag 16

Description: Class to describe an orthogonal surface, which can arise in face_taper.

"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.

"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.

“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.

“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.

path#

Purpose: Composes a law function with a tag for a path (e.g., edge or wire) used as an input argument.

Derivation: path_law_data : law_data : –

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

Description: This is an abstract data class that breaks down into either wires or edges. Therefore, it does not appear in the save file. The map law function accepts either wires or edges (i.e., bounded curves) as input arguments.

When a wire or edge is used as input into a law function requiring a path, it is always followed by an integer *n* that specifies its index into the input argument list. The index numbering starts at 1. For any given index number *n*, the argument list has to contain at least *n* arguments.

"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.

"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.

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.

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).

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.

“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.

“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.

“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.

"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.

"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.

“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.

"PI"

Purpose: Provides the representation for pi to the accuracy of the system.

Derivation: pi_law : constant_law : law : –

Data Elements: prim string — The word "PI" not already part of a larger string appears somewhere within this double quoted string.

Description: *pi* is used to denote the ratio of the circumference of a circle to its diameter; *pi* is 3.1415926545898....

"PIECEWISE"

Purpose: Permits laws to evaluate differently based on conditional definition statements.

Derivation: piecewise_law : multiple_law : law : –

Data Elements: prim string — The word "PIECEWISE" followed by something in parenthesis appears somewhere within this double quoted string.

Description: Permits an operation to be performed in a "piecewise" fashion, depending upon the conditions that were established. Both the conditions (e.g., cond1, cond2) and the laws (e.g., my_law1, my_law2, my_default_law) are normal law declarations. The number of laws in the statement has to be one more than the number of conditions, because the last law serves as the catchall "else" in the evaluation.

"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 ID Component 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.

“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
	ctrl	if_cond	if save_version_number is greater than or equal to DISCONTINUITY_VERSION
	prim	newline	
	prim	discontinuity_info	U Parameter values of discontinuities
	prim	newline	
	prim	discontinuity_info	V Parameter values of discontinuities
	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.

"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.

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.

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*-parameter, 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*-derivative vector, negate it if reverse_v is TRUE, and call it *v*-derivative 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*.

plus

Purpose: Composes a law mathematic function that uses the addition ("+") operator.

Derivation: plus_law : binary_law : law : –

Data Elements:

prim	string	The character "+" appears somewhere within this double quoted string and has elements preceding and following it.

Description: Parsing actually involves the "+" character. my_law1 and my_law2 can be any valid law mathematic function. Both my_law1 and my_law2 can be multiple dimensions; the smaller of the two is padded with zeros.

"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.

“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.

“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.

“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.

“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_domain_pt[ii]
	ctrl	repeat	Repeat for all apc_domain_dim
	prim	real	apc_base_pt[ii]
	ctrl	repeat	Repeat for all apc_domain_dim
	prim	real	apc_tang_vec[ii]

Description: For internal use only.

"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.

"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.