cadence®

Allegro® X ECAD-MCAD Library Creator User Guide

Product Version 23.1
September 2023

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

1

Introduction

Library Creator enables PCB librarians, ECAD, and MCAD designers to accurately and efficiently create and maintain high-fidelity physical libraries to serve both PCB layout and MCAD design requirements. Allegro ECAD-MCAD Library Creator combines powerful package modeling capabilities with advanced footprint generation and ECAD-MCAD library synchronization technology.

Library Creator supports two primary mechanisms for creating packages: parametric templates, and the import and featurization of existing geometric models. Over 100 package, body, and terminals templates and thousands of package instances are provided through the repository, covering many common package families. Library Creator is the first technology capable of directly leveraging the detailed 3D STEP models provided by leading connector vendors such as Tyco, Molex, and Samtec for use in ECAD footprint generation and ECAD-MCAD library synchronization. In addition to 3D models obtained from component vendors, the featurization process can be used to leverage existing MCAD models of mechanical components and hardware for PCB library population. Allegro ECAD-MCAD Library Creator enables both standard and non-standard package models to be used in both footprint generation and ECAD-MCAD library synchronization.

The Library Creator footprint generation technology enables footprints to be efficiently and accurately generated from the package description using technology-driven rules based on industry standard such as IPC-7351 or internal company practices. Through the ability to update and regenerate the footprint library to address multiple design technologies and evolving manufacturing requirements, investment in the physical library can be leveraged.

Library Creator enables ECAD-MCAD library interoperability through automated synchronization of ECAD library footprints and repository package models. Sophisticated searching and matching routines automatically identify packages with compatible contact areas from the repository and synchronize coordinate systems between the footprint and the package model. The resulting database of synchronized models is maintained by Library Creator to ensure library consistency and support ECAD-MCAD collaborative design.

2

Package

A 'Package' is a model of a physical component with sufficient detail and accuracy to support both PCB layout and MCAD design requirements. The model of the physical package is distinct from a footprint that may be generated from or associated with the package. The 'package' describes the physical component, while the 'footprint' contains all of the PCB design elements needed to support downstream manufacturing and assembly requirements.

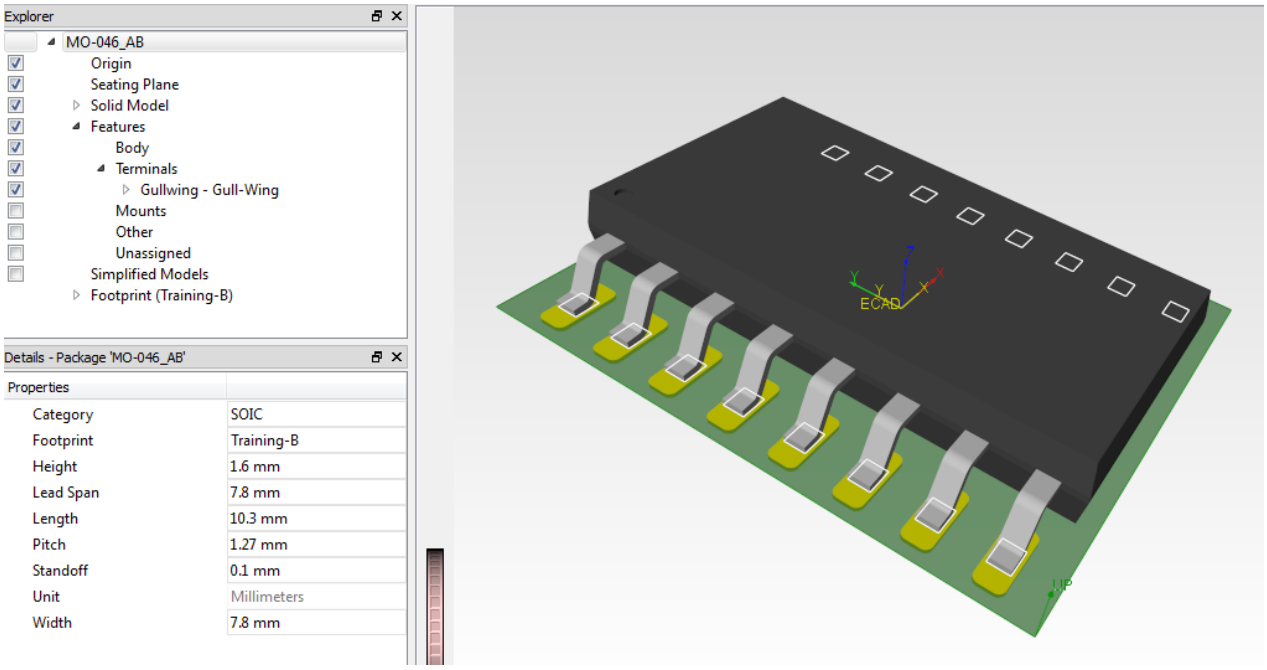
The package model can be used to represent traditional PCB component packages (i.e. Chip, SOP, QFN, BGA), non-standard components such as connectors, and mechanical components that impact PCB design, such as EMI shields, heat sinks, enclosures, and hardware components.

A package model typically contains a 3-dimensional representation of the physical component, as well as a number of features, such as the body, terminals, and mounts that play an important role in the interface between the package and the PCB. Package features may be associated with 3-dimensional geometry, or they may contain only a 2-dimensional contact area. The 2D contact areas are generated from the 3D geometry to ensure consistency between the ECAD and MCAD models, and are critical for both footprint generation and library synchronization. The seating plane defines the mounting orientation of the package with respect to the PCB.

Terminals and mounting features may have names (or pin numbers), and a footprint may be associated with the package model. In the case of an associated footprint, relationships between placements of padstacks and their associated terminal or mount feature is maintained. Identification of the primary pin is important for resolving orientation ambiguity, and is relied upon in library synchronization.

In addition to geometric information, a variety of properties are stored in the model. These properties may be top-level attributes associated with the package itself, or properties of individual features of the package.

The image below shows the top-level view of a typical package model. The *'Explorer'* tree outlines the key elements of the package model. When the package itself is selected in the *'Explorer'* widget, the top-level package properties are displayed in the *'Details'* panel.



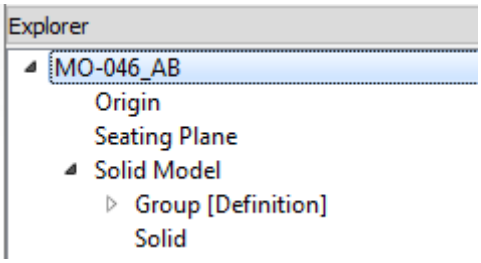
Package Unit

Packages are defined in one of the five available base units. The units of the active package are displayed as a top-level property on the package in the *'Details'* view and in the corner of the Footprint (2D) graphical context. The defined unit of repository packages is displayed as one of the available index columns in the package search view. The preferred units for repository index parameters and package import may be configured through the unit preferences dialog.

The unit system for an existing package cannot be changed, however, a copy of the package in an alternate unit system may be created using the *File->Copy in Session...* menu action or, in the case of a template based package, by creating a new template instance. The footprint export unit is independent of the unit within Library Creator and controlled by the drawing template.

Solid Model

The *'Solid Model'* contains the 3-dimensional geometric shapes in the current model. The solid model may contain a single solid or multiple solids. Elements of the solid model may be assigned directly to features or used to generate contact or face-based features.



Single-solid vs. Multiple-solid structure

Existing 3D models of electronic components that have been created internally or obtained from third party component vendors are likely to be represented as a single solid. 3D STEP models of connectors obtained from vendors are typically single-solid parts. An assembly-based model contains instances of multiple single solids. An assembly-based representation is conventionally created within a mechanical CAD environment to represent the assembly of discrete parts. However, an assembly-based model can also be used to represent individual components. For example, in a typical SOIC 14-lead package, a single solid (part) might be created for an individual gull-wing terminal, another solid might be created for the body of the package, and the complete package might be represented as an assembly containing one instance of the body solid, and 14 instances of the terminal solid, one for each terminal placement. Although both representations are supported in Library Creator, the assembly-based representation is typically more compact, and the association between 3D solids and features is more direct. For this reason, the Library Creator templates build models with an assembly-style geometric representation.

Features

A package feature is an element of the package that plays a role in either footprint generation or library synchronization. The common pre-defined package features are the body, terminal, and mounting features. In addition, there is an *'other'* feature type that can be used to establish new feature classifications based on internal conventions and practices. In additional to 3D geometry, the feature may have associated properties, section offsets, a 2D contact areas, an optional orientation, and a connect point.

Body

A package can contain only a single *'body'*. Typically, the package body represents the physical component extent excluding functional features such as terminals and mounts. In the case of a featurized external model, the body feature may represent the entire physical extent, including terminal and mount features. The body can be associated with an enumerated *'type'* attribute, and certain parametric data can be stored on the package body if desired. The *'Height'* parameter can be populated on the body, particularly if it is desired to explicitly represent body height as a tolerance dimension.

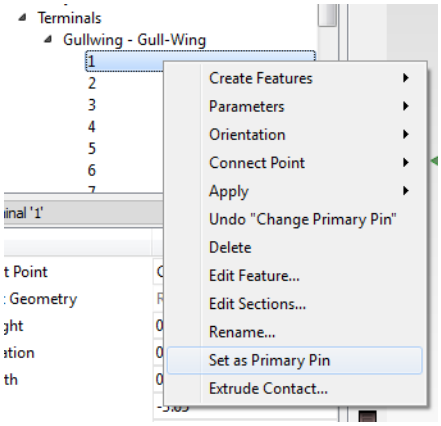
Terminals

Terminals represent physical interconnects, or pins. In addition to the three-dimensional physical representation, terminals typically contain a name (or pin number) that is unique within the context of the package. The lead form is an enumerated type that plays a critical role in both contact area and footprint generation. In the event that the lead form attribute is insufficient to categorize the terminal, the type attribute may be assigned. The terminal mount form (through vs. surface mount) is automatically derived from the lead form and cannot be set independently.

Primary Pin

Identification of the *'primary'* pin (or *Pin One*) is important for resolving orientation ambiguity and is relied upon in library synchronization. Whenever applicable, the *'Primary Pin'* attribute should be set for one terminal in the package. To set a terminal as the primary pin, select the terminal in the *'Explorer'* and use the right-click *'Set as Primary Pin'* context action. The primary pin is reflected in the package explorer by the (primary) indication after the pin name as well as in the properties of the terminal. Packages generated from templates will typically have a primary pin identified, however, in the case of a featurized external model, it is

necessary to set the primary pin explicitly. Through the Feature Types and Colors configuration object, it is possible to specify a unique color for the primary pin.



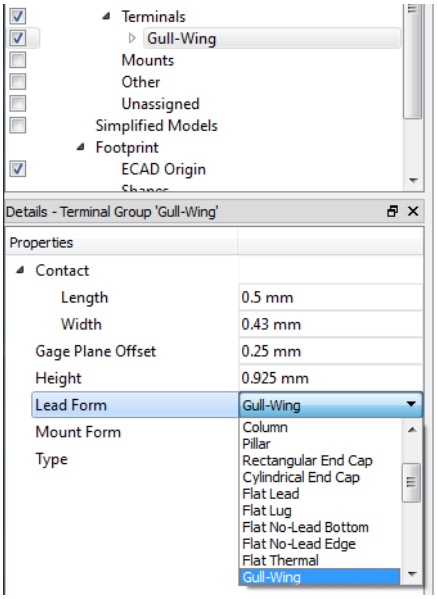
Terminal Groups

All terminals belong to exactly one terminal group. Within the Explorer tree, terminals will be nested under their respective group. The terminal group represents a collection of similar terminals, and must share a common leadform and/or type attribute. Parametric data can be assigned to individual terminals, however, terminals within a group will also inherit parametric data assigned at the level of the terminal group. In the event that the same parameter is assigned different values at the level of the group and individual terminal, the individual assignment will override the group-level assignment.

Lead Form

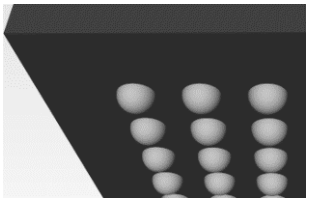
For template-based packages and footprint generation using standard IPC-7351 padstack rules, the *'Lead Form'* attribute controls which contact generator and which padstack rule will fire for a given terminal. Often, standard padstack rules will not be well-suited to pins on connectors and other non-standard packages. In these cases, additional specialized padstack and contact generation rules may be configured based on the terminal type attribute.

'Lead Form' is an attribute of the terminal group and is reflected in the name of the group in the *'Explorer.'* The lead form may be changed through the *'Details'* widget when the terminal group is selected.

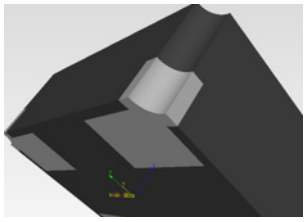


Below is an illustration of the common terminal lead form styles in Library Creator:

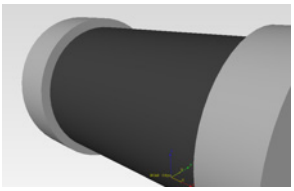
Ball



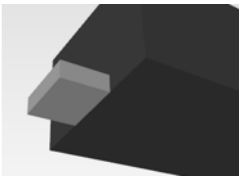
Corner - Concave



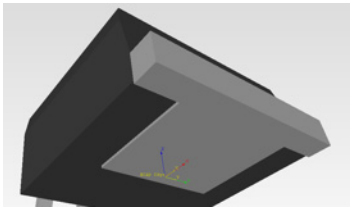
Cylindrical End Cap



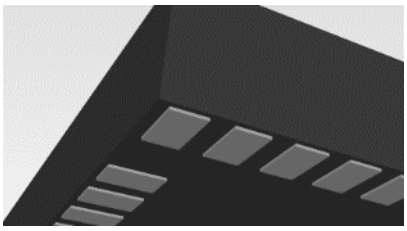
Flat Lead



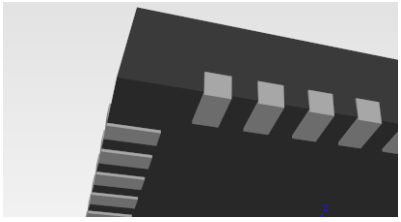
Flat Lug



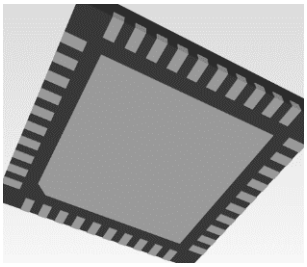
Flat No-Lead Bottom



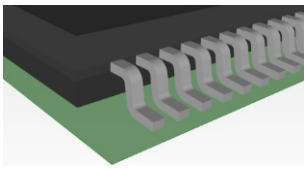
Flat No-Lead Edge



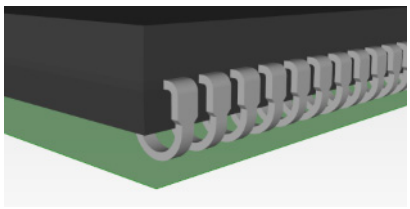
Flat Thermal



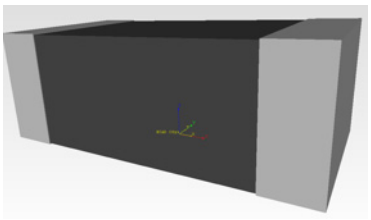
Gull-Wing



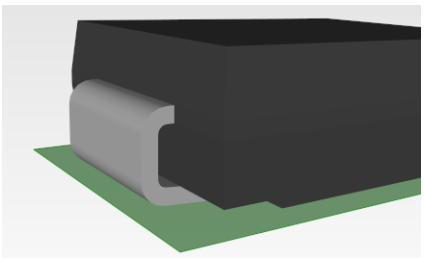
J-Lead



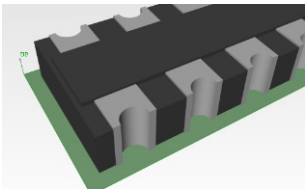
Rectangular End Cap



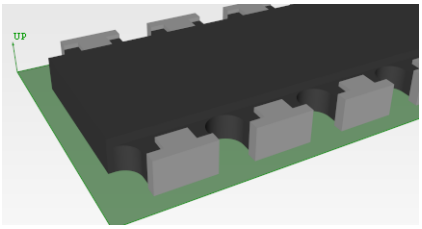
Ribbon L - Inward



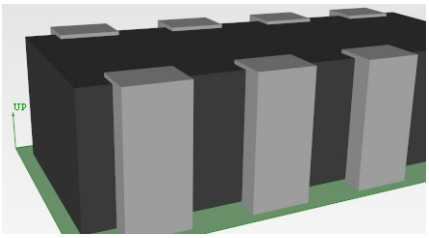
Side Lead - Concave



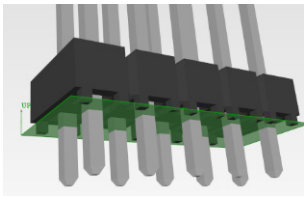
Side Lead - Convex



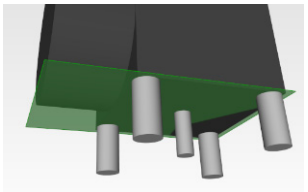
Side Lead - Flat



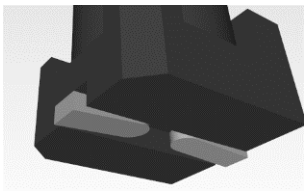
Through - Rectangular



Through - Round

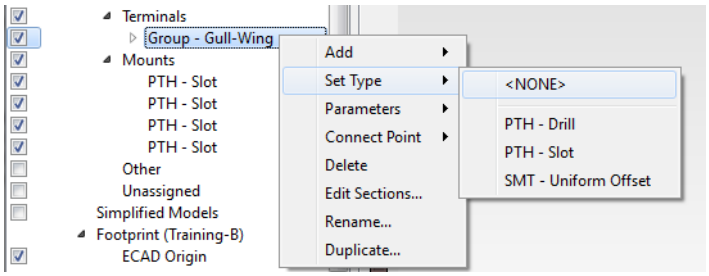


Under Body Outward L



Terminal Type

For certain footprint elements such as drills and slots, the lead-form may be insufficient to determine the role of the terminal in footprint generation. In these cases, terminals may be assigned an optional type in order to associate the feature with particular contact and/or padstack generators. The type may only be set on the terminal group, and applies to all terminals in the group.

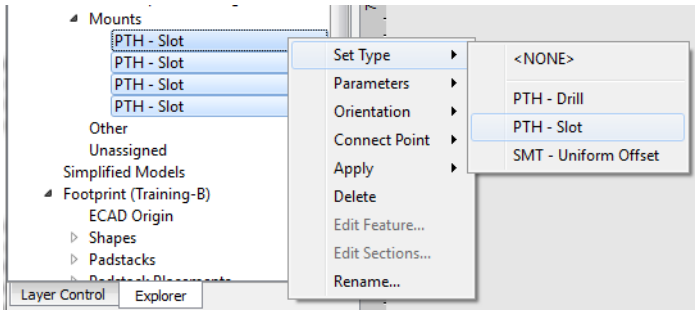


Mount Features

Mount features may be optionally used to model package interconnects for which an electrically connected terminal (i.e. pin) representation is inappropriate. In Allegro, mount features map to the concept of a mechanical pin. Typically, the mount feature should be assigned one of the enumerated mount *'types'* in order to support an association with contact and padstack generators applicable to the feature classification. In addition to the built-in mount parameters, user-defined parameters may be added as necessary to support application-specific requirements.

Mount Type

The mount *'type'* determines the role of the mount feature in footprint generation. An applicable type should be assigned to all mount features in order to associate the feature with particular contact and/or padstack generators. The type attribute may be set on individual or multiple mount features through the *'Set Type'* context action after selecting the features.



Other Features

In certain instances, it may be useful to create a package feature for which neither the body, terminal, or mount type is appropriate. The *'Other'* feature type may be used in such instances. The *'other'* features may be assigned an optional *'type'* in order to support an association with particular contact, padstack, and/or shape generator(s) applicable to the feature classification.

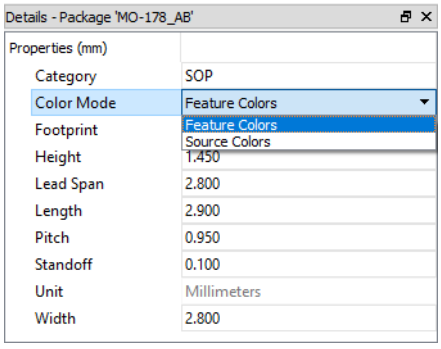
Unassigned Features

'Unassigned' features are an intermediate state in the process of creating features from existing 3D models. Unassigned features are not saved with the package model and should be assigned to one of the persistent feature types before upload to the repository.

Color Mode (3D)

There are two primary color schemes available for displaying 3D colors in Library Creator. The color mode is set as a top-level attribute on the package through the details pane as shown below.

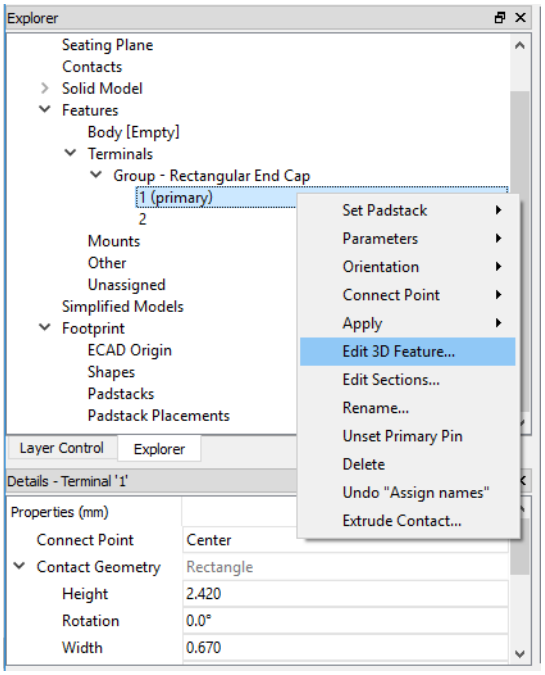
In *Feature Colors* mode, colors are assigned to 3D features in the package model based on the Feature Types and Colors configuration settings. In *Source Colors* mode, colors assigned to the 3D STEP model are displayed. The feature types and colors configuration options also support a hybrid display mode where feature colors may override the source colors for specified feature types as well as the primary pin.



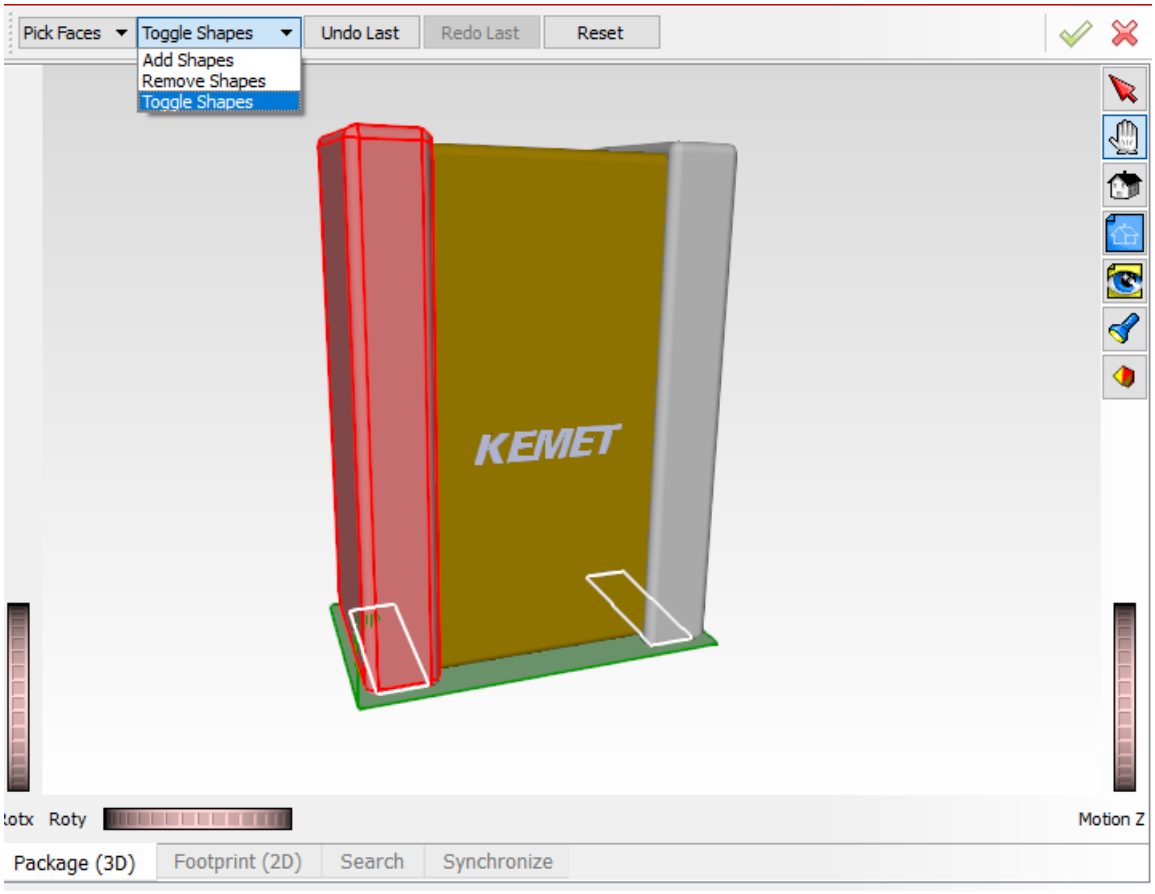
Edit 3D Feature

Packages created from templates typically have 3D solids associated with each of the features in the package. Packages created from STEP models may or may not have 3D faces and/or solids associated with the individual features depending on the both the structure of

the solid model as well as the method used to create the features. In particular, features created using the *'Contact Features'* procedures will typically not have 3D geometry associated with the features. 3D faces and solids may be added or removed from an individual feature through the *'Edit 3D Feature'* context action.

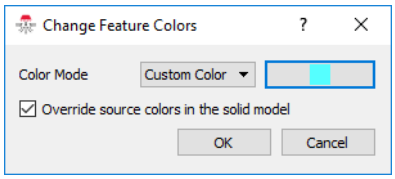


The *'Edit 3D Feature'* action will launch the 3D feature editing toolbar as shown below. Through the toolbar drop-downs, the tool may be configured to add / remove / toggle either 3D faces or solids. When in *'Select'* mode (red arrow), it is possible to add or remove either 3D faces or solids from the feature assignment. Use the *'Accept'* (green check) action to accept the changes and exit the tool.



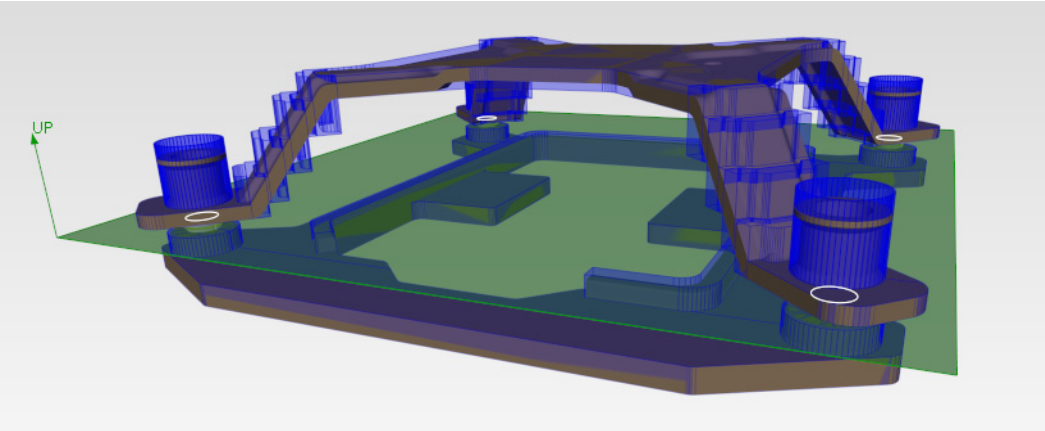
Set 3D Color

The *'Set 3D Color'* context action may be used to assign a custom color to an individual feature in a package independent of the feature based color configuration settings. The context action will bring up the *'Change Feature Colors'* dialog. In order to assign a custom color to the feature, set the color mode to *'Custom Color'* and click the color square to pick a custom color.



Simplified Models

'Simplified Models' are 3D shapes that may be added to the package model as alternate shape representations. Simplified models are created from the package through the application of a Simplified Model Generator. One of the common applications of a simplified model is visualization of the 2D place bound geometry. A *'Simplified Model Generator'* can be configured to extrude the 2D place bound shapes at their respective minimum and maximum heights for comparison and visualization with respect to the 3D package geometry. In the example below, the *'stair-step'* approximation of the 2.5D place bound regions with respect to a detailed 3D model can be inspected. The transparency can be enabled or disabled through the View Toolbar (3D).



Seating Plane

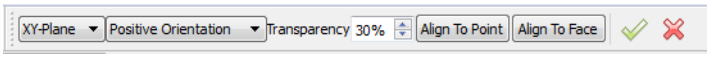
The seating plane is a datum plane that prescribes the relative position and orientation of the mating surface of the PCB with respect to the package. The seating plane is a critical element of the package model that determines the spatial relationship between the 3d dimensional package model and the 2d dimensional footprint representation. The seating plane is used in both contact generation and library synchronization.

The seating plane may be aligned with respect to any of the three principal axes (XY, XZ, YZ) and its orientation may be specified. A green vector (labeled *'up'*) indicates the orientation of the seating plane. This direction vector should point away from the mounting surface of the PCB.

Editing the Seating Plane

The seating plane may be edited through the right-click *'Edit'* context action on the seating plane in the *'Explorer.'*

The *'Edit'* action brings up the seating plane editing tool which enables specification of the alignment and orientation through pull-down options. The seating plane editing toolbar is shown below.



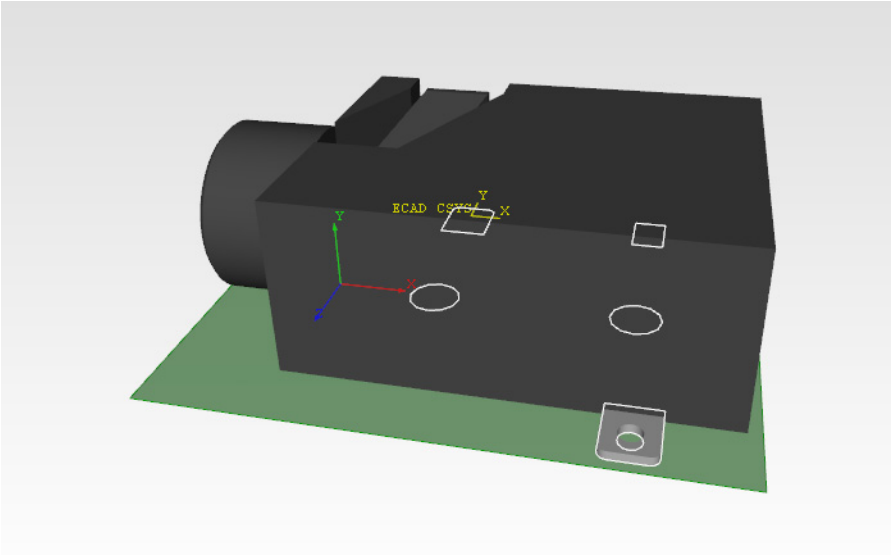
The seating plane can be aligned with either the *'Align To Point'* or *'Align to Face'* actions. *'Align To Point'* will adjust the offset of the seating plane while maintaining the currently selected orientation. *'Align To Face'* will snap the orientation to the closest of the principal axes while simultaneously adjusting the offset. After selecting one of the alignment buttons, ensure that you have the 3D Select mode enabled.

Origin

The package *'Origin'* is the origin of the *3D coordinate system*. The 3D origin is independent of the ECAD Origin. Library Creator internally maintains a relationship between these two coordinate systems for use in ECAD-MCAD library synchronization and footprint generation. The 2D ECAD (footprint) coordinate system is constrained to the seating plane. The position and orientation of the ECAD coordinate system may be changed for consistency with internal preferences and conventions. Library Creator template models typically have both the footprint and MCAD coordinate systems centered on the package with a common origin on the seating plane. The convention for Library Creator repository models is that the seating plane is the Z=0 plane for the MCAD coordinate system. The position and orientation of the 3D package coordinate system cannot be edited directly within Library Creator but can be made consistent with the ECAD origin using the Sync 3D Origin option.

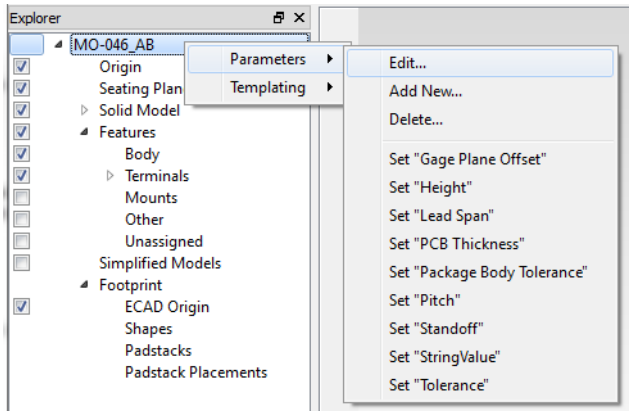
The figure below of a connector illustrates the situation when the ECAD and MCAD coordinate systems are inconsistent. In this package, the ECAD coordinate system is centered on Pin 1 and constrained to the seating plane. The MCAD coordinate system is

aligned with one edge of the body of the connector, and the X-Z plane of the MCAD coordinate system is parallel to the package seating plane.



Parameters

Parameters may be associated with the top-level package model as well as with lower level entities in the package such as features (body, terminals, etc.) and footprint elements (shapes, padstacks). In addition to predefined system parameters, additional arbitrary parameters may be added by the user. Values of existing parameters can be edited directly through the lower properties of the Details widget.

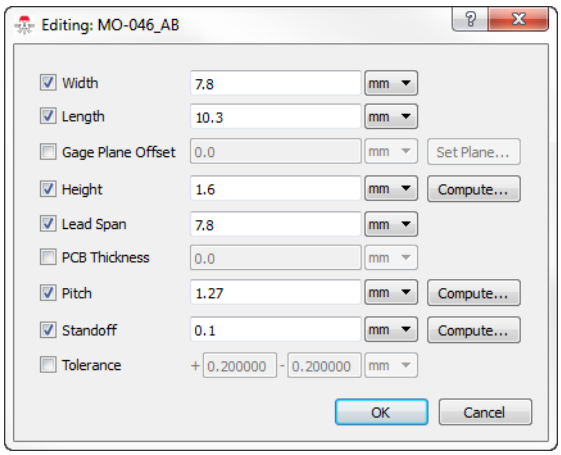


Edit / Add / Delete Parameters

Top-level package parameters may be edited, added, and deleted through right-click actions on the package explorer as shown below.

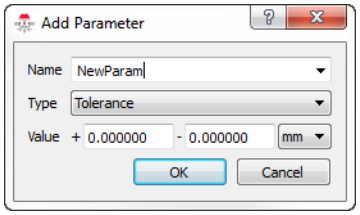
Edit Parameters

The *'Parameters -> Edit...'* action will bring up the parameter editing dialog as shown below. The dialog displays all currently existing parameters as well as the known system parameters. In order to add a parameter, select the corresponding check box. In order to remove a particular parameter, deselect the check box. Certain parameters have an associated auto-compute (*'Compute...'*) button that will populate an initial nominal value of the parameter.



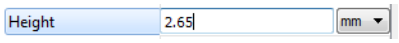
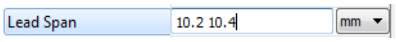
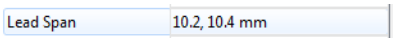
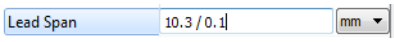
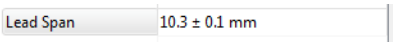
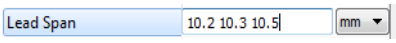
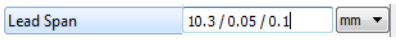
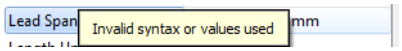
Add / Set Parameters

The Parameters -> Add New... action will bring up the Add Parameter dialog. The add parameter dialog is used to add a new user-defined parameter. When adding a new parameter, it is necessary to provide a name, parameter type, and value. Right-click menu short-cuts are provided to *'Set'* the value of individual parameters on the entity



Measurement and Tolerance Dimension

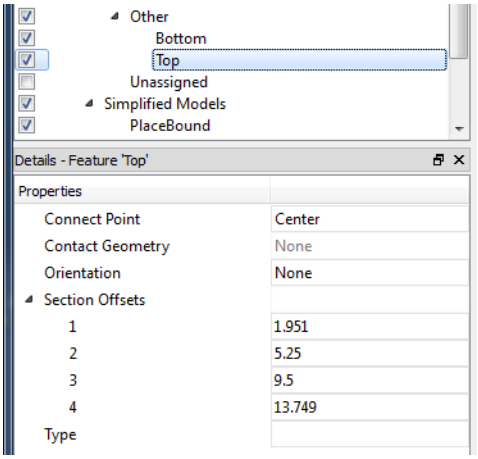
One of the most common parameter types is the measurement type. All measurements include a value and a unit. The value may be either a simple or a tolerances dimension. A tolerances dimension may be a limit (min, max) dimension, a nominal value with a symmetric (plus/minus) tolerance, or a minimum / nominal / maximum value representation to support non-symmetric tolerances. It is possible to omit either the minimum or maximum value in the non-symmetric tolerance. Below, the various entry formats and the corresponding display representations are enumerated.

Entry	Display
A simple measurement.	 
A nominal and maximum value.	 
A limit (min, max) dimension.	 
A symmetric tolerance.	 
A non-symmetric tolerance.	 
An invalid entry.	 

Section Offsets

'Section Offsets' are offsets from the seating plane that may be used during footprint generation to control the *'splitting'* or sectioning of the 3D shape in order to create a more accurate 2D approximation. Positive offsets represent sections to be taken above the seating

plane, while negative offsets are below. Section offsets are associated with individual features, and may be inspected in the *'Details'* view of a selected feature.



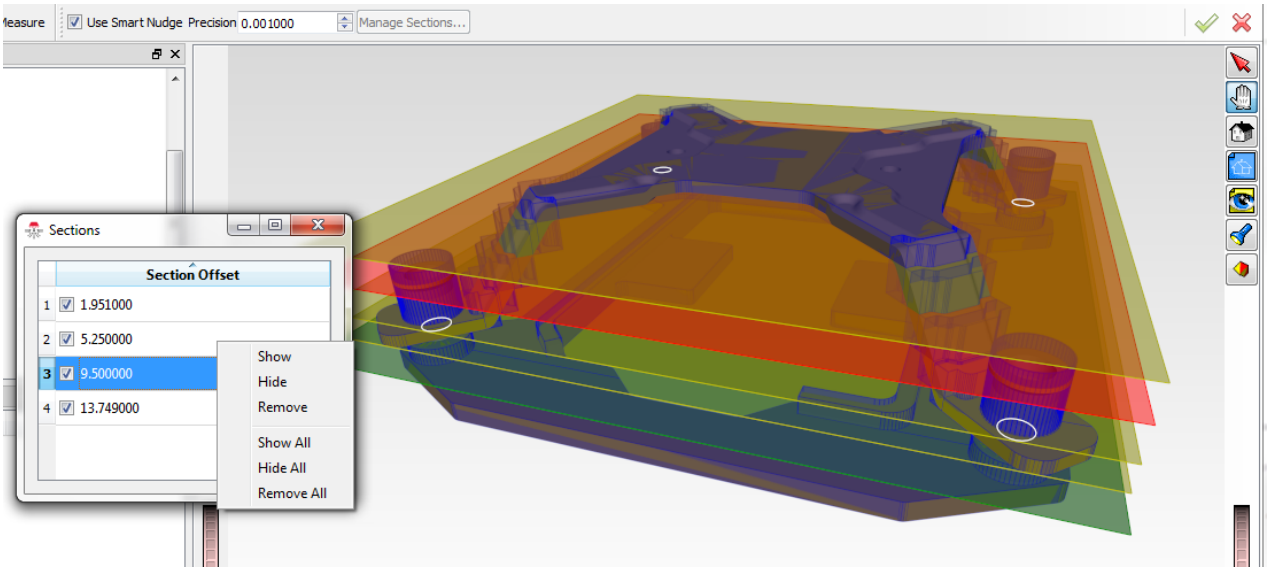
Edit Sections

The *'Edit Sections...'* context action launches the Section Editing tool for a selected feature. The tool is dismissed through the *'Accept'* or *'Cancel'* actions at the right side of the toolbar. New section offset values can be added graphically by clicking on the model while in the 3D Select mode.

It is not recommended to attempt sectioning exactly at surfaces parallel to the seating plane as the result is not well defined. To avoid this, the *Use Smart Nudge Precision* in the toolbar may be employed to incorporate a small offset *'nudge'* away from selected surfaces into the offset value.

The *'Manage Sections'* button will bring up the *'Sections'* widget that displays a list of the current section offset values for the given feature. Individual offset values can be manually edited by double-clicking to activate the editor, and values can be added and removed

through the context actions. The visibility of individual offset planes can be toggled with the corresponding check boxes.



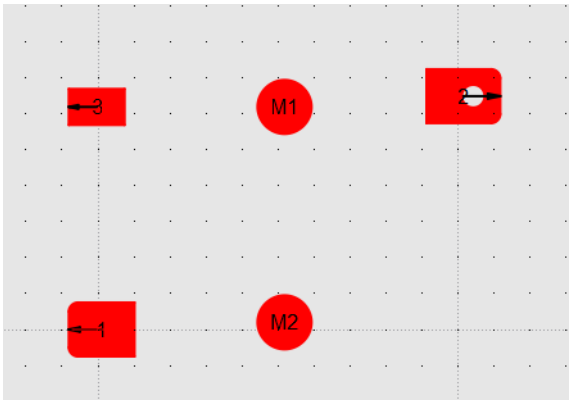
Contact Areas

Package features (body, terminals, mounts, and generic) may have an associated contact geometry that is generated directly from the 3-dimensional geometric model. In the case of a package terminal, the contact geometry is expected to reflect the physical area of contact between the terminal and the seating plane. The contact areas play a critical role in both footprint generation and library synchronization, as they serve as the link between the 3D package model and the 2D footprint. Library Creator includes default contact generation routines for each of the terminal lead form types that will create accurate contact regions for the vast majority of terminal configurations. The display of the contact area is controllable through the Layer Control widget.

It is not necessary that contact geometries represent a physical contact region. The contact geometry may be used to convey important geometric information from the 3D package model to be used in footprint generation. For example, a projected area of the package body may be used to create a keepout region in the footprint, or edges from mounting features may be projected as contact geometry into the seating plane for use as construction geometry in creating footprint symbols.

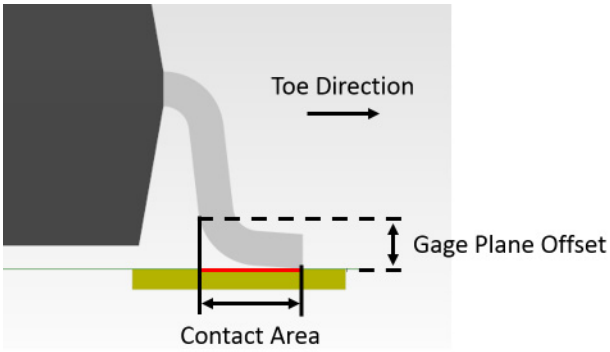
Rectangular or circular areas will be the most common contact areas, however, contact areas may also contain general 2D shapes and paths. The default contact generators will create rectangular contact regions for the majority of terminal lead forms. Certain lead forms will project the exact terminal area as the contact region.

For example, the contact regions for the connector in the image above are shown below. The contact area for the three SMT terminals is a direct projection of the extent of the terminals. A circular contact area has been generated for the two THT mount features based on a cross-section of the mounts with the seating plane.



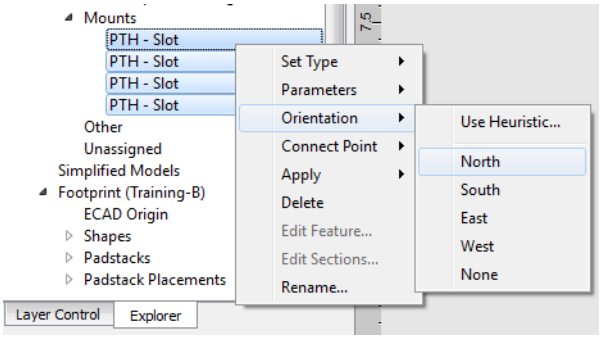
Gage Plane Offset

For terminals with a flat-bottom surface such as no-lead (SON, QFN) or rectangular end-cap (CHIP) terminations, the contact area is the area of contact between the terminal and the seating plane. Often, this is a simple projection of the bottom surface area of the terminal. For certain lead forms, such as the Gull-Wing, determining the effective contact area is more complex. JEDEC publication 95 (SPP-008) defines the *'Gage Plane Method'* for lead dimensioning. In Library Creator, the *'Gage Plane Offset'* parameter represents the height of the reference gage plane above the seating plane. This reference dimension is used in a number of the contact and contact feature generation routines. The image below illustrates the relationship between the gage plane and the contact area for a gull-wing terminal.



Contact Orientation

In addition to a 2D contact area, a feature may have an optional orientation direction. The orientation is limited to one of the four cardinal compass directions (North, South, East, West), and interpreted relative to the 2D ECAD coordinate system (i.e. East corresponds to the +X direction in the ECAD coordinate system) When assigned, the orientation is displayed in the 2D Footprint view as a vector, as well as a property of the feature in the *'Details'* widget. The contact orientation can be set on selected features through the *'Orientation'* context action, and can also be edited through the properties on the *'Details'*.



The display of the feature orientation is controllable through the Layer Control widget. When applicable, the orientation direction of the contact is intended to indicate the *'toe'* direction of the associated terminal or mount feature. Typically, this will be an outward facing orientation, although for certain lead forms such as the *'J-Lead'*, the toe-direction (and corresponding contact orientation) will be inward facing. For terminals and mount features that are associated with slots in the footprint, the orientation should be parallel to the slot orientation (i.e. the longer dimension of the slot).

Certain package features may not have a meaningful orientation. Examples of such features include terminals with lead forms including Ball, Bump, Flat Thermal, Through - Round, etc. When an explicit orientation is either unnecessary or inappropriate, the default *'None'* value may be used.

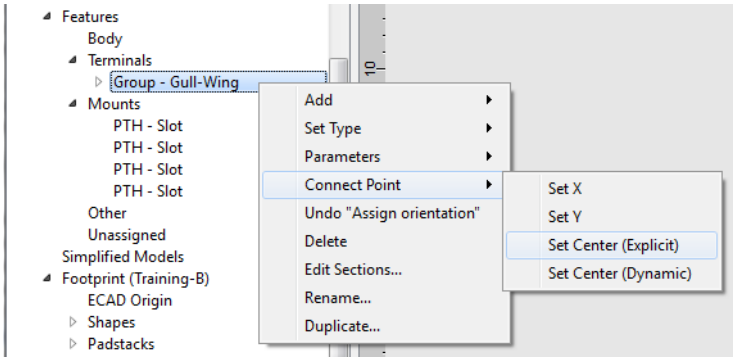
In the example above, the three SMT terminals have assigned orientation, while the two mounting pegs do not.

Extrude Contact

The *'Extrude Contact...'* context action can be used to create a 3D feature shape from an existing 2D contact. This can be used in update existing feature geometry in template-based models to reflect updates to the 2D contact areas made through editing. Alternately, it may be used to create new 3D geometry where non previously existed. The action brings up a dialog to specify the minimum and maximum heights for the extrusion.

Connect Point

The connect point is an optional attribute associated with the feature that enables explicit control over the placement of certain footprint elements. By default, the connect point of a contact is interpreted as the center of the two-dimensional bounding box of the geometry of the contact. This setting, denoted by the value *'Center'* is a dynamic location calculated from the contact geometry. The connect point can also be assigned an *'Explicit'* location that is independent of the contact geometry. An explicit connect point is displayed graphically as an *'x'* in the 2D Footprint view, and may be controlled through the Layer Control widget. The connect point can be set on selected features through the *'Connect Point'* context actions, and can also be edited through the properties on the *'Details'*. Connect points can also be moved explicitly using the Footprint (2D) editing tools, and will by default move with their associated feature (see the 2D context menu *'Options'*).



It is important to note that the role of the connect point in footprint generation is a function solely of the footprint generation configuration. Padstack generators may be configured to use the connect point to drive the padstack location, or may be configured such that the connect point plays no role at all in the footprint generation process.

3

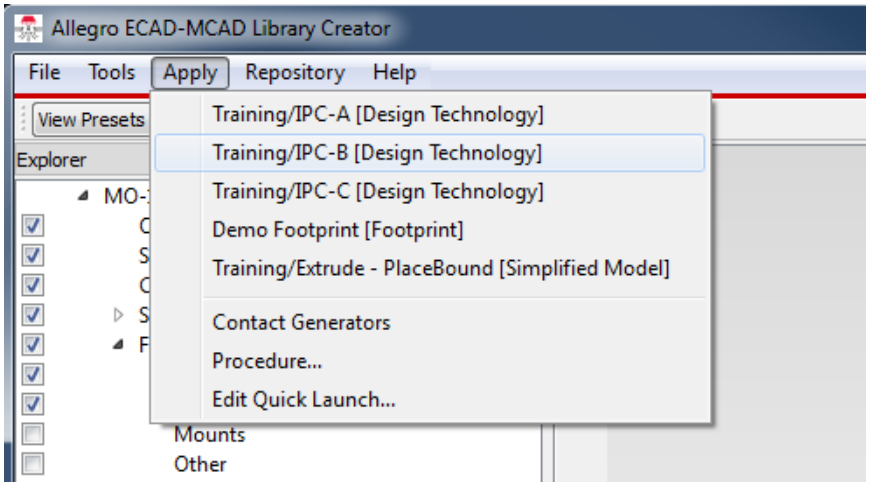
Footprint

Footprints are generated in Library Creator from the physical package model through the application of rules. Library Creator supports both fully automated footprint generation as well as interactive footprint generation combining rule automation with manual editing and manipulation of the footprint elements.

A footprint in Library Creator consists of an origin, padstacks, padstack placements, and shape, elements containing both geometry and text. Footprint details may be inspected and/or edited through the Explorer widget and the Footprint (2D) context. The visibility of various footprint elements and layers is controlled through the Layer Control widget.

Footprint Generation

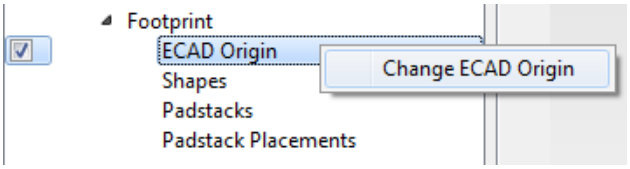
A footprint generation rule set can be applied to a given package through the *'Apply'* application menu. In most cases, an existing *'Design Technology'* or *'Footprint'* will be selected from those available in the Apply menu *'Quick Launch'* list. See for configuring the *'Quick Launch'* list.



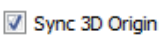
A variety of messages will be displayed in the and the generated footprint elements will appear in the Explorer tree and Footprint (2D) view.

ECAD Origin

The ECAD Origin is the 2D origin of the footprint in the Library Creator model. The location and orientation of the ECAD Origin can be manually edited using the *'Change ECAD Origin'* action in the Explorer to launch the *'Change ECAD Origin'* tool.

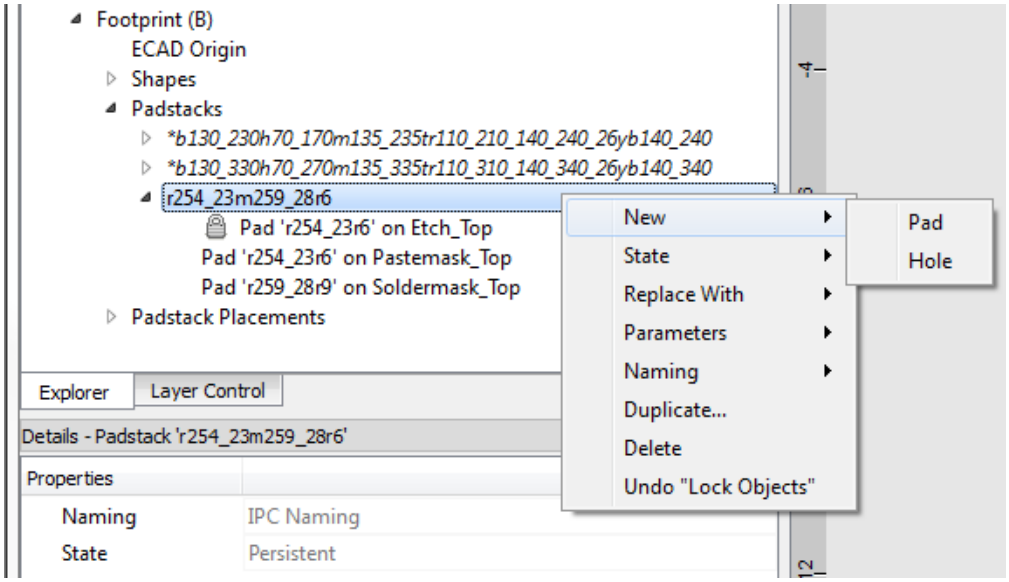


The *'Change ECAD Origin'* tool supports several centering modes as well as 90 deg. rotation through toolbar buttons. The ECAD origin is independent of the 3D package origin. When creating new packages and footprints within Library Creator, you will most often want to keep the 3D origin and footprint the same. This can be achieved by enabling the *Sync 3D Origin* option in the toolbar. Choose *'Accept'* or *'Cancel'* from the context menu or toolbar to conclude the editing action and terminate the tool. The available toolbar actions are as follows:

	Single Point Center	Set a new origin with a single point selected in the graphical context. The 'Coordinate...' context action can be used to directly enter a specific (x,y) value for the new origin.
	Two Point Center	Set a new origin at the center of two points selected in the graphical context.
	Shape Center	Set a new origin at the center of the bounds of one or more shapes selected in the graphical context.
	Rotate Clockwise 90	Rotate the origin 90 degrees clockwise.
	Rotate Counterclockwise 90	Rotate the origin 90 degrees counterclockwise.
	Sync 3D Origin	Update the 3D origin to be consistent with the 2D origin.

Padstacks

A padstack is a collection of pads along with an optional drill or slot. The padstack is placed in the footprint through a padstack placement. The pads and padstacks can be inspected and edited through the Explorer tree structure. The pads within the given padstack are visible within the padstack node of the tree. The name of the padstack is typically generated automatically during rule execution. Padstack names can be manually overwritten or changed by double-clicking on the padstack in the Explorer tree. A variety of editing actions are available through the Explorer padstack context menu. A new pad or drill/slot can be added to an existing padstack through the applicable 'Add' context menu action. In addition, the context menu provides access to padstack parameters and the persistence attribute.

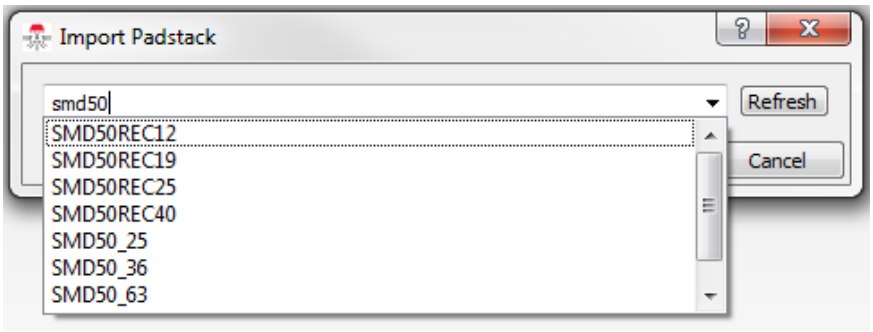


Padstack Import

In addition to creating new padstacks within Library Creator through the use of rules and interactive editing, it is also possible to leverage existing padstacks for use in new footprint. Existing padstacks may be imported into an existing footprint using the *'Import'* context menu action found on the *'Padstacks'* node in the Explorer tree, the *'Replace With'* menu on an existing padstack in use, as well as the *'Set Padstack'* menu available for package features. Any of these context menu actions will bring up the *'Import Padstack'* dialog. Note that the padstack import requires that the padstack be located on the Allegro search path.

The import padstack dialog has an auto-completer based on a list of the available padstacks. The name drop-down will filter the list of names to those containing the character substring

entered in the text box. Upon successful import, the padstack will be visible in the Explorer as an external padstack.

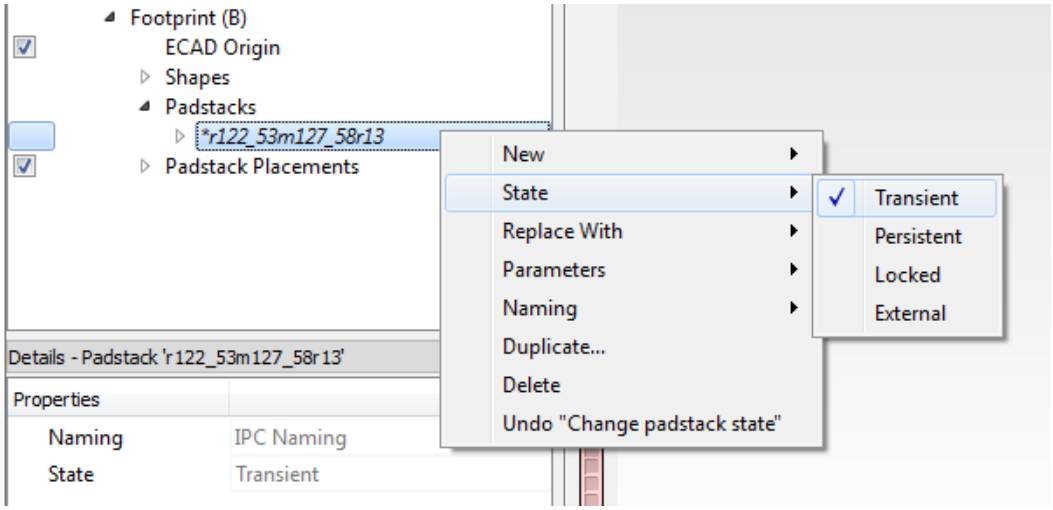


Padstack Export

Padstacks created by rules or interactive editing in Library Creator are not written to disk until Export is performed from Allegro. If desired, a padstack may be pushed to disk using the 'Export to Allegro...' context menu action. Note that when pushing a padstack to disk, the export directory is controlled through the configured export paths. This changes the state of padstack to external. Use the *Refresh* button to reload an external padstack from disk into Library Creator. External padstacks are locked for editing in Library Creator and can be edited using the Allegro padstack editor.

Padstack State

Padstacks in Library Creator are within one of four states, as described below. The state may be changed manually through the *'State'* context action.

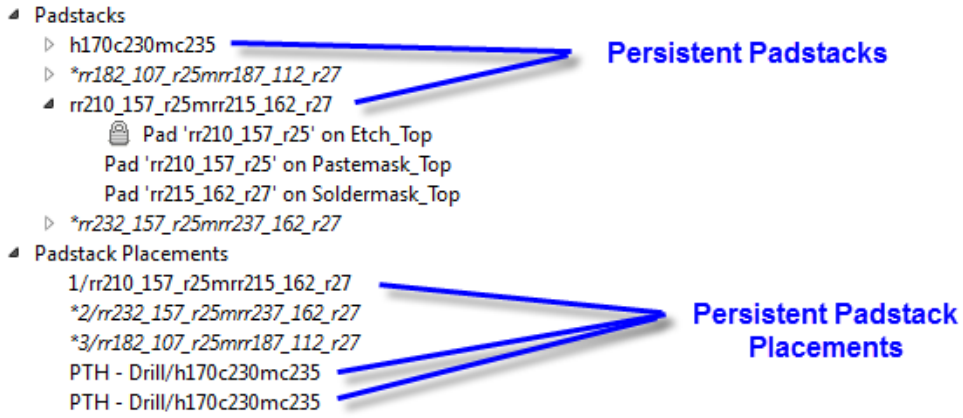


Transient

New padstacks created during rule execution will by default be in the *'transient'* state. Transient padstacks will be replaced with a newly generated padstack during rule execution. The transient state is reflected visually through the italic text style and asterisk (*'*'*) preceding the padstack name in the Explorer tree. Note that transient only refers to whether the padstacks will be replaced during footprint generation - all padstacks, including transient padstacks will be saved with the package on upload to the repository.

Persistent

Persistent padstacks will not be replaced with newly generated padstacks during rule execution. By default, newly generated padstacks will not be persistent. The persistent state is displayed in roman (non-italic) font in the Explorer. Persistence does not restrict a padstack from being modified by padstack modification rules, although pads and drills that are locked within a persistent padstack may not be modified by rule execution. Padstack placements for persistent padstacks will inherit the persistence attribute of the padstack.



Locked

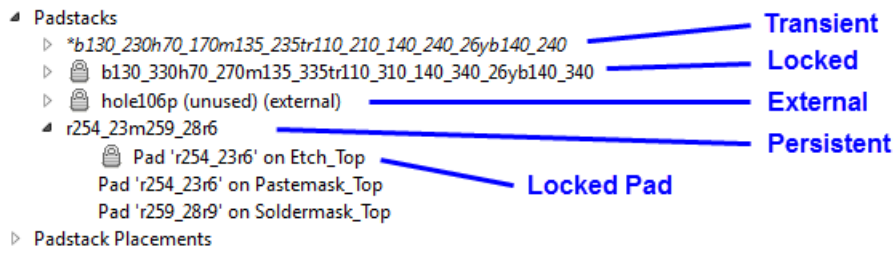
A locked padstack is one that will not be replaced or modified during rule execution. A locked padstack may still be manually edited by the user. The *'lock'* icon precedes the name of the padstack.

External

An external padstack is a reference to an existing padstack. If an existing padstack has been imported, the boolean *'External'* parameter will be set to *'True'*. External padstacks will not be overwritten during footprint export - the generated footprint will reference the existing

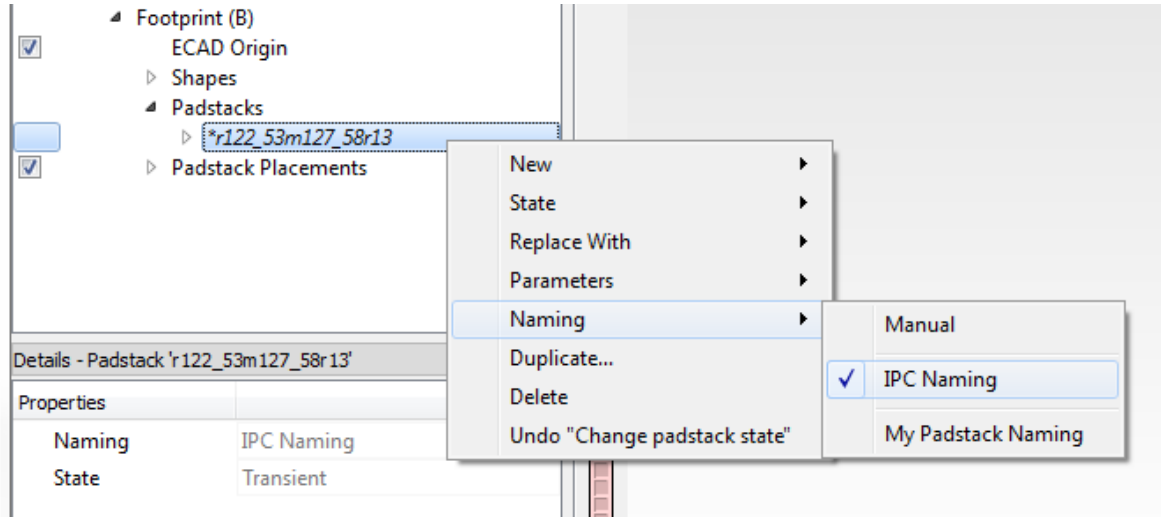
Allegro padstack of the same name. If an external Allegro padstack has been intentionally changed within Library Creator, the *'External'* attribute may be manually removed. Typically, in this case the name should be changed as well, in order to prevent inadvertently overwriting the existing padstack.

The state of the padstack is reflected in the Explorer tree as shown below.



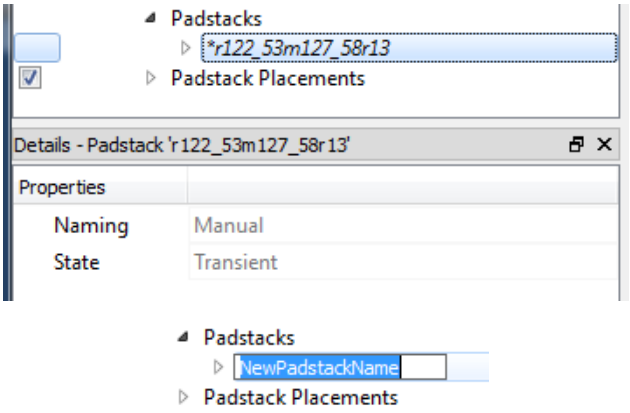
Padstack Naming

Padstacks generated by Library Creator may have an associated naming rule that auto-generates names for both pads and padstacks. The padstack naming rule is specified in the footprint settings and will be inherited by the padstack. When the padstack naming is set to a padstack naming rule, it is no longer possible to manually edit the name. When padstack details are changed manually, the name will be automatically updated based on the associated naming rule.



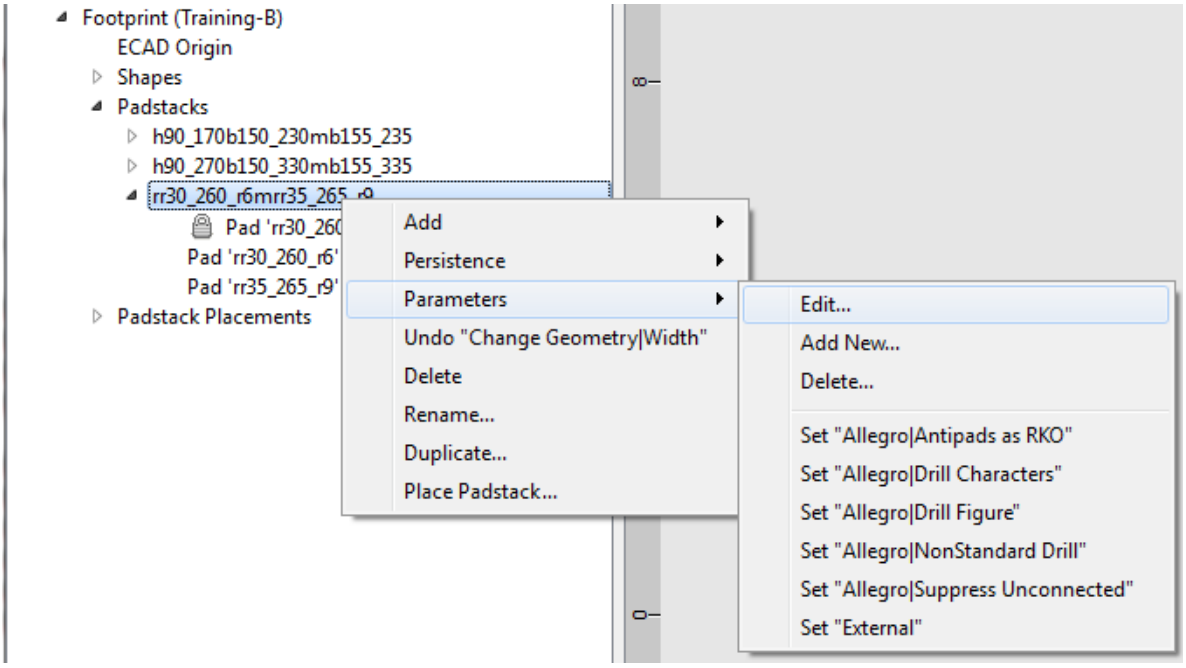
In order to change or override the automatically generated name of a padstack, the naming mode must be set to *'Manual'*. After changing the state to manual using the *'State'* context

action on a padstack, the *'Manual'* state should be visible in the Detail pane. It is then possible to manually change the padstack name by double-clicking to activate the editor for the name field as show below. External padstacks will be set to manual naming by default.



Padstack Parameters

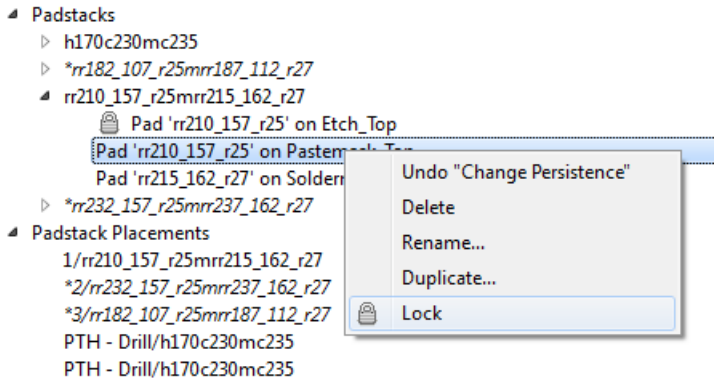
A variety of properties may be set on padstacks. Properties may be set through the *'Parameters'* context menu or edited directly in the *'Details'* widget.



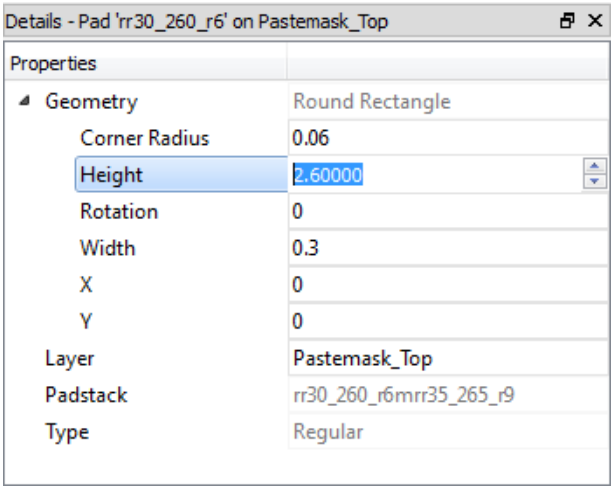
Pads

A pad represents a geometric area on a particular layer of the padstack. Pads are elements of padstacks. Pads may be one of the standard shapes or a non-standard shape. The standard shapes include circular, rectangular, donut, and a rectangle with rounded corners.

The pad context menu provides actions to delete, rename, duplicate, and lock the pad.



The details of a pad can be viewed through the Details widget. The layer of an existing pad may be changed by double clicking to activate the editor for the layer property. The image below shows the critical details of a rectangular pad on the Pastemask_Top layer.



Pads created through the footprint generation process may be edited manually. For standard shape pads, the geometry properties may be directly edited through the Details widget by double-clicking in the value field. A standard pad shape may be changed to a different standard pad shape through the Edit Properties dialog. Non-standard shapes (i.e. general polygons) may be edited directly with the shape editing tools available through the Toolbar (2D).

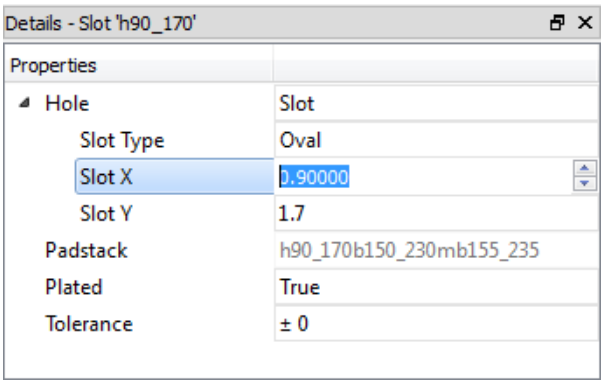
Locked Pads

Pads may be either *'Locked'* or *'Unlocked'*. The default state is unlocked. If a pad is locked, a lock icon will be displayed before the pad name in the Explorer tree. Pads may be manually locked or unlocked through a context menu action. Locking a pad will change the padstack to persistent.

'Locking' prevents a pad from being modified by rule application. This is useful in interactive footprint construction - manual edits of footprint details may be combined with rule automation to support efficient padstack and footprint generation.

Drills and Slots

A padstack may contain at most one drill or slot. The details of a drill or slot feature may be directly edited through the properties of the *'Details'* widget - it is not possible to edit drill or slot geometry using the 2D editing tools. The drill or slot shape, dimensions, tolerance and *'plated'* attribute are available for inspection and editing through the properties.



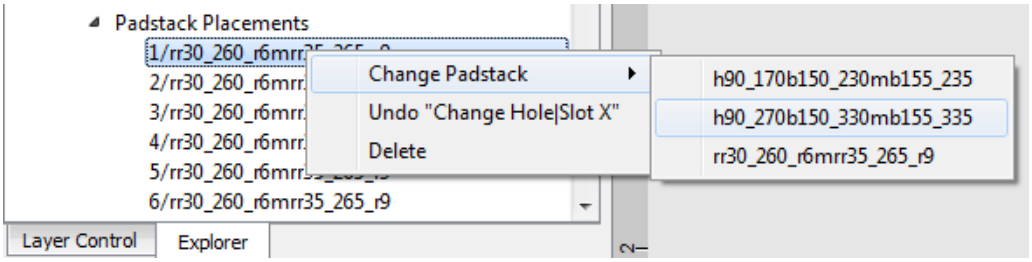
Locked Drills And Slots

Drills and Slots may be either *'Locked'* or *'Unlocked'*. The default state is unlocked. If a pad is locked, a lock icon will be displayed before the pad name in the Explorer tree. The interpretation and application of locked drills and slots is analogous to that of .

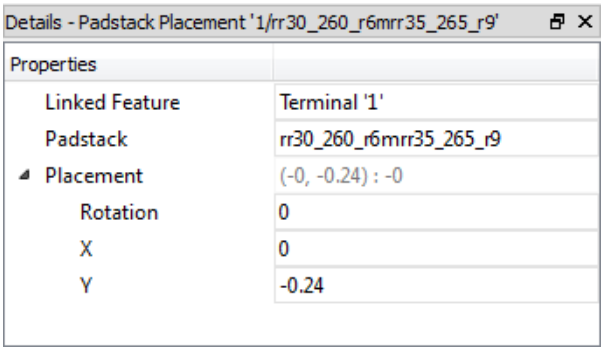
Padstack Placements

A padstack placement is an instance of a padstack that has been placed in the footprint A padstack placement includes a position (x, y) and a rotation, as well as a reference to the package feature (typically a terminal or mount feature) that it is associated with. Padstack placements inherited the *'persistence'* of their padstack.

An existing padstack placement may be assigned to a different padstack or deleted through the context menu.



The properties of a padstack placement may be edited through the *'Details'* widget.



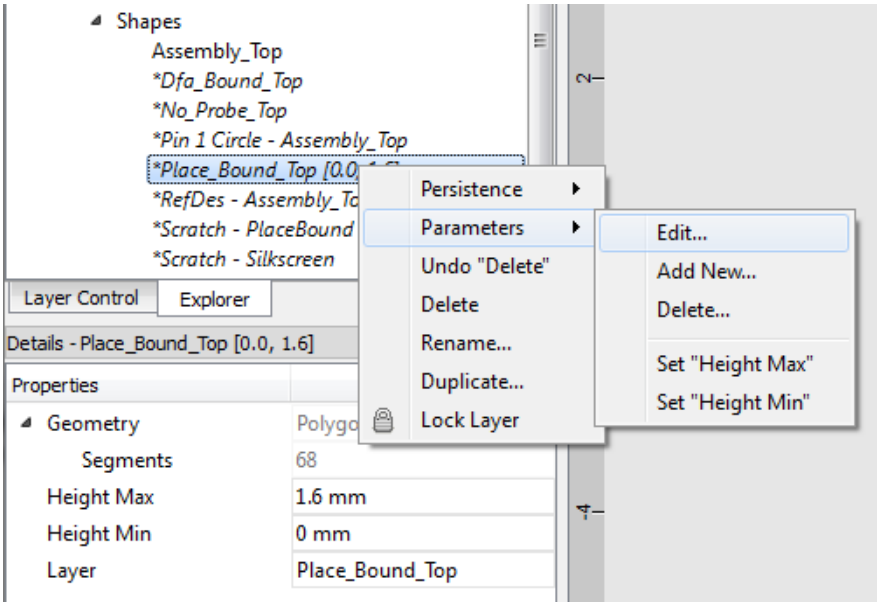
Shapes

The *'Shapes'* node of the *'Footprint'* in the Explorer widget contains all geometric elements of the footprint that are not part of a padstack. Examples of footprint shape elements includes assembly symbols, placebound shapes, keepouts, gang-clear soldermask cutouts, silkscreen symbols, reference designator text, etc. *'Shapes'* are generated by applying rules to the package as well as through manually editing. Each *'Shape'* element is associated with a particular layer of the footprint. A single shape can element can contain multiple geometric elements, including both path-based and area-based geometry as well as text. A shape may also have associated properties - the most common properties are the minimum and maximum heights.

The layer assignment can be inspected and edited through the *'Details'* widget of the selected shape. For shapes containing a single standard geometry element, the shape details may also be inspected and edited through properties of the *'Details'* widget. For shapes containing multiple geometric elements, individual geometries may be selected and edited through the Footprint (2D) editing context.

Shape Heights

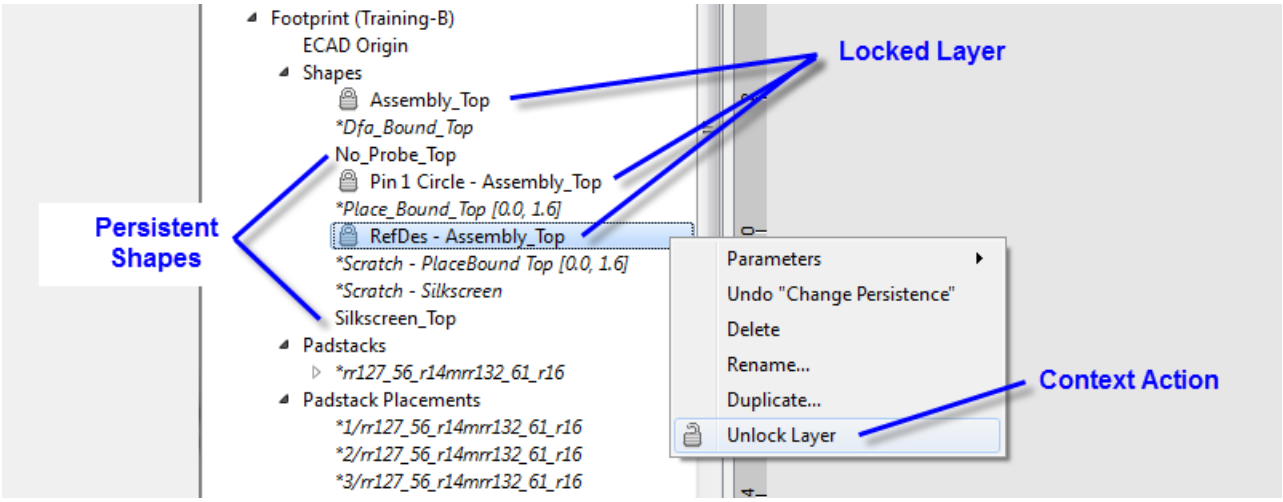
The optional Height Min and Height Max properties of a *'Shape'* element specify a minimum and maximum height for all geometry in the shape. Maximum and/or minimum heights may be assigned automatically to shapes during footprint generation on applicable layers such as Place_Bound and Dfa_Bound. Existing height values may be directly edited through the properties of the *'Details'* widget. Height properties may be added or removed from shapes through the *'Parameters -> Edit'* context menu actions.



Persistent Shapes

Shapes in a footprint may be either *'persistent'* or not. Persistence refers to whether or not the individual shape element is preserved when footprint generation rules are applied. By default, shapes generated by rules are not persistent. This is reflected in the italic text style and asterisk (*'**'*) preceding the shape name in the Explorer tree. Manually generated shapes are persistent by default. When a footprint generation rule is applied, all non-persistent shapes are removed from the footprint, and replaced with newly generated shapes. Note that all shape elements are saved with the footprint in the repository regardless of whether they are *'persistent'* or not - persistence applies only to the application of footprint generation rules. Note also that setting a given shape to *'persistent'* does not prevent duplicate or similar shapes from being generated when rules are applied. The most common application of a persistent shape would be to add additional geometry to the footprint that is not

automatically generated. For example, a board edge annotation might be added to the footprint.



Layer Locking

'Locking' a footprint layer both preserves the existing shapes on the layer, as well as preventing the footprint generation rules from creating any new shapes on the layer. *'Locking'* a layer can be useful in combination with automated footprint generation when manually editing is needed. *'Locking'* a layer sets the individual shapes to be *'persistent'*.

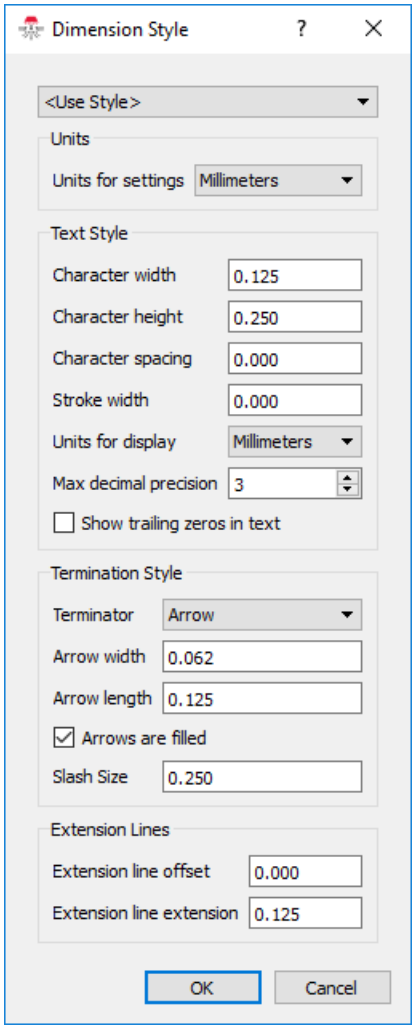
Dimensions

Dimensions can be added to a footprint as an aid in drafting, for measurements, and for documentation purposes. Library Creator supports both symmetric and relative dimensions. Horizontal, vertical, and diagonal dimensions of either type can be created through the applicable tools in the 2D Toolbar. Dimensions created in Library Creator are propagated to Allegro during export on a custom subclass named 'Dimension' within the 'Package Geometry' class. If it is desired to populate dimensions on a different class or subclass, this can be done by adding a mapping for the built in *Dimensions* layer through the configuration class or subclass mapping. The dimension value can be inspected and edited through the *Details* widget of the selected dimension.

Dimension Style

A variety of settings control the formatting and styling of a dimension. Library Creator provides a number of predefined built in style settings. Additionally, users can modify settings for individual dimensions as well as create custom dimension styles through the configuration

by creating an instance of a Dimension Style configuration object. When creating a new dimension, the currently active style will be used by default. During the creation process, the style settings may be changed through the *Set Dimension Style* context action. After the dimension has been created, the style may be changed through the *Properties* context action. The available style settings are shown below. To apply one of the existing style settings to the current dimension, select the desired style through the drop-down at the top of the dialog.



4

Repository

The Library Creator repository enables centralized configuration-controlled storage and management of package models. Package models may be instances of parametric templates or packages created from a featurized external model. Multiple users within a single organization share access to a common set of packages through an account. Additionally, users have access to templates and package instances provided by the system. In order to access the repository, a user must be signed in as an authorized user of a valid repository account.

There are numerous package attributes that are indexed in the repository. These attributes can be used in manual repository searches as well as automated search during library synchronization.

Searching the Repository

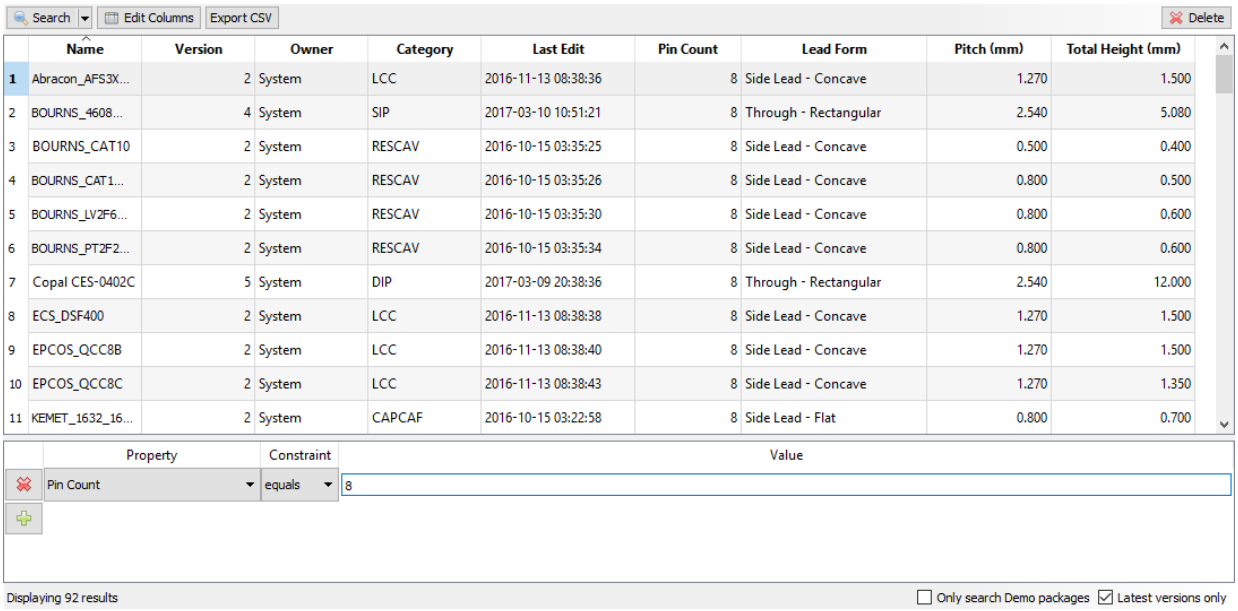
To initiate a search of the repository, select the 'Search' tab. The image below illustrates the primary controls used in a typical search. Individual search terms may be added and removed using the '+' and 'x' buttons at the left side of the property pane. Each search constraint involves a single repository index property.

The specific constraints available for a given index property are a function of the property type. After search terms have been specified, the search may be initiated by pressing *'Enter'* in one of the property value fields or through the *'Search'* button in the top-left corner of the search pane. The packages returned by the search will be the repository models that simultaneously satisfy all of the specified constraints. Dimensional index properties will be displayed in the preferred units.

The check boxes next to *Only search account packages* and *Latest version only* found in the lower right-hand corner of the search pane provide a global filter on the results returned. Selecting *'Only search account packages'* will restrict the search to only packages owned by the current account and exclude all *'system'* provided packages.

'String' and *'Enumerated String'* properties support *'contains'*, *'equals'*, *'startswith'*, *'endswith'*, and *'like'* constraints. All queries are case-insensitive. The *like* operator implements a SQL *'like'* query which permits wildcards (*'%'* for zero or more characters, *'_'*

for a single character). Integer and dimension (float) quantities support *'between'* and *'approx. equal'* constraints in addition to the common equality and inequality constraints.



Edit Visible Columns

The *'Edit Columns'* button at the top of the *'Search'* pane will bring up the dialog below. The visibility of columns in the search view can be toggled through the dialog.

Loading a Package

An individual package may be loaded into the session by double-clicking the selection in the search results, or through the right-click *'Load'* context actions. It is possible to switch between multiple packages in a session through the *'File -> Loaded Packages'* application menu action.

Deleting a Package

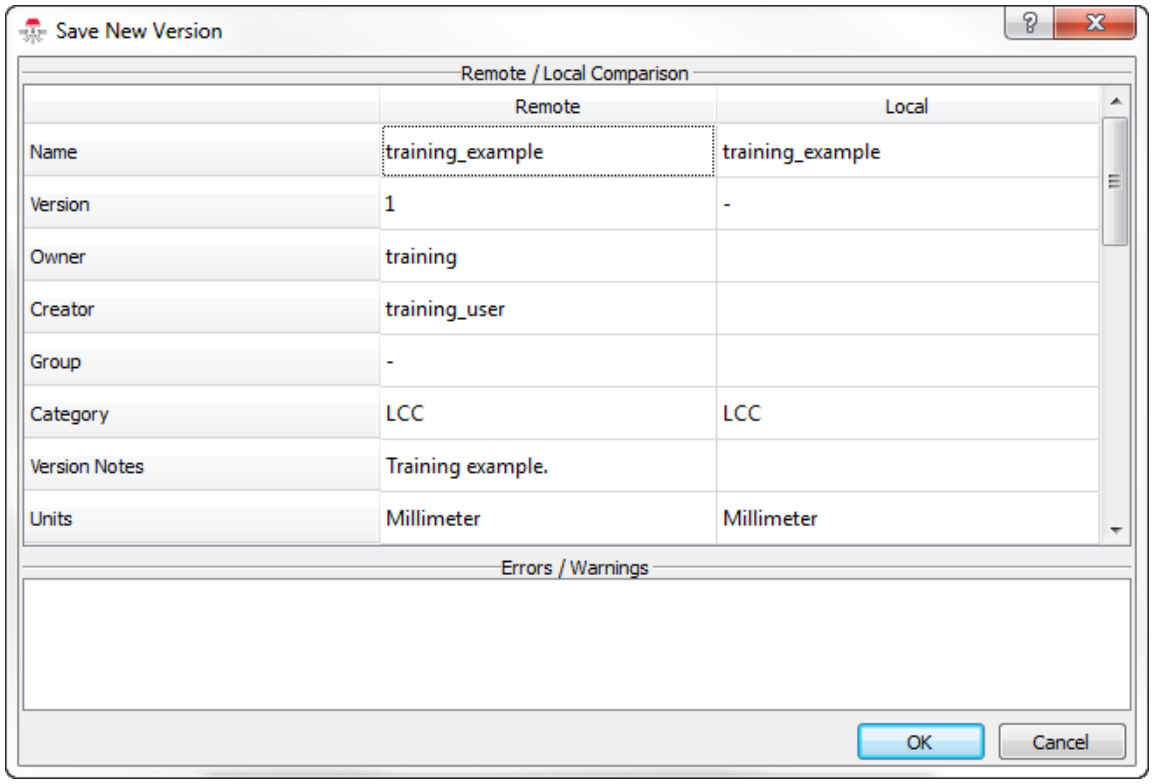
A package selected in the search results may be deleted from the repository with the *'Delete'* button in the upper-right corner of the search pane. By default, the *'Delete'* action is disabled. This action may be enabled by setting the needed *'Packages:Delete'* permission as described in the *'README'* file found in the application installation directly. Note that the delete action is irreversible and should be used with caution. Note also that only the selected version of the package is deleted (or loaded). If it is desired to operate on only the latest (most

recent) versions of a package, ensure that the *'Latest Versions Only'* check-box is selected in the lower-right corner of the screen.

Package Upload

Package models created either through templating or import and featurization of an existing geometric model can be uploaded to the repository. It is recommended that packages to be uploaded have contacts generated and that the primary pin is identified. To upload a package in session, select either the *'File -> Upload -> Save New Version...'* menu action. In certain cases, it may be desired to replace the current package version. The *'File -> Upload -> Replace...'* function is disabled by default. The *'Replace...'* action may be enabled by setting the *'Packages:Replace'* permission as described in the *'README'* file found in the application installation directly.

When uploading a package, an upload dialog similar to that shown below will be displayed. The calculated values of the index parameters may be seen in the *'Local'* column. If a prior version of the current package exists in the repository, the repository index details will be displayed in the *'Remote'* column. A variety of info, warnings, and /or error messages may be shown in the pane below.



Index Parameters

Repository models are indexed by a number of package attributes. The index properties can be used in manual repository searches and are also relied upon heavily in the automated searches performed during library synchronization.

The repository index properties are described below:

Index Property	Type	Description
Name	String	The name of the package model.
Version	Integer	The repository version of the package model.
Owner	String	The account owner of the package. This field is non-searchable, however, the *'Only Search My Packages'* check-box can be used to limit to packages owned by the current user account.
Creator	String	The name of the user who uploaded the package version. This field is non-searchable.
Category	Enumerated String	An enumerated classification of package families based on IPC-7351 naming conventions.
Version Notes	String	The comment entered by the user at the time of package upload.
Units	Enumerated String	The name of the package model unit.
Created	Date / Time String	A string representing the date and time that the model was initially created. The format is: YYYY-MM-DD HH:MM:SS. A substring search can be performed to return all models created on a certain date.
Last Edit	Date / Time String	A string representing the date and time that the model was initially created. The format is: YYYY-MM-DD HH:MM:SS.
Pin Count	Integer	The total number of terminals in the package.
Lead Form	Enumerated String	The lead form classification of the majority terminal group within the package.
Mount Form	Enumerated String	The mounting technology of the package. Possible values are *'Surface'*, *'Through'*, and *'Unset'*.

Pitch	Dimension (Float)	The center-to-center spacing distance for adjacent terminals.
Total Length	Dimension (Float)	Overall package length. The length is the longer dimension of the package measured parallel to the seating plane.
Total Width	Dimension (Float)	Overall package width. The width is the shorter dimension of the package measured parallel to the seating plane.
Total Height	Dimension (Float)	Overall package height. The height is measured from the seating plane.
Body Length	Dimension (Float)	Package body length. The length is the longer dimension of the package body measured parallel to the seating plane.
Body Width	Dimension (Float)	Package body width. The width is the shorter dimension of the package body measured parallel to the seating plane.
Body Height	Dimension (Float)	Package body height. The height is measured from the seating plane.
Contact Extent Length	Dimension (Float)	Longer dimension of the extent of the contacts within the seating plane.
Contact Extent Width	Dimension (Float)	Shorter dimension of the extent of the contacts within the seating plane.
Contacts Minimum Center Distance	Dimension (Float)	Radius of the largest circle centered on the bounding box of the contacts that is empty of contacts.
Signature	String	A unique string identifier for the package instance - primarily for internal application use.

User Accounts

In order to access the repository, a user must be signed-in as an authorized user of a valid repository account. A user can sign-in and sign-out of the repository, change password, and view status through the applicable actions under the *'Repository'* drop-down menu. All models added by a user are visible to all users of the same account (a single account is shared by all members of the organization who require access to the common set of repository models). Both user-supplied models as well as the Library Creator repository

models are accessible in both searching and library synchronization activities within Library Creator when signed in to a user account.

Pull Updates

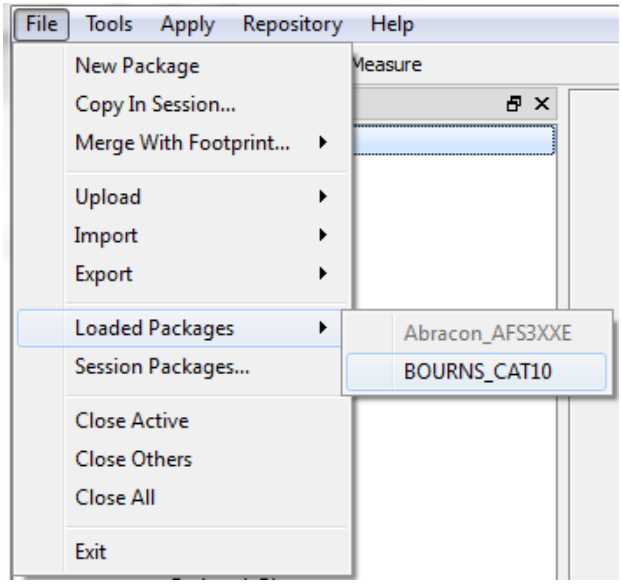
The *'Repository -> Pull Updates'* menu action will pull pending updates from the repository into the local session. Updates are automatically pulled at application start-up and it is seldom necessary to manually pull updates unless multiple people are simultaneously editing configuration data.

Rebuild Registry

Library Creator stores a locally cached version of all configuration data. The *'Repository -> Rebuild Registry'* menu action will regenerate this local cache from the repository. It is rarely necessary to force a registry rebuild. In certain instances, the registry will be rebuilt on the initial application launch following a software update.

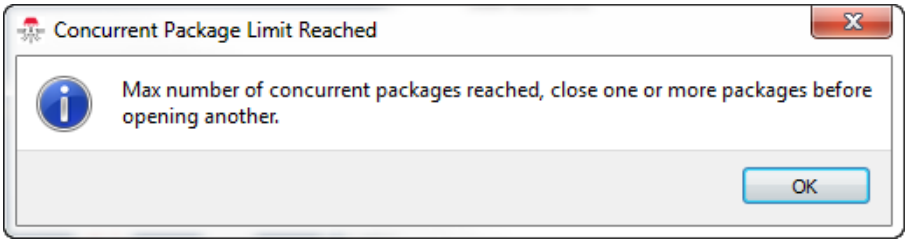
Packages in Session

Multiple packages can be simultaneously loaded in a single Library Creator session. Only one package in a session is *'Active'*. An alternate package can be made active using the *'File -> Loaded Packages'* menu action.



Concurrent Package Limit

There is a limit to the number of packages that may be simultaneously loaded in session. When this limit has been reached, a dialog will be displayed. At this point, one or more packages in the session must be closed using the applicable *'File -> Close'* menu action.



Permissions

Permissions control certain elements of the functionality as described below. User permissions are administered by either the account or site administrator of the repository server. Permissions are displayed through the *'Repository -> Show Status'* menu action. If user permissions are not visible in the dialog, file-based permissions are in effect.

The following permissions are supported by the application:

Configuration:Edit	Edit persistent configuration data managed through the Configuration UI (Tools -> Configuration...). This includes the ability to create new configuration objects, as well as to edit, rename, or delete existing configuration objects.
Configuration:View	View existing configuration data through the Configuration UI (Tools -> Configuration...). If this permission is not enabled the "Configuration..." action will not be visible in the Tools menu.
PackageTemplates:Edit	Edit template definition data managed through the Template Browser (Tools -> Templates...). This includes the ability to create new template objects, as well as to edit, rename, or delete existing template objects.
PackageTemplates:View	View template definition data managed through the Template Browser (Tools -> Templates...). If this permission is not enabled the "Templates..." action will not be visible in the Tools menu.

Packages:Create	Upload new packages as well as new versions of existing packages. This permission applies to both the upload of new packages and new package versions through the File -> Upload -> Save New Version... action as well as the regeneration of existing template instances with the "New Version" selection.
Packages:Delete	Delete packages and individual versions of packages.
Packages:Delete	Replace versions of existing packages through both the File -> Upload -> Replace... action as well as the regeneration of existing template instances with the "Update Current" selection.

File-Based Permissions

If user permissions are not visible through the *'Repository -> Show Status'* dialog, legacy file-based user permissions will be in effect. File-based permissions are controlled through a simple text file that is read by the client application at launch. After the initial launch of the client application, the default permissions will be written to the file:

```
<user_home>/Allegro ECAD-MCAD Library Creator/permissions
```

If desired, a permissions file can be explicitly provided through a file path specified through the "-DPermissions= " command-line argument to the client application executable. This enables centralized management of application permissions, and will take precedence over the local user permissions.

The permissions file may be edited to control permissions during subsequent launches of the client. The structure of the permissions file is a series of individual permission names, prefixed by either a "-" or a "+" to disable or enable the setting, respectively. After editing the permissions file, it will be necessary to exit and relaunch the application for the new permissions to take effect.

5

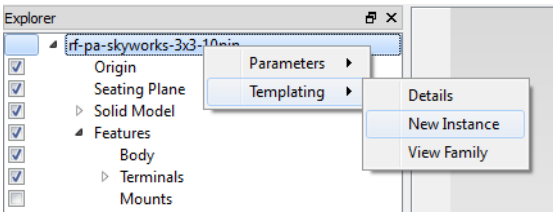
Templates

Library Creator provides an extensive collection of package, body, and terminal templates for the creation of packages from common component styles and families. Currently, over 100 package templates as well as numerous body templates and terminal templates for the majority of common terminal types are available.

The repository contains thousands of instances of common packages based on the system templates. A new template instance can be created either from an existing repository package or through the Template Browser.

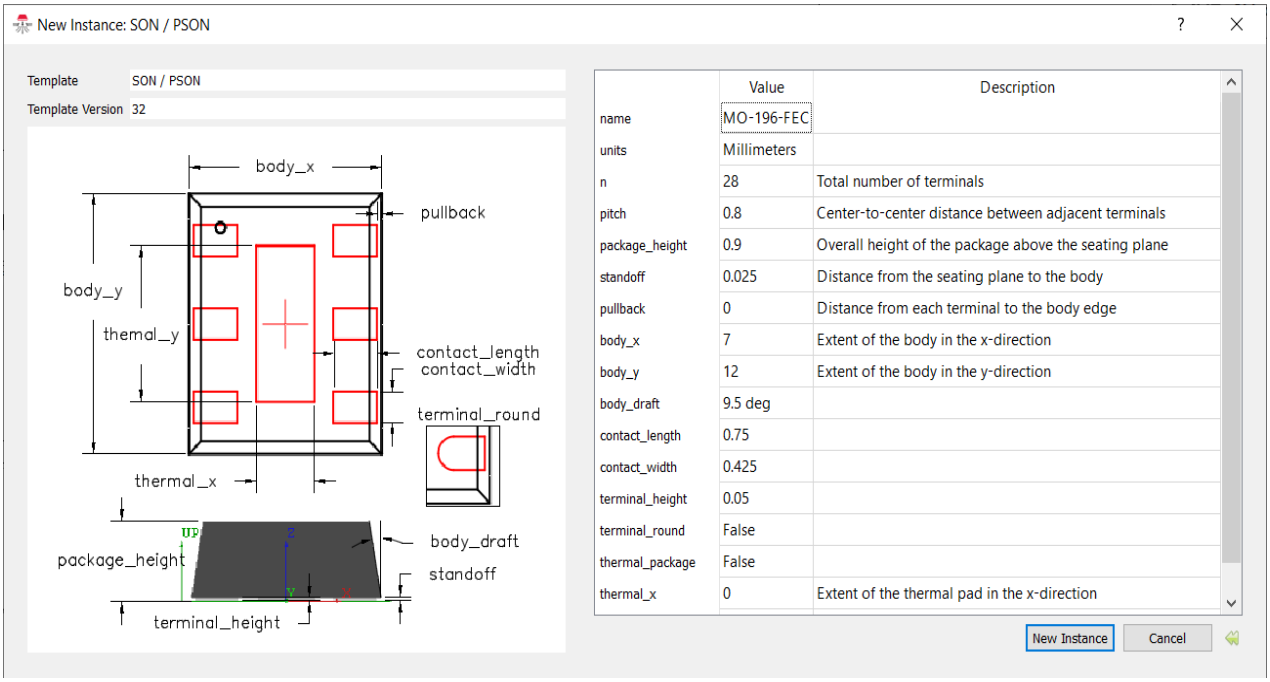
Creating an Instance of a Template

If an actively loaded package is an instance of a template, a new instance (or variant) of the package can be created through the *'Templating -> New Instance'* action on the right-click menu on a selected package in the *'Package Explorer'* as shown below.



This action will launch the *'New Instance'* dialog. Note that both the template name and current version are displayed at the top of the dialog. New values can be specified for any of the input parameters. Selecting the *'New Instance'* action will create a generate a new package instance with the given input parameter values. This package instance can then be

uploaded to the repository. For more information about uploading package models to the repository, refer to the Package Upload section.

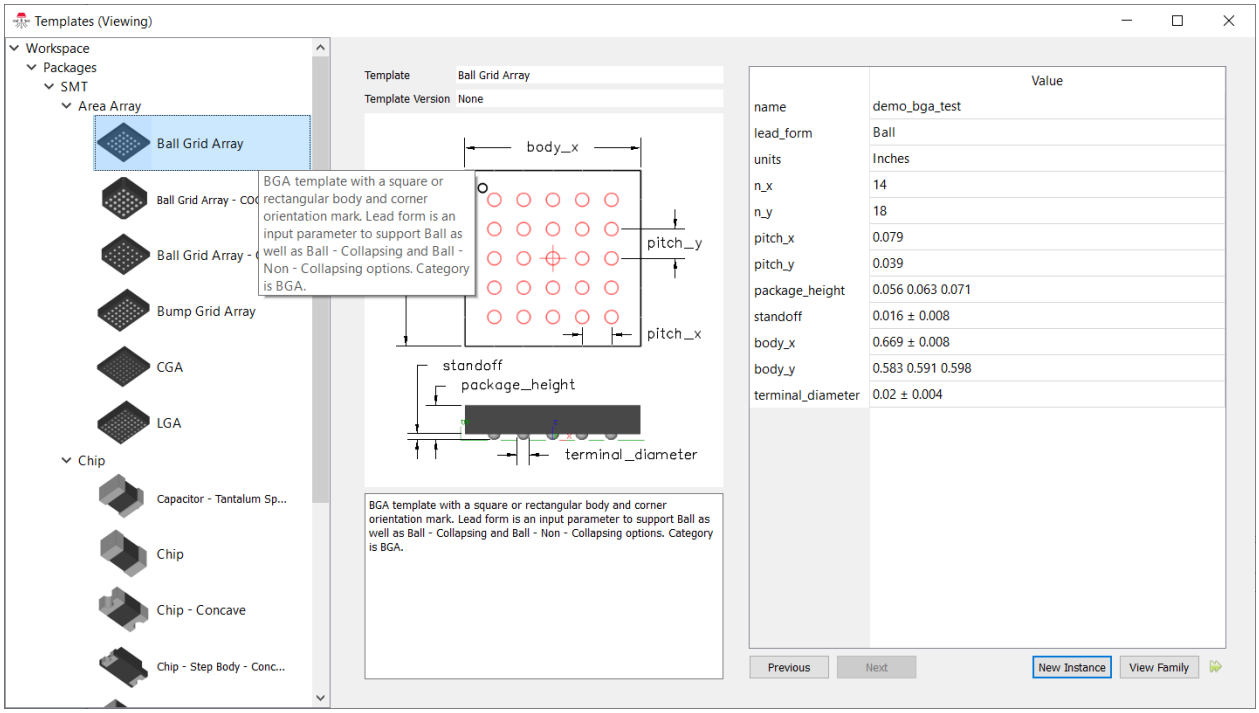


The syntax for entering tolerances on dimension and/or measurements is consistent with that used in editing package parameter descriptions and is described here. One important difference between template parameters and package parameters is that template parameters do not carry an independent unit. All template parameters are interpreted in the unit of the package being generated, which is typically one of the inputs to the template.

Template Browser

The Template browser allows you to browse through the available templates. Templates are organized by categories and displayed in a tree on the left side of the *Templates* dialog. The tree also displays a thumbnail view of the template. Selecting a particular template from the tree displays the image, description, template name and the parameter table. You can also

view the description by hovering the mouse cursor over the template in the tree view. These details are fetched from the most recently created instance.



The template browser maintains a local history of instance data that can be accessed through the *Previous* and *Next* buttons in the lower-left corner of the parameter table. You can also view and hide the description of the parameter values by clicking the icon.

Creating a New Template Instance

To create a new instance of a template:

1. Choose *Tools - Templates*.

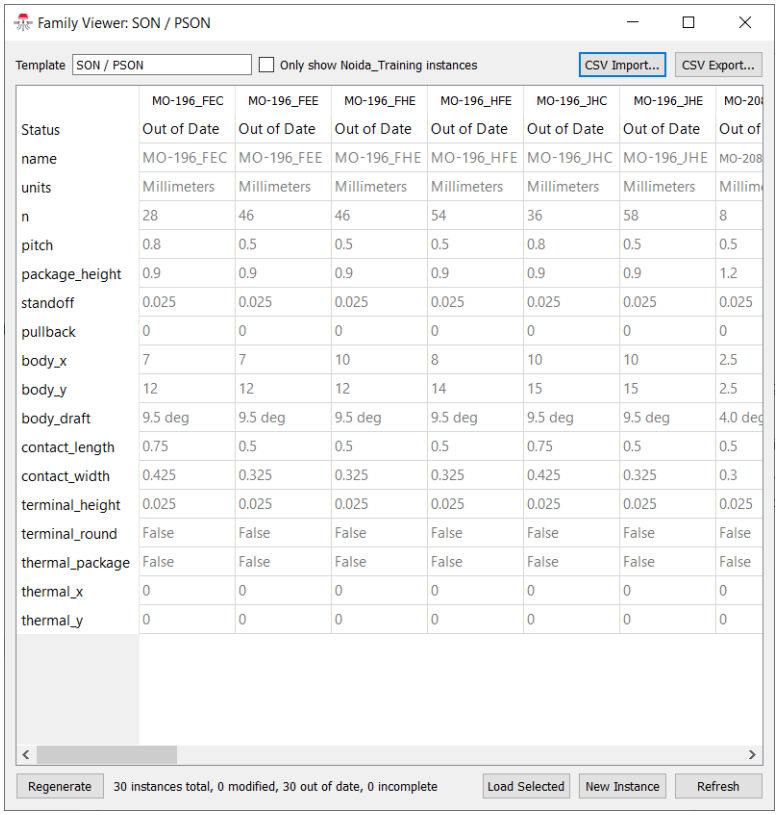
   Templates dialog is displayed.

2. Select a template from the templates tree.
3. Edit the required fields in the parameter table.
4. Click *New Instance*.

   A new instance of the selected template is created and displayed on the grid.

Family Viewer

Family Viewer for a given template can be accessed from the template browser through the *View Family* button. Display of the parameter descriptions column can be toggled through the arrow icon in the lower right corner of the parameter table.



	MO-196_FEC	MO-196_FEE	MO-196_FHE	MO-196_HFE	MO-196_JHC	MO-196_JHE	MO-20
Status	Out of Date	Out of Date	Out of Date	Out of Date	Out of Date	Out of Date	Out of
name	MO-196_FEC	MO-196_FEE	MO-196_FHE	MO-196_HFE	MO-196_JHC	MO-196_JHE	MO-208
units	Millimeters	Millimeters	Millimeters	Millimeters	Millimeters	Millimeters	Millim
n	28	46	46	54	36	58	8
pitch	0.8	0.5	0.5	0.5	0.8	0.5	0.5
package_height	0.9	0.9	0.9	0.9	0.9	0.9	1.2
standoff	0.025	0.025	0.025	0.025	0.025	0.025	0.025
pullback	0	0	0	0	0	0	0
body_x	7	7	10	8	10	10	2.5
body_y	12	12	12	14	15	15	2.5
body_draft	9.5 deg	9.5 deg	9.5 deg	9.5 deg	9.5 deg	9.5 deg	4.0 deg
contact_length	0.75	0.5	0.5	0.5	0.75	0.5	0.5
contact_width	0.425	0.325	0.325	0.325	0.425	0.325	0.3
terminal_height	0.025	0.025	0.025	0.025	0.025	0.025	0.025
terminal_round	False	False	False	False	False	False	False
thermal_package	False	False	False	False	False	False	False
thermal_x	0	0	0	0	0	0	0
thermal_y	0	0	0	0	0	0	0

It is also possible to manage template instances through a collective view of all instances of a given template. The *'Family Viewer'* can be accessed through either the *'View Family'* button from the template browser or through the *'Templating -> View Family'* context menu on a package. The family viewer is shown below. The family viewer enables bulk editing and regeneration of instances within a given template. Note the *'Only show instances'* check-box in the toolbar of the family viewer.

The *'Status'* cell in the top-row indicates whether a given instance is *'Up to Date'* with respect to the template. A template instance is generated from a specific version of the template. If changes are made to the template subsequent to the creation of the instance, the instance is no longer *'Up to Date'* with respect to the template. If desired, the existing instances can be *'regenerated'* against the most recent version of the template using the *'Regenerate'* button. From the family viewer, it is possible to either create new instances or edit existing instances.

6

Featurization

Library Creator supports the import and featurization of 3D STEP models for use in footprint generation and library synchronization. Using the internal featurization capabilities, 3D models may be rapidly augmented with the necessary package features and pin numbers for inclusion in the repository.

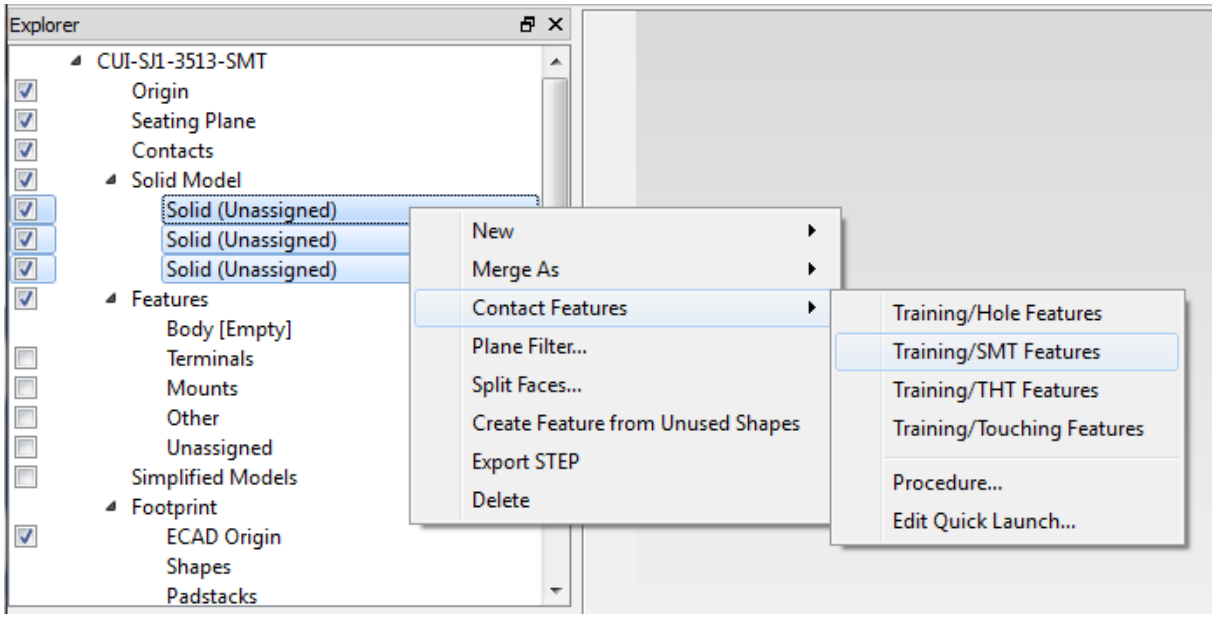
In order to begin the featurization process, import a 3D STEP model (File -> Import), and position the seating plane.

There are multiple methods for creating features from existing geometry in the solid model. In many cases, contact features may be directly generated from the solid model for terminal and mount features. When a single solid maps directly to a feature, such as the package body, the shape may be dragged directly from the solid model in the *'Explorer'* to the target feature. In certain cases, it may be necessary to create features from groups of faces in the solid model. A combination of feature assignment and creation methods may be employed. The internal structure of the geometric model will impact the decision.

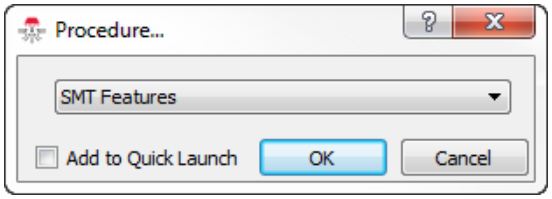
Generating Contact Features

The *'Contact Feature'* routines operate on the Solid Model and generate *'Unassigned Features'* with associated contact areas. These unassigned features can then be dragged into either the appropriate *'Terminals'* or *'Mounts'* groups in the *'Explorer'*. The contact feature routines can be applied to a single solid or multiple solids at a time. To apply a contact

feature routine, select one or more solids from the *'Solid Model'* of the package, and use the corresponding right-click *'Contacts Feature'* context action.



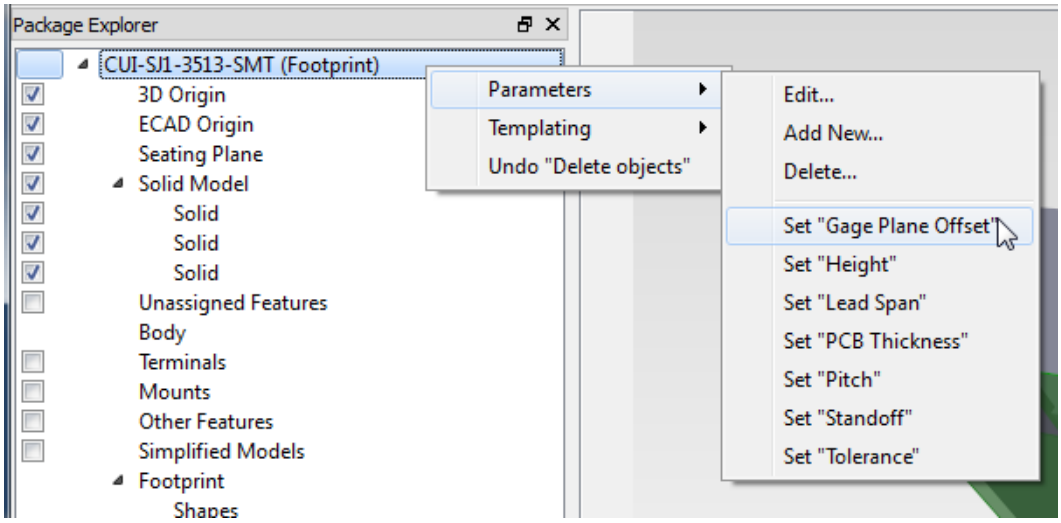
If the Hole, SMT, and THT options are not visible in the context menu, they can be found using the *'Contact Features -> Procedure...'* action and added to the drop down by selecting the *'Add to Quick Launch'* toggle. Alternatively, the Edit Quick Launch context action can be used.



SMT Features

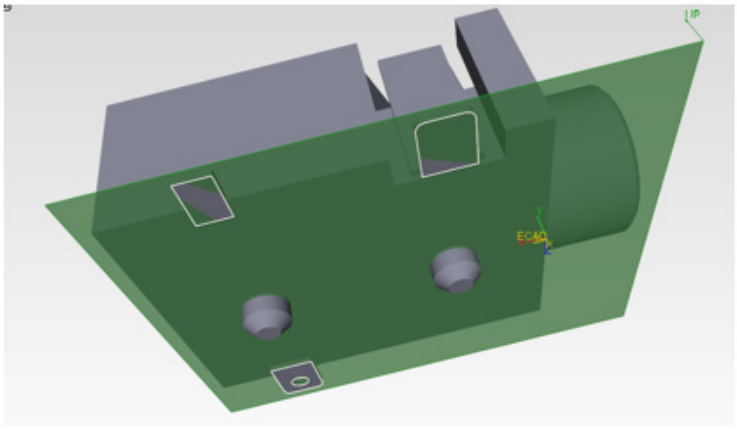
The *'SMT Features'* action will generate *'Unassigned Features'* within a specified distance of the seating plane, known as the Gage Plane Offset. If no value has been specified for the *'Gage Plane Offset'* a default value of 0.25 mm is assumed. While this value may be appropriate for certain common packages, there may be cases in which the default value of 0.25 mm results in multiple SMT features being treated as a single feature. In this case, the

user can manually set a smaller value for the Gage Plane Offset parameter on the package, or consider the Touching Features rule.



In the example of the CUI-SJ1-3513 connector, setting the Gage Plane Offset to a smaller value is necessary in order to separate the 3 SMT pins from the body. Setting the Gage Plane Offset to 0.001 mm, and running the *'SMT Features'* routine on all 3 solids results in the

creation of 3 unassigned features corresponding to the 3 SMT terminal contacts. These 3 features can be dragged into the *'Terminals'* node of the Package Explorer Tree:



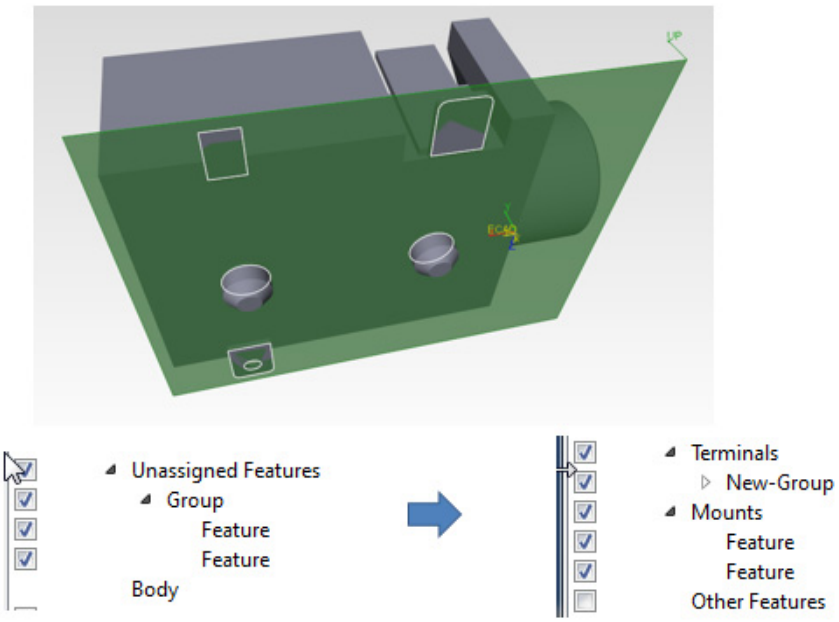
Touching Features

The *'Touching Features'* action will create features for geometry within a very small tolerance of the seating plane. This rule is suitable for flat bottomed features in contact with the seating plane. To include geometry within an effective distance above the seating plane, the SMT Features rule should be used with a suitable value for the *'Gage Plane Offset'*.

THT Features

The *'THT Features'* action will generate *'Unassigned Features'* based on a cross-section of the 3D geometry slightly below the seating plane. This function can be used to generate contact areas for THT features such as pins and mounting posts.

In the case of the CUI-SJ1-3513 connector, circular contact areas should be generated for the two mounting posts. These can be dragged from the *'Unassigned Features'* to the *'Mounts'* node of the Package Explorer Tree:



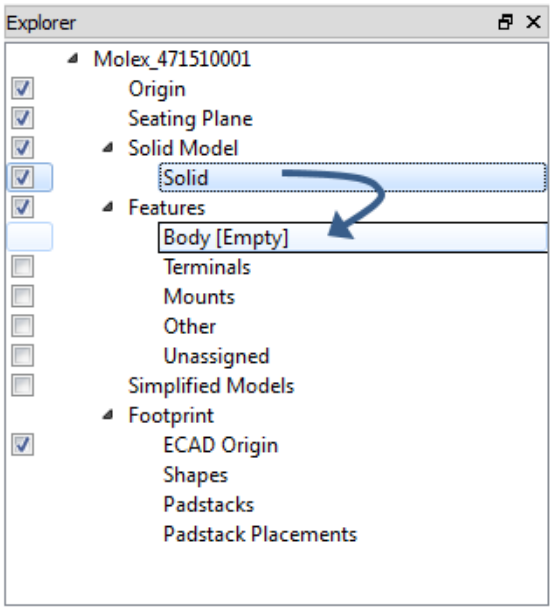
Hole Features

The *'Hole Features'* action will generate *'Unassigned Features'* based on a holes in the projected area of the solids above the seating plane. This function would typically be used to generate contact areas for mount features that will require a plated or non-plated THT padstack.

Drag and Drop Feature Assignment

The *'Explorer'* widget supports drag-and-drop between the solid model and package features. Single or multiple solids selected from the *'Solid Model'* can be dragged directly

into one of the feature types (including *'Unassigned'*). Existing features can also be dragged into a different feature node in order to change their type.

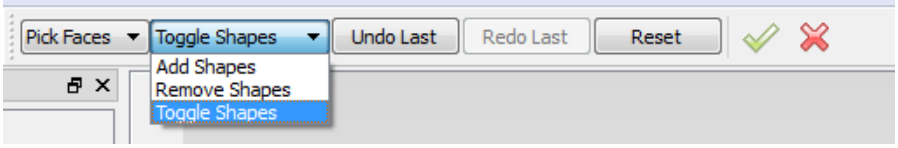


Generating Face-Based Features

Library Creator also supports the creation and representation of package features based on a collection of 3D faces from the solid model. These *'face-based'* features may be used in instances where a feature cannot be represented by a complete solid, and it is important to include the 3D geometry in the feature itself. In this case, a variety of tools are available to create features based on collections of faces.

Manual Feature Editing

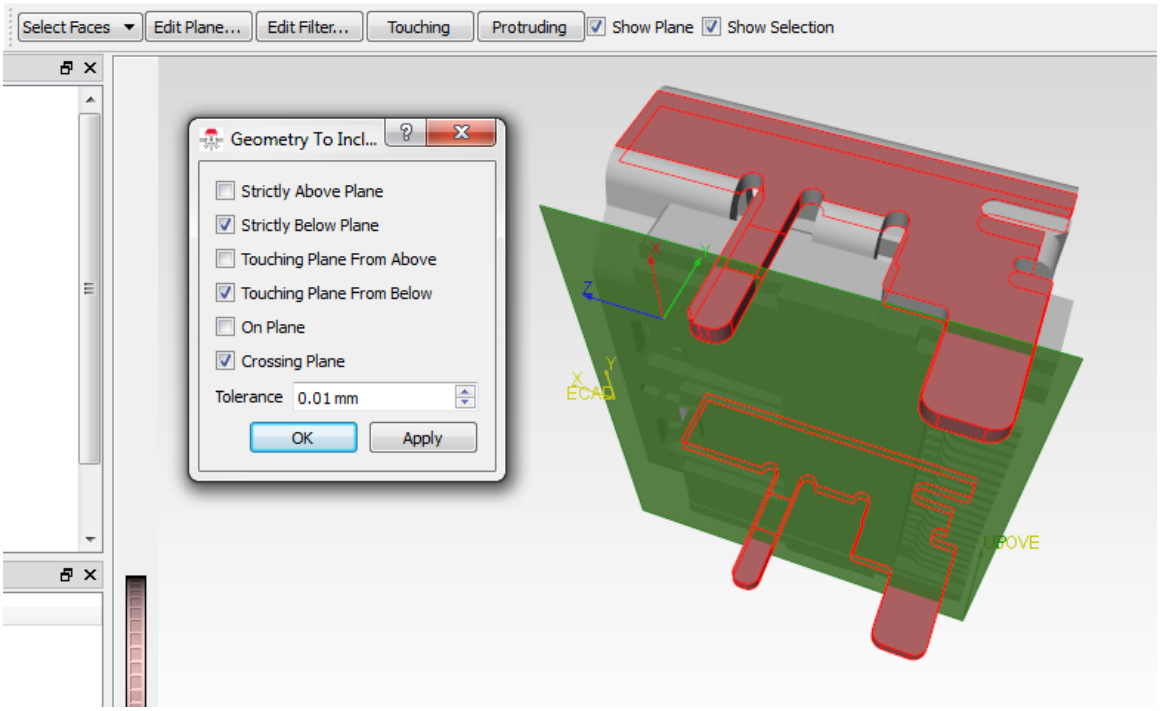
The 'Edit Feature..." right-click context action on a feature will bring up the manual feature editing toolbar shown below. Through the editing tool, the user can manually pick either solids or faces to add and/or remove from the active feature. Once the selections have been completed, the user can either accept or cancel the edit.



The Plane Filter

The 'plane filter' tool is launched through the 'Create Features -> Plane Filter' context action available on both solids and existing features. The plane filter tools is shown below. The plane filter creates groups of connected faces based on their relationship to a reference plane. There are two planes visible within the plane filter wizard - the seating plane (green) and the wizard filter plane (yellow). The plane filter initializes with these two reference planes coincident - often the seating plane is the intended reference plane.

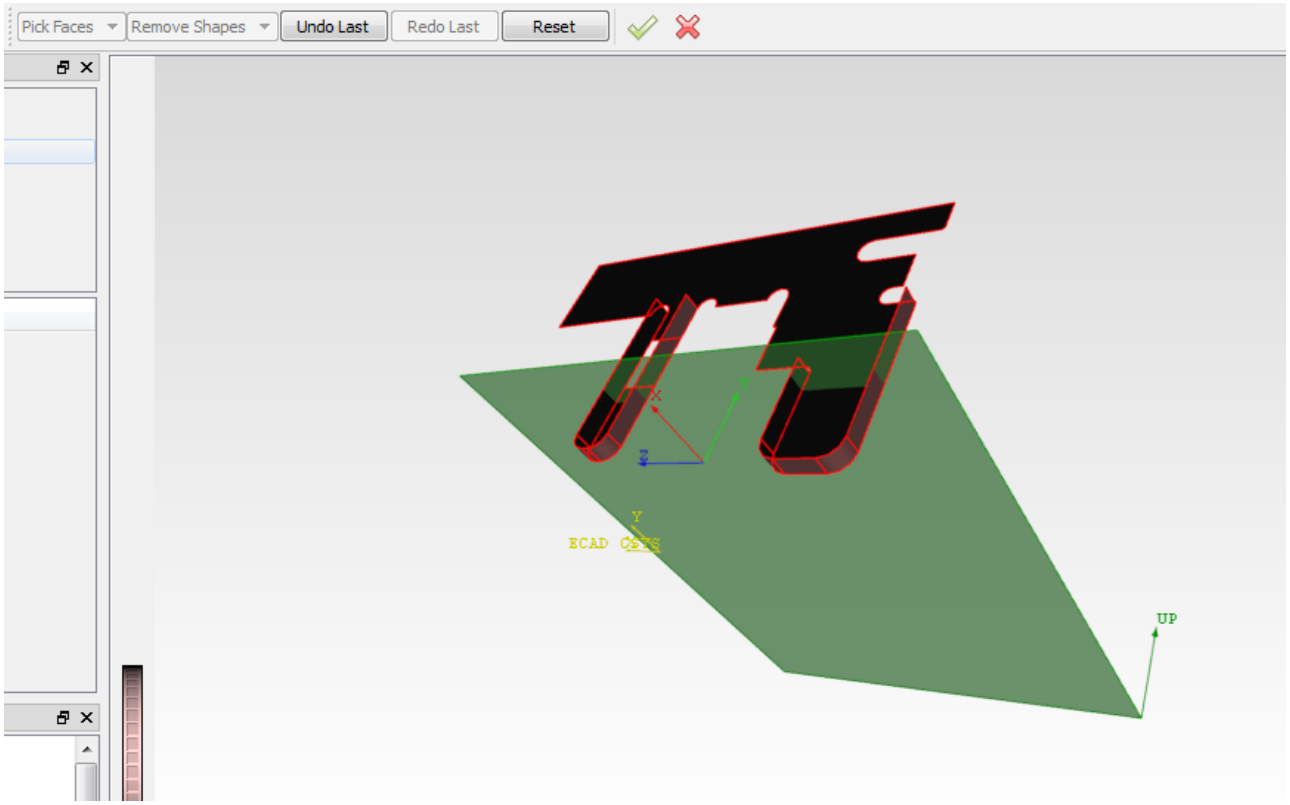
It is possible to edit the reference plane using the 'Edit Plane...' button. The most common actions in the plane filter are the 'Touching' and 'Protruding' actions triggered by the corresponding buttons. The 'Edit Filter' dialog (shown below) enables more fine-grained control over the faces to be included in the feature groups. Once the appropriate groups have been selected, the accept action  may be used to create the resulting 'Unassigned' features. The resulting features can be either dragged into a feature node or further refined through additional editing actions.



Feature Splitting

In certain cases, it is necessary to 'split' features created by the plane filter. In the example above and below, a single face spans discrete THT mount features. The 'Create Features -> Splitting' action can be used to manually remove the spanning face(s) resulting in the desired individual features. The Feature Split wizard may be seen below. Selection of individual faces

removes them from the target feature. The accept action will result in the generation of new 'Unassigned' features.



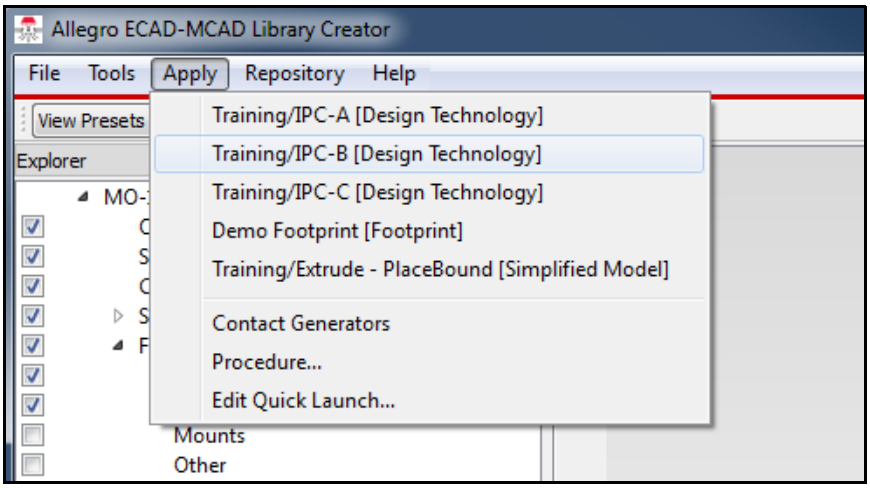
7

Applying Rules

Apply Menu

The *'Apply'* menus are used to apply rules to packages or elements of a package. There are different types of rule objects that may be applied, depending on the context. The *'Apply'* menu provides the ability to fire rules that have been added to the applicable *'Quick Launch'* list, or to choose individual rules from those available.

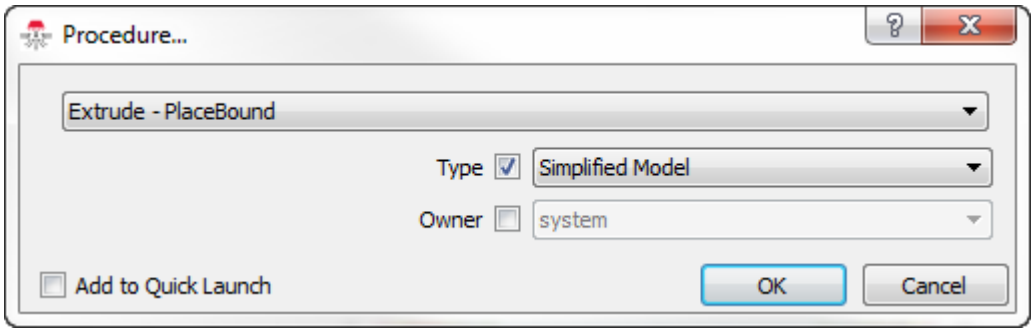
The top-level *'Apply'* application menu enables the selection and application to rules to be applied to the entire package. Context-specific *'Apply'* menus are available for solids and features through the Explorer.



Apply Procedure

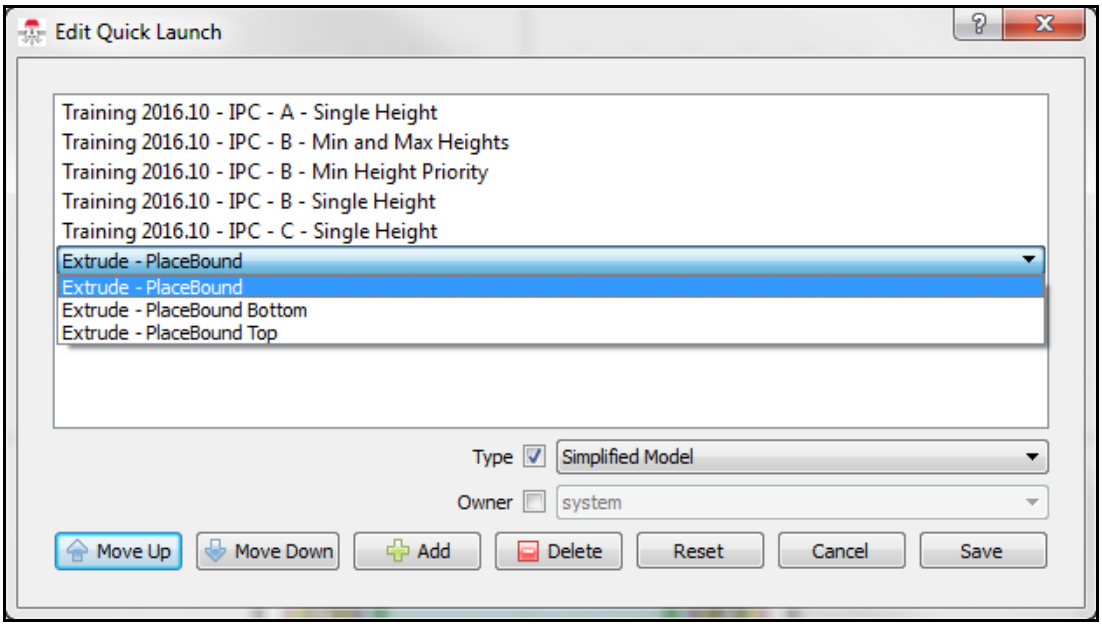
The *'Procedure...'* action on the *'Apply'* menu provides a drop-down for selecting an individual rule to apply. The *'Type'* and *'Owner'* filters can be enabled through the check box to restrict the available choices in the drop-down by rule object type and ownership. By

selecting the *'Add to Quick Launch'* check box, the currently applied rule will be added to the Quick Launch list.



Edit Quick Launch

The *'Quick Launch'* list is a set of rules available to fire directly from the *'Apply'* menu. Typically, it will be convenient to add commonly used rules to the *'Quick Launch'* list. Each *'Apply'* menu maintains an independent quick launch list. The *'Quick Launch'* list may be edited by using the *'Edit Quick Launch...'* action at the bottom of the *'Apply'* menu.



The *'Edit Quick Launch'* dialog allows rules to be added and removed from the list, and the list sequence to be configured through the applicable dialog buttons. Note that individual entries in the list are editable - a double-click will bring up a drop-down with the optional *'Type'* and *'Owner'* filters limiting the available selection. The *'Reset'* button can be used to revert to the default rule list for the particular menu.

Rule Objects

Contact Generator	A Contact generator generates 2D contact areas from 3D package features. *'Contact'* rules can be applied to individual features or groups of selected features through the Explorer *'Apply'* context menu.
Contact Feature Procedure	A Contact Feature procedure generates *'contact only'* features from elements in the solid model of the package. Typically, contact feature rules will be available for *'SMT'*, *'THT'* and *'Hole'* features. *'Contact Feature Procedure'* rules can be applied to individual solids or should of selected solids from the *'Solid Model'* of the package through the Explorer 'Contact Features context menu.
Design Technology Rule	A Design Technology is a collection of footprint generation rules. The intention of a design technology is to group together applicable footprint rules for a design or manufacturing technology, such as IPC-7351 or flex vs. rigid. A *'Design Technology'* can be applied to the package through the top-level *'Apply'* menu.
Feature Shape Rule	A Feature Shape is a footprint shape generator that fires on one or more package features. *'Feature Shapes'* may be applied either from the top-level *'Apply'* menu or through the context menu on features in the Explorer. When a *'Feature Shape'* is applied at the top-level, it will fire on all applicable package features, as determined by its configuration triggers.
Footprint Rule	A Footprint generator is a collection of padstack and shape rules configured to build either a complete or partial footprint for the given package. A *'Footprint'* rule may be tailored to a particular package family, or broadly applicable to all packages within a particular design technology. A *'Footprint'* rule can be applied to the package through the top-level *'Apply'* menu.
Package Procedure	A Package Procedure is a class of rules reserved for modifications to the package itself. *'Package Procedures'* are fired from the top-level *'Apply'* menu.
Simplified Model Generator	A Simplified Model is a 3D shape generator footprint shape generator that adds a *'Simplified Model'* to the package. *'Simplified Model'* rules are fired from the top-level *'Apply'* menu.
Shape Rule	A Shape is a footprint shape generator that fires on the package as a whole. *'Symbol'* rules are fired from the top-level *'Apply'* menu.

Rule Owner

Rules are provided either through a repository account or as standard *'system'* rules provided with the Library Creator application. Rule *'ownership'* refers to the provider of the rule. The *'Owner'* filter can be used to limit visible rules to those that are *'system'* provided or those that are local to the current repository account.

Apply Contact Generators

The *'Apply Contact Generators'* action is available from the top-level *'Apply'* menu. This action will fire all applicable *'Contact'* rules on all features on the package. Default contact generators exist for the standard terminal leadforms. Additional *'Contact'* rules may be configured in the account configuration to add additional contact generators and/or override the default contact generators.

8

Configuration

The configuration stores procedures and rules for generating contacts, footprints, and simplified models in Library Creator, as well as configuring model parameters, layers, and certain search and scoring details used in library synchronization. Library Creator users have access to system provided rules, as well as the ability to create custom rules and procedures through the configuration.

Much of the configuration is focused on rules for footprint generation. Library Creator provides powerful footprint generation capabilities for both standard packages and non-standard components such as connectors and mechanical hardware. A variety of pad, padstack, and shape generation routines enable precise generation of footprints, keepouts, assembly, and design symbols derived directly from the detailed geometric model. Both area-based and line-based footprint elements can be automatically generated from the package geometry, and pads and padstacks can be positioned precisely with respect to terminal contact areas. The footprint generation routines are extensively configurable by the end user to enable consistency and compliance with internal company practices, EMS requirements, and established standards such as IPC-7351.

Because the configuration is centralized and shared by all users of a given repository account, it is recommended that a single user maintain responsibility for the shared Library Creator configuration within their organization.

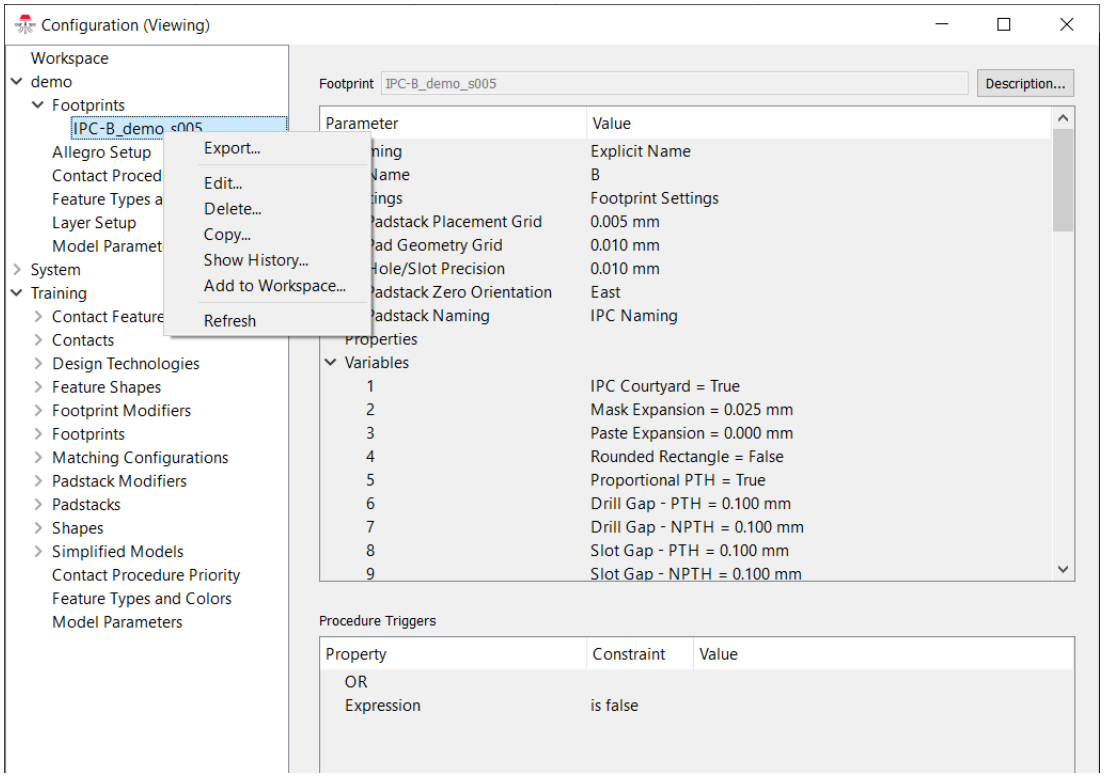
Registry

All configuration data is maintained within a centralized registry shared by all users of a given account in the Library Creator repository. Users have access to configuration objects created and provided through the common *'System'* and *'Training'* accounts, as well as the ability to create and edit configuration objects within their own account. Objects within the registry are organized in a tree structure. The top-level nodes correspond to the account that owns the object. Within each account, objects are organized by type. There are many configuration object types that can be created, edited, and managed by the user. Configuration objects may reference other configuration objects, including objects from other accounts. Configuration objects can be applied to a given package through the applicable Apply Menu.

The configuration registry is accessed through the Tools->Configuration... menu action. A view of the configuration registry may be seen below. The tree-structure in the left navigation panel can be used to navigate through the registry objects. A single object can be selected at a time. When selected, the details of the object are displayed in the right panel. In this case, a *'Padstack Modifier'* rule within the *'Demo'* account is displayed.

Users may create new objects of various types, edit existing objects, as well as create and modify copies of existing objects. The top-level object management actions are accessed through right-click context menu actions from the navigation pane as seen below.

The right panel is divided into two regions. The configuration details are displayed in the top, while the bottom contains the details of the procedure triggers that controls which individual features and/or packages the procedure will be applied to (fired on). Each of the configuration object types is described in additional detail below.



Editing Permission

In order to edit the configuration registry, it is necessary to first enable the Configuration:Edit permission. By default, this permission is disabled.

Workspace

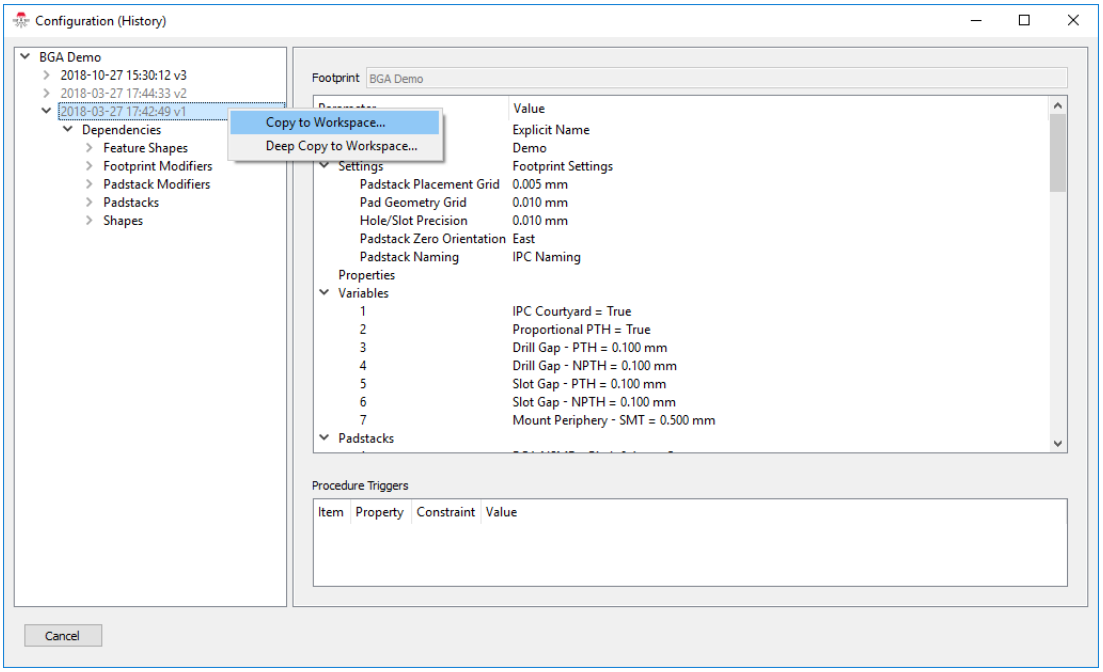
The creation and editing of configuration objects occurs in the Workspace. The workspace contains a local copy of all pending edits and changes prior to publishing changes to the repository. New configuration objects may be created directly in the Workspace through the right-click *'New'* contact menu action.

Existing configuration objects in the user account may be edited through the *'Edit'* context menu action. Editing an existing object brings a copy of that object into the workspace for editing. Existing objects in the user account or the global *'System'* and *'Training'* accounts may be copied into the workspace for editing. Objects in the workspace are stored locally and will not be available to other users. Changes may be tested locally in the workspace. Once testing is complete, the user may publish the changes in the workspace to the local repository account through the *'Publish Workspace...'* context action. In addition, it is possible to import and export configuration objects from the workspace to an external file using the *'Import...'* and *'Export...'* contact actions. The available context actions are described below.

History

It is possible to view the edit history of configuration objects as well as to copy past versions of objects to the workspace through the *'Show History...'* context action. The history view displays all repository commits of the given object, as well as the dependencies of each of the objects. Selecting a given version of a configuration object displays the object content in the right-hand pane. The user who committed the object version and the version notes are available through a tool-tip display. It is not possible to revert configuration objects to a prior

state. However, it is possible to create a copy of a past version of an object, as well as its dependencies, to the workspace (see available actions below).



Two actions are available on a previously committed version of an object through the history view as show above:

Copy to Workspace...	Create a copy of the given version of the object in the workspace.
Deep Copy to Workspace...	Create a copy of the given version of the object as well as all of it is dependencies in the workspace.

Actions

The following actions are available through the configuration object context menu:

Import...	Import configuration objects from a `.cfg` file into the workspace. The Import action is available from the Workspace context menu.

Export...	Export configuration objects to a `.cfg` file. It is possible to export configuration objects from the workspace as well as the user account. The export action may be applied to individual objects or multiple selections of objects within the workspace and user account, as well as to the top level workspace and user account nodes.
Add to Workspace...	Bring an object from the local user account into the workspace without performing edits. This action is only available to objects within the local user account. Objects in the global Training and System accounts can only be brought into the workspace by creating a copy.
Publish Workspace...	Commit all objects in the workspace to the user account on the repository.
Show History...	Display the version history of the configuration object.
Discard Workspace...	Discard all objects in the workspace.
New	Create a new configuration object of the specified type in the workspace and enter editing mode for the new object. Prior to the initial save, the name of the object can be changed.
Copy...	Create a copy of the currently selected object within the current user account and enter editing mode for the new object. Prior to the initial save, the name of the object can be changed.
Edit...	Enter editing mode for the currently selected object. Editing an object will bring both the object and any dependent objects into the workspace.
Delete...	Flag the currently selected object for deletion in the workspace. The objects will be persistently deleted when the changes in the workspace are published to the repository.
Discard Changes...	Remove this object from the workspace and discard any edits made within the workspace. Does not delete the object from the registry.

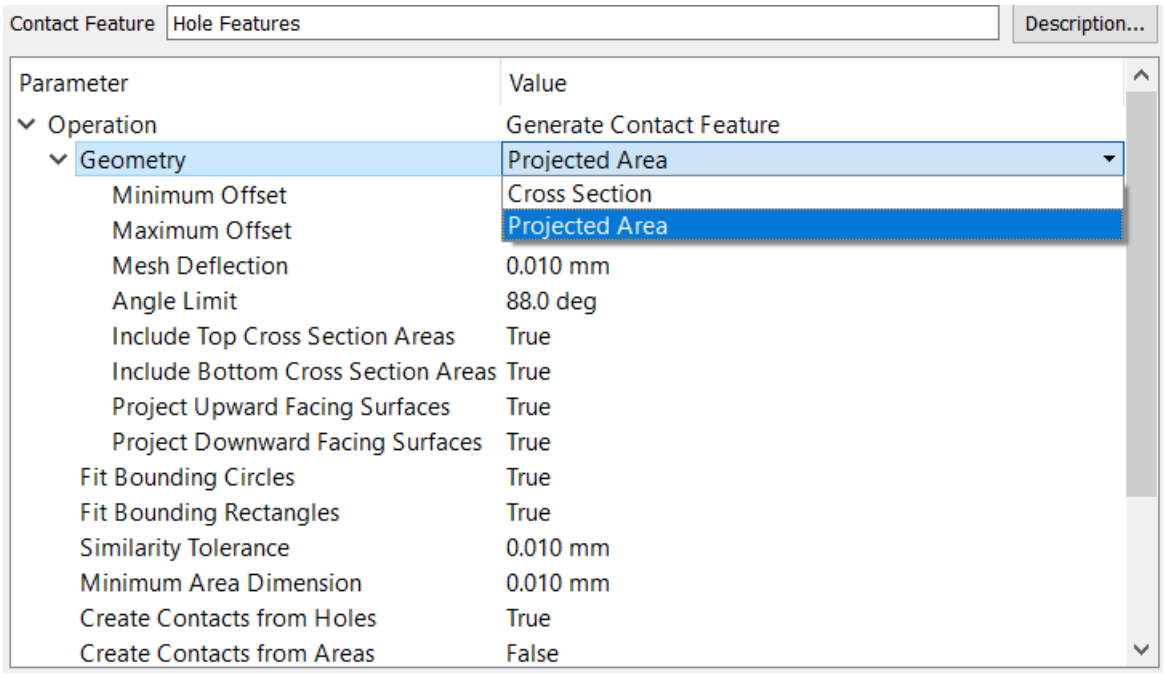
Editing Objects

Configuration objects cannot be edited while in registry viewing mode. Note: *'Configuration (Viewing)'* is displayed at the top of the registry window when in viewing mode, '*Configuration (Editing)*' is displayed when in editing mode. Editing mode is initiated by either creating a new object, copying an existing object, or editing an existing object.

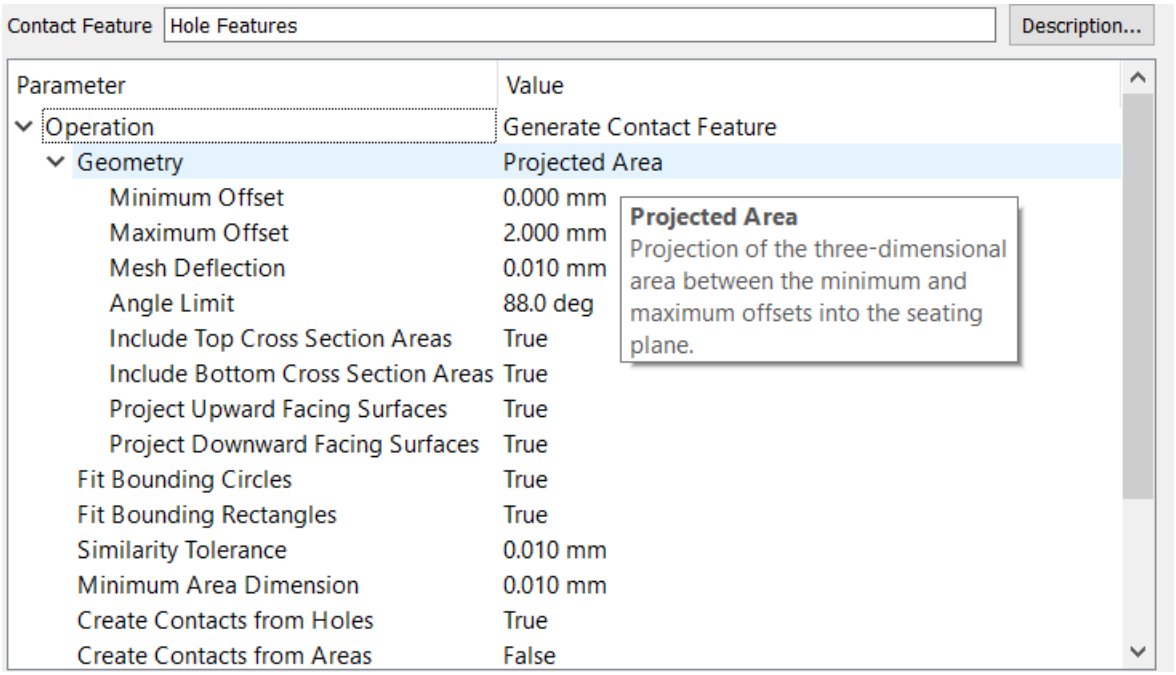
Populating a new object or editing an existing object is carried out through specifying options through drop-down menus (double-click to activate) and direct entry of parametric values within the applicable fields in the definition of the configuration object. As configuration

objects are selected required child attributes will appear within the parent node, in turn requiring population.

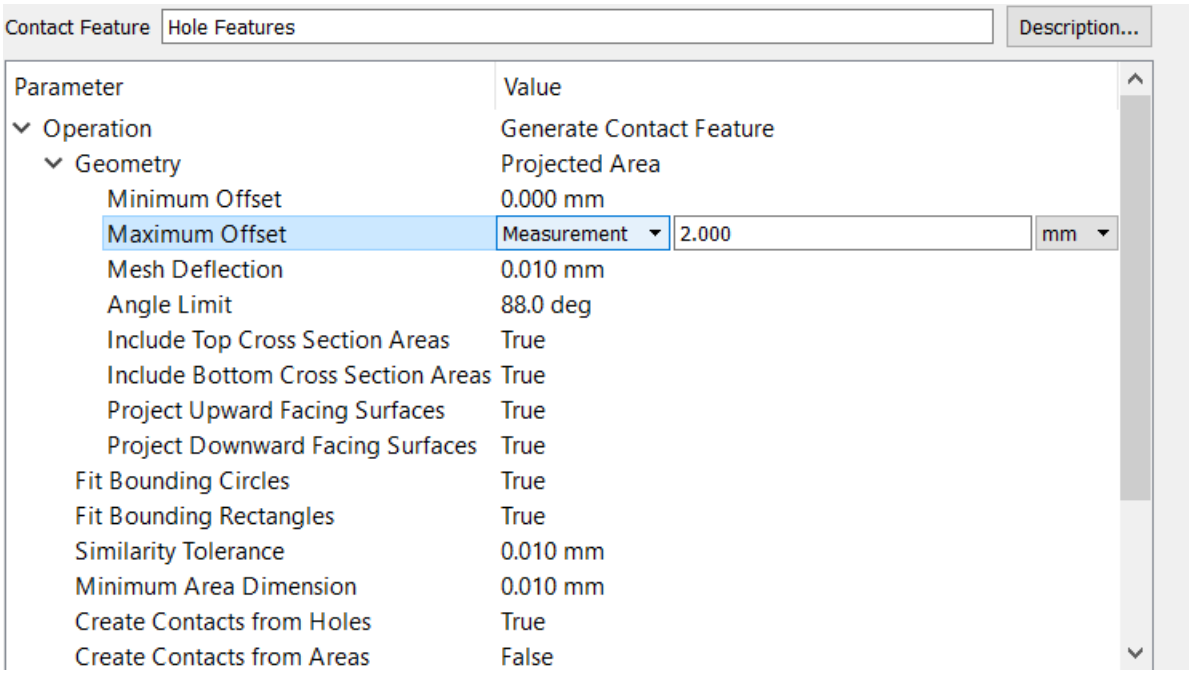
The example below represent an in-process definition of a contact feature generation object. The top-level operation 'Generate Contact Feature' requires selection of one of two options for the 'Geometry' procedure. Selection of the 'Projected Area' option, in turn, requires specification of the 'Minimum Offset' and 'Maximum Offset' attributes.



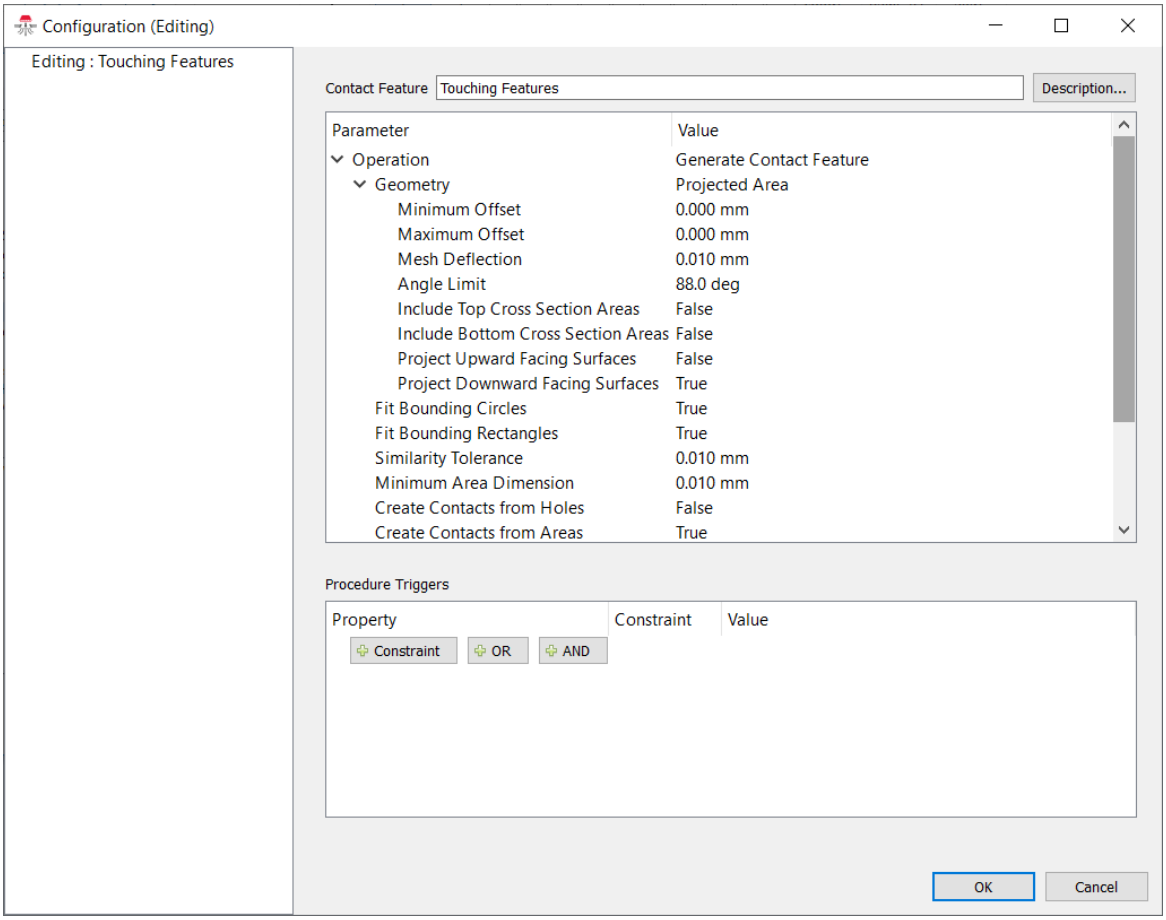
Additional detail regarding the individual procedures and input attributes is often provided in a 'tooltip' box when hovering over the field in question with the mouse, as shown below.



Values may be entered for required attributes by just clicking to activate the value editor.



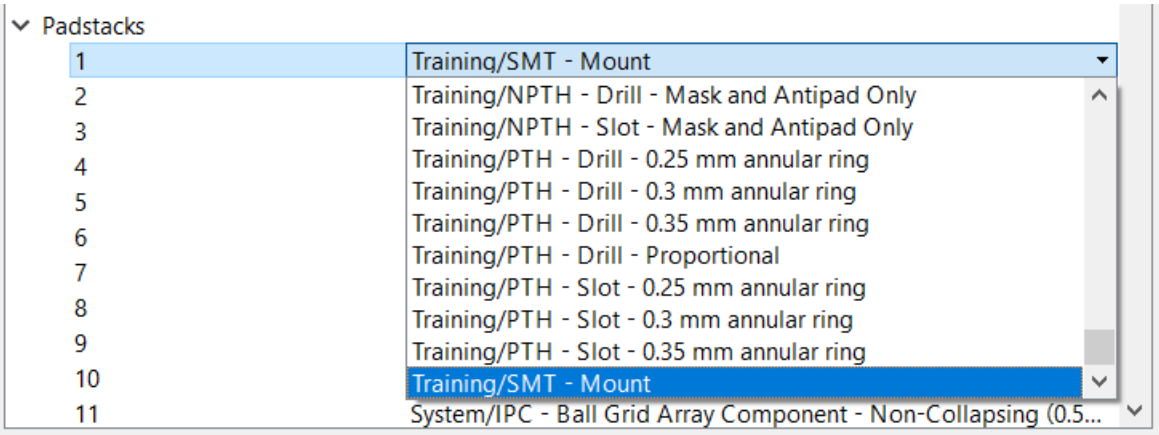
After required attributes have been specified, select 'OK' to save the edits to the workspace or 'Cancel' to discard the edits and exit editing mode.



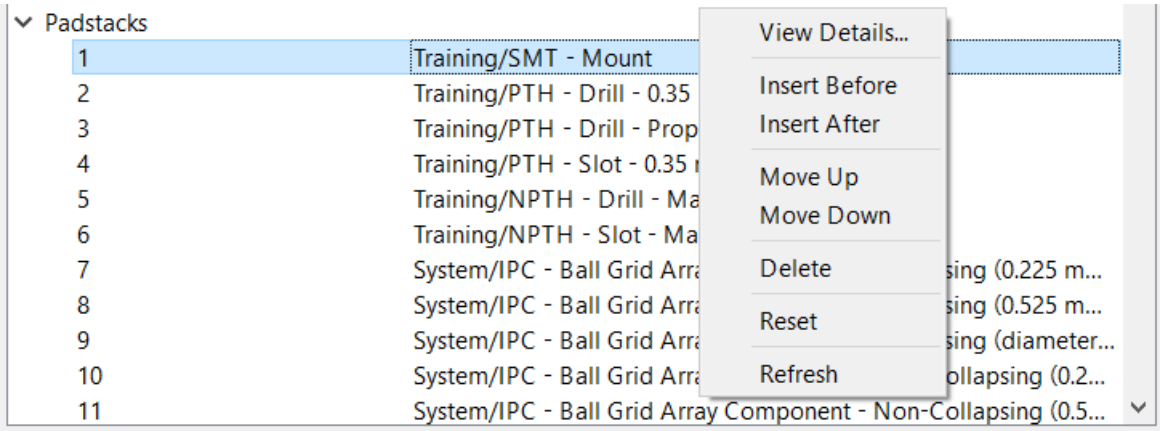
Lists

Lists are common elements of configuration objects, and are edited through a common series of actions. Lists are initially empty. In order to add an initial element to an empty list, use the *Add Item* button:

Once an initial element is visible, it is value may be populated by clicking in the value field to activate the editor for the list element. The editor will vary depending of the type of the list element. In some cases, it will be a drop-down of fixed choices as in the example below.



After a list contains an initial element, it is possible to add, remove, insert, and reorder the list through a series of context menu actions available on any list element.



Objects

There are a many configuration object types that can be created, edited, and managed by the user. Configuration objects may reference other configuration objects, including objects from other accounts. Certain configuration objects, such as design technologies and footprints can be applied to a given package through the applicable Apply Menu, while others, such as padstacks and footprint modifiers are only fired from within a top-level rule object. Below, the available configuration object types are summarized.

Contact

A 'Contact' rule creates two-dimensional contact areas from three-dimensional package features (body, terminals, mounts, and other). Individual contact generators can be configured with procedure triggers that determine which features they will fire on. Most common is to configure the trigger for contact generators based on terminal lead-form and/or the feature type attribute. Default contact generators have been configured for all terminal lead-forms. These default contact generators are visible within the configuration registry under the 'System' account and contain names such as '[Default] Gull-Wing'. In addition to the default contact generators, users may created, modify, and customize their own contact generators. User-created contact generators must be added to the ' contact procedure priority ' list before they will be used by Library Creator during contact generation. User-created contact generators will be executed in the sequence defined by the priority list, and all contact generation routines in the priority list has precedence over the default contact generators. This enables the user to completely override the default routines if desired. A variety of specialized contact generation procedures have been developed and may be configured by the end user. The available contact generation procedures are selected through the 'Geometry' attribute, listed below:

3D Basis + 2D Postprocess	Contact generated through the combination of either a projection or cross-section basis from the three-dimensional feature geometry and a post-processing step to generate the final two-dimensional contact shape. For complex 3D shapes, the cross section and projected area geometry options are recommended. Primarily used for fitting circles and rectangles to regular feature shapes. Also useful for projecting areas of flat (planar) surfaces parallel to the seating plane.
Ball Diameter	A contact circle equal to the full ball diameter. Diameter calculated based on cross-section taken at the reference gage plane. Terminal must be spherical and tangent to the seating plane for the diameter to be calculated accurately.
Bounds Below Gage Plane	A rectangular contact encompassing the bounds of the terminal below a reference gage plane.
Clipped Ball Diameter	A contact circle limited by the diameter at the intersection with the package body or the full ball diameter, whichever is smaller. Diameter calculated based on cross-section taken at the reference gage plane. Terminal must be spherical and tangent to the seating plane for the diameter to be calculated accurately.

Cross Section	A generalized cross-section implementation suitable for complex shapes. Cross section is computed at a specified offset to the seating plane. Use negative offset for values below the seating plane. Options to fit circles and bounding rectangles based on a similarity tolerance.
Gullwing Method	A rectangular contact spanning from the toe to the heel at the point of cross section with the reference gage plane. Suitable for regular terminal geometries such as those generated from a template. When the details of the 3D terminal shape are unknown, the projected area method is recommended.
Horizontal Cylinder Diameter	A contact rectangle based on the projected full diameter of the terminal. Diameter calculated based on cross-section taken at the reference gage plane. Terminal must be horizontal and tangent to the seating plane for the diameter to be calculated accurately.
No Shape	No shape will be generated.
Projected Area	A generalized area projection implementation suitable for complex shapes. Vertical extent of the area included in the projection is controlled by a minimum and maximum offset. These may be set to extreme values if the complete 3D area is to be included (no clipping). Options to fit circles and bounding rectangles based on a similarity tolerance.

Contact Feature

A 'Contact Feature' rule creates new features with two-dimensional contact areas directly from the solid-model. Most commonly, the contact feature generator is configured to create 'Unassigned' features to be manually assigned by the user. Procedure triggers are not relevant for contact feature rules, as the rule may only be applied to elements of the solid model. Several common configurations of the contact feature generators are provided in the 'Training' account as described here. These contact generators may be used directly or as a starting point for further customization. The contact areas are generated using either a 'Projected Area' or 'Cross Section' method through the 'Geometry' attribute as described below. A variety of geometric details may be configured within the rule, including the minimum and maximum offsets for the area projection, which elements of the geometry to preserve in the generated contacts, and a number of parameters pertaining to the accuracy of the generated projection.

Cross Section	A generalized cross-section implementation suitable for complex shapes. Cross section is computed at a specified offset to the seating plane. Use negative offset for values below the seating plane. Options to fit circles and bounding rectangles based on a similarity tolerance.
Projected Area	A generalized area projection implementation suitable for complex shapes. Vertical extent of the area included in the projection is controlled by a minimum and maximum offset. These may be set to extreme values if the complete 3D area is to be included (no clipping). Options to fit circles and bounding rectangles based on a similarity tolerance.

Design Technology

A 'Design Technology' is the top-level object for organizing and grouping footprint generation rules. A design technology must contain at least one footprint generation rule, and may optionally contain additional procedures that are applied to all packages. The intention of a design technology is to permit applicable footprint rules for a design or manufacturing technology, such as IPC-7351 density level A vs. B or flex vs. rigid. In the default 'Training' configuration, the design technologies contain only a single footprint rule that applies to all packages. In certain contexts, it may be preferable to create footprint rules that are tailored to certain package families. For example, a design technology might include a footprint rule that applies to BGA components, and another that applies to CHIP packages, etc. It is possible to directly apply footprint rules to individual packages through the 'Apply' menu. However, bulk export and the matching configuration (used for synchronization) require a design technology.

Dimension Style

A *Dimension Style* is a specifications of the settings to control the formatting and style of a dimension object. Instances of the *Dimension Style* object created in the configuration are be available through the drop-down in the *Dimension Style* dialog.

Drill List

A 'Drill List' is a list of specific drill diameters that are to be used when creating a drill element of a THT padstack. A configured drill list may be referenced from within a Drill element of a THT padstack rule through the 'Use Drill List' choice for the drill 'Diameter Calculation'. The use of a drill list is optional. Alternate methods for calculating the drill diameter are based on an explicit calculation in combination with the 'Minimum Diameter' and 'Diameter Step Interval' attributes. When using a drill list, the diameter will be taken to be the smallest element in the list that satisfies the specified gap.

Feature Procedure

A 'Feature Procedure' is a routine that fires on individual package features (body, mounts, terminals, other) outside the context of a footprint generation rule. The rule may be invoked from either the top-level 'Apply' menu or from the Apply' context menu action on individual features. There are currently two operations supported by the feature procedure object:

Add Radial Pin Contact Pattern	Add a radial pattern of terminals with contacts of specified diameter.
Decompose Area	Add a pattern of rectangular shapes on a specified layer to the footprint based on a target basis area.

Feature Shape

A 'Feature Shape' fires on individual package features (body, mounts, terminals, other) to creates shapes on a specified footprint layer. The 'Feature Shape' rule may be included in the list of 'Geometry Procedures' within the footprint rule or may be invoked directly from the 'Apply' menu on a package or individual feature. When included in a footprint rule or applied to the package itself, the rule will be applied to each feature in the package sequential and will fire on each feature satisfying the trigger constraints. The feature shape rule generates shapes on a target layer. A wide variety of shape algorithms are available through the 'Geometry' attribute:

Annulus - Absolute	Place an annulus (donut) shape of inner and outer radius at the location of the specified basis (contact, drill, pad).
Annulus - Basis Defined	Place an annulus (donut) shape for the specified basis (contact, drill, pad). The annulus will encompass the basis geometry with a specified air gap and ring width (difference between the outer and inner radius).
Bounding Circle - 2D Basis	Create a circle that bounds the specified 2D basis (contact, drill, pad). The bounding circle will be of minimum size to encompass the basis. An optional expansion may be specified.
Bounding Circle - 3D Basis	Create a circle that bounds the 3D shape of the feature. Either the total or sectioned 3D shape may be specified through the *'Basis'* attribute. The *'Height Max'* and/or *'Height Min'* of the 3D shape may optionally be included in the footprint shape.

Bounding Rectangle - 2D Basis	Create a rectangle that bounds the specified 2D basis (contact, drill, pad). The rectangle will be of minimum size to encompass the basis. An optional expansion may be specified.
Bounding Rectangle - 3D Basis	Create a rectangle that bounds the 3D shape of the feature. Either the total or sectioned 3D shape may be specified through the *'Basis'* attribute. The *'Height Max'* and/or *'Height Min'* of the 3D shape may optionally be included in the generated shapes.
Circle - Absolute	Place an circle shape of prescribed radius at the location of the specified basis (contact, drill, pad).
Clearance Offset	Enables the positioning of the specified *'Geometry'* element to be offset with respect to the *'Clearance Basis'*. The geometry element is placed based on the given feature basis, and then offset in the direction of the feature orientation (or specified heuristic).
Coplanar Face Projection	Project faces of the associated 3D feature shape into the seating plane. This projection applies only to flat (planar) faces that are parallel to the seating plane. Preserves arcs in the projected geometry when possible. The *'Projected Area'* geometry option should be used for generalized 3D area projection.
Edge Projection	Project edges of the associated 3D feature shape into the seating plane. Preserves arcs in the projected geometry when possible.
HLR Projection Hull [Deprecated]	Deprecated - the *'Projected Area'* geometry option should be used for generalized 3D area projection.
HLR Projection Linework	Discrete orthographic projection of visible edges of the associated 3D feature shape into the seating plane. Edges not visible from the projection direction are eliminated.
Keep Basis Geometry	Create a copy of the 2D basis geometry (contact, drill, pad) as a shape on the target layer.
Mesh Projection [Deprecated]	Deprecated - the *'Projected Area'* geometry option should be used for generalized 3D area projection.
Projected Area	Project the 3D feature shape into the seating plane. Supports sectioning of the 3D shape through the basis and sectioning attributes. The *'Height Max'* and/or *'Height Min'* of the 3D shape may optionally be included in the generated shapes.
Text	Create a text shape on the target layer. Supports a variety of text types (RefDes, Literal, etc.) as well as positioning options.

Uniform Offset	Create a copy of the 2D basis geometry (contact, drill, pad) as a shape on the target layer with an optional expansion.

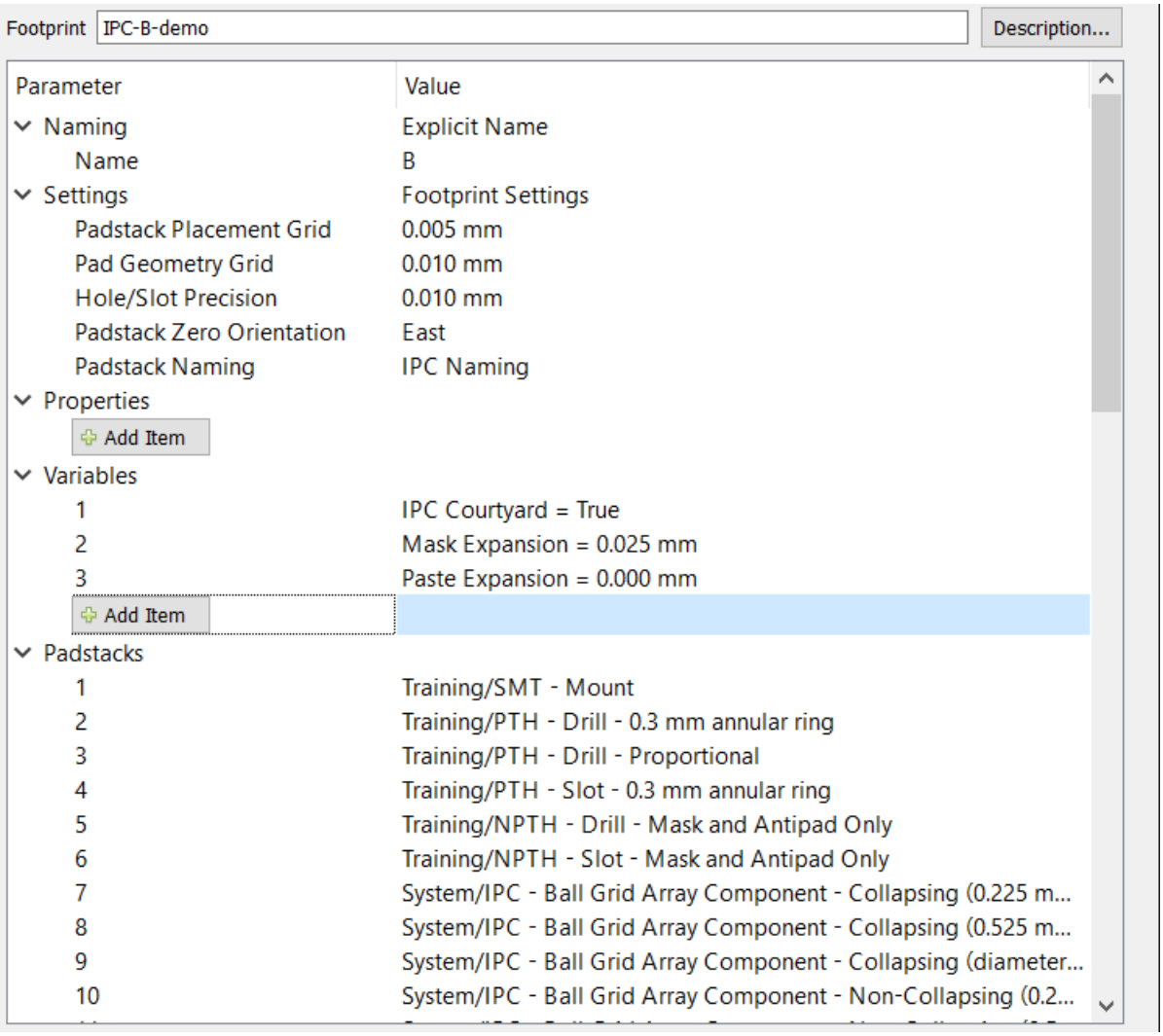
Footprint

A *'Footprint'* generator is collection of padstack and shape generation and modification rules that collectively can be used to generate elements of a footprint or a complete footprint for a given package. The primary components of a footprint generator are described below.

You can be easily add rules to any of the sections using the *Add Item* button available at the bottom of the respective section and select a rule from the drop-down list.

Settings

The *'Settings'* section of the footprint generator controls a number of global settings for the footprint. These include geometric grid settings and padstack orientation and naming.



Properties

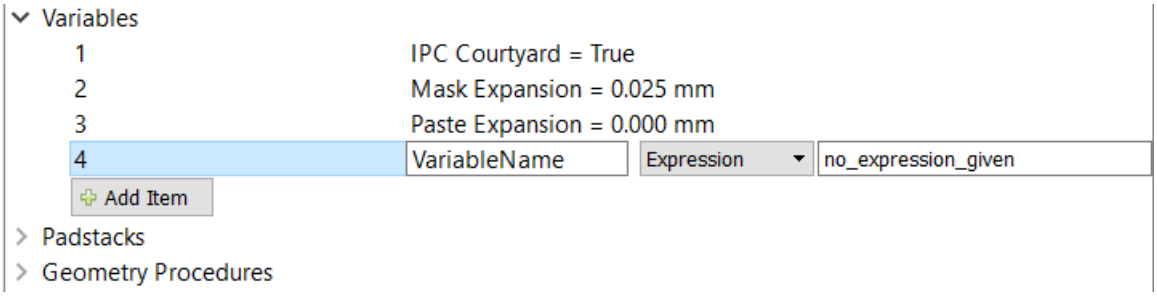
'Properties' are optional attributes that can be populated in the generated footprint. The primary reason for populating properties on the footprint is for propagation to the target Allegro footprint.

You can easily add properties using the *Add Item* button selecting a property from the drop-down list.

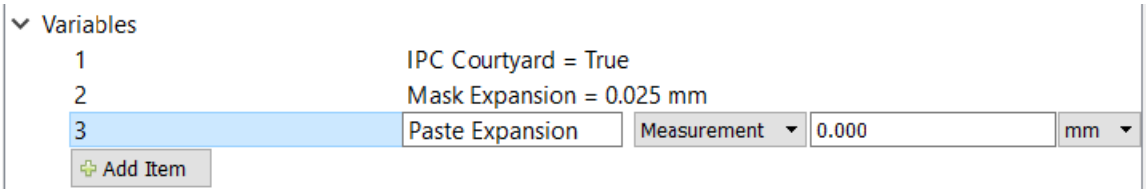
Variables

Variables are an optional mechanism to define values or configuration settings for use during the footprint generation process. Footprint variables enable top-level settings and parameters to be defined once and accessed by all padstack and shape routines for that the footprint is dependent on. In the training rules, a number of global attributes, such as the mask and paste expansion values, are defined as footprint variables. The values of these variables may then be accessed through an attribute value expression with the applicable padstack and padstack modifier rules.

Footprint variables are added using the *Add Item* button. A new variable is defined by first typing the variable name and then selecting the variable type through the drop-down in the value column.



After selecting the value for the type, you can add or select the value of the selected type from the rightmost drop-down list. The name, type, or value of an existing variable can be edited by just clicking the variable to make it editable.



Padstacks

Padstack objects can be added to the footprint using the *Add Item* button. During footprint generation, Library Creator iterates over each feature of the package. Library Creator then attempts to apply each padstack against the feature, starting with the first padstack in the list. Once a padstack fires on a given feature, execution procedures to the next feature in the package. It is important to note that if multiple padstack rules are applicable for a given feature, the sequence that they are included in the list will impact footprint generation. You can also define variables in padstack objects.

Geometry Procedures

After applicable padstack rules are fired for each feature in the package, Library Creator will execute each of the elements of the geometry procedures list in turn. The geometry procedures list is edited using the list editing actions. When adding to the list, all applicable rule objects will be displayed in the drop-down. The following object types may be included in the geometry procedures list:

- Padstack Modifier
- Footprint Modifier
- Feature Shape
- Shape

Because the geometry procedures operate on the current state of the footprint as it is being constructed, the sequence of elements in geometry procedures list is important. For example, if a footprint shape is intended to include the pad areas, it is necessary that all padstack modifier and trimming rules be completed before the shape is generated.

Footprint Modifier

A *'Footprint Modifier'* rule modifies some aspect of the footprint under construction. Footprint modifiers may modify both pads and footprint shapes. There are several **trimming** procedures available for both pads and shapes, as well as a couple of shape modification procedures available through the top-level *'Procedure'* attribute:

Connect Holes	Adds vertical and/or horizontal channels to existing shapes with holes in order to create *'simple'* polygons consisting of only a single boundary and no interior (hole) boundaries. Commonly used to create valid shapes for ECAD layers that do not permit holes. Operates on all shapes on the target layer.
Extract Corners	Modify all existing path geometries on the target layer to extract transition regions and corners. May be applied to a continuous boundary to result in corner segments suitable for silkscreen or other visual layer marks.
Pad to Pad Trimming	Trim pads on the target layer to achieve a specified clearance. Suitable for side, toe, and heel trimming, as well as 45 deg. corner trimming of quad-style pad layouts.
Shape Trimmer	Trim footprint shapes on the target layer to achieve a specified clearance. Suitable for paste mask shapes that are not included in the padstack geometry.

Under Body Pad Trimming	Trim pads on the target layer to achieve a specified clearance from the package body. Suitable primarily for template-based packages where the 3D body shape is distinct from the 3D terminals.

Matching Configuration

The *'Matching Configuration'* object enables configuration of the search and scoring characteristics used in library synchronization. A matching configuration is required for both the auto search and scoring functions. When scoring a footprint against a package, Library Creator, generates a target footprint based on the *'Design Technology to use as offsets'* in the matching configuration and then compares the target footprint with the current footprint in order to generate a score. Default matching configurations are available in the *'Training'* account based on the provided IPC-7351 design technologies.

Package Procedure

A *'Package Procedure'* is an operation that modifies the package itself (as opposed to the footprint). Currently, there is only a single operation available through this configuration object. Package procedures may be applied to the package through the top-level *'Apply'* menu and added to the *'Quick Launch'* list.

Rename Package	Renames the active package based on the IPC-7351 package naming convention. Note that it is not possible to rename a package that is already persistent in the repository. In order to rename template-based packages, a new instance should be created. Other packages can be renamed only after creating a copy (File -> Copy in Session).

Padstack

The *'Padstack'* object creates new padstacks for features that satisfy the trigger conditions. If an existing transient padstack is present in the footprint, it will be replaced by a newly generated padstack. However, if there is an existing persistent, locked, or external padstack

associated with the feature, the padstack generator will not fire. Padstacks may include optional properties. The following types are available through the top-level padstack attribute:

External Padstack	Allows rule based placement of an existing library padstack for the target feature. The library padstack must be identified by name, and may be placed based on either the center of the contact area, or the connection point. Specifying the **Native Orientation** enables the placed padstack to be oriented correctly with respect to the contact orientation of the feature. The native orientation should indicate the *'toe'* direction of the library padstack at 0 deg. rotation.
Simple SMT Padstack	Includes Top Signal, Top Solder Mask and Top Solder Paste pad definitions. Unwanted pads can be removed by setting to *'None'*.
Full SMT Padstack	Includes Top Signal, Top Solder Mask and Top Solder Paste pad definitions and an optional Antipad and Thermal Relief pad. Unwanted pads can be removed by setting to *'None'*.
Generic SMT Padstack	An arbitrary list of single, mask, and paste pads with optional antipad and thermal relief. The Generic SMT Padstack is necessary for creating bottom side SMT padstacks.
Simple THT Padstack	Includes a Drill / Slot definition. Includes optional Top/Bottom Signal, Top/Bottom Solder Mask and Top/Bottom Solder Paste pad definitions. Assumes that top and bottom pads are equivalent and does not support internal pads. Unwanted pads can be removed by setting to *'None'*.
Full THT Padstack	Includes a Drill / Slot definition as well as optional top, bottom, and internal pads , including antipads and thermals. Top and bottom pads may be equivalent or independent. Unwanted pads can be removed by setting to *'None'*.
Generic THT Padstack	Includes a Drill / Slot definition as well as an arbitrary list of single, mask, and paste pads with optional antipad and thermal relief.

Pads

A wide variety of pad generators are available for use in both padstacks and padstack modifiers. In the majority of cases, the pad is constructed and positioned based on an underlying basis geometry. The basis may be either the contact shape, another pad shape, or the drill/slot shape. In an SMT padstack, for example, the top signal pad may often be constructed from the contact shape, while the solder mask pad is constructed from the top signal pad. The contact shape is the basis for the top signal pad, while the top signal pad is the basis for the solder mask pad. The primary pad generation methods are outlined below.

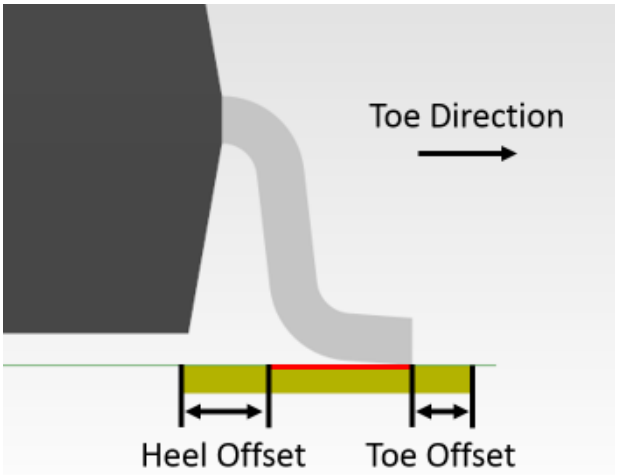
Fixed Shape Absolute and Basis Defined Pads

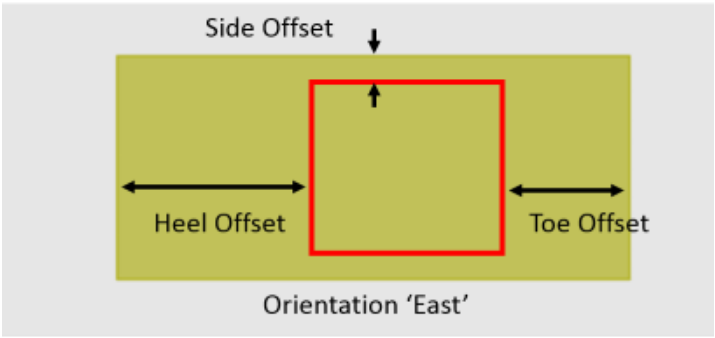
The most basic pad generators are limited to a fixed pad shape and expansion value. These generators have names such as *'Round Rectangle - Absolute'* and *'Round Rectangle - Basis Defined'*. These generators are unable to position pads offset with respect to the contact shape. In many cases, alternate approaches, such as the toe/heel/side generators will provide additional flexibility.

Toe/Heel/Side Pad

Many SMT pad generation rules rely on the concept of toe, heel, and side offsets between the physical terminal contact and the pad extents. These offsets are intended to enable formation of a suitable solder fillet during the reflow process. The contact orientation is used to distinguish between the toe and heel of the terminal. If a contact orientation is not set explicitly on the feature, many of the pad generators provide a **Toe Heuristic** that the rule will use to deduce the orientation. When using a heuristic, the **Toe Reversed** boolean should be

set to True for terminals where the toe is oriented opposite to that deduced by the heuristic. The toe/heel/side generator supports the common pad shapes for standard terminals.





Toe/Heel Pad with Explicit Width

The *'Toe/Heel Pad width Explicit Width'* offers similar options to the Toe/Heel/Side pad, however the pad width is fixed to a specified value, independent of the contact width. This pad generator is often used in combination with triggers or expressions based on the terminal pitch.

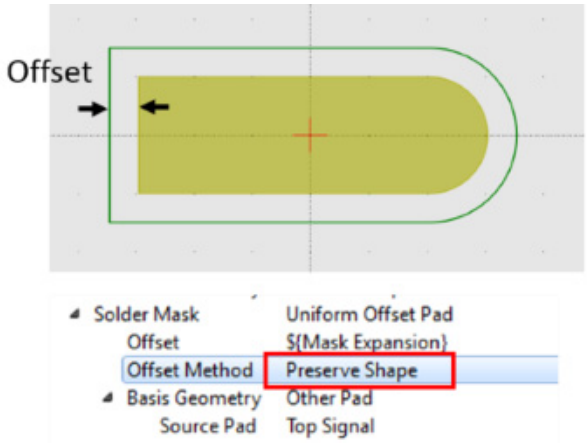
IPC Toe/Heel/Side Pad

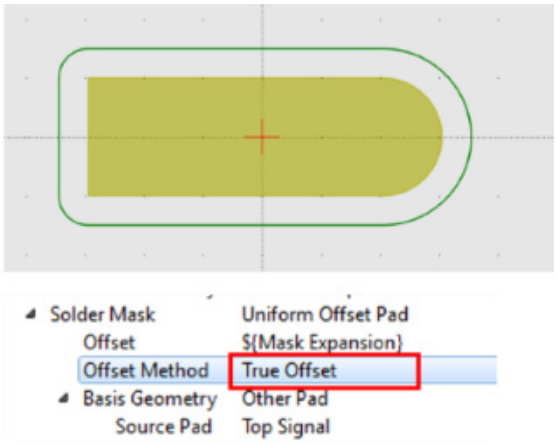
The *'IPC Toe/Heel/Side Pad'* is used by the majority of IPC-7351 SMT pad generators in the System account. This generator incorporates the IPC prescribed calculations for incorporating tolerances in the offsets. Tolerances on the system-defined Contact|Length, Contact|Width, and Lead Span model parameters are used in the pad calculations based on the methodology detailed in IPC-7351. The *'P'* and *'F'* attributes are the placement and fabrication tolerances, while *'jToe'*, *'jHeel'*, and *'jSide'* are the toe, heel, and side fillet

targets. Default values for the lead span, contact width, and contact length tolerances can be provided through the *'cL'*, *'cW'*, and *'cT'* attributes if desired.

Uniform Offset Pad

A *'Uniform Offset Pad'* is based on a uniform expansion (contraction if negative) of the basis shape. Uniform offset pads are commonly employed in generating mask and paste pads using the top signal pad as a basis. The uniform offset pad may also be used in generating copper from a contact area for certain non-directional terminals. The uniform offset pad has an option to *'Preserve Shape'* when creating the offset pad. This option will ensure that a standard shape symbol, such as the *'D'* or bullet shape is maintained, as illustrated below.

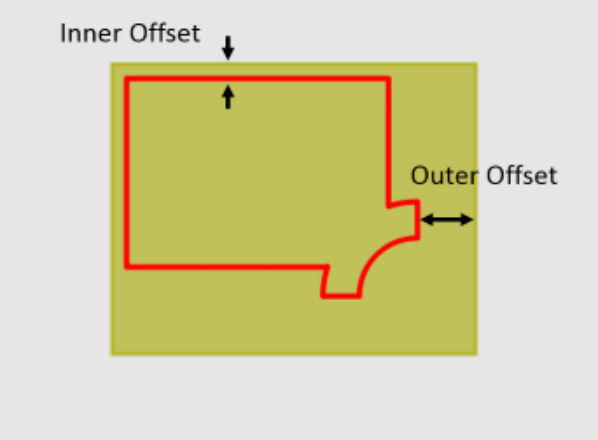




Corner Offset Pad

The *'Corner Offset Pad'* addresses the specialized case of corner terminations in which an inner offset is to be applied to the two interior edges, while an outer offset is applied to the

two outward facing edges. This generator relies on a heuristic to determine the corner orientation.

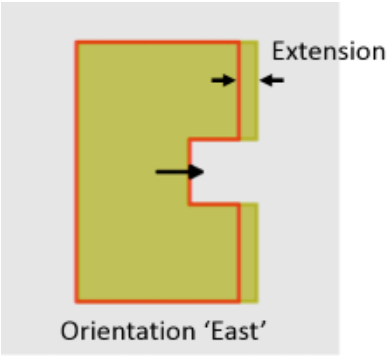


Corner Toe Heel Side Pad

The 'Corner Toe Heel Side Pad' generator incorporates concepts from both the 'Corner Offset Pad' and the 'Toe/Heel/Side Pad. This rule relies on both the corner heuristic as well as the contact orientation in order to determine the offsets to be applied in the toe, heel, and inner and outer side directions.

Edge Extension Pad

An 'Edge Extension Pad' is intended primarily for use with no-lead style terminations that are aligned with the body edges. Only a single extension offset is supported. This pad generator could be combined with a uniform offset pad modification for additional flexibility.



Drills and Slots

The THT padstack types have a single 'Drill' attribute that is used to specify the drill or slot for the padstack. Drills and slots may be either plated or unplated, and contain an optional tolerance. The slot type is either 'Oval*' or '*Rectangular*'. Both both drills and slots, there are both absolute and basis defined options.*

Absolute Drill

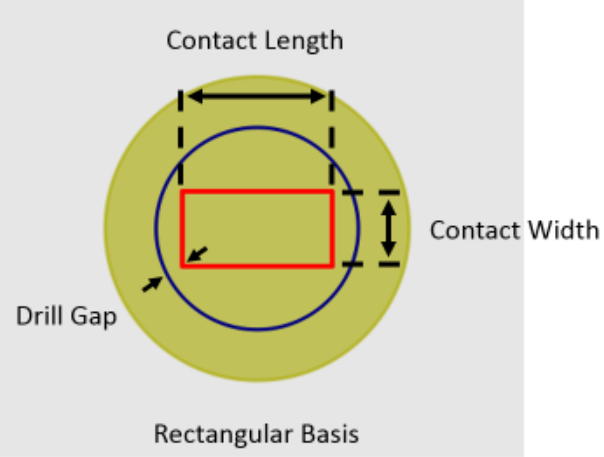
In an absolute drill, the diameter is specified as either an explicit value or through an expression. Custom diameter calculations can be achieved through the use of an expression in combination with model parameters and footprint variables. The drill may be placed at the contact center, connect point, or center of another pad.

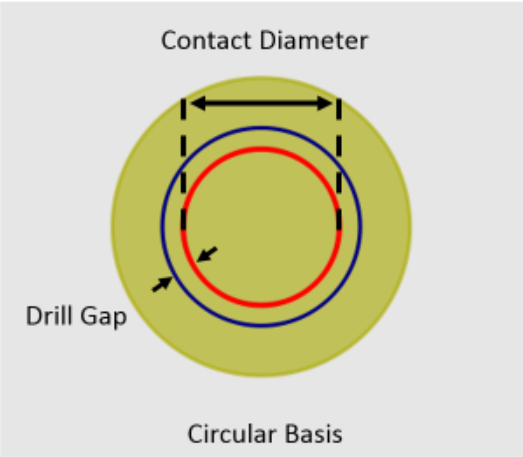
Absolute Slot

In an absolute slot, the length and width are specified as either explicit values or through an expressions. Custom calculations can be achieved through the use of an expression in combination with model parameters and footprint variables. In the majority of cases, slots are oriented parallel to the contact orientation through the 'Use Feature Orientation' value for the slot orientation. The slot may be placed at the contact center, connect point, or center of another pad.

Drill From Basis

The 'Drill From Basis' enables the drill diameter to be determined based on the combination of a basis shape (typically the contact shape) and a required drill gap. Several options are available for the diameter calculation. The 'Include Contact Tolerances' will automatically incorporate tolerances from the Contact|Length, Contact|Width, and Contact|Diameter model parameters. Depending on which parameters are populated on the model, either a rectangular or circular basis shape will be used for the diameter calculation, as illustrated below. In the case of contact shapes that are not circular and rectangular or rectangular, the diameter calculation will be conservative. In the event that the contact parameters are inconsistent with the contact geometry, a warning will be generated, and the values from the model parameters will be used. The drill will be placed at the center of the basis shape.





Slot From Basis

The 'Slot from Basis' enables the slot length and width to be determined based on the combination of a basis shape (typically the contact shape) and a specified gap. Options are provided for either a uniform gap, or independent length and width gaps. Both the slot width and length calculations have the option to include contact tolerances. When contact tolerances are included, either a circular or rectangular basis will be used for the length and width calculations as described above.

Padstack Modifiers

Padstack modifiers may optionally be included in the geometry procedures section of a footprint generation rule. Padstack modifiers enable rule-based modifications to be applied to padstacks that were either existing in the footprint or have been generated by the padstack generations in the current footprint generation context. Padstack modifiers will fire on transient padstacks, and unlocked pads of persistent padstacks. Padstack modifiers may add, remove, or modify existing pads. The padstack modifier contains a list of modifications. The following modification operations are supported:

Create/Modify Pad	Apply one of the available pad generations to either create a new pad or replace an existing pad. The target pad type and layer is specified through the pad attribute. When replacing an existing pad, it is possible to apply a modification to the existing pad. For example, the top signal pad could be replaced with a new pad using itself as the basis. In cases where the pad modification does not use itself as the basis, there is no distinction between creating a new pad or modifying an existing pad.
Remove Pad	Removes the pad on the specified layer and type, if it exists.

Padstack Naming

The 'Padstack Naming' configuration object supports a limited set of user-configurable options to the automated padstack naming implementation. In addition to the ability to specify a unit system and decimal precision value, a number of variations on the naming implementation are supported. After creating an instance of the padstack naming object, it will be available through the 'Padstack Naming*' attribute within the* footprint settings.

Shape

A 'Shape' rule generates new footprint shapes on the specified layer based on existing elements of the footprint and/or package. The triggers for the shape rule are package-level - when included in the list of 'Geometry Procedures' within a footprint generator, the shape rule will fire once if the trigger constraints are satisfied. A shape rule may also be invoked directly from the 'Apply' menu on a package in session. A wide variety of shape algorithms are available through the 'Geometry' attribute. Because a shape generator always creates new shapes rather than modifying existing shapes, it may sometime be useful to use temporary

or scratch layers as either the input or output of a shape generator to support multiple step shape processing requirements.

Axis-Aligned Approximation	Generates a simplified 'boxy' shape that bounds the elements included in the 'approximation basis'. A rectangular *'axis-aligned'* bounding box is generated for each of the basis elements, and the rectangular elements are unioned together to create the axis-aligned approximation. Optionally, a set of elements may be included in the *'subtraction basis'* to be subtracted from the generated approximation shape. The *'axis-aligned approximation'* may be useful when it is desired to maintain simply shapes for certain footprint elements such as placebounds or keepouts. See also the *'Axis-Aligned Merge'* option within the *'Union Shape'* geometry attribute.
Bounding Circle	Generates a single bounding circle that encloses all elements in the specified basis.
Bounding Rectangle	Generates a single bounding rectangle that encloses all elements in the specified basis.
Copy Shapes from Layer	Creates a copy of all existing footprint shapes in the specified *'From Layer'* on the target shape layer.
Corner Marking	Generates corner marks based on rectangular bounds of the 3d package or body extents. For generating corner marks for complex package outlines, the *'Extract Corners'* footprint modifier is recommended in combination with a path outline generated by the *'Projected Area'* feature shape.
Fill Between Pads	Creates rectangular shapes to fill between (and optionally over) neighboring pads. Fill shapes will be created when the distance between neighbors is less than the specified *'Threshold'* distance. Options are provided to expand the fill shapes in both the side and end directions (relative to the pad orientation). Commonly used to create RKOs between pads.
Gang Clear	Creates rectangular bounding gang clear regions around neighboring pads. See also the *'Fill Between Pads'* option described above when additional configuration flexibility is needed.
HLR Projection Hull	Deprecated - the *'Projected Area'* feature shape generator should be used for generalized 3D area projection.
HLR Projection Linework	Discrete orthographic projection of visible edges of the 3D package or body shape into the seating plane. Edges not visible from the projection direction are eliminated.

Resolve Overlaps	Creates new shapes on the target layer that cover the same area as the input layers while eliminating overlapping regions. Useful in creating multiple height placebound or area approximations from the raw shape output generated by the *'Projected Area'* feature shape generator.
Subtract Shapes	Create new shapes on the target layer based on subtracting the specified *'Subtracting Elements'* from the shapes on the *'Basis Layer'*.
Text	Create a text shape on the target layer. Supports literal text as well as text types such as refdes, part number, etc. Text may be positioned with either an *'Absolute'* or *'Area Based Placement'* option. The *'Area Based Placement'* enables X and Y alignment relative to the bounds of the specified basis area.
Union Shapes	Create new shapes on the target layer based on a union operation applied to the input elements. Input elements may be detailed pad and footprint shapes as well as bound elements. The *'Discrete Union'* method is recommended for complex geometry outlines. The *'Explicit Union'* will maintain arcs in the output shapes. The *'Axis-Aligned Merge'* is useful for creating axis-aligned approximations from input elements not supported by the *'Axis Aligned Approximation'* shape generator.

Simplified Model

A *'Simplified Model'* generator creates simplified models from the existing package and/or footprint. The name of the simplified model may be assigned by the generator. Several geometry options are available:

Axis-Aligned 3D Approximation	Generates a simplified 3D *'boxy'* shape that includes the specified 3D features. The expansion and height values will typically be driven by expressions. An option is provided to merge neighboring features.
Bounding Box - Explicit Height	Generates a single 3D bounding box that included the full extent of the package geometry. The expansion and height values will typically be driven by expressions.
Compound	Generates a simplified model as a compound representation of the 3D shape generators specified in the list of *'Elements'*. Enables extrusion of multiple shape layers, or combinations of shape layers and alternate 3D bounding approximations to be included in the generated simplified model.

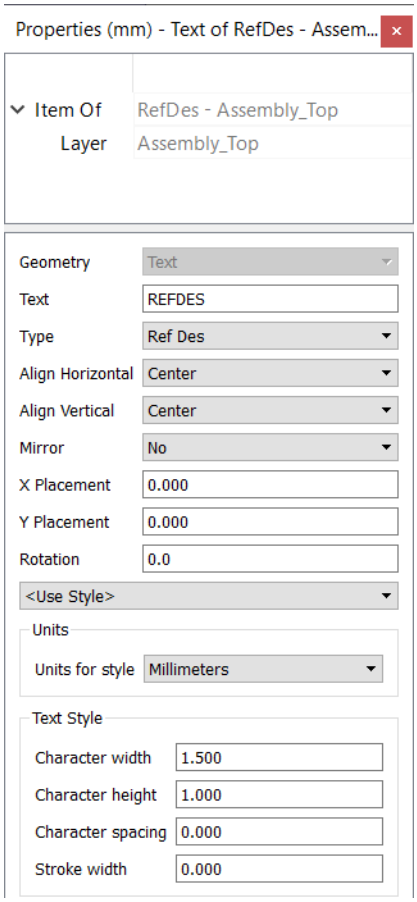
Shape Extrusion	Extrudes the target layer as a 3D simplified model. The *'Offset'* attribute enables a horizontal clearance to be incorporated into the extrusion. The *'From Shape'* options for the min and max height attributes enable 3D visualization of shapes containing height data. To interpret the height values for bottom side shapes, the *'Bottom Side Symbol'* attribute should be set to true. The *'Offset From Shape'* height options may be used to include an additional vertical clearance or to compensate for the *'PCB Thickness'* for bottom side footprint shapes.
Union	Generates a simplified model through a boolean union of the 3D shape generators specified in the list of *'Elements'*. Enables extrusion of multiple shape layers, or combinations of shape layers and alternate 3D bounding approximations to be included in the generated simplified model. The *'Compound'* option is generally recommended for complex shape.

Text Style

Text Style is a specification of text dimensions for use in text generation procedures. When instances of the text style object exist in the configuration, they will be available through the *'Text Style'* drop-down for objects that require text specification. Use of the text style is optional but recommended to facilitate rule maintenance by having a single point of definition for common text attributes.

You can change the text styles of characters, such as width, height, spacing, stroke width from the *Text Style* section in the *Properties* window. The *Units* section allows you to set the unit

in which the text styles are controlled. The available units are *Centimeters*, *Millimeters*, *Microns*, *Inches*, and *Mils*.

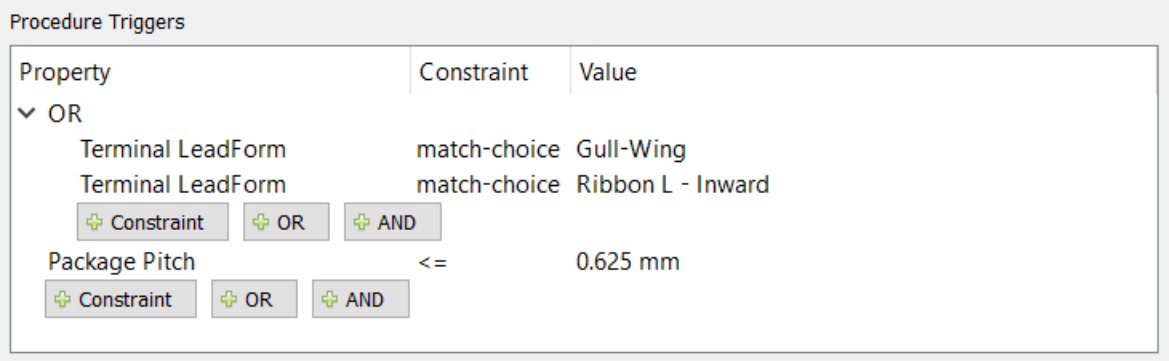


You can also apply the text styles created using the *Configuration* dialog from the *Use Style* drop-down list.

Procedure Trigger

The majority of the configuration object types include the option to specify triggers that limit the package features and/or packages that the procedure will be applied. to. If no triggers are specified, the procedure will be applied to all objects on which it is called. For example, if a particular padstack generation routine is included in a footprint generator, the routine will be applied to all terminals, mount, and generic features in the footprint. Typically, this is not the desired behavior. Triggers can be established to limit the application of the padstack generator to a variety of feature and package attributes such as the terminal lead form, mount type, or package pitch.

A procedure trigger consists of one or more individual constraints. Each constraint applies to a particular property and contains a specified value. Logical combinations of constraints can be created within *'AND'* and *'OR'* constraint groups, which can also be nested. All top-level constraints must be independently satisfied in order for the procedure to fire. In the example below, the configured procedure will trigger when the package pitch is less than or equal to 0.625 mm and the terminal lead form is either *'Gull-Wing'* or *'Ribbon L - Outward'*.



New constraints and constraint groups can be created using the *Constraint* button within the procedure trigger pane. Add a nested constraint or logical group, or insert a new constraint can also be done using the OR and AND buttons. You can delete a constraint, nested constraint or a logical group from the right-click menu.

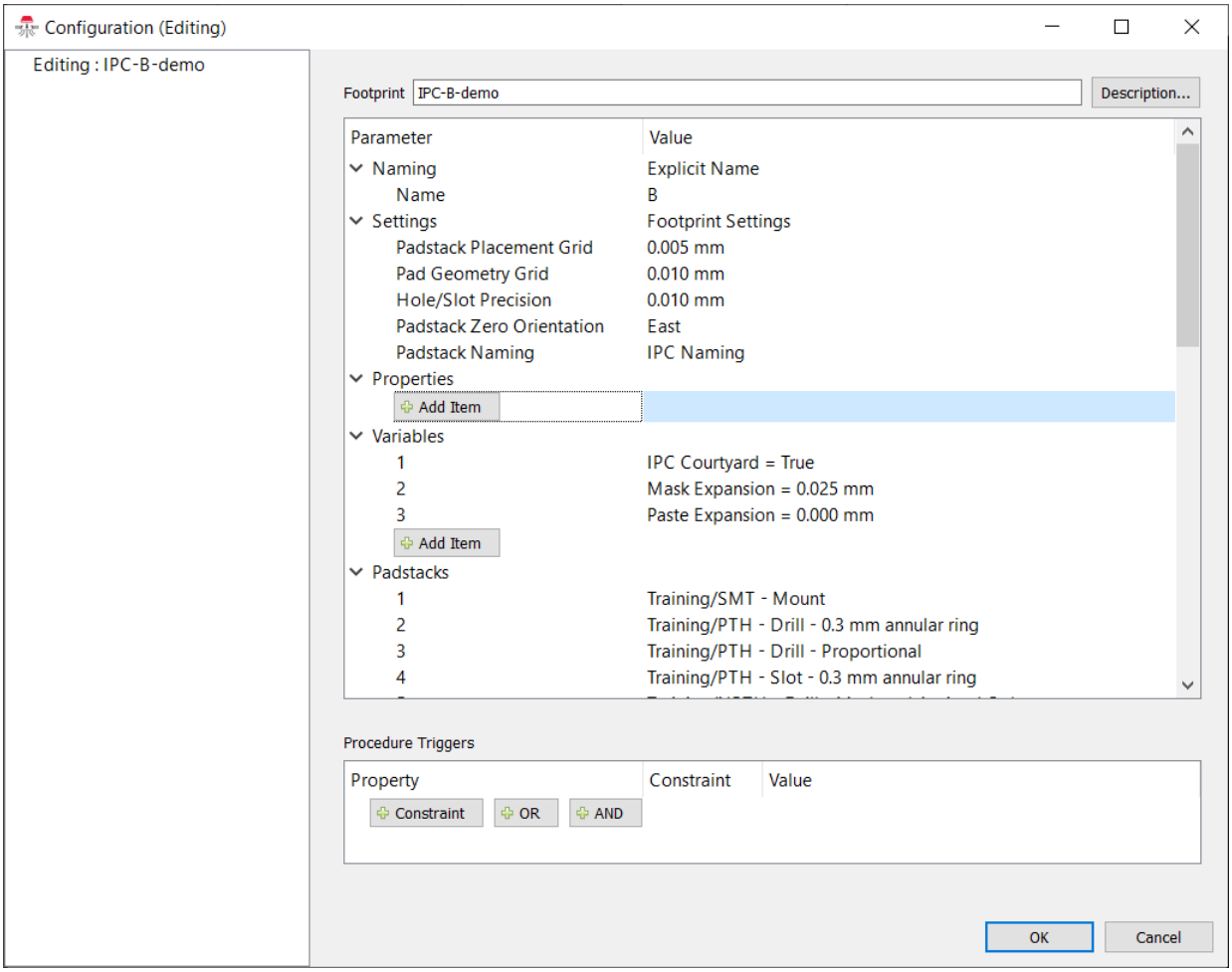
Trigger Constraint

The available constraint properties within the procedure trigger depend on the type of configuration object. For example, the trigger properties within a Padstack object will not be the same as those available within a Footprint object. The specific properties available can be seen and selected within the drop-down list available through a triple-click selection on the *'Property'* field of a constraint entry. The available constraint operations are dependent on the type of the constraint property. For example, the constraint operations for Package.Height (a dimensional attribute) will be different than the operations available for Terminal.LeadForm which is an enumerated string property. The available constraint operators can be seen and selected within the drop-down list available through a triple-click selection on the *'Constraint'* field of an individual constraint entry. Finally, a value must be entered in the *'Value'* field of the constraint. For enumerated strings, the *'match-choice'* constraint operator combined with drop-down selection of the associated value is preferred, in order to avoid inadvertent misspellings of the target strings.

Dependents

Configuration objects often reference other configuration objects. For example, a footprint contains lists of padstack objects and shape generator objects. Objects that are referenced by another object are referred to as *'dependent'* objects or *'dependents'* for the referencing object. For example, if a particular padstack object is included in a particular footprint object, the padstack is a dependency of the footprint. When editing or deleting objects, it is important to consider the impact on dependent objects.

For this reason, the list of *'Dependents'* is displayed when editing a configuration object and prior to deleting an object. During these actions, a list of dependent objects (if any) will be displayed in the left pane of the Configuration window as shown in the image below. If it possible to view and edit the details of the dependent objects by double-clicking on the object name in the dependents list. In this example, the *'IPC-B-demo'* object is being edited. After reviewing the dependencies, the user has the option to either cancel or proceed with the edit or delete action via the *'Cancel'* and *'OK'* buttons at the bottom of the right-hand pane.



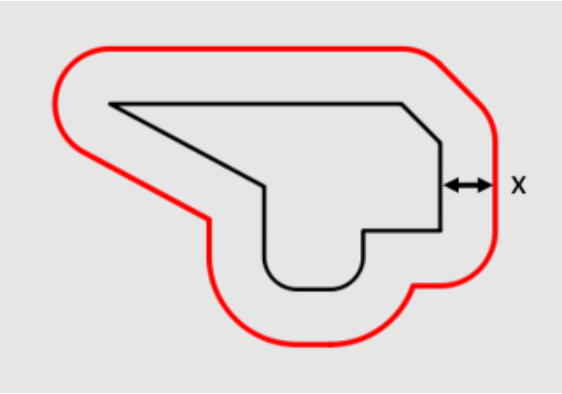
When an object with dependents is saved, a new version of all of the dependent objects is created as well. When an object with dependents is deleted, a new version of the dependent objects will be created as well. In most cases, an object with dependencies should not be deleted unless the dependent objects are updated to remove their dependence on the objects being deleted.

Shape Expansion

Many of the shape and feature shape geometry procedures provide an option to expand the generated 2D shape. Several expansion procedures are available as described below. Note that not all expansion options may be available for a given geometry procedure.

Explicit

An *'Explicit'* expansion is an analytic offset of all of the linear and arc geometry in the underlying shape. Arcs will be preserved in an explicit expansion, and their radius will increase (or decrease) by the value of the expansion (*'x'* in the image below). Sharp corners will result in circular arc segments during an outward expansion. The image below illustrates the result of the explicit expansion (red) applied to a portion of a polygon (block) consisting only of linear edges.

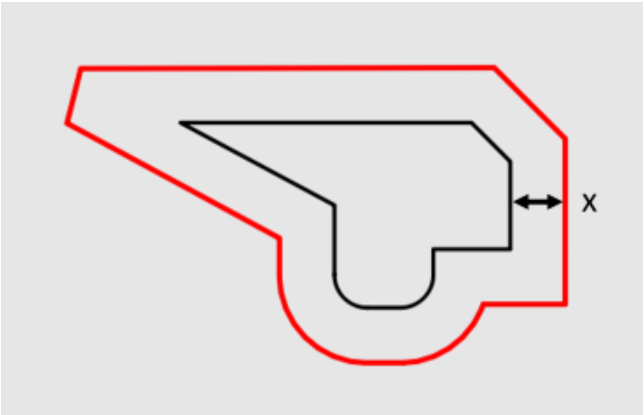


Discrete Buffer

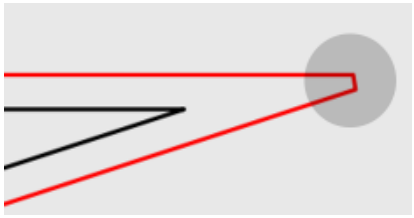
A Discrete buffer expansion creates a new offset shape consisting solely of linear path segments.

Mitre Join

The *'Mitre'* join option will preserve sharp corners in the offset geometry.

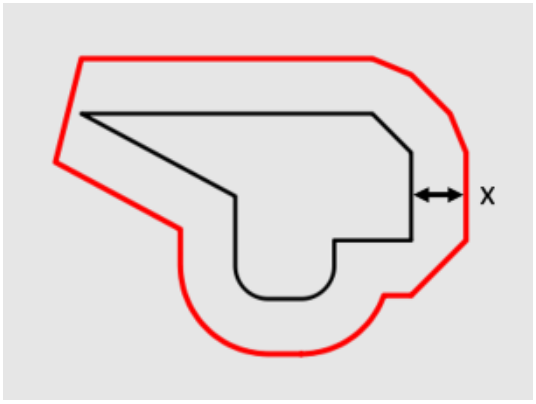


When using the *'Mitre'* join option, the **Mitre Ratio Limit** limits the ratio of the distance from the corner to the end of the mitred offset corner. Corners with a ratio exceeding this limit will be beveled, as illustrated below.



Bevel Join

Segments will be joined with a bevel (at 45 degrees) to the underlying segment intersection.

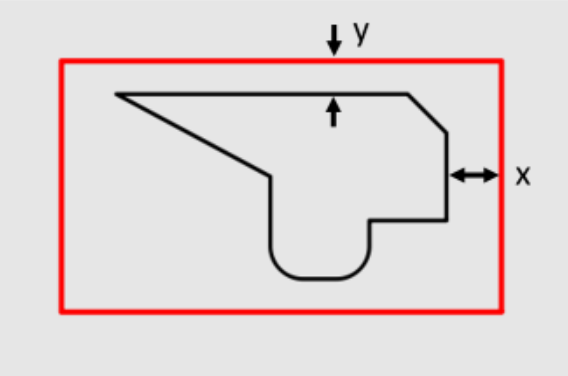


Round Join

Segments will be joined with a discrete approximation to a circular arc.

Rectangular Bounds

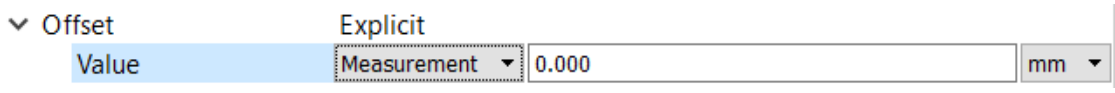
A rectangular bounds expansion will create a rectangular shape that bounds the horizontal and vertical extents of the input shape. The *'X Expansion'* and *'Y Expansion'* values are not required to be the same. The image below shows the result (in red) of a rectangular bounds expansion being applied to a polygonal shape with rounded corners. The expansion is applied to each side of the shape so that the overall increase in the horizontal extent will be twice the *'x'* expansion value, while the vertical extent will increase by twice the *'y'* expansion value.



Expressions

Expressions can be used in place of explicit values for certain object attributes and trigger constraints. An expression can reference one or more model properties and footprint variables and perform calculations and logical operations in order to generate a result of the required type. Most commonly, expressions are used within the configuration objects in place of measurement and boolean (True/False) values.

An editor for an value attribute is initially activated in *Measurement* mode. When in measurement, the value may be specified only as a constant value of the expected type. To switch to expression mode, select *Expression* from the drop-down list.



When in expression mode, the value of the attribute must be an expression that returns a value of the appropriate type. To switch back to the constant value entry mode, select *Measurement* from the drop-down list.

Syntax

The syntax of a variable expression is that of the Python programming language. The standard Python operations and majority of the built-in functions of the Python interpreter are available for use in expressions. In addition, syntax extensions have been provided to enable referencing of package model parameters and footprint variables as well as explicit measurement values in either Inch or Millimeter units.

Context

Configuration object expressions are evaluated during rule execution within the context of either a particular package feature or the package itself. In addition, an expression may be evaluated within the context of a particular footprint rule. The values of package or feature parameters as well as footprint variables can be accessed within expressions. The context of the expression evaluation determines which parameters and variables are within scope.

Expressions in configuration objects that are elements of a particular footprint rule can reference the defined footprint variables. The footprint rules provided in the *'Training'* account define a number of top-level variables that are used in the subsequent padstack and shape generation rules. Certain configuration objects are evaluated within the context of a particular feature within a package. For example, Contact generators, Feature Procedures, Feature Shapes, Padstack generators and Padstack Modifiers are applied to individual features within a package and can reference parameters on the feature. Other configuration objects, such as Contact Feature generators, Footprint generators, Shape generators, and Simplified Model generators are applied to the package itself. For this reason, only package level parameters (and footprint variables, when applicable) will be accessible within the evaluation context.

The *'$'* preceding an attribute name enclosed within braces is used to reference the value of an parameter or variable within the context of the expression evaluation. A *'?'* preceding an

attribute name enclosed within braces is a query that returns the boolean True or False depending on whether the attribute is defined within the current evaluation scope.

${Contact|Length}	Returns the nominal value of the 'Contact|Length parameter on the current feature.
${Mask Offset}	Returns the value of the *'Mask Offset'* footprint variable.
$pkg{Gage Plane Offset:0.25mm}	Returns the value of the *'Gage Plane Offset'* attribute on the package, it if exists. If it does not exist, returns the default value of 0.25 mm
${Height}	Returns the nominal value of the *'Height'* parameter on either the feature or the package. If the expression is evaluated within the context of a feature, the Height parameter on the feature will be returned. If the expression is evaluated in the context of a package, the Height parameter on the package will be returned.
$pkg{Height}	Returns the nominal value of the *'Height'* attribute on the package. The *'$pkg'* syntax specifies that the package level attribute should be accessed. This syntax is valid in either the feature or package context, although unnecessary in the package context.
$body{Height}	Returns the nominal value of the 'Height' attribute on the body. The '$body' syntax specifies that the attribute on the body feature should be accessed. This syntax is valid in either the feature or package context.
$tpl{terminal_thickness}	Returns the nominal value of the 'terminal_thickness' template parameter of the package. The '$tpl' syntax is used to access the value of a template parameter for a package that is a template instance. This syntax is valid in either the feature or package context.

Values of Measurement Parameters

Measurement parameters are often used in expressions. Because a measurement parameter may represent either a simple measurement value or a toleranced measurement, a number of access functions are provided. The access functions described below can be

applied to any expression value that returns a measurement, including those accessed through the scope modifiers described above.

${Height}	Returns the nominal value of the *'Height'* attribute. This syntax is valid for either a simple measurement or a toleranced measurement.
${Height}.plus	Returns the *'plus'* value of the tolerance for the measurement parameter, if it exists, or None. This syntax should be used with caution, as it is only applicable for a toleranced measurement that has either an explicit maximum value defined, or a nominal value and a *'plus'* tolerance value.
${Height}.minus	Returns the *'minus'* value of the tolerance for the measurement parameter, if it exists, or None. This syntax should be used with caution, as it is only applicable for a toleranced measurement that has either an explicit minimum value defined, or a nominal value and a *'minus'* tolerance value.
${Height}.range	Returns the tolerance range for the measurement parameter. This syntax will always return a value for a valid measurement. In the case of a simple measurement, the value will be 0. In the case of a toleranced measurement, the difference between the maximum and minimum value will be returned.
${Height}.max	Returns the maximum value of the *'Height'* attribute, if it exists, or None. This syntax should be used with caution, as it is only applicable for a toleranced measurement that has either an explicit maximum value defined, or a nominal value and a *'plus'* tolerance.
${Height}.max_value	Returns the maximum possible value of the *'Height'* attribute. This syntax will always return a value for a valid measurement. In the case of a simple measurement, the value will be returned. In the case of a toleranced measurement, the maximum value will be returned.
${Height}.min	Returns the minimum value of the *'Height'* attribute, if it exists, or None. This syntax should be used with caution, as it is only applicable for a toleranced measurement that has either an explicit minimum value defined, or a nominal value and a *'minus'* tolerance.

${Height}.min_value	Returns the minimum possible value of the *'Height'* attribute. This syntax will always return a value for a valid measurement. In the case of a simple measurement, the value will be returned. In the case of a toleranced measurement, the minimum value will be returned.

Measurement Constant

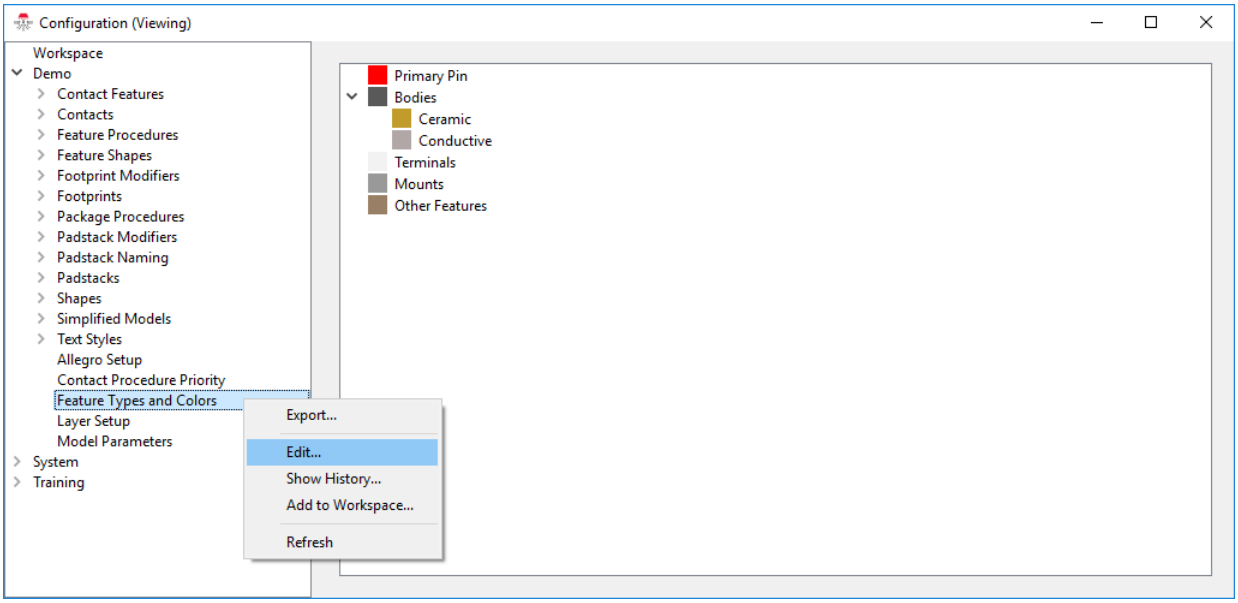
A physical measurement constant is represented as a floating point value followed immediately (with no whitespace) by one of the measurement unit types. The following are valid physical measurement constants for use in an expression.

0.1mm	A value of 0.1 mm.
0.1mil	A value of 0.1 mil.
0.1in	A value of 0.1 in.
0.1um	A value of 0.1 um.
0.1cm	A value of 0.1 cm.

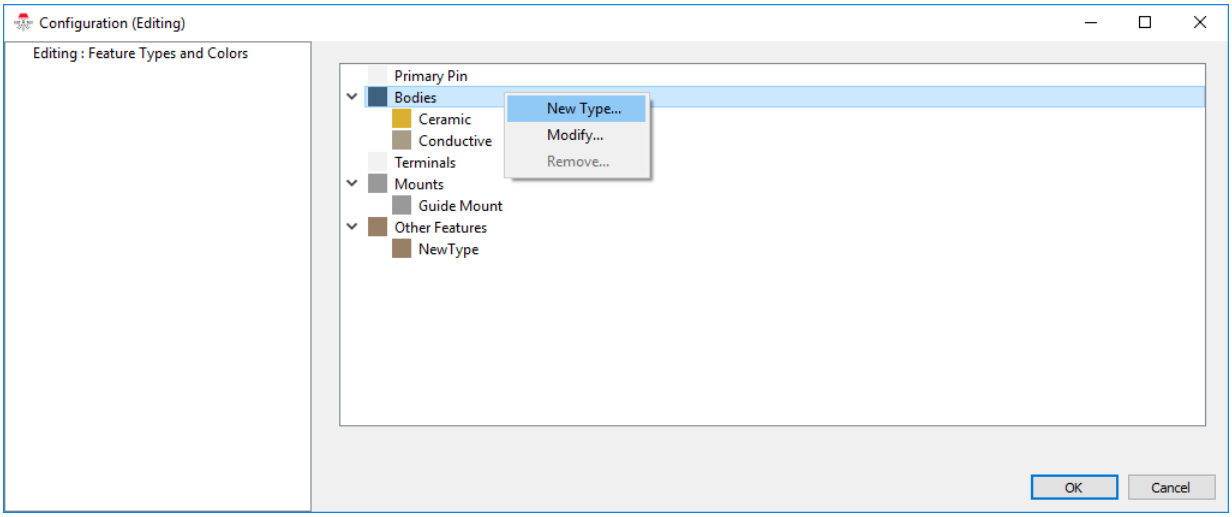
Feature Types and Colors

The feature types and colors configuration object enables the specification of feature types as well as the 3d colors that are assigned to the features. Feature types may be configured either to support specialized rule behavior or to support a particular color scheme, or a combination of the two. It is possible to create feature types within any of the four feature classes - Bodies, Terminals, Mounts, and Other Features.

Editing of the *'Feature Types and Colors'* configuration object can be initiated by using the *'Edit...'* context action as shown below.

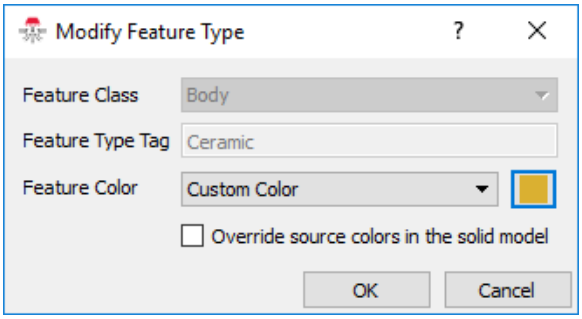


The *'New Type...'*, *'Modify...'*, and *'Remove...'* context actions may be used to create, modify and remove a type within one of the four feature classes.

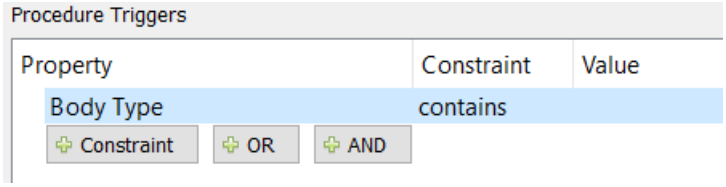


Set the *'Feature Color'* drop-down to either *'Default'* or *'Custom Color'* to inherit the 3D color from the feature class or specify a custom color. When the drop-down is set to *'Custom Color'*, click within the color box to edit the custom color through the *'Select Color'* color pick

dialog. The *'Override source colors in the solid model'* option enables the color to be applied to the feature when the package color mode is set to *'Source Colors.'*



Feature types may be used in procedure triggers for feature-based rules such as padstacks and feature shapes as shown below.



You can easily add constraints using the *Constraint* button. Also, you can create logical combinations of constraints using the *OR* and *AND* buttons.

Primary Pin (3d color only)

The *'Primary Pin'* within the *'Feature Types and Colors'* may be used to specify a custom color to be applied to the 3D geometry of the terminal marked as the primary pin.

Body Types

Body types may be created either to allow refinement of 3D color visualization, or to support functional rule behavior. For example, rules related to standoff height and copper pad trimming may vary depending on whether the package body is conductive. Additionally, a user may want to change the default body color for conductive package bodies.

Terminal Types

Terminals have an optional *'Type'* attribute that may be used to create a categorization for use during the application of rules. The terminal type is typically used in instances where the lead form is insufficient to drive padstack and / or footprint rules. For example, the *'Through*

- Rectangular' lead form describes the physical lead style. However, it may be necessary to use the supplemental *'Type'* attribute to identify whether a circular drill should be created rather than a slot. Terminal types defined with the user account will be appended to those defined in shared accounts (the *'System'* and *'Training'* accounts).

Mount Types

The *'Mounts'* type is used to categorize mount features for use during the application of rules. Because mount features do not have a lead form attribute, the mount type is generally a required for controlling padstack generation. The *'Mount Types'* allows the definition of a set of string tags or labels analogous to that of the body and terminal types.

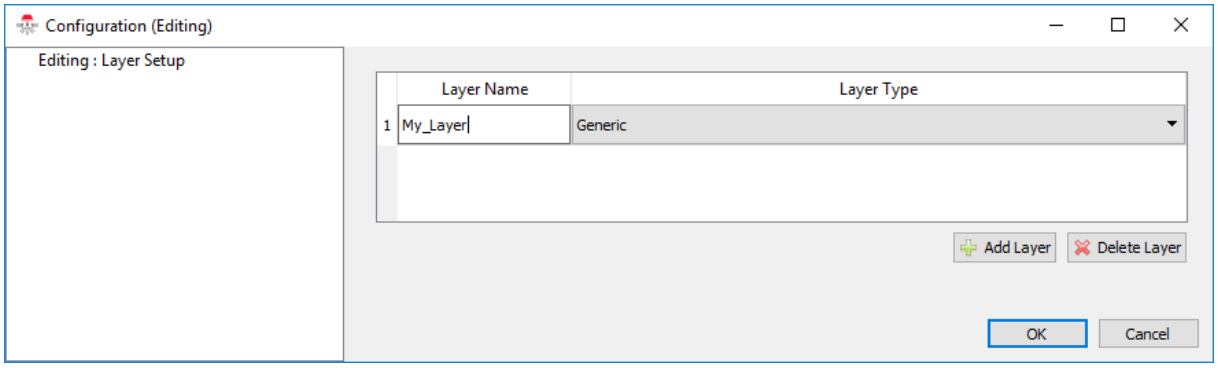
Other Feature Types

The *'Other Features'* type may be used to categorize package features for which the body, terminal, and mount labels are not appropriate. The *'Other Features'* item found in the *'Feature Types and Colors'* configuration object allows the definition of a set of string tags or labels analogous to that of the body and terminal types.

Layer Setup

'Layer Setup' is used to define additional layers for use in footprint representation. If a layer is to be mapped to Allegro, it must first be added to the layer setup. Use the right-click *'Edit'* action on the *'Layer Setup'* within *'Settings'* to enter editing mode.

The layer editing dialog is shown below. Layers may be added or removed using the *'Add Layer'* and *'Delete Layer'* buttons. A layer is defined by a name and a type, which is either *'Conductive'* or *'Generic'* for all non-conductive layers. After completing the edits *'OK'* will apply the edits to the workspace.

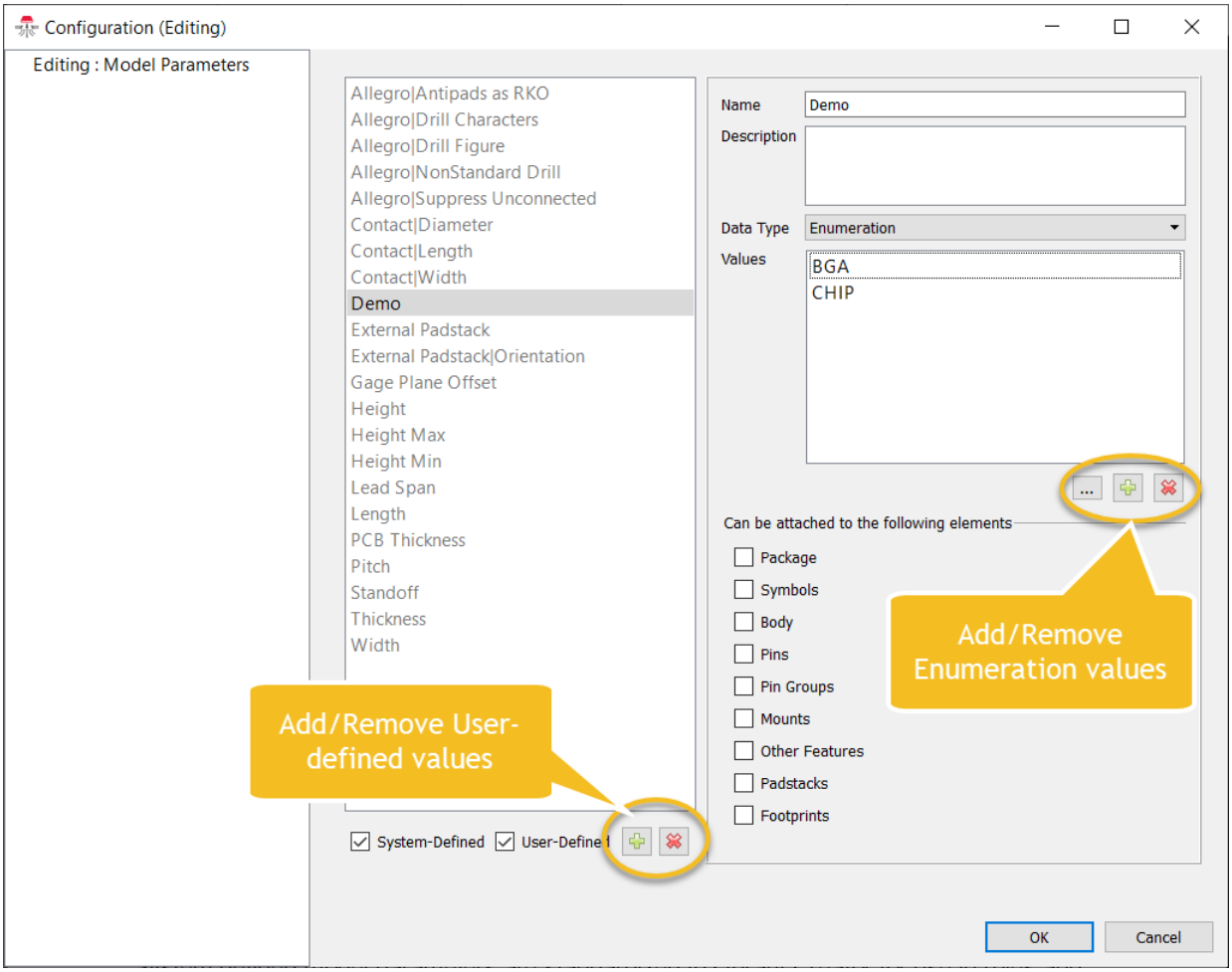


Model Parameters

Model parameters defined in the configuration may be attached to the package as well as elements of the package and footprint. The definition of a model parameter consists of a name, a data type, and a default value (or values in the case of an enumeration). In addition, the definition specifies what package and/or footprint elements the parameter may be assigned to. Model parameter may be either system or user defined.

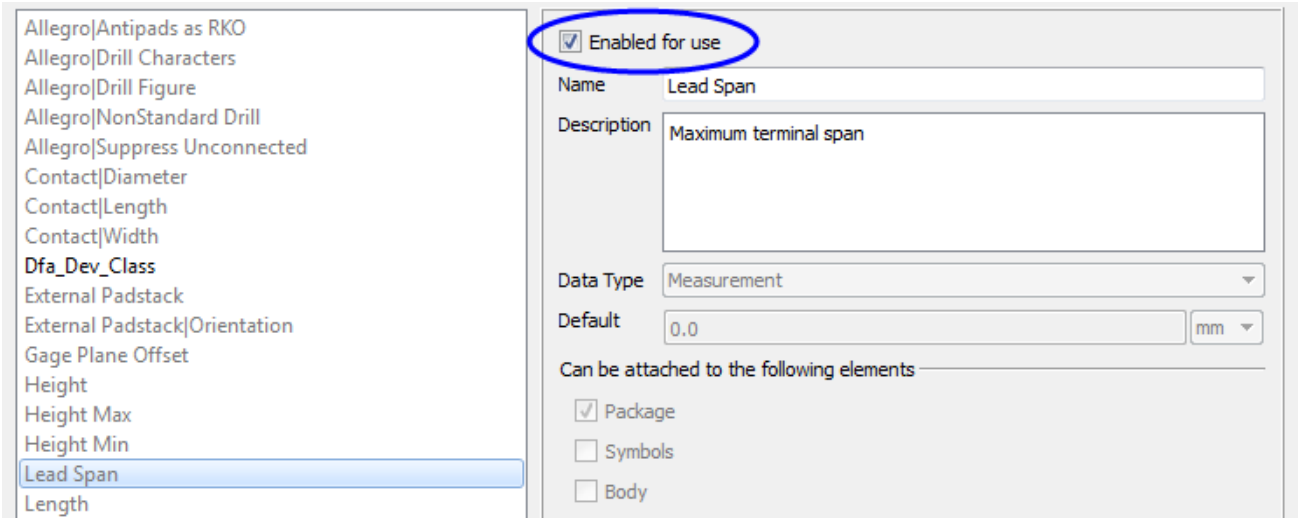
In order to edit the model parameter definitions, enter the editing mode with the right-click *'Edit'* context action.

The model parameter editing dialog is shown below. Multiple parameters may be edited in a single editing session. When editing is complete, hit the *'OK'* button to save changes to the workspace or *'Cancel'* to dismiss.



System-Defined

System defined model parameters are standardized in Library Creator for use in rules and repository functions. It is recommended that system-defined parameters be used whenever possible to ensure consistency and conformity with Library Creator conventions. While it is not possible to delete or change system-defined parameters, it is possible to disable them within the application. To disable a system-defined parameter, select the parameter and deselect the *'Enabled For Use'* check box before saving the edit. Disabled system defined parameters will no longer be visible in the editing dialogs and actions.



User-Defined

User-defined parameters provide extensibility to the package and footprint data model. User-defined parameters may be added to store a wide variety of package and footprint data. This data may be used in rules as well as propagated to Allegro.

To add a new user-defined parameter, use the *'+'* button and provide a unique name for the new parameter. Choice an appropriate data type and provide a default value, if applicable. In the case of an enumeration, as show above, it is possible to define a list of string values that will serve as drop-down choices when setting the value of the parameter.

9

Library Synchronization

Library Synchronization refers to the task of creating associations between ECAD footprints, Library Creator packages, and (optionally) part numbers. The synchronization process may be used to verify consistency between the footprint and the package or to add an associated 3D model in a consistent coordinate system for use in Allegro PCB Layout or ECAD / MCAD collaborative design.

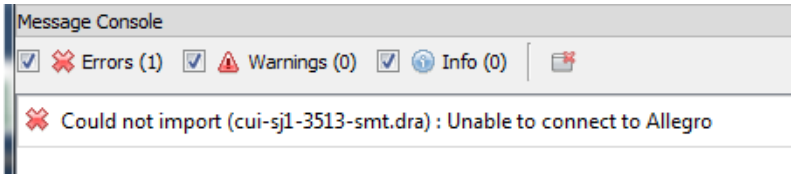
Synchronization associations are referred to as *'records'*. There are two types of records that can be created in Library Creator:

- A 'footprint default' record involves only a footprint name and a package name. The association will be used by Library Creator for all instances of an ECAD footprint in a PCB layout that do not have an over-riding part-number association.
- A 'part-footprint' record involves a part number, a footprint name, and a package name. This association will be used only for the specific combination of part number and footprint in a PCB layout.

Note that it is not necessary that both *'footprint default'* and *'part-footprint'* records be employed. An organization may choice to synchronize their library using only *'footprint default'* records, only *'part-footprint'* records, or a combination of both. Existing records may be viewed, and new records may be created within the *'Synchronize'* tab of the Library Creator application.

Many of the actions carried out during synchronization require Library Creator to load a footprint from the target ECAD library. Library Creator currently provides support for synchronization with an existing Allegro library (see *'ECAD Source...'* button at the top of the *'synchronize view'*). During the synchronization process, the footprints to be used must be located in a directory on the Allegro library Allegro search path.

If Library Creator is unable to access the Allegro footprint, an error similar to the following will be generated in the Library Creator message console:

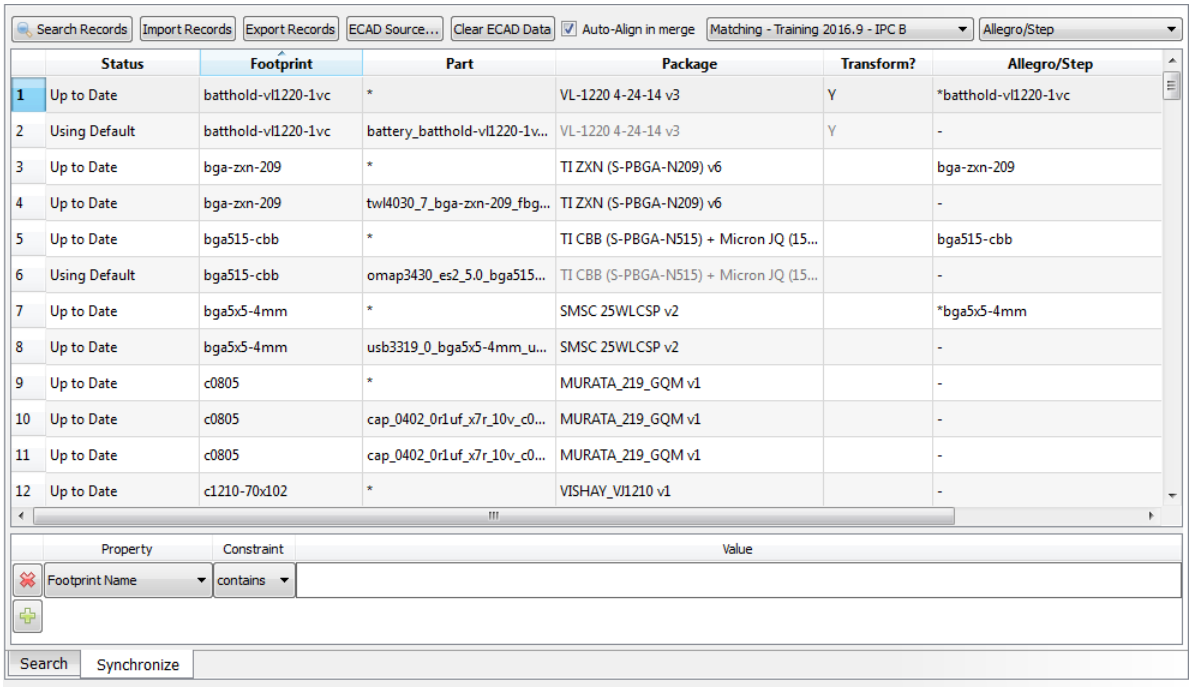


After resolving the communication issue, the process will need to be repeated.

Searching and Viewing Existing Records

Synchronization records are stored in the repository. In order to search existing records, search properties can be added in the *'Synchronize'* tab using the *'+'* button on the left side of the search properties pane. The available search properties include the footprint name, package name, and part number which may be search using string constraints such as *'contains'*, *'startswith'*, etc. There are also *'Part Assignment'* and *'Package Assignment'* properties which have only the Boolean constraint options *'exists'* and *'does not exist'*. Since both the *'footprint default'* and the *'part-footprint'* records must always contain a footprint name, a search property *'Footprint Name'* with the constraint *'contains'* and an empty value field will return all existing synchronization records. The search is executed by hitting enter in the value field or pressing the *'Search Records'* button at the top-left corner of the synchronize view.

The image below illustrates some of the possible record types as well as the information displayed in the synchronization view.

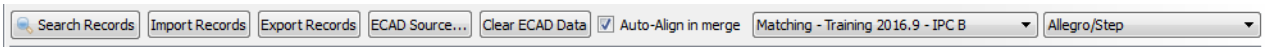


Records displayed on lines 4 and 6 in the image above both represent complete *'part-footprint'* records. The records contain a footprint name, a part number, and an associated package. A *'Y'* entry in the transform column indicates that Library Creator has stored a transformation needed to align the coordinate systems of the footprint and the 3D package model. The *'Up to Date'* in the Status column indicates that the version of the package currently associated with the part and footprint names specified is the newest version available in the repository. Records displayed on lines 1 and 3 are examples of *'footprint-default'* records. This is indicated by the *'*'* in the part column. A record with a status of *'Unassigned'* indicates that there is no package assigned for the particular record. The record displayed on line 2 is a *'part-footprint'* record with no assigned package. However, since there is a *'footprint-default'* for the same footprint (*'batthold-vl1220-1vc'*, line 1), the associated *'footprint-default'* package is displayed on line 2 in the grey color indicating that there is an inherited relationship for this part. The *'Using Default'* value indicates that the package assignment has been inherited from a *'footprint-default'* record.

Creating New Records

There are several methods that may be used to create new synchronization records. Optionally, *'footprint-default'* synchronization records may be automatically created during export of a generated footprint to Allegro. Alternatively, new records may be created from a

variety of import sources using the *'Import Records'* action in the Synchronization view toolbar.



Automated Creation of Synchronization Records

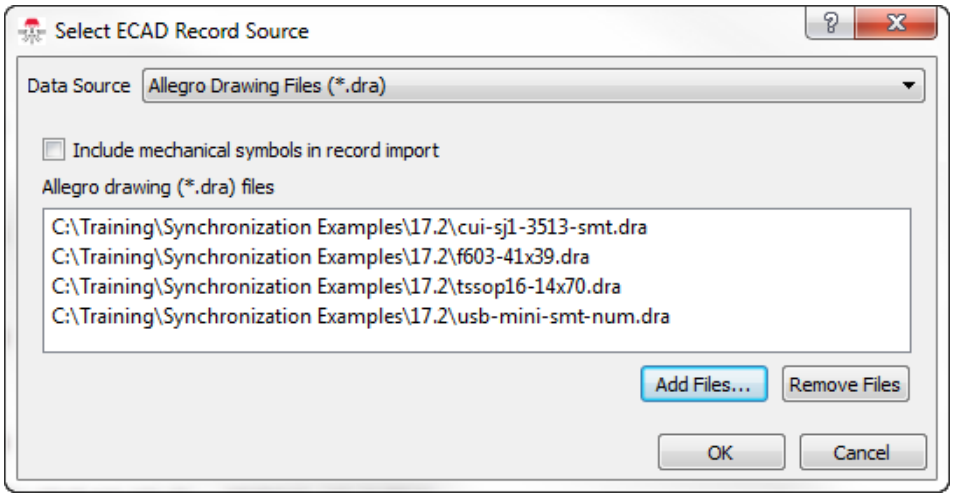
'Footprint-default' synchronization records may be created automatically during export of footprints generated by Library Creator to the target ECAD system. To enable automated creation of synchronization records during export to Allegro, select the *'Automatically create synchronization relationships'* option in the *'Advanced Export Options'* tab of the Allegro Options dialog.

When a footprint is generated directly from a package model in Library Creator, consistency between the package and footprint is ensured. When relying on automatically created synchronization records, it is required that the package be uploaded to the repository prior to footprint export, in order to ensure that the synchronization record is consistent with the package stored in the repository. After the footprint default synchronization record has been created, it may be desired to create part number associations for this same footprint-package combination - see Modifying Existing Records.

Importing Records - Allegro Source

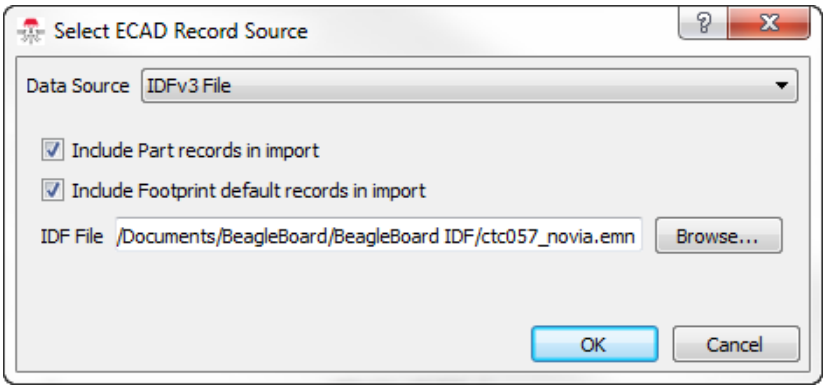
'Footprint default' records can also be created from an existing Allegro source file (`.dra`) or library folder - select *'Allegro Drawing Files (*.dra)'* or 'Allegro folder (*.psm, *.bsm) Files' from the Data Source drop down in the 'Import Records' dialog. All records created from an Allegro source will be 'footprint-default' records created with a status of '*Unassigned*'. It will

be necessary to add the package associations to the individual records later - see Modifying Existing Records.



Importing Records - IDF source

Records can be created from an existing IDF source file (select IDFv3 File) from the Data Source drop down in the *'Import Records'* dialog. During record creation from an IDF file, the user has the option of creating footprint default matches, part-footprint matches, or both (see dialog below). All records created from an IDF source will be *'unassigned'* records. It will be necessary to add the package associations to the individual records later - see Modifying Existing Records.



Importing Records - CSV Source

Records can be created from an existing CSV source file (select *'CSV File'* from the Data Source drop down in the *'Import Records'* dialog. The first row of the CSV file must contain

the case-insensitive column headings *'Footprint'*, *'Part'*, and *'Package'*. There is no required sequence to the columns, and data from columns with any other column heading (or an empty column heading) will be ignored. The CSV file may be created using a spreadsheet application such as Microsoft Excel (Save as CSV), or a standard text editor. A sample CSV source file is shown below in both Excel and a text editor.

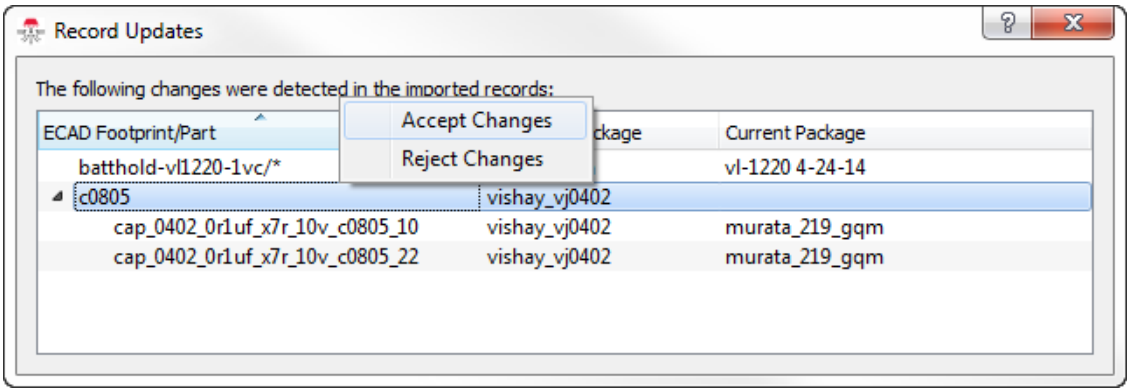
The first record (row 2) specifies the creation of a *'footprint default'* records, while the following two records represent *'part-footprint'* relationships. In the final record (row 5), no footprint is specified. This is not a valid synchronization record, and will be ignored during import.

	A	B	C
1	Footprint	Part	Package
2	batthold-vl1220-1vc	*	VL-1220/VCN
3	C0805	CAP_0402_0R1UF_X7R_10V_C0805_22	VISHAY_VJ0402
4	C0805	CAP_0402_0R1UF_X7R_10V_C0805_10	VISHAY_VJ0402
5		CAP_0402_0R1UF_X7R_10V_C0805_10	VISHAY_VJ0402
6			

Sample_Synchronization_Records.csv

```
1  Footprint,Part,Package
2  batthold-vl1220-1vc,*,VL-1220/VCN
3  C0805,CAP_0402_0R1UF_X7R_10V_C0805_22,VISHAY_VJ0402
4  C0805,CAP_0402_0R1UF_X7R_10V_C0805_10,VISHAY_VJ0402
5  ,CAP_0402_0R1UF_X7R_10V_C0805_10,VISHAY_VJ0402
6
```

After the *'Import Records'* action is initiated with a CSV record source, the *'Detected Record Updates'* interface will appear. The user must manually accept or reject the proposed change using the applicable right-click action, as shown below.



Prior to creating a new association the package and footprint will be brought together through the merge process for manual review and inspection.

Modifying Existing Records

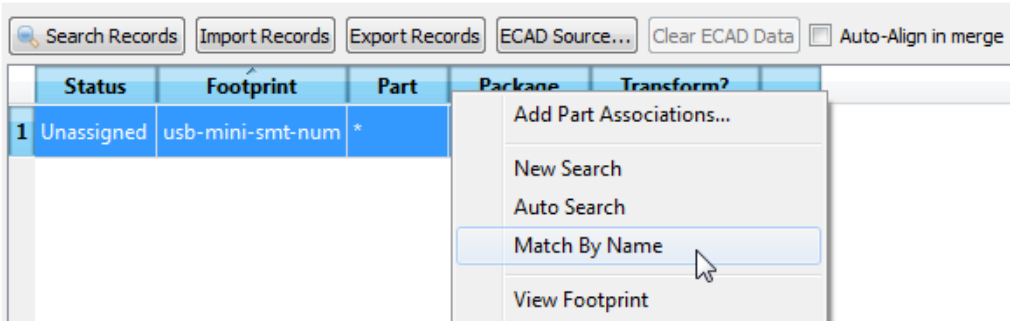
Regardless of the method of creation, a *'part-footprint'* record can be created from a *'footprint default'* record through a right-click action on the record in the synchronize view. Note that this does not replace the existing *'footprint default'* record. If it is desired to eliminate the footprint default record, it must be explicitly deleted through the right-click *'Delete Records'* action.

Assigning Package To Record

A package can be associated with an existing record by either specifying a package manually or by using the automated search functions to attempt to find compatible packages for the given footprint.

Match By Name

If you know the exact package to be used for synchronization, you can specify it directly by selecting the record and using the right-click *'Match By Name'* context action.

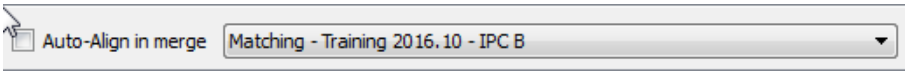


Then, enter the exact name of the package into the dialog. This will initiate the merge process.

Auto Search and Scoring

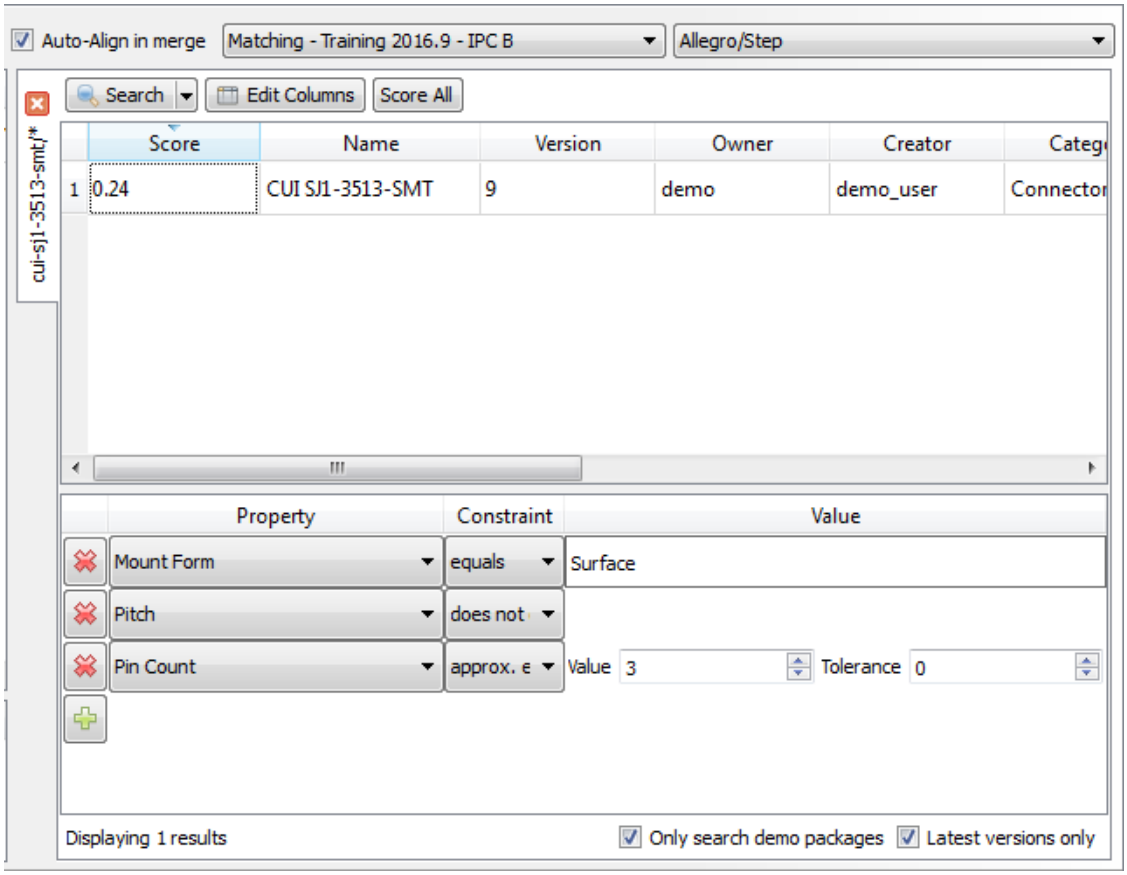
The *'Auto Search'* action can be used to find candidate packages from the repository that may be compatible with the given footprint. The *'Auto Search'* function extracts details from the footprint and constructs automated search queries for the repository.

The automated search and scoring functions rely on a matching configuration, which is selected through a drop-down in the synchronization toolbar. Prior to executing these functions, ensure that the configuration is set to an appropriate value:

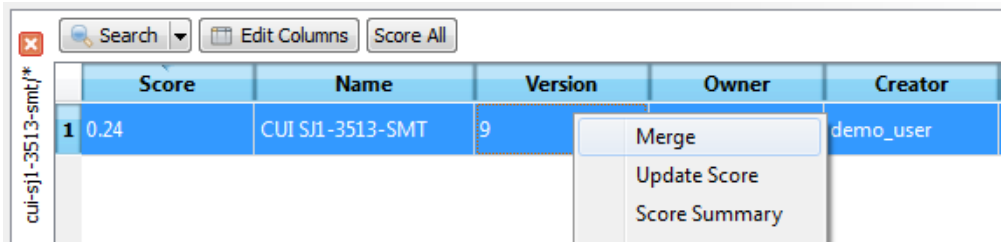


In order for the *'Auto Search'* to function, the footprint must be available on the Allegro search path. If this is not the case, you may see an error similar to the following:

If Library Creator is successfully able to access the footprint from Allegro, a new search pane will appear on the right-side of the synchronization view. Pressing the *'Search'* button at the top will generate a repository search based on the automatically generated search terms, visible in the bottom of the search pane. Additionally, you can trigger the scoring through the *'Score All'* button, or a right-click *'Update Score'* action on an individual result. Note the check box at the bottom of the search view that controls whether the search is limited to only local account packages.



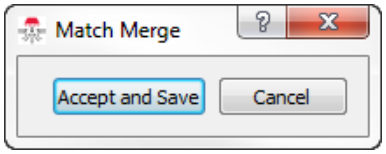
A footprint can be synchronized with a particular package from the search results using the right-click *'Merge'* action. This has the same result as the *'Match By Name'* action described above and will initiate the merge process.



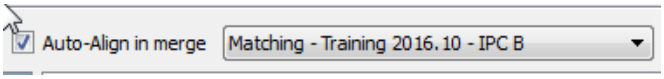
Note that the automatically generated search terms will sometimes be either too restrictive or not restrictive enough. In this case, the search terms can be manually manipulated, and the search process repeated.

Merge Package and Footprint

The final step in associating a package with a synchronization record is the *'Merge'* between the package and the footprint. During this step, the footprint and package are brought together in a unified view for visual inspection and confirmation. In the merge view, the Allegro footprint is brought together with the package model, and it is possible to adjust the relationship between the coordinate systems. The relationship can be inspected in both the 2D and 3D views, prior to accepting or cancelling the merge through the *'Match Merge'* dialog (in cases of multiple monitors, the *'Match Merge'* dialog may be positioned off screen).



If the Auto-Align in merge check box in the synchronization toolbar is selected, Library Creator will attempt to automatically align the footprint and package.

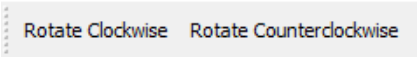


This process requires that the terminal and mount features have associated contact areas. If the automated alignment is unsuccessful, or incomplete, it is always possible to manually adjust the relationship between the coordinate systems.

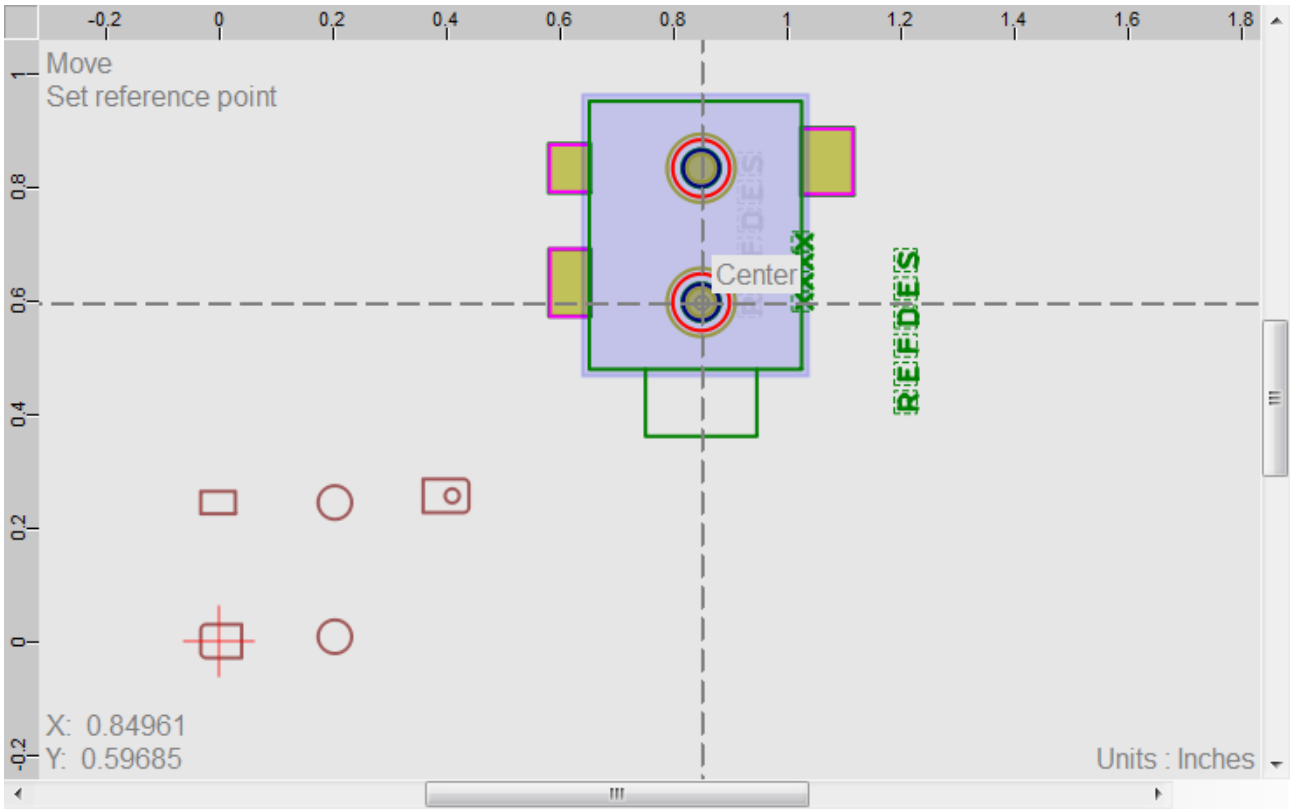
Edit Merge

In cases where automated alignment is not possible or successful, the alignment can be manually edited using the *'Edit Merge'* action available on the right side in the toolbar of the Footprint (2D) merge view.

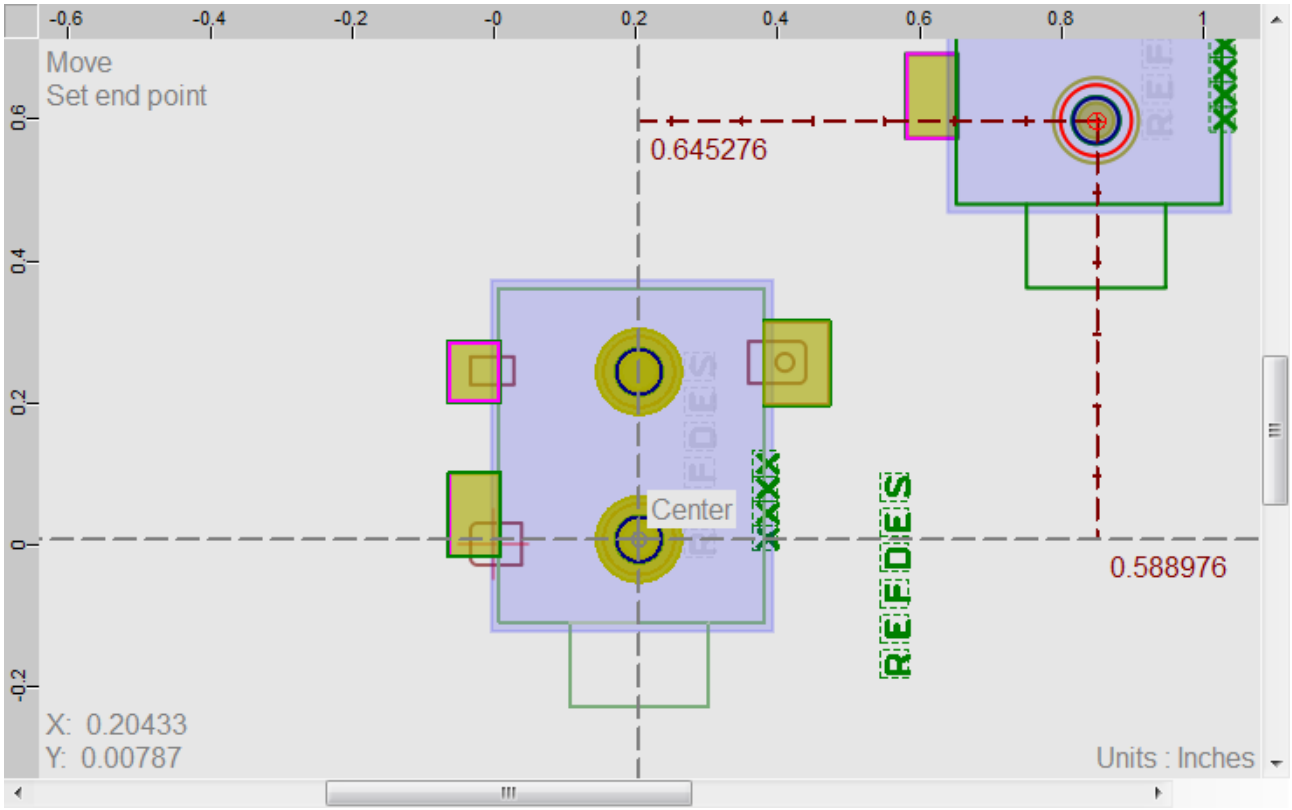
Within the *'Edit Merge'* tool, you can rotate the coordinate system 90 degree clockwise or counter-clockwise, as well as translate the coordinate system of the footprint relative to the package. Typically, it will be easier to adjust the rotation first, using the buttons at the top of the *'Edit Merge'* toolbar.



Once the rotation has been corrected, the translation can be corrected by providing an initial reference point and an end point. Note the prompts in the upper left corner of the graphics view. Often it will be possible to use a snap point on the footprint for the reference point, and the corresponding snap point on the package contacts as the end point in order to achieve precise alignment without entering coordinate values directly. In the example below, the center point of the padstack is aligned with the center point of the contact.



After the reference point has been specified, the tool prompt indicates that the *'end point'* should be set.

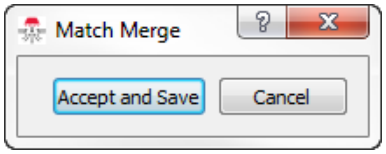


Note that in the above example, the automated alignment indicates a problem with the Allegro footprint relative to the physical 3D package model. Although the mounting pegs are properly aligned, the pad for pin 1 is misaligned with represent to the terminal. Resolving this problem would require that the Allegro footprint be corrected to be consistent with the physical package.

In more complex cases, it may be necessary to align x and y separately. This may be done in two steps, toggling the snap controls and vertical and horizontal restriction modes as needed using the buttons on the Snap context menu or right toolbar.

When necessary, explicit coordinate and/or delta values can be entered during the move command using the *'Coordinateâ?¦'* and/or *'Deltaâ?¦'* actions on the right-click context menu.

After alignment has been completed, exit the *'Edit Merge'* mode with the *'Accept'* action from the context menu or top toolbar. Finally, accept or cancel the merge through the *'Match Merge'* dialog.



Propagate the Synchronized 3D Model

At this stage, the synchronization record has been updated to reflect the fact that a package is now associated with the footprint. The relative transformation (if any) between the ECAD and MCAD coordinate systems is stored with the record. Hovering over the record in the synchronization view will display record details.

The synchronization record is stored persistently in the repository. If desired, it can be deleted with a right-click *'Delete Records'* action. Existing records can be searched for in the synchronization view.

Push STEP to Allegro

Use the right-click *'Push STEP to Allegro'* action on one or more synchronization records to propagate the 3D STEP models to Allegro and create mapping relationships with the footprints. Note that this action will modify the target Allegro footprints. Confirm that the Allegro Options have been properly configured for STEP export.

Push STEP to Disk

Use the right-click *'Push STEP to Disk'* action on one or more synchronization records to propagate the associated 3D STEP models to disk in a consistent coordinate system.

10

Allegro Interface

The Allegro interface supports import and export of footprints, padstacks, and associated shape symbols to and from Allegro X PCB Editor to support the population and synchronization of the PCB library. Allegro import and export is triggered through the *File – Import – Allegro* and *File – Export – Allegro* menu actions. Prior to Allegro import and export, ensure that the settings has been enabled through the daemon and the settings have been configured.

The majority of standard Allegro footprint layers have been pre-configured in Library Creator. If you need to use the custom subclasses and the less commonly used Allegro class or subclasses, additional layers can be configured and mapped.

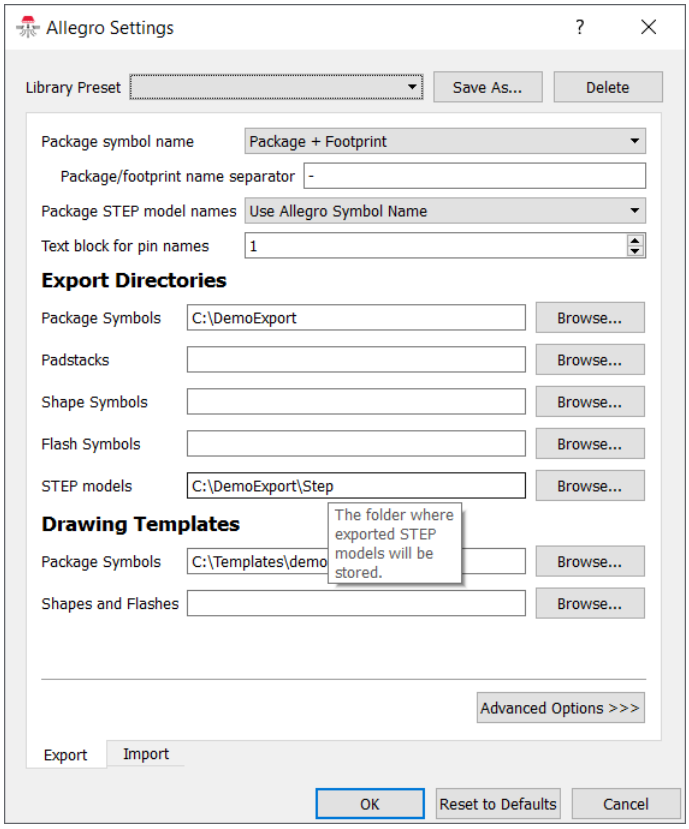
Note: To export to a custom subclass, it is necessary to configure a template containing the subclass.

Allegro Settings

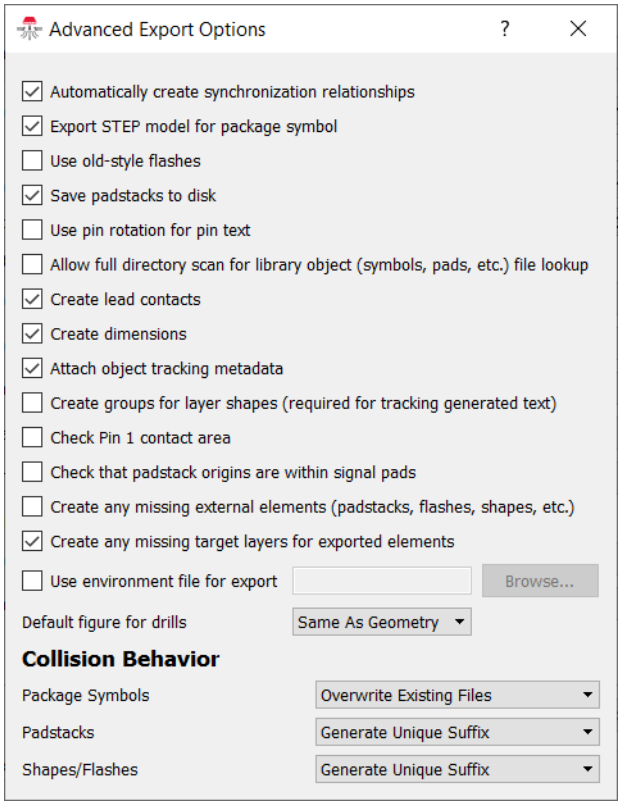
Allegro import and export settings are configured through the *Allegro Settings* dialog. The dialog is launched through the *Tools – Allegro Settings* menu. The settings are organized into the Import and Export tabs in the dialog. Multiple settings may be configured using the *Save As* button at the top of the *Allegro Settings* dialog. Alternate settings may be used to support configurations for multiple target libraries. It is recommended that the default settings be reviewed prior to exporting Allegro footprints from Library Creator.

Export Settings

The export settings are configured through the *Export* tab of the *Allegro Settings* dialog.



The *Advanced Options* button on the *Export* tab contains a series of additional export option configuration settings.



Use this Option	To
Automatically create synchronization relationships	Automatically create a synchronization between the package and the created Allegro X PCB Editor symbol when the footprint is written to disk. A synchronization record is created and the name of the generated PCB Editor package symbol is mapped to the package being generated.

Use this Option	To
Export STEP model for package symbol	Include the detailed 3D solid geometry of the package with the Allegro X PCB Editor package symbol. The STEP model is written to the directory specified as the export directory for STEP models and the mapping relationship is automatically populated in the package symbol.
Use old-style flashes	Force old-style flash support for Allegro X PCB Editor packages. If this option is enabled, all exported flashes are created in old-style. Deselect this option if you do not need the old-style flash support.
Save padstacks to disk	Save all created padstacks to disk. The padstacks are written to disk in the directory specified as the export directory for padstacks. If no directory is specified, the padstacks are written to the export directory specified for *Package Symbols*. If this option is disabled, generated padstacks are only created in the package symbols and are not written to disk.
Use pin rotation for pin text	Pin text follows the padstack orientation. The baseline of the generated pin text is equivalent to the local positive X-axis for the corresponding pin.
Allow full directory scan for library object (symbols, pads, etc.) file lookup	Allegro interface falls back to full directory scanning to resolve names of library elements if they cannot be found using direct lookup. Enabling this option can have significant performance penalties in cases where a large library may be hosted remotely on network machines. This option is useful when the library files are hosted on a case-sensitive filesystem and the incoming data (pad names, padstack names, etc.) have case inconsistencies with the library file elements.

Use this Option	To
Create lead contacts	Pin contact geometry is populated in the generated Allegro X PCB Editor package symbol and visible in the Lead Editor and the *Package Geometry /Component_Lead* class and subclass.
Create dimensions	Populate the dimensions in the *Package Geometry* or *Dimension* class and subclass.
Attach object tracking metadata	Populate the properties in the Allegro X PCB Editor package symbol referencing the source package model and the rules used to generate Allegro X PCB Editor data elements. The populated properties are prefixed with `Lc_`.
Create groups for layer shapes (required for tracking generated text)	Populate the shapes within groups in the Allegro X PCB Editor package symbol. This is required to support the population of object tracking metadata for text.
Check Pin 1 contact area	Check that the pin 1 contact area overlaps the associated signal pads for the corresponding padstack placement.
Check that padstack origins are within signal pads	Check that the padstack origin is contained within all signal pad geometries for the corresponding padstack placement.
Create any missing external elements (padstacks, flashes, shapes, etc.)	The default behavior and assumption of padstack elements in the external state is that they already exist in the target library and are not generated by Library Creator. Also, padstacks, shapes, and flashes in the external state that are not found in the Allegro search path are created and written to disk.

Use this Option	To
Create any missing target layers for exported elements	Automatically create any missing target layers in the Allegro X PCB Editor symbol during export. If this option is disabled, missing target layers are treated as an error and it will be necessary to add them explicitly to the export template `.dra`. Leaving this disabled also allows explicit control of what layers are populated in the generated Allegro X PCB Editor symbol.
Use environment file for export	Allow an `.env` file to be used to set or override desired environment variables in the Allegro X PCB Editor environment used for export.
Default figure for drills	Specify the drill figure for generated drills and slots.
Collision Behavior	These options control the export behavior when a library element (footprint, shape, flash, padstack, etc.) is generated that has the same name as an existing library element. Note that whenever possible, Library Creator re-uses the existing element of the same name. If the existing element is equivalent, it is re-used and the collision behavior specified in the *Package Symbols*, *Padstacks*, and *Shapes/ Flashes* fields is not required. The available options are: ■ *Overwrite Existing Files*: The existing element on the disk is overwritten with the current element of the same name. ■ *Generate Unique Suffix*: A unique suffix is appended to the name of the current element and written to the disk to avoid a name collision. ■ *Abort Export*: The export is aborted in case a name collision with an existing library element.

Export Directories

When exporting a footprint to Allegro, a variety of files are generated - the top level package symbol (footprint), padstacks, and possibly shape and flash symbols. The export directories are configured under the *Export Directories* heading of the *Export* tab. At a minimum, a target directory must be specified for the *Package Symbols*. If only the package symbols path is set, all files will be created within this single directory. If desired, alternate paths may be configured for padstacks, shape, and flash symbols, as well as STEP models.

STEP Export

The 'STEP models' export directory controls the destination directory for STEP models generated by Library Creator during export to Allegro. Library Creator will automatically create the mapping relationship between the Allegro footprint and the STEP model (see 'Setup -> Step Package Mapping...' in PCB Editor). By default, Library Creator will export and map STEP models with footprints.

If it is desired to disable this behavior, it can be done by deslecting the 'Export STEP model for package symbol' option under the *Advanced Export Options* dialog. Note that the directory containing the 3D STEP models must be included in the Allegro 'steppath' in order for PCB Editor to load the associated 3D models.

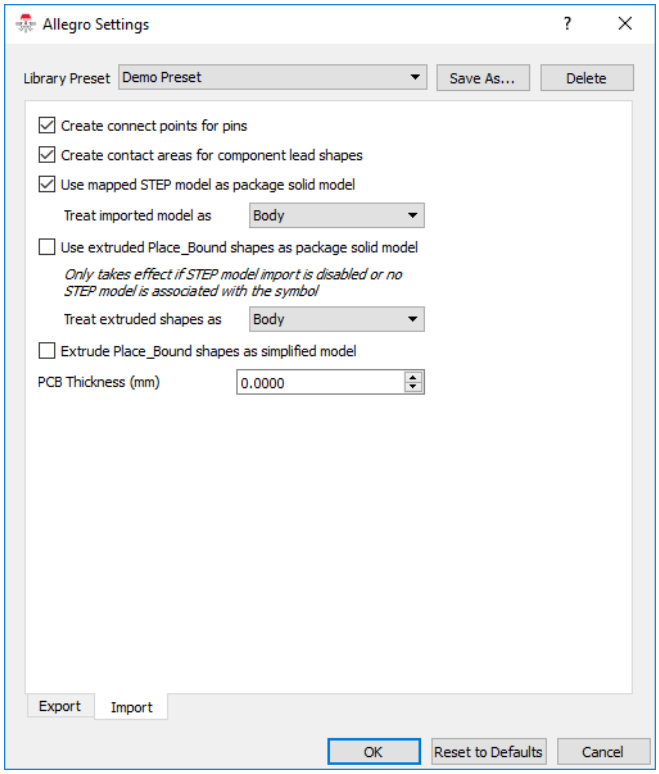
Drawing Templates

Drawing templates are Allegro .dra files that are used as a starting point for constructing Allegro footprints and shape symbols during export. Drawing template settings are configured under the 'Drawing Templates' heading of the 'Export' tab. The footprint unit is controlled by the design units of the drawing template. A drawing template may contain preferred settings for library elements such as units, layer and visibility settings, text blocks, user-defined properties, and custom subclasses. The package template may also contain static elements such as an origin marking. If desired, an alternate template may be specified for creating shape symbols. The shape symbol template must not contain any geometric elements - if the package export template contains any geometric elements, it is necessary to configure a separate template `.dra` for generation of shape and flash symbols.

Import Settings

The import settings are configured through the 'Import' tab of the dialog as shown below. By default, STEP models that are mapped to Allegro footprints are imported with the footprint. The imported 3D shape can either be set as the body (default behavior) or a Generic (Other) feature type. When a STEP model is not associated with the footprint, or if the STEP import

is disabled, it is also possible to create extruded 3D shapes from the Place_Bound regions in the footprint.



Library Paths

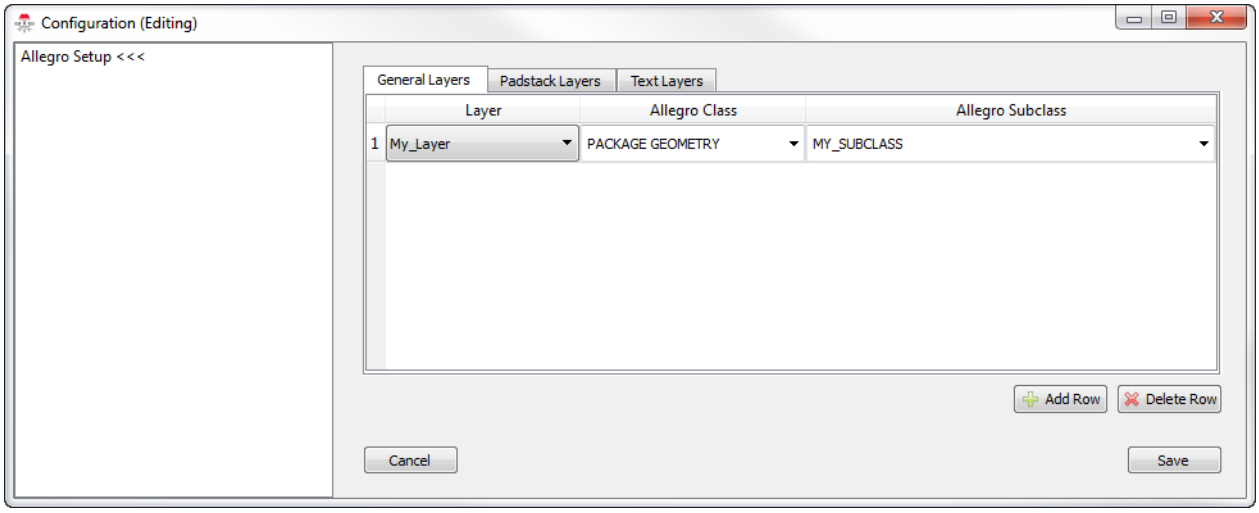
It is important to ensure that footprints, padstacks, and STEP models for use in Library Creator are available on the defined library paths. Library Creator uses the same search path mechanism as Allegro for the import of footprints, padstacks, shape symbols, and STEP models. The Allegro configuration is loaded by Library Creator at application launch. If the configured search paths have been changed, it is necessary to relaunch Library Creator in order to refresh the settings.

Allegro Padstack Editor

External padstacks are locked for editing in Library Creator. It is possible to invoke the Allegro padstack editor on an external padstack in Library Creator using the 'Edit external padstack...' context action. After completing the desired edits, the padstack may be saved. Upon closing the padstack editor, the external padstack will be automatically refreshed in Library Creator.

Class/Subclass Mapping

The majority of the layers commonly used in Allegro package and mechanical symbols are predefined in Library Creator and mapped automatically to the applicable Allegro class and subclass. However, it is possible to add additional layers in Library Creator though the Layer Setup. These added layers may be explicitly mapping to an Allegro class and subclass through the *'Allegro Setup'* dialog in the configuration *'Settings.'* Use the right-click *'Edit'* action on the *'Allegro Setup'* within *'Settings'* to enter editing mode. Use the *'Add Row'* and *'Delete Row'* buttons to add and remove a layer mapping. Select the Library Creator layer from the *'Layer'* drop down, and select the mapped Allegro Class and Subclass. If the class and/or subclass is not available in the drop-down, the name may be entered directly. Mapping to user-defined subclasses is supported, however, the subclass must be defined in the drawing template prior to export.

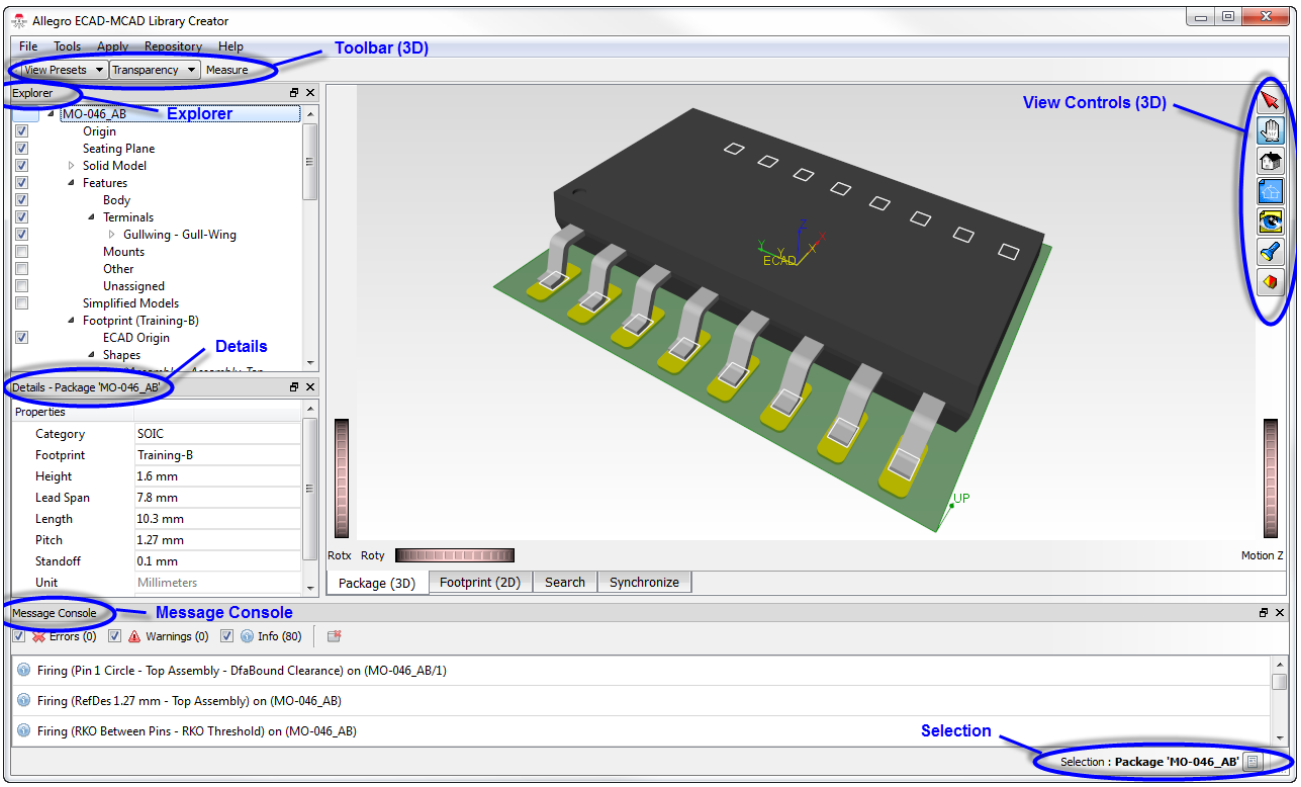


11

User Interface

Package (3D) View

The Package (3D) View is activated through the *'Package (3D)'* tab. When in the package view, it is possible to view and inspect the 3D package. The package view is also used to assign or modify package features. The View Controls (3D) and View Toolbar (3D) are unique to the Package (3D) View, while the Explorer, Details, and Message Console are common to both the Footprint (2D) and Package (3D) views. The primarily UI components of the Package (3D) view are labeled in the image below.



View Controls (3D)

The toolbar at the upper-right corner of the 3D pane enables manipulation of the 3d view, as well as selection. The tools are as follows:

 **Select**	Select mode. Used to select faces and/or solids during featurization or seating plane alignment. Also required for measurements within the Measure wizard.
 **Rotate / Pan / Zoom**	Rotate / pan / zoom mode. Left mouse button to rotate, press center button / scroll wheel to pan, and scroll wheel to zoom.
 **Return to Home**	Return the 3D view to the current home position.
 **Set Home**	Set the current view and orientation as the home position.
 **Zoom To Fit**	Zoom to fit to the current window. Orientation unchanged.
 **Set Focus (center of rotation)**	Set focus on a point in the 3D view. Focus point becomes the center of rotation. View is transformed to center on the focus point selected.
 **Orthogonal vs. Perspective**	Toggle between orthogonal vs. perspective view for the 3D view.

View Toolbar (3D)

The View Toolbar (3D) is shown below. It's default position is at the upper-left corner of the main window, but it can be used as a floating toolbar, as well as docked on the top, bottom, left, and right edges of the main window. To move the toolbar, drag the left edge of the toolbar with a left-click mouse action.

The toolbar actions are as follows:

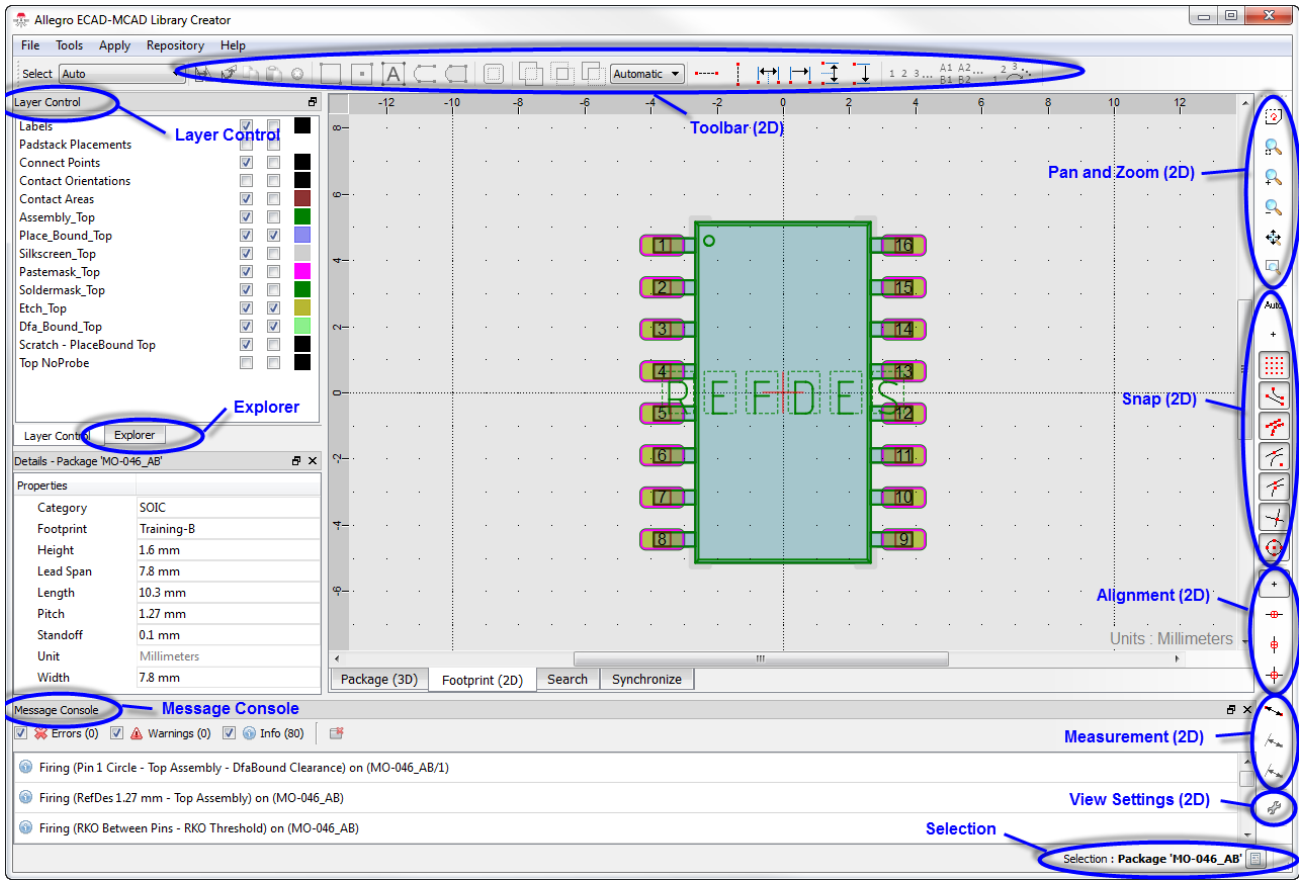
View Presets	Drop-down provides access to 12 orthogonal view presets (+XY, +XZ, +YX, etc.) The view will be transformed such that the first axis specified is rightward facing, and the second is upward facing. For example +XY represents a top-down view from the positive Z axis, (positive X axis to right, positive Y axis oriented upward), while -XY represents a bottom up view from the negative Z axis, with the positive X axis facing left and the positive Y axis oriented upward.
Transparency	Enable / disable transparency of all applicable elements of the 3D, including the seating plane, gage plane, section offsets, and simplified models.
Measure	Initiates the 3D measurement tool. Note that the 3D measurement tool must be dismissed through the 'Accept' action at the right-side of the toolbar before proceeding to other UI interactions. Select a measurement mode from the 'Distance Mode' drop-down in the toolbar. While in Select mode (View Controls 3D), initiate a measurement with a left-mouse click. A second mouse click will complete the measurement. The distance is displayed near the measurement. Measurements will be made between points on the surface of the 3D model. To achieve accurate measurements, it is often helpful to switch to one of the view presets combined with orthogonal view mode (View Controls 3D).

The Footprint (2D) View

The 2D Footprint View is activated through the *'Footprint'* tab below the main center panel. When in the footprint view, it is possible to view, inspect, and edit the footprint. The footprint view is also used to name, copy, and delete pins (terminals) of the package. The primary UI components that rare unique to the footprint view are the View Toolbar (2D), the Layer Control, the Pan and Zoom Controls (2D), the Snap Controls (2D), the Alignment Controls (2D), and the Measurement Tools (2D). The elements are labeled in the image below, and described in additional detail in the following sections.

While in the footprint (2D) view, the mouse may be used for panning of the current viewpoint while holding down the left mouse button. The center mouse thumb wheel may be used to

zoom in and out.In the majority of modes, panning (translation) of the view is accomplished while holding down the left mouse button.

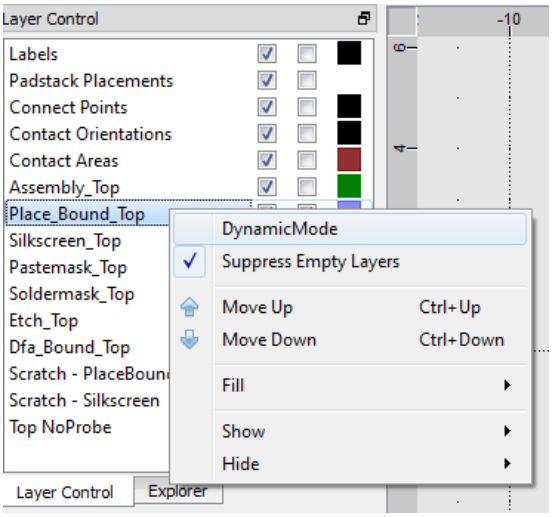


Layer Control

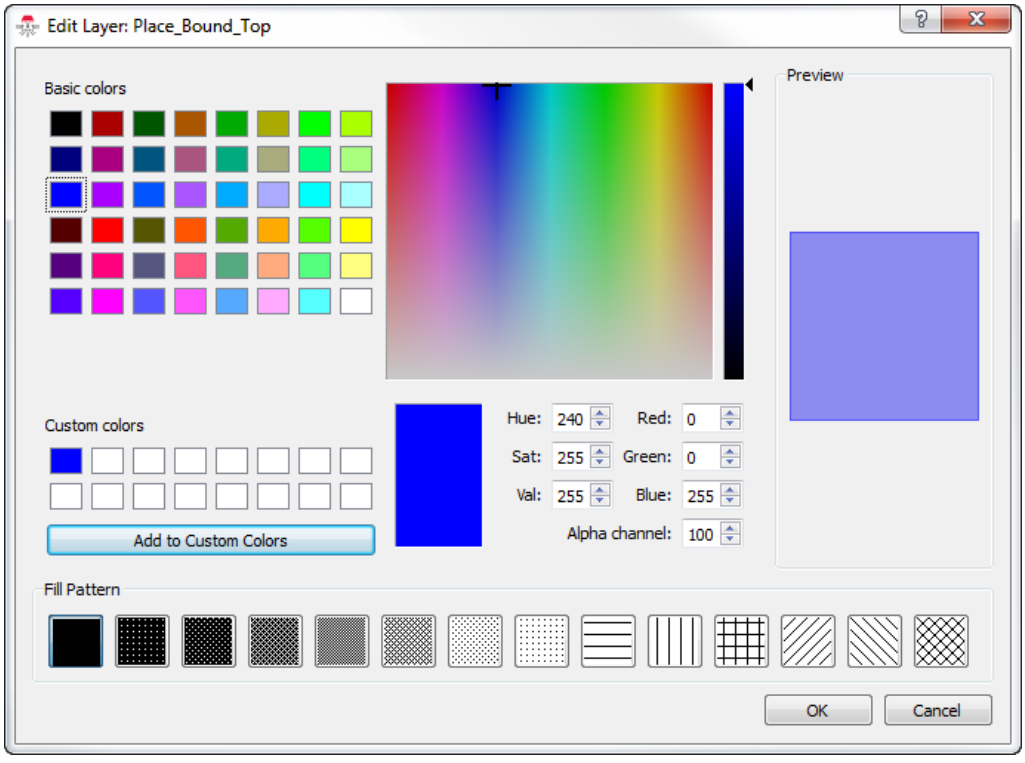
The dockable Layer Control Widget controls the visibility and display settings of the package and footprint layers in the Footprint (2D) view. The first check box column toggles layer visibility on and off, the second column toggles the layer fill, and the third column launches the Layer Color Editor. The right-click context menu provides additional configuration and display options. The rendering stack-up follows the sequence of layers in the widget. This can be changed through *'Move Up'* and *'Move Down'* actions available through the context menu, and the corresponding keyboard short-cuts.

In Dynamic Mode, available through the context menu, layer visibility follows the selection of layers in the widget. *'Shift'* and *'Ctrl'* click selection can be used to rapidly set / clear visibility

of multiple layers. When a single layer is selected, the up and down arrows can be used to rapidly step through a sequence of visible layers.



The layer visibility settings include color, transparency, and optional fill pattern. The Edit Layer Color widget enables configuration of the visibility settings for a given layer. The layer transparency is specified through the *'Alpha channel'* setting, which ranges from 0 (fully transparent) to 255 (fully opaque).



Pan and Zoom Controls (2D)

Redraw	Refresh the current graphics view.
Auto Zoom	Fit the contents of the footprint view to the current window size.
Zoom In	Zoom in (or scroll with center mouse thumb wheel).
Zoom Out	Zoom out (or scroll with center mouse thumb wheel).
Pan Zoom	The standard mode for 2D pan and zoom. Hold down the left mouse button to pan. Scroll with center mouse thumb-wheel to zoom.
Window Zoom	Zoom to a selected region. After selecting 'Window Zoom' drag a rectangular region with the left mouse button to define the zoom extents.

Snap Controls (2D)

Auto Snap	Enable all snap options. Snap to grid, reference points, end points, middle points, and entity boundaries.
Free (No Snap)	Disable all snapping.
Grid Snap	Snap to the predefined grid.
Endpoints Snap	Snap to endpoints of linear and circular arc segments defining standard symbols as well as paths and polygon boundaries.
On Entity Snap	Snap to entity boundaries. Often useful in combination with alignment restrictions.

	**Center Snap**	Snap to center position of standard shape symbols. For segments in paths and polygons, snaps to the center point of arc segments, and center of linear segments.
	**Middle Snap**	Snap to midpoints of linear and circular arc segments defining paths, polygons, and standard shape symbols.
	**Intersection Snap**	Snap to intersections between linear and circular arc segments defining paths and polygons.
	**Reference Point Snap**	Snap to control points of standard symbols (four corners and center) as well as endpoints of path and polygon shapes.

Alignment Controls (2D)

	Restrict Nothing	No alignment restrictions during editing or measurement.
	**Restrict Horizontally**	Restrict to horizontal alignment. May be used in both editing and measurement modes.
	**Restrict Vertically**	Restrict to vertical alignment. May be used in both editing and measurement modes.
	**Restrict Orthogonally**	Restrict to horizontal or vertical alignment. May be used in both editing and measurement modes.

Measurement Tools (2D)

	Distance Point to Point	Point to point distance measurement. The measurement tool can be initiated regardless of the selection state. The tool context text will prompt for alternating selection of first and second points for the linear distance measurement. The tool will remain active until terminated with the '*Cancel*' context action or by initiating another tool. The snap and alignment controls may be used in combination with the measurement tool to restrict measurement to targeted points and/or dimensions.
	Distance Entity to Point	Entity to point distance measurement. The measurement tool can only be initiated with a single selected entity. Once initiated, the tool will report distance from the entity to the current cursor position. The tool may be terminated through the 'Cancel' context action or through a selection action.
	Distance Entity to Entity	Entity to entity distance measurement. The measurement tool can only be initiated with a single selected entity. Once initiated, the tool context text will prompt for selection of a second entity. Distance to multiple entities can be measured. The tool may be terminated through the '*Cancel*' context action or through a selection action.

Toolbar (2D)

The Toolbar (2D) is shown below. It's default position is at the upper-left corner of the main window, but it can be used as a floating toolbar, as well as docked on the top, bottom, left, and right edges of the main window. To move the toolbar, drag the left edge of the toolbar with a left-click mouse action.

The toolbar actions are as follows:

	Select	Limit selection to the object type specified in the drop-down. Only applies to graphical selection through the 2D View context. The 'Auto' option will enable selection of all applicable types. Equivalent to the Selection Filter. See also the Select Other_context menu action.
	Move Entity	Move a group of selected entities. Enter a reference point and end point as prompted. Note the tool context menu and context options.
	Rotate Entity	Rotate a group of selected entities. Enter a rotation point and angle as prompted. Note the tool context menu and context options.
	Copy Entity	Copy a group of selected entities to the clipboard.
	Paste Entity	Paste entities from the clipboard into the current editing context. Specify an insertion point as prompted.
	Delete Entity	Delete the currently selected entities.
	Add Shape	Add a rectangular shape to the current editing context by specifying two corner points as prompted. Use the properties context action to change the initial rectangle to an alternate standard shape.
	Add Shape	Add a rectangular shape to the current editing context by specifying a center point and a corner points as prompted. Use the properties context action to change the initial rectangle to an alternate standard shape.
	Add Text	Add a text object to the current editing context.
	Draw Path	Add a path object to the current editing context using the Path Tool.
	Draw Polygon	Add a polygon object to the current editing context using the Polygon Tool.
	Modify Corners	Round, chamfer, or miter corners with the Corner Tool.

	Buffer	Buffer the currently selected object.
	Union	Union the currently selected objects. Only available when more than one shape from the same context is selected. Replaces the selected shapes with their geometric union.
	Intersection	Intersect the currently selected objects. Only available when more than one shape from the same context is selected. Replaces the selected shapes with their geometric intersection.
	Subtraction	Perform a geometric subtraction. Available when one or more shapes from the same context is selected. The shapes selected prior to launching the subtraction tool form the basis for the subtraction. Select additional shapes as prompted to subtract from the basis. The basis shapes are modified by the operation.
	Boolean Mode	Set the active mode for boolean operations. The default 'Automatic' mode will preserve circular arcs during boolean operations. Setting the boolean mode to 'Discrete' will result in circular arcs being discretized during the operation. The resulting shapes will be composed only of linear segments.
	Horizontal Guide	Create a horizontal guide to be used in drafting or measurement operations. Entry mode can be switched between 'From Center' and 'From Endpoints' using tool context actions.
	Vertical Guide	Create a vertical guide to be used in drafting or measurement operations. Entry mode can be switched between 'From Center' and 'From Endpoints' using tool context actions.
	Horizontal Symmetric Dimension	Create a horizontal symmetric dimension to be used in drafting or measurement operations. Specify the entry points as prompted. The value can be entered directly using the *Set Dimension* context action or edited after dimension is created using the *Properties* attribute in the *Details* widget.

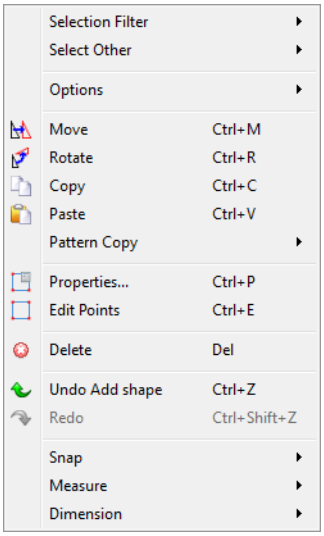
	Horizontal Relative Dimension	Create a horizontal relative dimension to be used in drafting or measurement operations. Specify the entry points as prompted. The value can be entered directly using the *Set Dimension* context action or edited after dimension is created using the *Properties* attribute in the *Details* widget.
	Vertical Symmetric Dimension	Create a vertical symmetric dimension to be used in drafting or measurement operations. Specify the entry points as prompted. The value can be entered directly using the *Set Dimension* context action or edited after dimension is created through the *Properties* attribute in the *Details* widget.
	Vertical Relative Dimension	Create a vertical relative dimension to be used in drafting or measurement operations. Specify the entry points as prompted. The value can be entered directly using the *Set Dimension* context action or edited after dimension is created through the *Properties* attribute in the *Details* widget.
	Diagonal Symmetric Dimension	Create a diagonal symmetric dimension to be used in drafting or measurement operations. Specify the entry points as prompted. By default, the orientation (angle) of the dimension will snap to a fixed interval. The snap internal may be set after the first point is specified using the 'Set Angle Snap...' context action. In addition the orientation may be toggled between the snap and lock mode using the 'Angle Snap' and 'Angle Lock' toggles through the context menu. A specific orientation may be provided through the 'Set Angle Lock...' context action when in lock mode. The value can be entered directly using the 'Set Dimension...' context action or edited after dimension is created through the 'Properties' attibute in the Details widget.

	Diagonal Relative Dimension	Create a diamgonal relative dimension to be used in drafting or measurement operations. Specify the entry points as prompted. By default, the orientation (angle) of the dimension will snap to a fixed interval. The snap internal may be set after the first point is specified using the 'Set Angle Snap...' context action. In addition the orientation may be toggled between the snap and lock mode using the 'Angle Snap' and 'Angle Lock' toggles through the context menu. A specific orientation may be provided through the 'Set Angle Lock...' context action when in lock mode. The value can be entered directly using the 'Set Dimension...' context action or edited after dimension is created through the 'Properties' attibute in the Details widget.
	Linear Naming	Apply sequential naming to the selected pins using the Linear Naming Tool.
	Array Naming	Apply grid-style naming to the selected pins using the Array Naming Tool.
	Radial Naming	Apply radial naming to the selected pins using the Radial Naming Tool.
	Sequential Naming	Apply sequential naming based on user picks using the Sequential Naming Tool.

Context Menu (2D)

The Context Menu is available in the Footprint (2D) graphical context through a right-click mouse action. The actions available through the context menu are based on the current selection and active tool. The top-level context menu provides short-cuts to many of the toolbar actions and snap, alignment, and measurement tools. Certain actions are available solely through the top-level context menu, as described below. Keyboard short-cuts are available for many context actions. Note that the Footprint (2D) graphical contact must be in focus for the keyboard short-cuts to be active.

The top-level context menu:



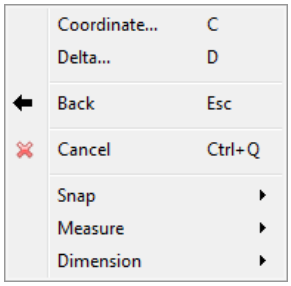
The actions below are available solely through the top-level context menu:

Selection Filter	Limit selection to the specified object type. Only applies to graphical selection through the Footprint (2D) view. The 'Auto' option enables selection of all applicable types.
Select Other	Available only when there is an active selection. Provides a list of alternate selection entities based on the prior selection. If applied, the current selection will be replaced with the alternate.
Options	Options available when moving and copying applicable entities. Enable Update Solids to move and/or copy associated 3D shapes with the feature. Update Padstacks will move and/or copy associated padstack placements, and Update Connect Points will move and/or copy associated connect points.
Pattern Copy	Pattern copy may be applied to one or more actively selected entities. For both the Linear Pattern and the Grid Pattern, the patterned instances are generated based on incremental offsets from the position of the lead entity. The Radial Pattern generates a full-circle radial pattern based on a specified center point and a count.
Properties	Edit the properties of a standard shape. Can be used to change both the shape type and parameters. Launches the Edit Properties dialog.
Edit Points	Edit the reference points of an existing shape.

Undo	Undo the last action. The context menu prompt will indicate the name of the last action.
Redo	Redo the last action. The context menu prompt will indicate the name of the last action.

Tool Context Menus

Many of the editing tools provide additional context menu actions while the tool is active. The example below shows the context menu actions available from the *'Move'* tool:



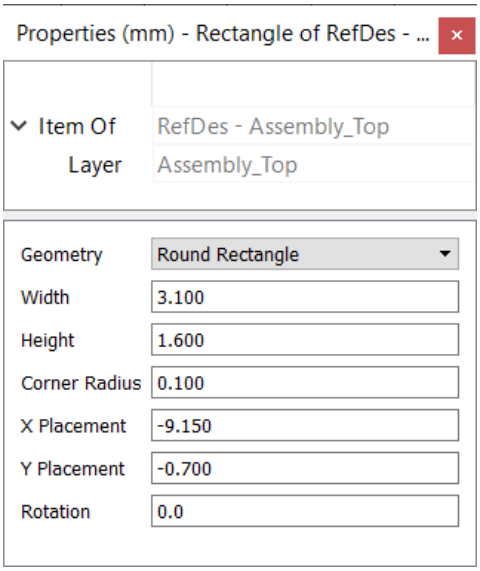
The *'Coordinate'* and *'Delta'* actions described below are common to several tools and provide useful alternatives to graphical selection in certain instances.

Coordinate...	The 'Coordinate' context action enables direct entry of (x,y) coordinate values in place o the next reference point requested by the tool.
Delta...	The 'Delta' context action enables direct entry of an (x,y) delta (relative) offset. Available 'Move' in which a relative position is applicable.

Properties (Standard Shape)

The *Properties* dialog enables direct editing of standard shapes. After selecting a standard shape, the *Properties* dialog is launched through the *Tools - Properties* menu. You can also change the shape type from the *Geometry* drop-down list and the properties of the

selected shape from this dialog. This dialog is not available for non-standard (polygon) shapes.

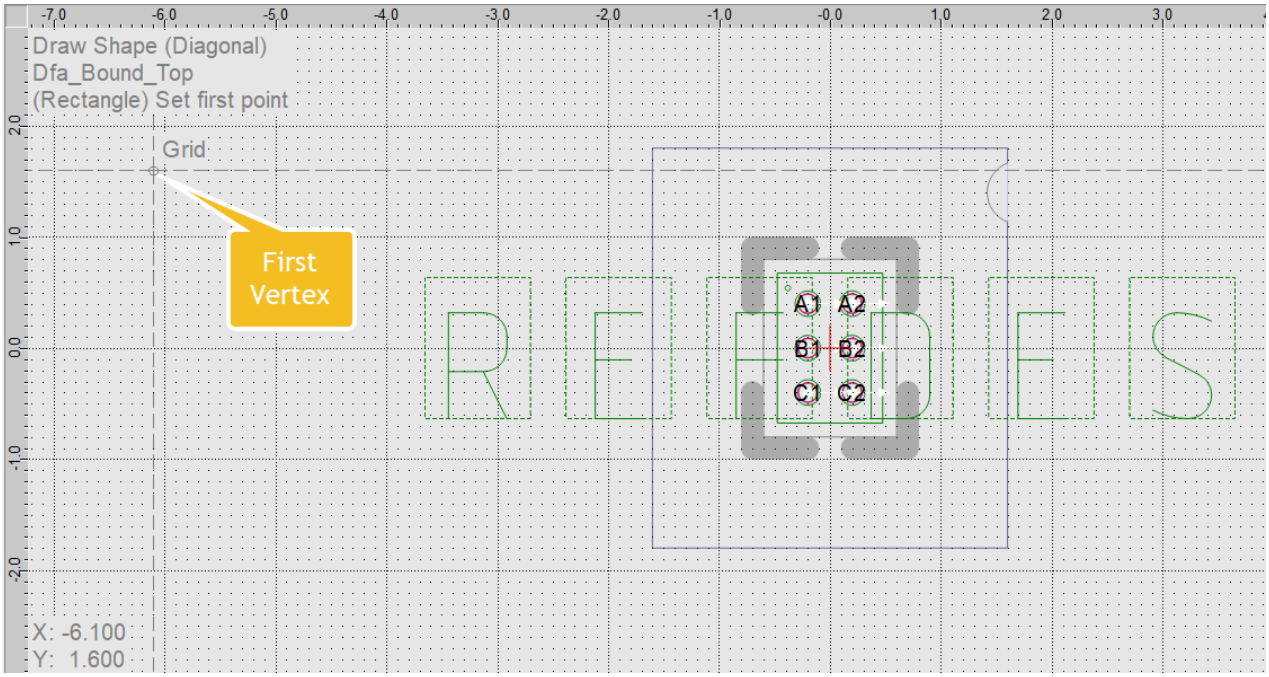


Adding Shapes

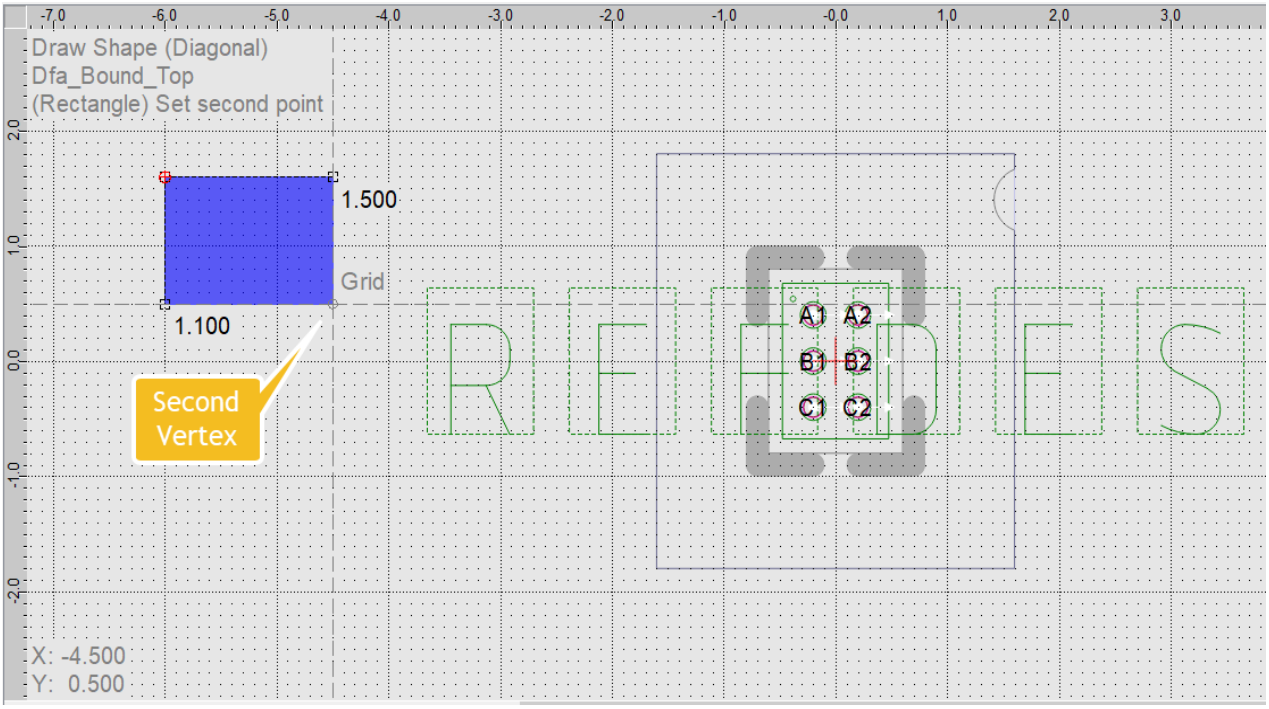
Library Creator allows you to easily add shapes to the canvas, such as a square, rectangle, polygon or circle. To add a shape:

1. Click on the toolbar.
2. (Optional) To change the shape type to *Circle*, right-click anywhere on the canvas and select *Circle*.

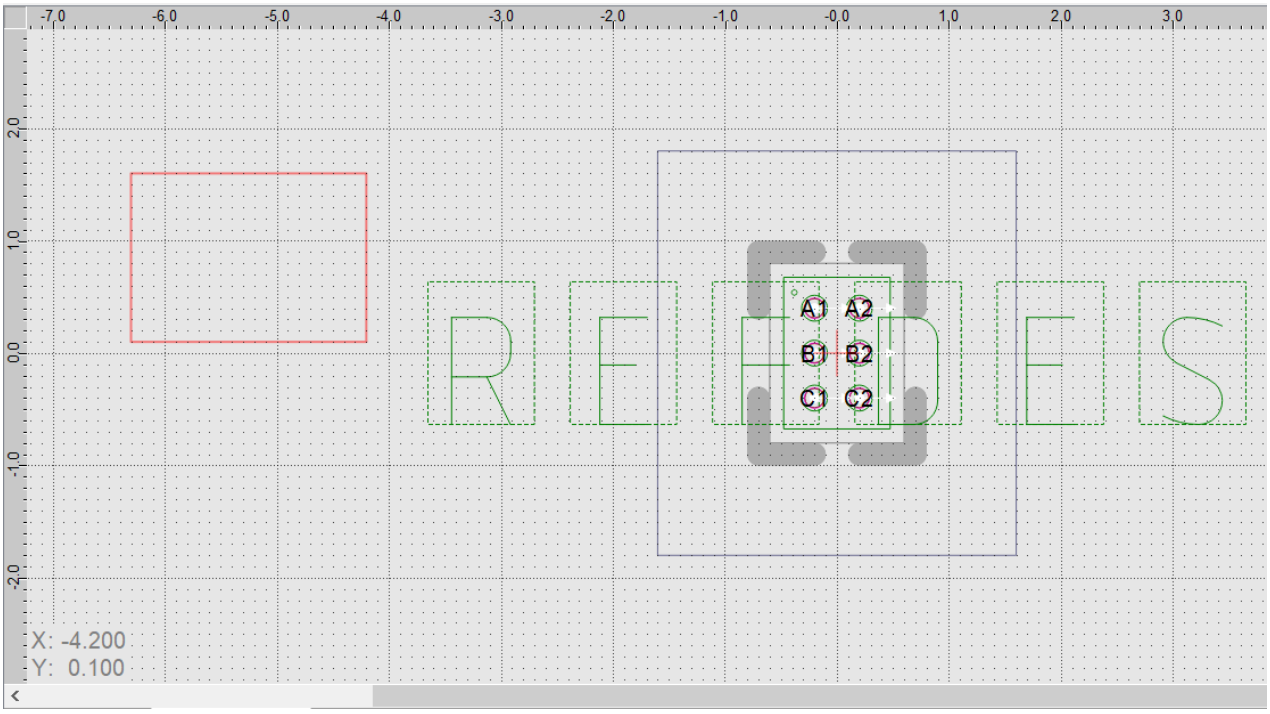
3. Click anywhere on the canvas to place the first vertex of the shape.



4. Drag the mouse and click to place the second vertex.



The shape is created.

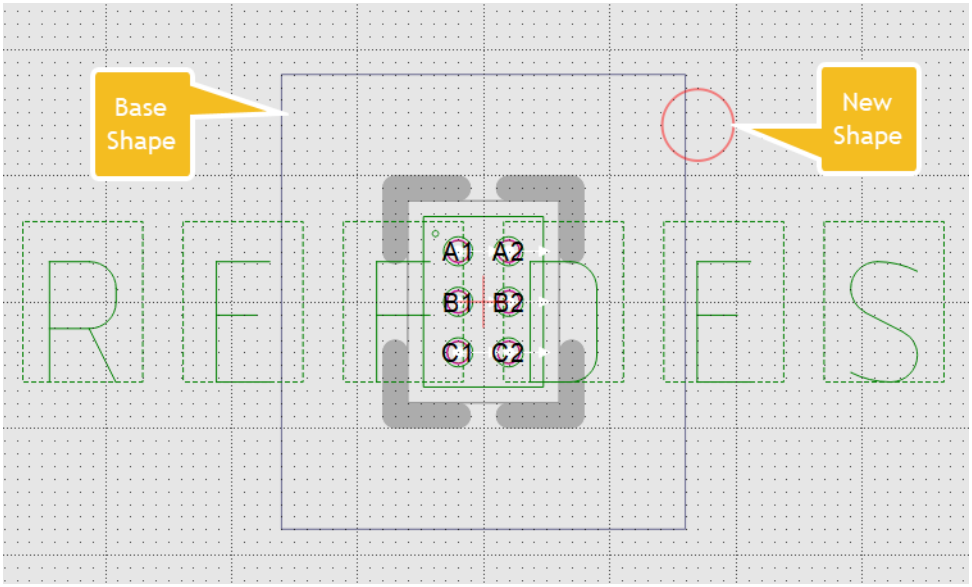


Subtracting Shapes

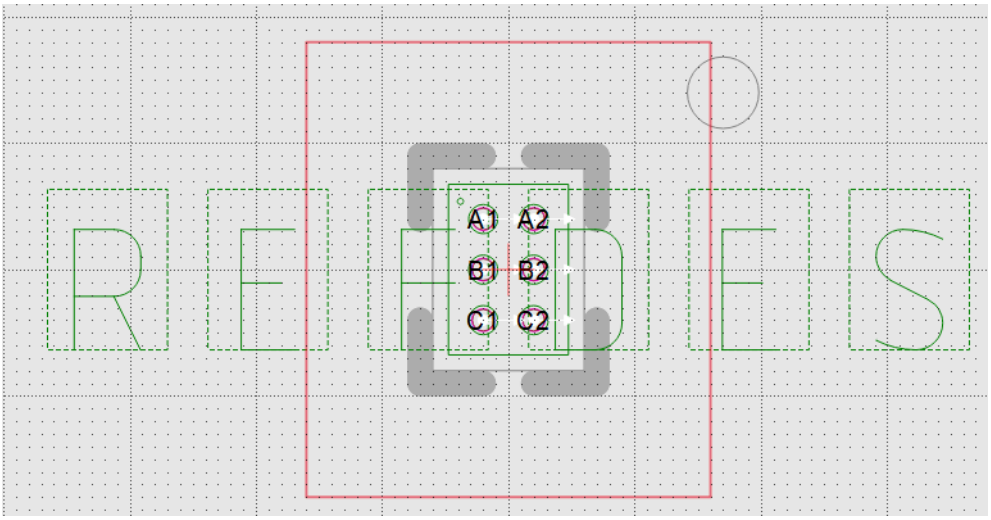
You can subtract one or more shapes from another shape in Library Creator. Subtraction of shapes is possible when shapes from the same context are selected. For example, subtracting a standard shape from another standard shape is supported, but, subtracting a path based shape from a standard shape is not supported.

To subtract shapes:

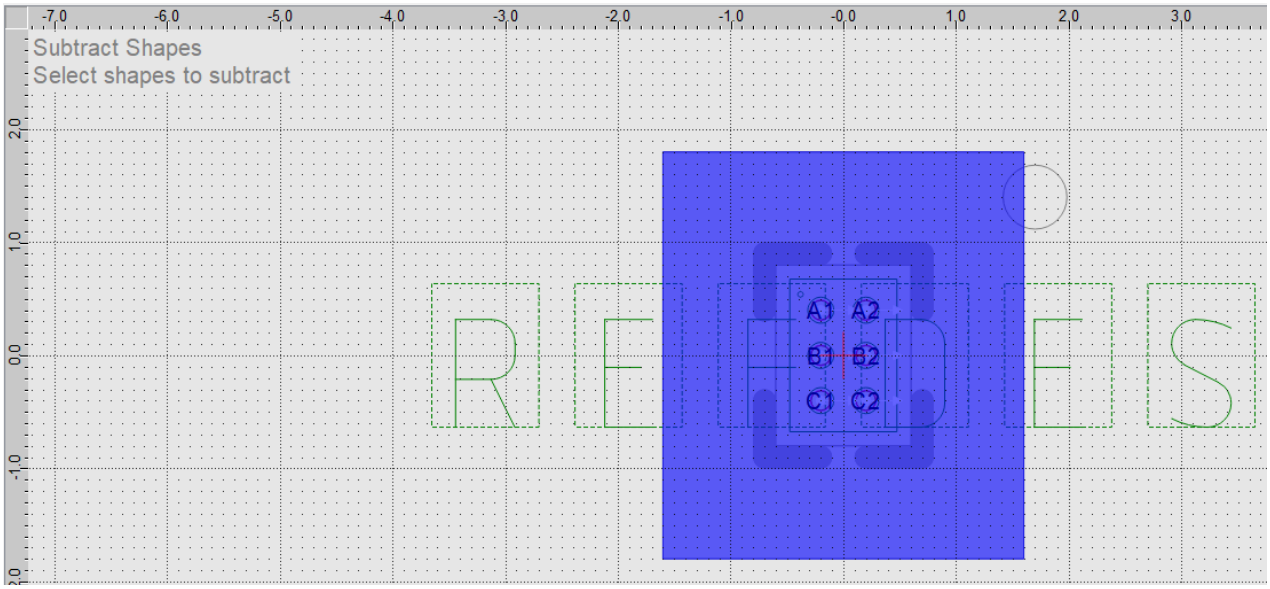
1. Place a new or drag an existing shape onto the base shape from which subtraction is to be done.



2. Select the base shape.

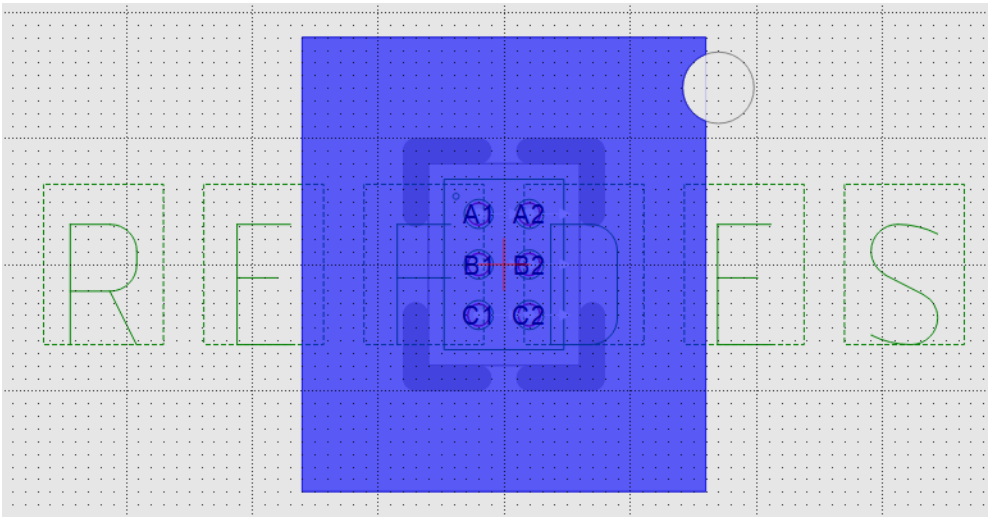


3. Click on the toolbar.



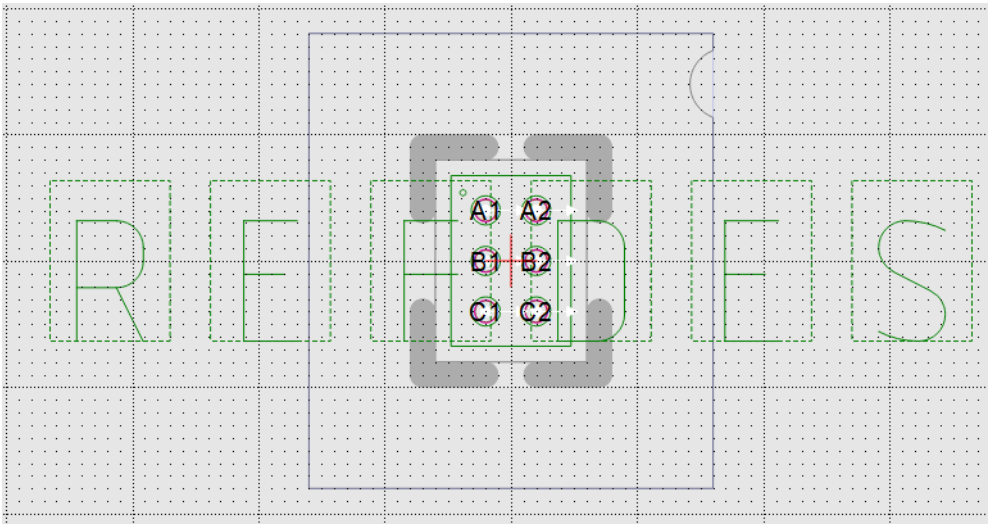
4. Click the new shape.

The overlapping area of the new shape is subtracted from the base shape.



5. Right-click anywhere on the canvas and choose *Accept*.

6. Delete the new shape.



Path Tool

The *Path Tool* adds a new path shape to the current editing context. A path is constructed as a combination of linear and arc segments with a specified width. The current segment type (*Lines* vs. *Arcs*) can be selected through the *Segment* drop-down list.

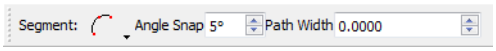
Creating Path Objects using Lines

When composing a path with linear segments, it is also possible to specify a segment join style. The available segment join styles are *Sharp Corners*, *Round Corners*, and *Chamfered Corners*, which can be selected from the *Join* drop-down list. The round join is controlled through the *Max Radius* or *Max Trim* options. The chamfer join is constrained by either a *Max Chamfer Length* or *Max Trim* distance. The chamfer and round join styles will insert a chamfer and/or arc transition between linear segments.

Creating Path Objects using Arcs

When the segment mode is set to *Arcs*, the join style is unavailable. The arc segment is specified by placing the end point and an additional arc point. The angle snap precision can

be used to aid in the construction of arc with a regular sweep. The path construction is terminated with the (*Accept*) or the (*Cancel*) buttons from the toolbar.



Adding a Vertex

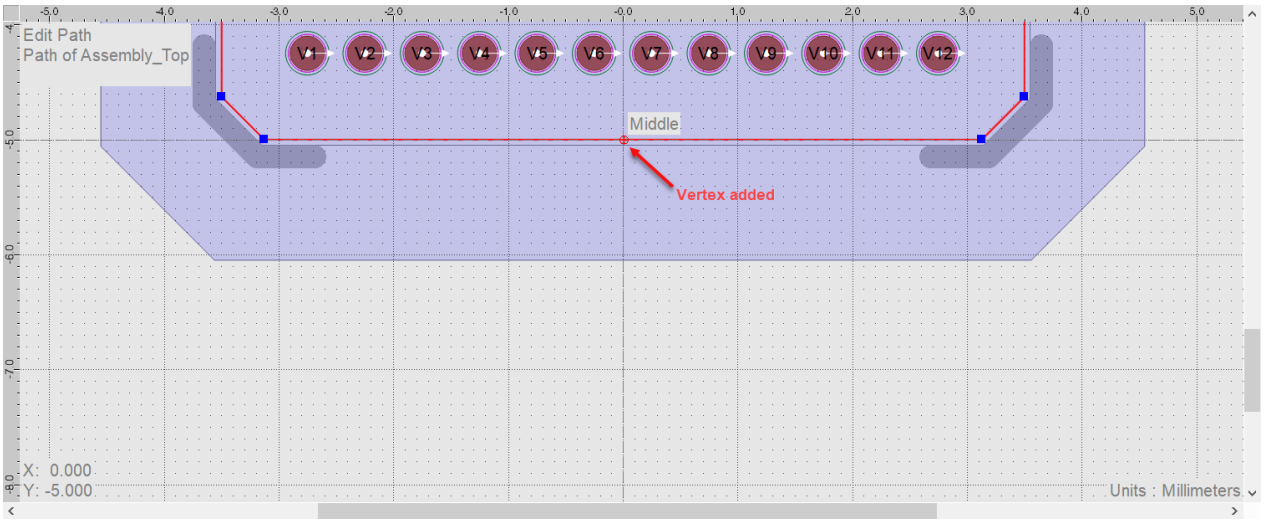
You can also add a vertex to an existing polygon to change its shape. To add a vertex to a polygon:

1. Right-click a polygon and select *Edit Points* or press *Ctrl* + *E*.

   This puts the polygon into edit mode. Moving the mouse over a polygon line in this mode displays the *Insert Vertex* tool tip where you can add a vertex.

2. Click the line of the polygon to which you want to add a vertex.

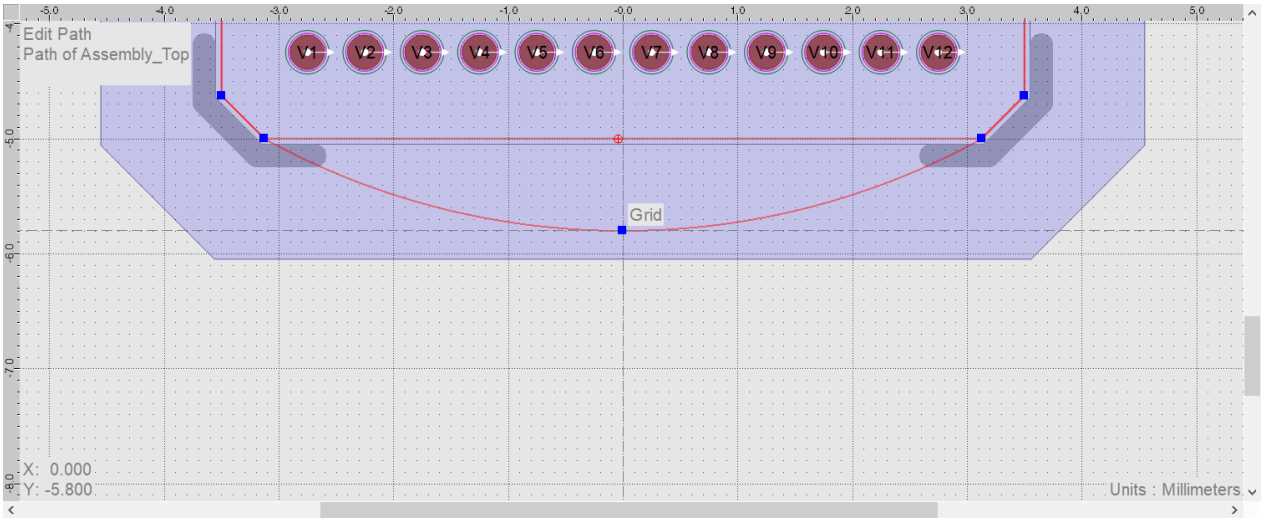
   A vertex is added to the line.



   By default, the vertex segment is lines.

3. Right-click the vertex and select *Insert Arcs*.

4. Drag the mouse pointer till you are finished creating the shape and click once.



5. Click on the toolbar to accept the changes.

Deleting a Vertex

To delete a vertex:

1. Right-click the polygon and select *Edit Points* or press *Ctrl* + *E*.

   This puts the polygon into edit mode and displays all the vertexes added to the shape.

2. Right-click the vertex and choose *Delete*.

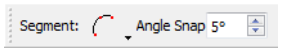
   The vertex is deleted and the shape of the polygon is change to its previous state.

Polygon Tool

The *'Polygon Tool'* adds a new area shape to the current editing context. A polygon is constructed as a combination of linear and arc segments defining the outer boundary of the shape. The current segment type (line vs. arc) can be selected through the toolbar drop-down, the context menu, or the associated keyboard shortcut. When composing a boundary with linear segments, it is also possible to specify a segment join style. The round join is controlled through either a maximum radius or maximum trim distance through the toolbar or context menu options. The chamfer join is constrained by either a maximum chamfer length

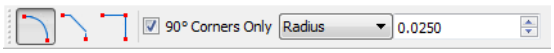
or trim distance. The chamfer and round join styles will insert a chamfer and/or arc transition between linear segments.

When the segment mode is set to arcs, the join style is unavailable. The arc segment is specified by placing the end point and an additional point on the arc. The angle snap precision can be used to aid in the construction of arc with a regular sweep. The polygon construction will terminate automatically on completion of a closed boundary, or alternatively may be cancelled with the *'Cancel'* action from the toolbar or context menu.



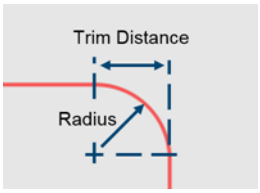
Corner Tool

The *'Corner Tool'* allows modification to path or area shapes through the insertion or remove of miter, round, or chamfered transitions between neighboring linear segments. The current modification mode is selected through the toolbar or context menu. When the *'90 deg. Corners Only'* option is selected, the actions will only be applicable to neighboring linear segments that are perpendicular to each other. The *'Round'* and *'Chamfer'* actions can be applied to either a vertex or an existing transition segment. The *'Miter'* mode can be applied only to an existing transition segment, and will replace the transition with a linear extension of the two neighboring segments.

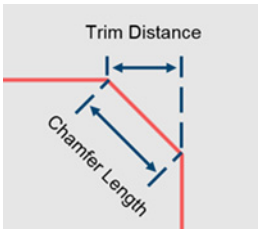


The Round is controlled by specifying either a *Radius* or *Trim Distance* through the toolbar or context menu options. The Chamfer is controlled by specifying either the *Chamfer Length* or Trim Distance as illustrated below.

Round



Round may be controlled by specifying either the Trim Distance or Radius. In the case of a 90 deg. corner, these are equivalent.

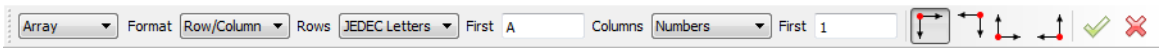
Chamfer		Chamfer may be controlled by specifying either the Trim Distance or Chamfer Length

Linear Naming Tool

The *'Linear Naming Tool'* allows a sequential numbering pattern to be applied to a group of pins. After selecting a group of pins, naming is initiated through the *'Linear Naming'* action on the Toolbar (2D). Various numbering options may be configured through the toolbar. The *'First'* pin in the sequence is indicated with a red box around the pin contact. To change the first pin, select the *'1st'* toolbar action and select the new pin. The available sequence patterns will dynamically update based on the pin layout and first position. Select a sequence pattern to preview of the naming result. The tool is terminated with either the *'Accept'* or *'Cancel'* action.

Array Naming Tool

The *'Array Naming Tool'* allows a grid-based numbering pattern to be applied to a group of pins. After selecting a group of pins, naming is initiated through the *'Array Naming'* action on the Toolbar (2D). Various numbering options may be configured through the toolbar. Pin numbers will update dynamically after selecting a sequence pattern to enable a preview of the naming result. The tool is terminated with either the *'Accept'* or *'Cancel'* action.



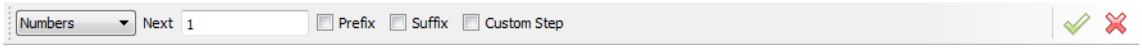
Radial Naming Tool

The *'Radial Naming Tool'* allows a radial numbering pattern to be applied to a group of pins. After selecting a group of pins, naming is initiated through the *'Radial Naming'* action on the Toolbar (2D). Various numbering options may be configured through the toolbar. The *'First'* pin in the sequence is indicated with a red box around the pin contact. To change the first pin, select the *'1st'* toolbar action and select the new pin. To move the center point for the radial sequence pattern, select the origin action from the toolbar and pick a new center pint. Select

either the clockwise or counter-clockwise sequence pattern to preview the naming result. The tool is terminated with either the *'Accept'* or *'Cancel'* action.

Sequential Naming Tool

The *'Sequential Naming Tool'* allows a sequential naming pattern to be applied to a series of pins based on user selection. Various naming options may be configured through the toolbar drop-downs and options. The *'Next'* name to be applied is displayed in the toolbar and may be reset manually if desired. The subsequent pick will apply the current *'Next'* value and increment to the next value in the sequence. The naming may be terminated with either the *'Accept'* or *'Cancel'* action.



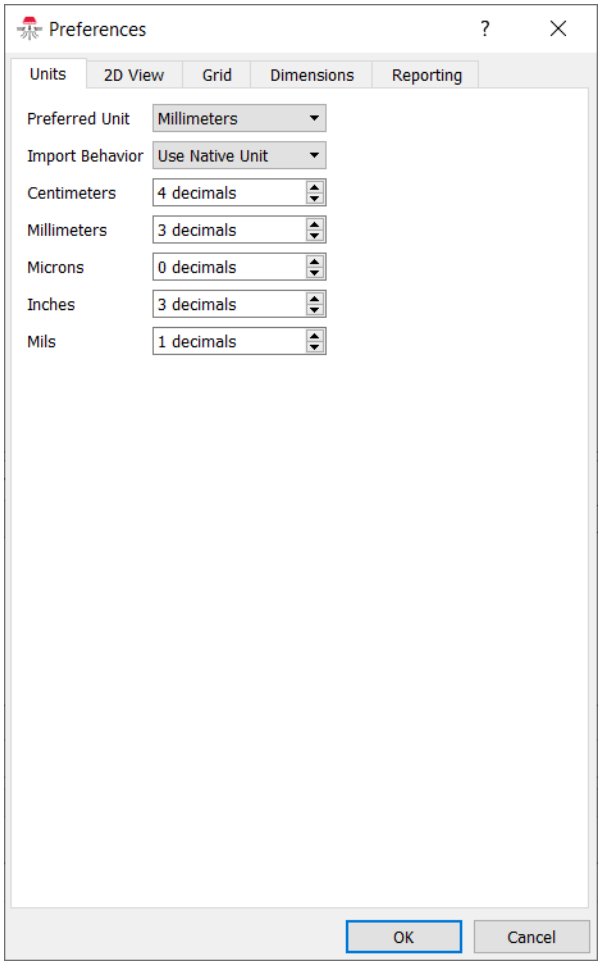
Preferences

Local application settings related to units, the 2d view and the grid are configured through the Preferences dialog. The Preferences dialog can be accessed through the *Tools - Preferences* application menu or through the gear icon below the measurement tools in the 2D View.

Unit Preferences

The preferred unit system, the import behavior, and the displayed decimal precision may be configured through the *'Units'* tab of the preferences dialog (Tools->Preferences) shown below. The preferred unit controls the display of dimensional index parameters in the repository search view as well as the default unit when creating a new package using the *'File->New Package'* menu action. The *'Import Behavior'* drop-down enables specification of whether imported packages should be converted to the selected preferred units or maintained in their original native unit. This preference applies to both imported STEP models as well as existing repository packages. For each of the five defined units, *Centimeters*,

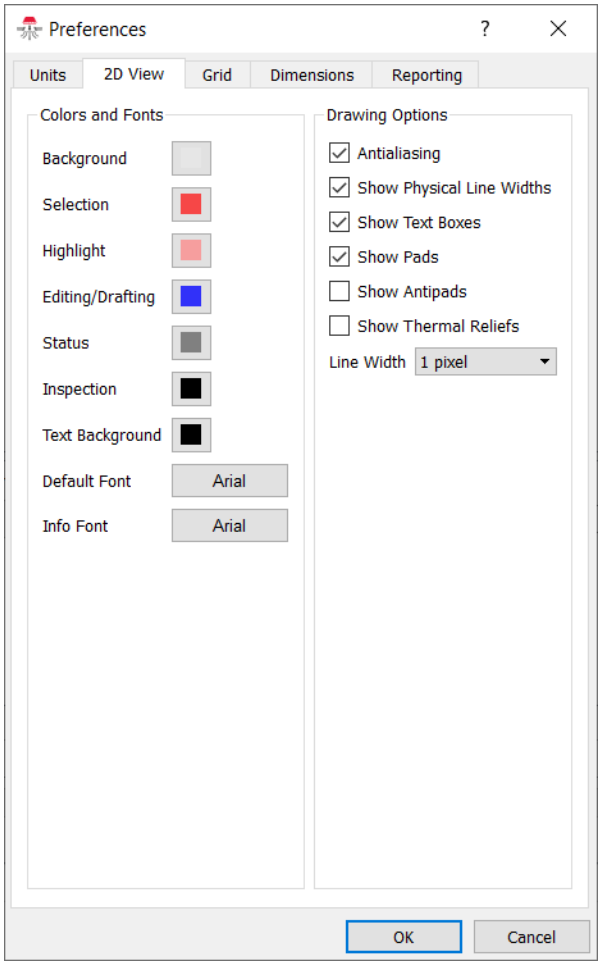
Millimeters, *Microns*, *Inches*, and *Mils*, the number of decimal digits to display can be configured through the *Unit* tab of the *Preferences* dialog.



2D View Preferences

Display preferences related to the 2D view may be configured through the *'2D View'* tab of the preferences dialog (Tools->Preferences) as shown below. It is important to note that these local settings reflect only display preferences and do not impact the model itself. Through the

dialog, local settings related to layer color and transparency, display fonts, and a variety of additional display preferences may be configured.



Grid Preferences

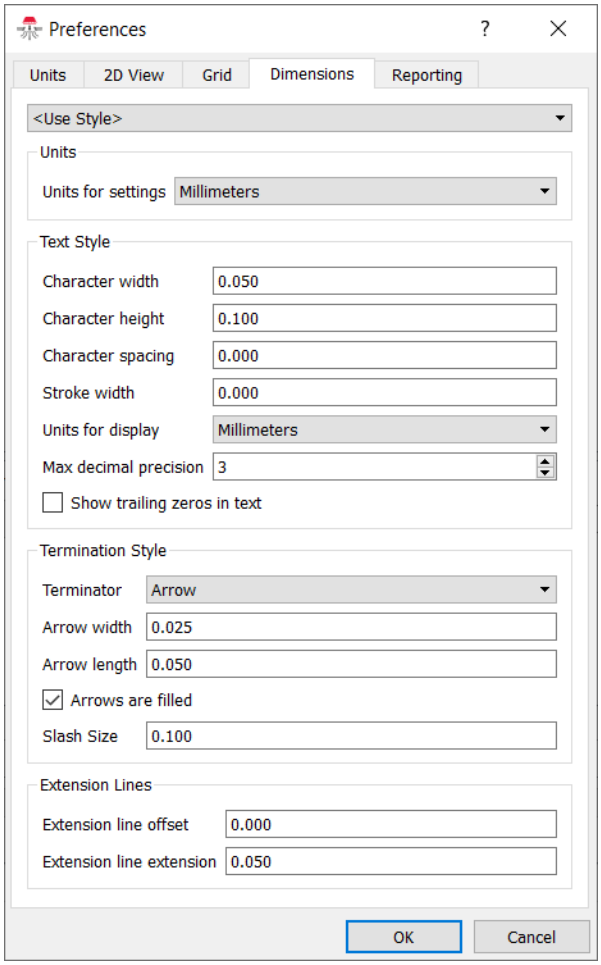
The local user preferences related to the 2D grid may be configured through the *'Grid'* tab of the preferences dialog (Tools->Preferences). Note that the grid settings are configured for the current package unit. The grid settings for the current package unit are configurable through

the dialog, however, individual settings are stored and maintained for each of the five available units.

Dimension Preferences

The dimension unit, style, and formatting settings may be configured through the 'Dimensions' tab of the preferences dialog (Tools->Preferences). The preferenes settings will

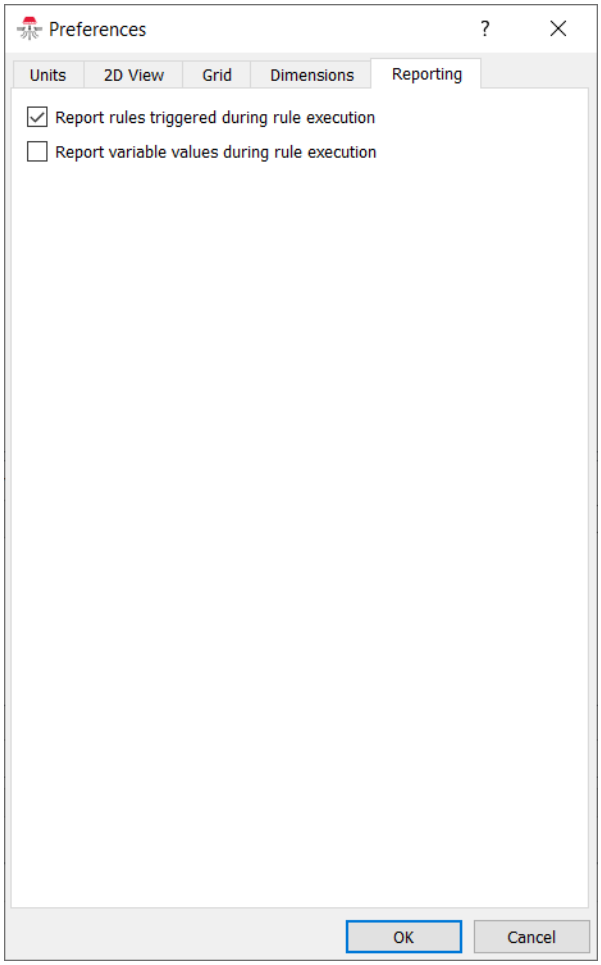
apply for newly created dimensions. Changing the preferences will not impact existing dimensions.



Reporting Preferences

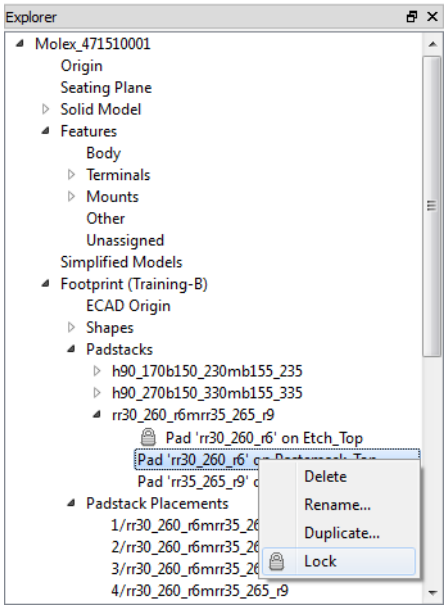
The reporting preferences enable or disable the generation of certain messages during the application of rules. Reporting the specific rules triggered and variable values during rule execution can be valuable in validating and debugging rule implementations. In production

use, it may be desired to disable one or both of these options to reduce the number of messages generated.



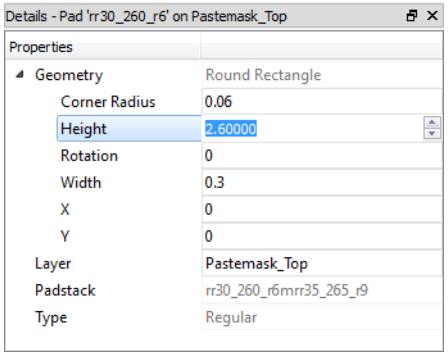
Explorer

The Explorer provides an interface for inspecting, manipulating, and editing elements of the package model and associated footprint (when present). Many actions are available on applicable elements of the package model through right-click context menu.



Details

The *'Details'* widget displays properties associated with a selected entity. Properties will be displayed only when a single entity is selected. Certain properties can be directly edited in the properties table. The editor is activated for a given property by double-clicking on the value field.



Message Console

The Message Console displays error, warning, and information messages generated by the application. The most recently generated messages appear at the top of the console display. Visibility of error, warning, and information messages can be toggled through the corresponding check boxes. Toggling the display of information messages may often be helpful to allow focus on warnings and errors. Because messages accumulate during the course of application run, the *'Clear Messages'* action (to the right of Info check box) may be used to clear the console messages prior to executing a new command in order to focus on only newly generated messages.