Chapter 4. Architecture

Topic: *Software Architecture

A *software component* is a functionally specialized unit of software—a collection of software items (functions, classes, etc.) grouped together to serve some distinct purpose. It serves as a constituent part of a whole software system or product. A *product* is one or more software components that are assembled together and sold as a package. Components can be arranged in different combinations to form different products.

The ACIS product line is designed using *software component technology*, which allows an application to use only the components it requires. In some cases, more than one component is available (either from *Spatial* or third party vendors) for a given purpose, so application developers can use the component that best meets their needs. For example, several rendering components are available from *Spatial*, and developers use the one that works best for their platform or application.

Components

Topic: *Software Architecture

Each software component maps to a top level directory in the ACIS installed directory tree. The component directories in your installed directory tree will be a subset of all available component directories, based on your platform and which product(s) you purchased.

Component directories contain one or more subdirectories in which the component's files are located (no code is in the top level component directory). Each of these subdirectories generally corresponds to an object library (refer to the *3D ACIS Application Development Manual* for information on object libraries). Most component directories also include subdirectories containing Scheme extensions (subdirectory name `*_scm`).

Table 4-1 lists all the components, alphabetically by their directory names, available for the ACIS product line. For more information about a component, refer to the corresponding Component Manual.

Table 4-1. Software Components

Directory	Component Name (Abbreviation)	Description/Comments
abl	Advanced Blending (ABL)	Blending beyond standard blending (BLND)
adm	ACIS Deformable Modeling (ADM)	Free–form 3D sculpting operations on a curve or surface
admgi	ADM Graphic Interaction (ADMGI)	Bridge between ADM and GI for drawing; illustrates how to use GI rendering with ADM
phlv5	Precise Hidden Line Removal V5 (PHLV5)	Calculates hidden line data and draws hidden line representations of the model
apfill	PowerFill (PFILL)	Covers circuits in solid or wire bodies using advanced surface deformation technology
amfc	ACIS MFC (AMFC)	Support for Microsoft Foundation Classes (MFC) based applications (NT platforms)
base	Base (BASE)	Provides some very low–level common functionality that is used by all ACIS components, including memory management, error handling, some basic data types, etc.
blnd	Blending (BLND)	Standard blending operations
bool	Boolean (BOOL)	Unite, intersect, and subtract operations
br	Basic Rendering (BR)	Rendering supplied with ACIS
catia	CATIA Translator (CATIA)	Read CATIA model files and CATIA Export files and translate the geometry to ACIS
clr	Clearance (CLR)	Determine minimum distance between bodies or faces
covr	Covering (COVR)	Cover wires and sheets (all boundaries specified)
cstr	Constructors (CSTR)	Basic topology construction; wireframe construction and editing; analysis (area, length, mass properties)
ct	Cellular Topology (CT)	Divide larger regions up into smaller subregions or cells
ds	Standalone Deformable Modeling (SDM)	Standalone component used by ADM for sculpting operations

eulr	Euler Operations (EULR)	Expand, flatten, separate, and combine lumps
examples	Examples XMP	Contains Scheme extensions that illustrate API usage
fct	Faceter (FCT)	Generate faceted (polygonal) representation
ga	Generic Attributes (GA)	Attributes that allow applications to exchange data
gi	Graphic Interaction (GI)	Commonly needed graphic display functionality
gl	OpenGL (GL)	Rendering for Windows NT platforms using OpenGL
heal	Healing (HEAL)	Fix models—usually imported from other modeling systems into ACIS—in which tolerance problems affect how ACIS interprets the model
iges	IGES Translator (IGES)	Bidirectional transfer of data between IGES format and ACIS format
igl	Interactive OpenGL (IGL)	Interactive viewing interface to the ACIS space warping functionality using OpenGL on NT platforms; intended for NT-based testing and demonstration
ihl	Interactive Hidden Line (IHL)	Creates views of ACIS model objects with hidden lines removed
intr	Intersectors (INTR)	Curve/curve, curve/surface, surface/surface intersectors; ray testing; silhouettes; parameter lines; point classification; body checking; curve and surface extension
kern	Kernel (KERN)	Spline interface; basic entity and attribute support; topology and geometry ENTITY classes; construction geometry classes; math classes; save and restore support; history and roll support
law	Laws (LAWS)	Provides symbolic representations of equations to solve complex problems
lop	Local Ops (LOP)	Locally manipulating models

lopt	Local Op Tools (LOPT)	Provides tools used in local operations
ofst	Offsetting (OFST)	Wire and face offsetting
oper	Operators (OPER)	Spline conversion
part	Part Management (PART)	Support for grouping entities
phl	Precise Hidden Line (PHL)	Hidden line removal
pid	Persistent ID (PID)	Attach identifiers that persist across saves
proe	Pro/E Translator (Pro/E)	Read Pro/Engineer part files and translate them into ACIS
rbase	Rendering Base (RBASE)	Interface common to all renderers
rbi	Repair Body Intersections (RBI)	Repairing self-intersections in a body
rem	Remove Faces (REM)	Removing unnecessary faces, such as after a local operation
sbool	Selective Booleans (SBOOL)	Selective Boolean operations (unite, intersect, subtract) using graph theory
scm	Scheme Support (SCM)	Scheme Interpreter; basic Scheme extensions; Scheme AIDE interface to rendering, part management, etc.
shl	Shelling (SHL)	Create shelled (hollow) bodies
skin	Advanced Surfacing (AS)	Various techniques (including skinning and lofting) for fitting a surface through a set of curves
step	STEP Translator (STEP)	Bidirectional transfer of data between STEP format and ACIS format
stitch	Stitch (STITCH)	Stitches a list of faces and bodies into a single body
swp	Sweeping (SWP)	Sweep a profile along a path
trans	Translator Utility (TRANS)	Utilities for translators
vda	VDA–FS Translator (VDA–FS)	Bidirectional transfer of data between VDA–FS format and ACIS format

vm	VisMan (VM)	Visual display and manipulation of models, using an object architecture that supports interfaces, co-classes, and class factories (cannot be used with GI)
warp	Space Warping (WARP)	Uses the ACIS law functionality to warp (twist, bend) entities based on law definitions
xgeom	Translation Geometry (XGEOM)	Geometry translation for translators

Component Dependencies

Topic: *Software Architecture

A software component may be dependent on others. All ACIS components are ultimately dependent upon the Base Component. Figure 4-1 shows a *dependency graph* for the “core” ACIS components, using the abbreviations of the component names (as shown in Table 4-1). The core components are those components of ACIS that provide modeling functionality and are not simply support or interface components (such as for Scheme, or MFC support).

The component dependencies are indicated by lines flowing from one component down to the component(s) on which it is dependent. The ADM component is codependent on the SDM component, which is reflected in the dependency graph with a loop going back up into ADM. Except for this codependency loop, all dependency paths flow down.

Note *Some dependencies include a path from the component to a numeral within a circle. This circled-numeral is then shown as dependent upon some other component. This is just a convention for simplifying the diagram.*

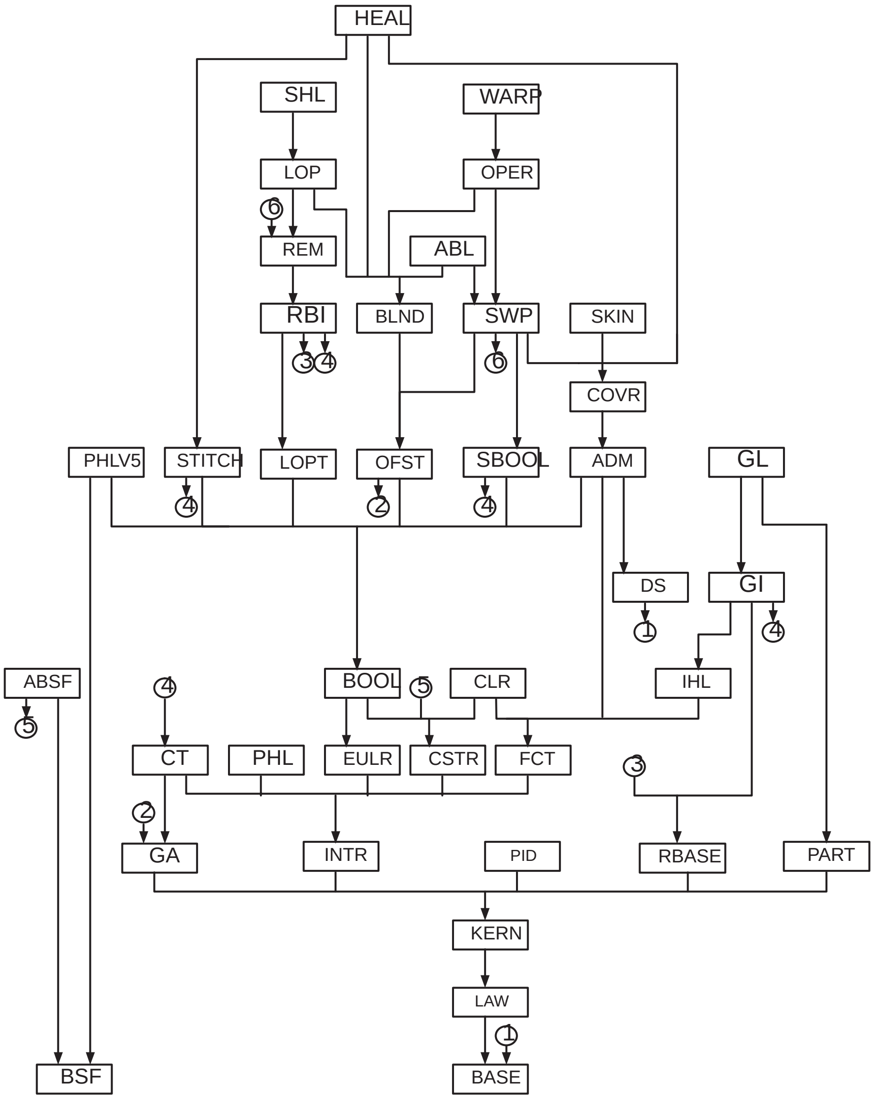

Figure 4-1. ACIS Core Component Dependencies

Object Libraries

Topic: Software Architecture, *ACIS Overview

The source code for a component is compiled and built into one or more object libraries. An application must link in the object libraries for any component it references. The core ACIS libraries are available as either static link libraries or shared (dynamic link) libraries.

Note *A shared library is called a* Dynamic Link Library *(DLL) on some systems (such as Windows). When referring to a shared library in a context that is not specific to a platform,* Spatial *uses the term* shared/DLL *library.*

Refer to the *3D ACIS Application Development Manual* for information about the available object libraries and using shared/DLL libraries.