



# Ciranova Python API Version 4.0 Code Conversion Guidelines

Copyright © 2006-2007 Ciranova, Inc.  
<http://www.ciranova.com>

Doc name:	Code Conversion 4.0
Doc rev:	3
Software rev:	4.0.1
Doc date:	Jan 31, 2007

# CONTENTS

Introduction.....	3
Existing Python API Class Changes.....	3
ParamArray class .....	3
Point class .....	3
Range class.....	3
Box class .....	4
Grid class .....	4
Bar class .....	4
Pin class .....	4
Route class .....	5
Instance class .....	5
InstanceArray class .....	5
Contact class .....	6
ShapeFilter class .....	6
Tech class.....	6
Layer class .....	6
DloGen class .....	6
PhysicalComponent class.....	7
Parameter changes for FG methods (ShapeFilter).....	7
Existing Python API Global Function Changes.....	8

# Ciranova Python API Version 4.0 Code Conversion Guidelines

## Introduction

Ciranova has incorporated many enhancements to the Ciranova Python API for the F406 release. These include both new additions, which provide improved functionality, and also changes to existing classes and methods. The latter changes will require updates to existing PyCell Python code before it will work with the F406 release.

This document contains a brief description of these incompatible changes between the F306 and F406 releases, along with suggestions for how to update existing code written with release F306. Although no attempt is being made to encode this information in the form of conversion scripts for existing PyCell Python source code, this listing of class method changes should still be used as a guideline for manual conversion.

## Existing Python API Class Changes

### ParamArray class

The "attribute style" syntax for the ParamArray class is no longer supported, as it can cause subtle bugs when parameter values are accessed. This "attribute style" syntax is of the form "params.value", such as "params.length" or "params.width". Instead, the "dictionary style" syntax should be used: "params['length']" or "params['width']".

### Point class

Several new methods were added, including place(), getSpacing(). The snap() method has been replaced by 4 methods: snap(), snapX(), snapY() and snapTowards(). These snap() methods use a grid parameter from the updated Grid class.

### Range class

The overloaded merge() method has been replaced by two separate methods: merge() and mergeCoord(). If the merge() method was being used with a coordinate value, the mergeCoord() method should be used instead. This was done to allow Python keyword parameters to be used with the merge() method.

## **Box class**

The overloaded `merge()` and `contains()` methods have been replaced by separate methods: `merge()`, `mergePoint()`, and `contains()`, `containsPoint()`. If the `merge()` or `contains()` method was being used with a point value, then the `mergePoint()` or `containsPoint()` method should be used instead. This was done to allow Python keyword parameters to be used with the `merge()` and `contains()` methods.

## **Grid class**

The Grid class has been improved in several ways. The creation method has been changed, so that it does not depend upon the Tech class; it also allows for different X and Y coordinate values. The existing `snapType()` method should be changed to use the new SnapType class values, instead of the existing Grid UP, DOWN, NEAREST, and ROUND\_UP enumerated values.

There are also several new methods for this class: `getSize()`, `getXSize()`, `getYSize()`, `getSnapType()`, `setSnapType()`, `setSize`, `setXSize()`, `setYSize()`.

## **Bar class**

The `addToPin()` method for the Bar class has been removed; this is now handled by the `addShape()` method for the Pin class. In addition, the new methods `getRect()`, `extendTo()` and `getRoutePathIntersectBox()` were added to the Bar class.

Also note that the Bar class no longer automatically merges contacts, since the merged contacts did not have the same size as the original contacts. If it is desired to have merged contacts, then this will need to be done by the user.

## **Pin class**

The `addInstPin()` method was added to the Pin class; this method can be used to replace the DloGen `addToPin()` method, which has been removed (see DloGen class section).

## **Route class**

There have been several changes to the Route class, which has been renamed to the RoutePath class to better reflect this improved functionality. These changes are summarized as follows:

(1) Instead of a single straightLine() method with an optional Bar parameter, there is now a separate method to route to a Bar, the StraightLineToBar() method, in addition to the existing StraightLine() method.

(2) All method calls using "Route" should now use "RoutePath"; for example, "Route.ZShape()" should be renamed "RoutePath.ZShape()".

(3) The position parameter for the Route LShape() and ZShape() methods has been changed from being of type double to being of type Point; this allows more precision in the alignment of the center of the Route to a known reference.

(4) Contact generation is now optional; contacts used to be automatically generated. There is now a "genContact" parameter to control the automatic generation of contacts by methods in the RoutePath class. This allows users to use their own contacts, if desired.

Note that in addition to LShape and ZShape, a CShape is now provided; this is very useful when creating routing for multi-fingered interdigitated device layout.

## **Instance class**

The firstInstPin() and instPin() methods for the Instance class were replaced with a single method, the findInstPin() method. This was changed since these two methods were used for iteration, which is no longer handled in this manner. Note that the findInstTerm() method was also added to the Instance class.

## **InstanceArray class**

The creation method for the InstanceArray class was changed, so that there are no longer any default values for the *dx*, *dy*, *numRows* and *numCols* parameters. It is now necessary to provide specific values for each of these different parameters for the creation method.

## **Contact class**

The stretchTowards() method for the Contact class was renamed to stretchToCoord(), since it is an absolute position, instead of a relative offset (as used by the PhysicalComponent moveTowards() method). The name stretchTowards() will be reserved for a future method for the Contact class which uses relative offset.

## **ShapeFilter class**

There is a new creation method for the ShapeFilter class; this was done to avoid problems with overloaded creation methods in Python (keyword parameters can not be used). In addition, several methods were added to this class, to allow existing ShapeFilter objects to be modified; this was not possible in the F306 release. These methods are the new include(), exclude() and isIncluded() methods.

## **Tech class**

The getGridResolution() method for the Tech class no longer accepts a layer parameter; instead, it returns the default manufacturing grid value, which is the value used when no layer specific grid resolution value has been defined.

## **Layer class**

The Layer class has a new method, getGridResolution() to return any grid resolution which has been defined for the particular layer. If no layer specific grid resolution has been defined for this layer, then the default manufacturing grid value used for all layers will be returned instead.

## **DloGen class**

The three addToPin() methods for the DloGen class have been removed, as has the addToPin() method for the Bar and MultiPath classes. Instead, the DloGen addPin() method, as well as the addShape() and addInstPin() methods for the Pin class should be used for this purpose.

A large number of methods for the DloGen class have been moved into the PhysicalComponent class. These methods operated on one or more physical components, and had no need to access DloGen design data directly. In addition, these methods are more naturally contained in the PhysicalComponent class, in terms of basic OOP inheritance principles. These methods include the following:

abut(), alignEdge(), alignEdgeToPint(), alignLocation(), alignLocationToPoint(), fgAbut(), fgAddEnclosingPolygon(), fgAddEnclosingRects(), fgAnd(), fgFill(), fgMinSpacing(), fgNot(), fgOr(), fgPlace(), fgSize(), fgXor(), getSpacing(), place()

### **PhysicalComponent class**

Several methods related to physical components have been moved from the DloGen class to the PhysicalComponent class. This was done for several reasons, as described in the section on the DloGen class. These methods include the following:

abut(), alignEdge(), alignEdgeToPint(), alignLocation(), alignLocationToPoint(), fgAbut(), fgAddEnclosingPolygon(), fgAddEnclosingRects(), fgAnd(), fgFill(), fgMinSpacing(), fgNot(), fgOr(), fgPlace(), fgSize(), fgXor(), getSpacing(), place()

Note that when these methods are used as methods for the PhysicalComponent class, the parameters will be different from their use as DloGen methods; namely, the parameters referring to physical components will be changed, since the first physical component will not be necessary (since it refers to the "self" physical component). However, many of these methods will also exist as global functions, in the "cni.geo" module; these global functions will have the same parameters as were used for the DloGen class version.

Several parameter names for the above methods have also been changed to be more consistent. Namely, the following convention is now being used:

Physical component: comp  
Direction: dir, refDir (versus referenceDir)  
Location: loc, refLoc (versus referenceLoc)  
Physical component: refComp (versus reference)  
ShapeFilter: refFilter (versus referenceFilter)

In addition, the handling of parameters has been changed for several of these methods; in the case where the physical component and the reference component are the same, an exception will now be raised. Previously, this was allowed and had the affect of moving the physical component. This change affects the abut(), alignEdge(), alignLocation(), place(), fgAbut() and fgPlace() methods.

### **Parameter changes for FG methods (ShapeFilter)**

FG functions now have a second ShapeFilter parameter for the reference component, as well as the component being moved; the fgPlace() and fgMinSpacing() methods are impacted. In addition, an optional align parameter flag was added.

## Existing Python API Global Function Changes

There have been several changes to the global functions in the Ciranova Python API, which are summarized as follows:

- (1) The `fgDrc()` method for the `DloGen` class is now a global function, `fgDrc()`.
- (2) The global function `getMinSpacing()` was removed; instead use the `getSpacing()` or `fgMinSpacing()` methods for the `PhysicalComponent` class.
- (3) The global function `genPolygonCircle()` was removed; instead use the `genPolygonPoints()` method for the `Ellipse` class.
- (4) The global utility function `roundOff()` was removed, since it only worked for floating-point database units, and not for integral user units; calculations are best performed directly in user units, rather than in database units.
- (5) Also, the `snapGrid()` global utility function was removed; instead use the updated `Grid` class and the new `SnapType` class methods.