



Walt Disney World Swan and Dolphin Resort
Orlando, Florida

Advanced Topics for Dynamic Blocks in AutoCAD® 2006

John Beltran - Autodesk

and K.B. Lee (Assistant); Heinz Baumann (Assistant)

GD41-1L This hands-on lab provides AutoCAD users who create, maintain, or use AutoCAD block libraries a complete introduction to the dynamic blocks technology introduced in AutoCAD 2006. Understand how dynamic blocks are an extension, and not a replacement, for blocks in earlier AutoCAD releases. Become familiar with the Block Authoring mode of AutoCAD 2006, and how to use it to create AutoCAD blocks with custom properties and grips that allow one block to be drawn many different ways. Learn how to create blocks that automatically generate part numbers, or have multiple views. Understand the power of dynamic blocks, how to incorporate them into your existing block libraries, and how your existing customizations can be enhanced to work with dynamic blocks. To take full advantage of this lab attendees should either have taken the "Hands-On Introduction to Dynamic Blocks in AutoCAD 2006" or should have some familiarity with dynamic blocks.

About the Speaker:

John has worked for 12 years in the AutoCAD development team at Autodesk. Most recently he was one of the lead engineers behind on the Dynamic Blocks feature in AutoCAD 2006, but has worked on many AutoCAD features, including plotting and the Properties Palette. He started using AutoCAD with Release 12 in 1991 while he was working for an Autodesk developer before joining Autodesk. John holds a Master of Science degree in Structural Mechanics from the University of California at Los Angeles.

Creating Blocks that Have Alignment Grips and Specifying Insertion Point Cycling

You can add alignment parameters to dynamic blocks. Alignment parameters define grips that align the block reference to other geometry in your drawing. Alignment parameters do not define custom properties, cannot have associated actions, and always act on the entire block reference rather than on objects within the block.



When adding an alignment parameter to a block definition, you must define an alignment axis and alignment type (perpendicular or tangent). When moving a dynamic block using an alignment grip and the grip passes over other geometry in the block, AutoCAD rotates the entire block about the grip so that the alignment axis is either perpendicular or tangent to the geometry under the grip. The alignment grip also snaps to the nearest point on the underlying geometry.

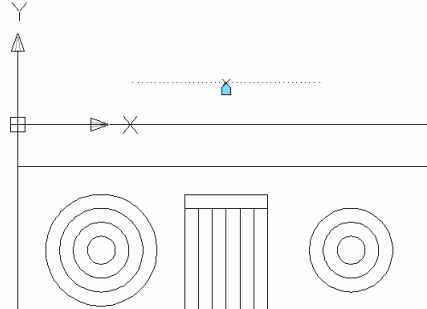
Dynamic blocks can have multiple insertion points, which you can cycle through by pressing the CTRL key when inserting the block reference. All dynamic block grips have a Cycling property. When this property is set to Yes, it can act as an insertion point for the block and can be cycled to while inserting the block.

This Cycling property is set to Yes by default for new grips on point, linear, XY, polar, and alignment parameters. You can change the property for any grip and define the order in which AutoCAD cycles to them using the Insertion Cycling Order dialog box (BCYCLEORDER command).

Lesson 1 Creating Blocks that Have Alignment Grips and Specifying Insertion Point Cycling

Objective: Learn how to use the alignment parameter to align blocks to other geometry. Learn about specifying insertion point cycling and the Insertion Cycling Order dialog box (BCYCLEORDER command).

1. Open the sample drawing *Wall Base.dwg*.
2. Open the “Range” block.
 - Enter **bedit** on the command line to display the Edit Block Definition dialog box.
 - In the Edit Block Definition dialog box, select the “Range” block from the list.
 - Click OK to close the dialog box and open the Block Editor.



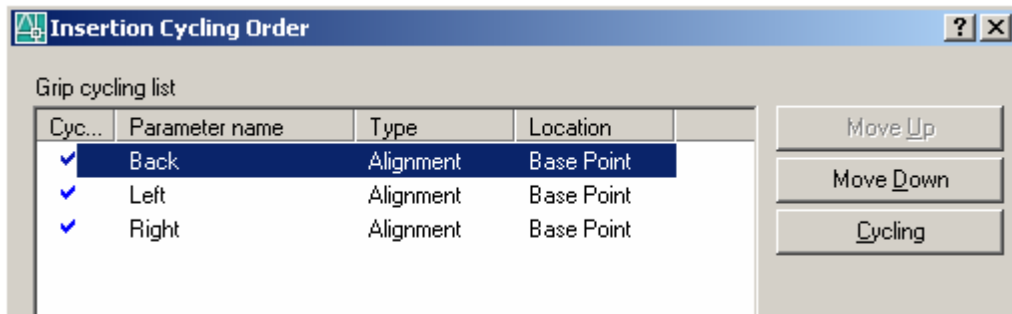
3. Add an alignment parameter to the back of the stove to align the stove to adjacent objects while maintaining a 3” offset from the other objects.
 - On the Parameters tab of the Block Authoring Palettes window, click the Alignment Parameter tool.
 - Specify the base point as the midpoint of the back of the stove, offset 3” along the Y axis. (1’-3”, 0’-3”).
 - Specify the alignment direction along the X axis (2’-0”, 0’-3”). Notice the caret-shaped grip that displays, along with a line indicating the alignment direction.
 - Select the alignment parameter and change the Parameter Name property to “Back”.

Tip: The alignment direction determines the axis to which the block reference is aligned. When grip-editing a block using an alignment grip, the block is rotated so that the alignment axis is either perpendicular to the normal to other curves under the grip, or tangent to other curves under the grip. The Alignment Type property defaults to Perpendicular but can be changed to Tangent when placing the alignment parameter in the block definition or by changing the Alignment Type property in the Properties palette after placing the parameter.

4. Add an alignment parameter to the left side of the stove.
 - On the Parameters tab of the Block Authoring Palettes window, click the Alignment Parameter tool.
 - Specify the base point as the midpoint of the left of the stove, offset -3” along the X axis. (- 0’-3”, - 1’-0”).
 - Specify the alignment direction along the Y axis (- 0’-3”, 0’-0”).
 - Select the alignment parameter and change the Parameter Name property to “Left”.
5. Add an alignment parameter to the right side of the stove.
 - On the Parameters tab of the Block Authoring Palettes window, click the Alignment Parameter tool.
 - Specify the base point as the midpoint of the left of the stove, offset -3” along the X axis. (2’-9”, - 1’-0”).
 - Specify the alignment direction along the Y axis (2’-9”, 0’-0”).
 - Select the alignment parameter and change the Parameter Name property to “Right”.

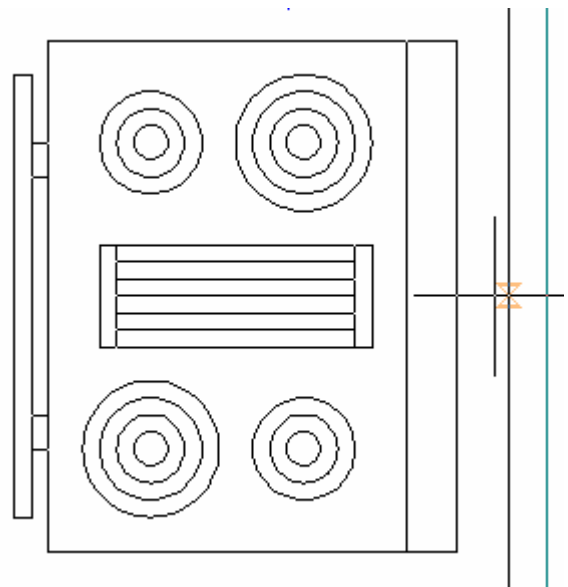
6. Specify the order that insertion points are cycled: (1) Back, (2) Right, and (3) Left

Tip: AutoCAD 2006 supports multiple insertion points for blocks. When using the mouse to select the insertion point for a block reference, you can cycle between insertion points by pressing the CTRL key. Each time you press the CTRL key, AutoCAD cycles to the next insertion point in the block reference. When you cycle past the last insertion point, AutoCAD returns to the first (default) insertion point. The default insertion point for a block reference is always the first insertion point listed in the Insertion Cycling Order dialog box. Custom grips you add to the block can also be used for insertion points. By default, all grips except those on rotation, flip, lookup, and visibility parameters are enabled for cycling. You can change this after inserting a parameter by selecting the grip and changing the Cycling property, or by selecting the grip in the Insertion Cycling Order dialog box (BCYCLEORDER command) and clicking the Cycling button to toggle cycling for the grip on and off.



- Enter **bcycleorder** on the command line to display the Insertion Cycling Order dialog box,
 - Select the Right parameter, and then click Move Up once to move it from the bottom of the list up one level.
 - Click OK to accept the changes.
7. Save your changes and return to the floor plan.
 - Enter **bclose** on the command line and save your changes to the block.
 8. Insert a new range in the drawing.

- Make the “Stove” layer current.
- Enter **insert** on the command line.
- In the Insert dialog box, select “Range” from the Name drop-down list.
- In the Scale and Rotation areas, clear the Specify On-screen check boxes, but leave it selected under Insertion Point.
- Click OK to close the dialog box.
- While dragging the block, press the CTRL key to cycle through the insertion points. Notice how the insertion point starts at the default point, then cycles to (1) back, (2) right, and (3) left sides of the stove, in order, then back to the default point again.



- Use the CTRL key to cycle to the insertion point at the back of the stove.

- Move the mouse close to other geometry in the drawing. Notice how the stove “snaps” to align to other geometry as you move over the geometry with the mouse.

Tip: The block is aligned to the side of the geometry approached by the mouse. For example, if you approach a vertical line from the right, the block snaps to the right side of the line. If you approach the line at the same point from the left, the block snaps to the left side of the line.

Creating Blocks with Multiple Views

You can define sets of objects in a block that become invisible when a special property of the block changes. This special property is called a *visibility* property and is added to the block when you add a visibility parameter to the block. The visibility parameter also adds a triangular grip to the block. When you click this grip on a block reference in a drawing, a drop-down menu of the allowed visibility property values (called visibility states) is displayed. A block can only contain one visibility parameter, and hence, only one visibility property or grip.



When you add a visibility parameter to a block the visibility state controls on the Block Editor toolbar are enabled. These toolbar buttons and controls allow you to change the current visibility state in the Block Editor and allow you to add or remove objects from the current visibility state. You can also manage visibility states using the Visibility States dialog box (BVSTATE command).



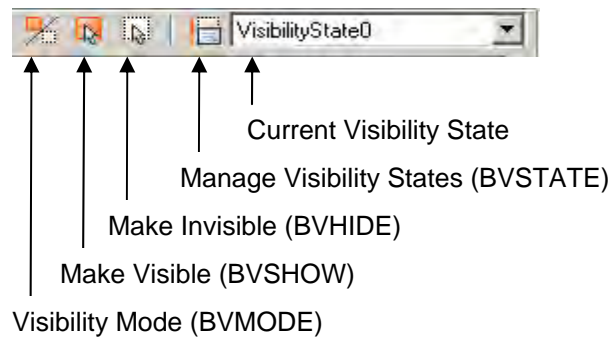
Lesson 2 Creating Blocks with Multiple Views

Objective: Learn how to use a visibility parameter to control visibility of parts of a block. Learn about the Visibility States dialog box (BVSTATE command). Learn about the BVMODE system variables and the visibility controls on the Block Editor toolbar.

1. Open the sample drawing *Wall Base.dwg*.
 - Open the “Range” block in the Block Editor.
 - Enter **edit** on the command line to display the Edit Block Definition dialog box.
 - In the Edit Block Definition dialog box, select the “Range” block from the list.
 - Click OK to close the dialog and open the Block Editor.

2. Add a visibility parameter to the block

- On the Parameters tab of the Block Authoring Palettes window, click the Visibility Parameter tool.
- Place the parameter at the center of the stove, just below the door (1'-3 1/16", - 2'-5").
- Notice how the Current Visibility State drop-down list becomes enabled on the Block Editor toolbar.

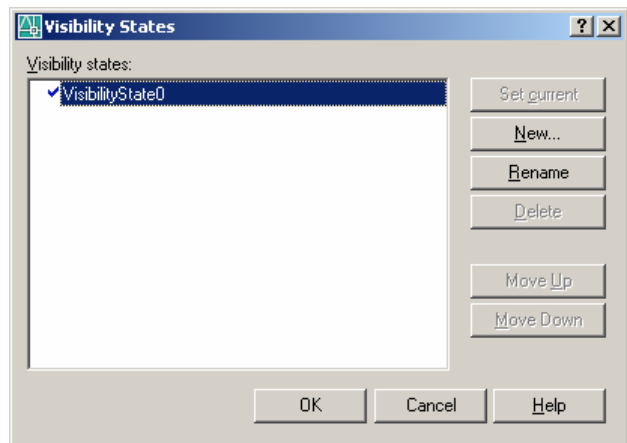


Tip: A block can contain only one visibility parameter. The Current Visibility State drop-down list remains disabled until the block contains a visibility parameter. Every block has a default visibility state called “VisibilityState0” that initially contains all of the geometry, parameters, and actions in the block when you add a visibility parameter to the block definition. This visibility state can be renamed.

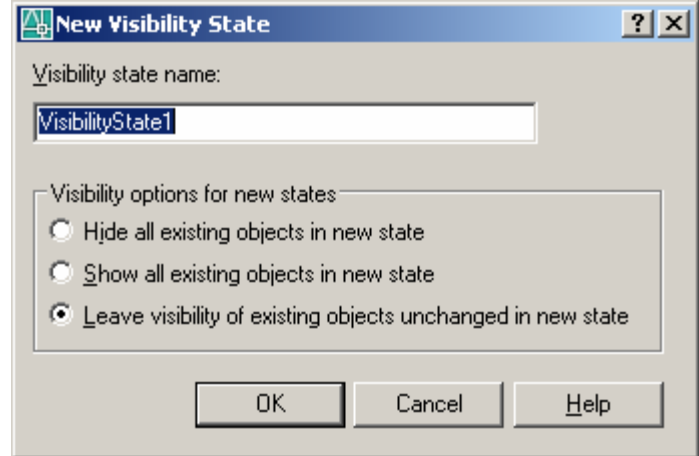
- Select the visibility parameter in the block definition, and change the Visibility Label to “Range Type”.

3. Rename the default visibility state “Electric”.

- Enter **bvstate** on the command line, or click on the Manage Visibility States button on the Block Editor toolbar.
- In the Visibility States dialog box, select “VisibilityState0”, click Rename, and rename the state “Electric”.

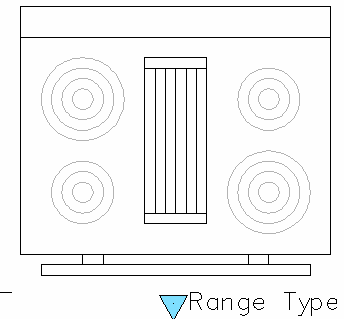


4. Add a new visibility state "Gas".
 - Click New...
 - In the New Visibility State dialog box, type "Gas" for the visibility state name.
 - Select Show All Existing Objects in New State.
 - Click OK to close the New Visibility State dialog box. In the Visibility States dialog box, notice that the new "Gas" visibility state is the current state.
 - Click OK to close the Visibility States dialog box.



5. Make the electric coils on the range top invisible in the current (Gas) visibility state.
 - Click the Make Invisible button on the right side of Block Editor toolbar or enter **bvhide** on the command line.
 - Select all of the circles in the drawing that represent the electric coils.
 - Notice that the coils are no longer displayed.

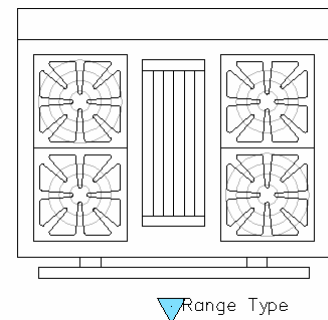
Tip: Objects in a block can be in more than one visibility state, but only objects in the current visibility state are normally visible in the Block Editor. You can control the visibility of objects not in the current visibility state by changing the BVMODE system variable setting.



- Click the Visibility Mode button on the Block Editor toolbar to toggle the visibility of objects in other visibility states on and off. Notice how the electric coils display in a dimmed color when toggling BVMODE on. Leave the electric coils visible for now.

6. Add grids for a gas range top.

- Enter **insert** on the command line and insert four copies of the "Grille" drawing. Place each grille at the center of one of the electric coils.
- Click the Visibility Mode button to toggle BVMODE off and hide the electric coils.
- Use the Visibility States drop-down to toggle between the Electric and Gas visibility states of the block. Verify that the electric coils are only visible in the Electric state, and the gas grilles are only visible in the Gas state.

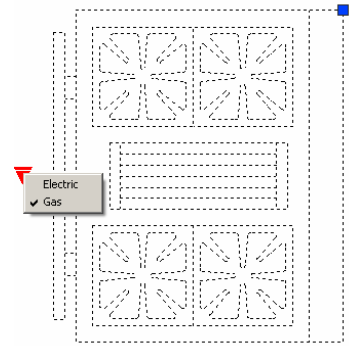


7. Save the block and return to the model tab.

- Enter **bclose** on the command line and save your changes to the block.

8. Change the stove from electric to gas in a drawing.

- Select the stove block in the drawing. Notice the light blue grip that appears near the stove. Notice the Range Type custom property in the Properties palette.
- Click on the grip and change the Range Type from Electric to Gas. Notice how the stove block updates to show the grilles instead of the coils.



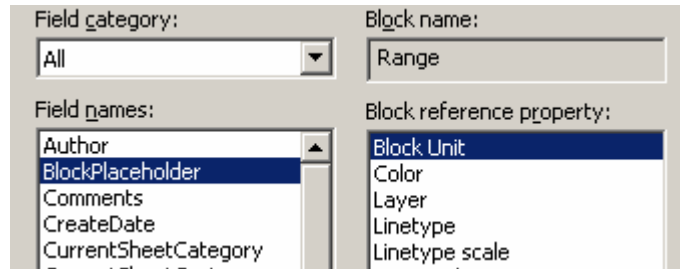
Displaying Custom Properties as Attributes

AutoCAD 2006 enhances attributes and fields to work closely with dynamic blocks.

You can control whether attributes in blocks have their own grips through the new Lock Position in Block property. This property is set to True by default, which prevents attributes from having their own grips.

Lock position in block

You can also display properties of blocks, including the new dynamic block custom properties, in attributes using the new BlockPlaceholder field. The BlockPlaceholder field contains a reference to a block property. When you insert or change the block AutoCAD re-evaluates any BlockPlaceholder fields in attributes contained in the block and updates them to reflect the new block properties. Using BlockPlaceholder fields is one way to display custom block properties as text in the drawing.

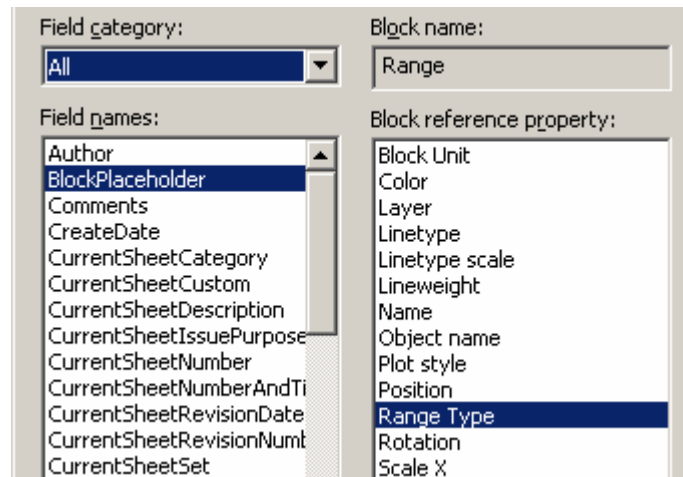


Lesson 3 Displaying Custom Properties as Attributes

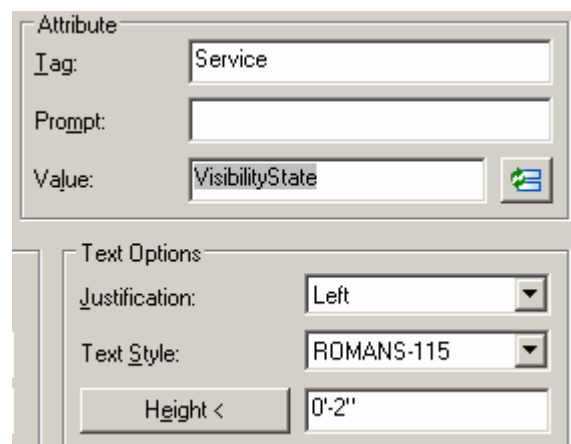
Objective: Learn how to use BlockPlaceholder fields and attributes to display custom block properties as text.

1. Open the sample drawing *Wall Base.dwg*.
 - Enter **insert** on the command line.
 - Click Browse....
 - In the Select Drawing File dialog box, select the *Range.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box, and click OK to re-define the existing Range block in the draw.
 - Place the block.
2. Open the new “Range” block in the block editor
 - Enter **bedit** on the command line to display the Edit Block Definition dialog box.
 - In the Edit Block Definition dialog box, select the “Range” block from the list.
 - Click OK to close the dialog and open the Block Editor.
3. Create a new attribute called “Service” that will display the current value of the Range Type property for the block

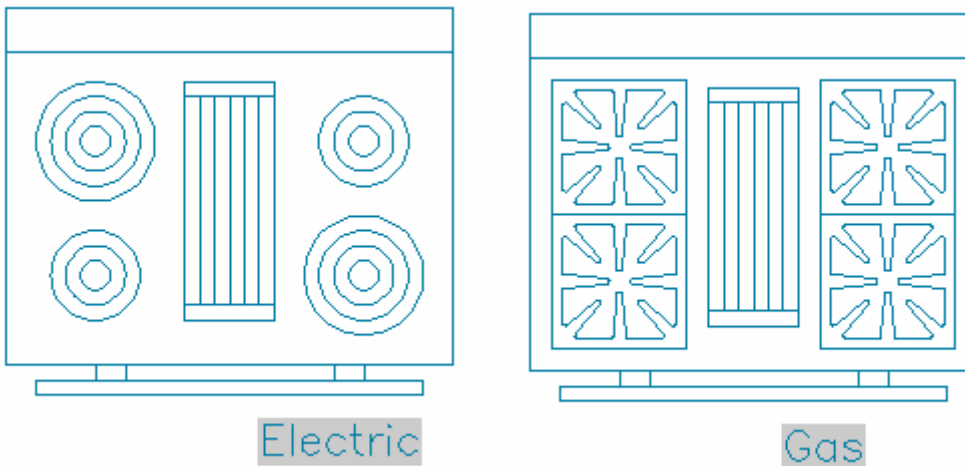
- Enter **attdef** on the command line to display the Attribute Definition dialog box.
- In the Attribute Definition dialog box, click the Insert Field button next to the Value field.
- In the Field dialog box, select BlockPlaceholder from the Field names list, and Range Type from the Block reference property list.
- Click OK to close the Field dialog box.



- Enter “Service” for the attribute Tag.
- Enter “2” for the text height.
- Click OK to close the Attribute Definition dialog box.
- Place the attribute just to the right of the visibility state grip (1'-4", -2'-6").

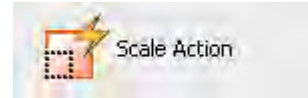


4. Add the new attribute to all visibility states
 - Enter **bvshow** on the command line.
 - Select the Service attribute definition.
 - Add it to all visibility states by entering **all**.
5. Save your changes to the block.
 - Click the Close Block Editor button. Click OK to save your changes to the block.
6. Use the ATTSYNC command to update the existing block references with the attribute you just added. Unlike custom properties and other geometry, AutoCAD does not automatically update existing block references with new or changed attributes.
 - Enter **attsync** on the command line, select the **Name** option, and type "Range".
7. Notice how the existing Range blocks now display the new attribute. Verify that the attribute contains the name of the current visibility state. Experiment with changing the Range Type property and verify that the field updates to display the new block property (you may have to use the REGEN command after changing the block to display the updated field values)



Scaling and Mirroring Block Geometry

You scale dynamic block geometry using a scale action. You can only associate scale actions with parameters that define a distance property: linear, polar, and XY parameters. You must define the base point for the scale action, and the geometry it acts upon.



AutoCAD executes scale actions when the distance property on the associated parameter changes. The scale factor is computed by the relative change in the associated distance parameter. For example if the distance doubles, the scale factor applied by the action is 2.0. If the distance is reduced by 1/3, the scale factor is 0.66. Scale actions always apply uniform scale factors and cannot be used to scale objects non-uniformly.

You mirror block geometry using flip parameters and flip actions. Flip parameters define one custom property describing the flipped/not flipped state of the parameter. You can customize the names displayed in the Properties palette for the flipped/non-flipped states to something appropriate for the block (e.g., Closed/Open, Left/Right, On/Off). Flip parameters also define a single arrow grip that toggles the mirrored state of the parameter. When placing a flip parameter you must also define a reflection line. This line defines the axis about which any associated flip actions will mirror objects.

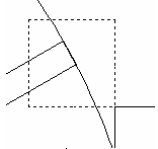
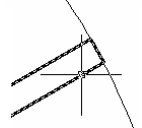


Flip actions can only be associated with flip parameters. Flip actions define the objects that are mirrored about their associated parameter's reflection line when the state of the parameter changes.



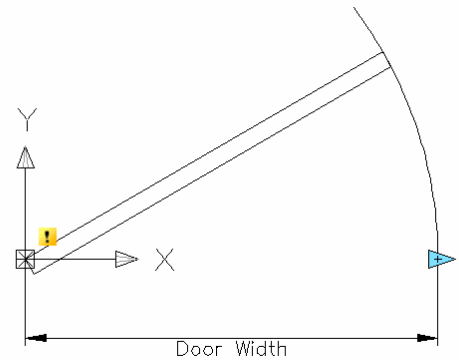
Lesson 4 Scaling and Mirroring Block Geometry

Objective: Learn how to create complex blocks by associating multiple actions with a single grip or parameter. Learn how to scale objects using a scale action. Learn how to mirror objects using a flip parameter and action.

1. Open the sample drawing *Wall Base.dwg*.
2. Edit the “Door – Left” block in the Block Editor.
 - Double-click on a door block in the drawing or enter **bedit** on the command line.
 - Select “Door – Left” from the list of blocks and click OK to edit the block.
3. Create a linear parameter that controls the door width.
 - On the Parameters tab of the Block Authoring Palettes window, click the Linear Parameter tool.
 - For the start point, click on the leftmost corner of the door (the corner at 0,0).
 - For the endpoint, click on the lower-right end of the arc.
 - For the label location, pick a point below the geometry.
4. Select the parameter, and change its properties in the Properties palette.
 - Change the Number of Grips property to 1. The grip at the origin should be removed, leaving the grip near the end of the arc.
 - Change the Distance Label and Parameter Name to “Door Width”.
5. Create a stretch action that stretches the door.
 - On the Actions tab of the Block Authoring Palettes window, click the Stretch Action tool.
 - For the parameter, select the grip on the Door Width parameter, or select a point on the line close to the right end of the parameter. Confirm that the red crosshairs appear at the right end of the parameter. If the crosshairs don’t display, use the Start Point/Second point command line options or pick with the cursor so that the red crosshairs display at the correct end.
 - For the stretch frame, pick a window encompassing the right end of the door.
 
 - For the objects to stretch, select the rectangle that forms the door. *Do not select the arc.*

 - Place the stretch action.
6. Change the axis of the stretch action to be parallel with the door.

Tip: Most actions default to moving/rotating/stretching objects in the same direction or axis as the grip or parameter with which they are associated. Some actions can be customized so that they act in a direction that is relative to, but not parallel to, the motion of the grip or parameter. These are called *action overrides*. In this case, we need the axis of the stretch action to be along the axis of the door, not along the axis of the Door Width parameter. This is a 30 degree offset.

 - Select the stretch action.
 - Change the Angle Offset property in the Properties palette to 30.



7. Create a scale action that scales the arc.

- On the Actions tab of the Block Authoring Palettes window, click the Scale Action tool.
- For the parameter, select the Door Width parameter. Note how the scale action does not require you to pick which end of the parameter to use.
- Select the arc for the object to scale.

Comment: You might be wondering why we didn't just scale both the arc and door and do away with the stretch. The reason is because we don't want to scale the thickness of the door, only its width. The scale action scales objects uniformly in all directions, if we scaled the door about the origin it would have the right width, but its thickness would also change as we changed the door width.

- Place the action icon.

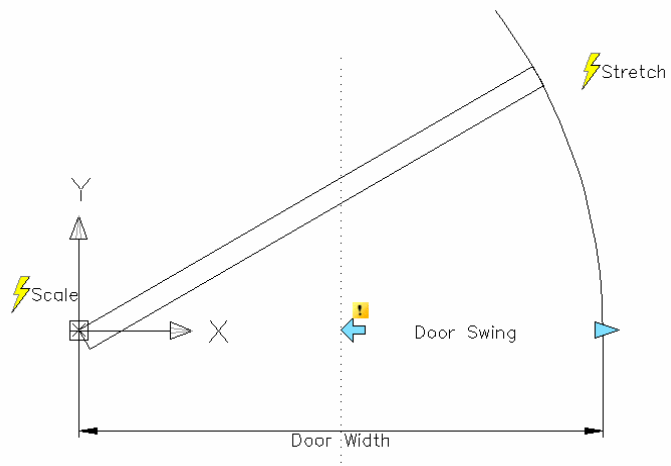
8. Create a flip parameter that controls the door swing.

- On the Parameters tab of the Block Authoring Palettes window, click the Flip Parameter tool.
- Pick (18,0) for the base point of the reflection line. (This is the midpoint of the door width.)
- Pick a point along the Y axis for the endpoint of the reflection line (e.g., 18,6).
- Pick a point to the right for the label location.

9. Select the parameter and change its properties in the Properties palette.

- Change the Flip Distance Label and Parameter Name to "Door Swing".
- Change the Base State Label property to "Left Swing".
- Change the Flipped State Label property to "Right Swing".

Tip: The state labels appear as choices for the flip property in the Properties palette when you select the block reference in a drawing. Giving them meaning in the context of the block will make the block easier to use and understand.



10. Create a flip action that mirrors the entire block about the door swing reflection line.

- On the Actions tab of the Block Authoring Palettes window, click the Flip Action tool.
- For the parameter, select the Door Swing parameter.
- You want to flip the door geometry, scale action, stretch action, and Door Width parameter.

Tip: While it is tempting to simply type "All" when selecting objects for an action, you should consider it a best practice to always think about what objects you want action applied to, and avoid simply windowing everything on the screen or typing "All". Doing so can lead to unexpected consequences when you inadvertently include other actions or parameters in action selection sets.

- Place the flip action.

11. Save your changes to the block and return to the floor plan.

- Enter **bsaveas** on the command line and save your changes to a new block named "Door".

12. Experiment with some of the door blocks in the drawing.
 - Enter **insert** on the command line and insert a new instance of the “Door” block.
 - Verify that moving the linear grip changes the door width by stretching the door and scaling the arc.
 - Verify that clicking the flip grip flips the door swing, including the linear grip. Notice how the Door Swing property in the Properties palette correctly displays the door swing, and how changing the value in the Properties palette flips the block.
13. **Extra Challenge**:: Fix the block so that the flip grip remains centered in the door opening as the door width changes.
 - Open the “Door” block in the Block Editor.
 - Add a move action, associate it with the grip controlling the Door Width parameter, and select the flip parameter as the move action’s selection set.
 - Select the move action, and in the Properties palette, change the Distance Multiplier value to 0.5.

Tip: Most actions default to moving/rotating/stretching objects proportional to the change in the location of their associated grip. Some actions can be customized so that they act as a multiple of the grip motion. In this case we want to only move the flip action and parameter half as far as the linear grip that drives the action, so that they will remain centered in the door opening.

 - Save the block and close the Block Editor.
 - Verify that the flip parameter remains centered in the door opening when the door width changes.

Arraying Objects with the Array Action

You array objects in your block by adding an array action to a linear, polar, or XY parameter. When associated with a linear or polar parameter the objects are arrayed in one direction only. When associated with an XY parameter the objects can be arrayed in two directions.



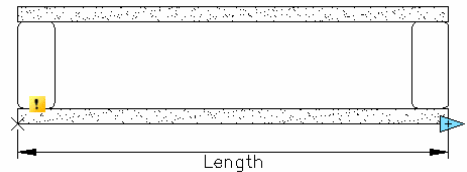
When you add an array action, you specify the objects to array, and the distance between columns and rows of your array. The array action divides the column distance and row distance into the respective distance properties on the associated parameter to determine the number of columns and rows and makes one copy of the selected objects for each column in every row.

Lesson 5 Arraying Objects with the Array Action

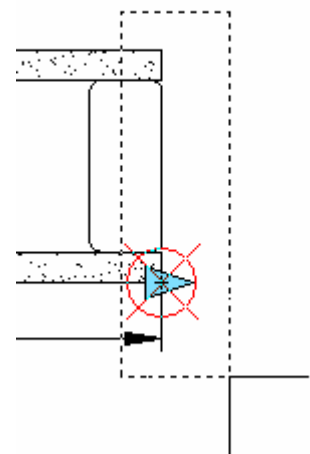
Objective: Learn how to use an array action.

In this lesson, you will add parameters to an existing wall block to stretch the length of the block. The block shows a plan view of a typical wood frame interior wall section containing studs with 16" spacing and 5/8" drywall on either side of the wall.

1. Open the sample drawing *Wall Base.dwg*.
2. Insert the *Wall.dwg* drawing into *Wall Base.dwg*.
 - Enter **insert** on the command line.
 - Click Browse..., and then, in the Select Drawing File dialog box, select the *Wall.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
3. Open the "Wall" block in the Block Editor.
 - Double-click on the block you just inserted to open the Edit Block Definition dialog box.
 - Ensure that the "Wall" block is highlighted in the list.
 - Click OK to close the dialog box and open the Block Editor.
4. Add a linear parameter to control the length of the wall.
 - On the Parameters tab of the Block Authoring Palettes window, click the Linear Parameter tool.
 - Click the lower-left corner of the wall for the start point and the lower-right corner of the wall for the endpoint. Click below the wall to place the parameter.
 - Select the parameter. In the Properties palette:
 - a. Change the Number of Grips to 1
 - b. Change the parameter Name and Label to "Length".
 - c. Change the Dist Minimum to 17 1/2".

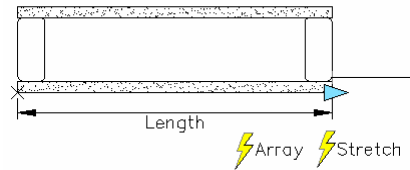


5. Add a stretch action to stretch just the drywall rectangles with the Length parameter.
 - On the Actions tab of the Block Authoring Palettes window, click the Stretch Action tool.
 - Click the Length grip, then click again to confirm the parameter point.
 - Click a point above and just to the left of the top right corner of the wall for the first stretch frame corner.
 - Click a point below and just to the right of the bottom right corner of the wall for the second stretch frame corner.
 - Select the top drywall rectangle and the bottom drywall rectangle. *Do not select the stud (large rectangle inside the wall).*
 - Click to place the stretch action icon.



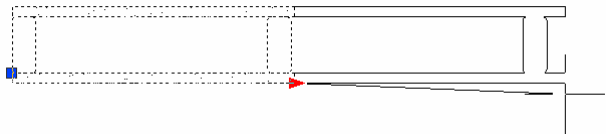
6. Add an array action that adds a new stud every 16" along the wall.
 - On the Actions tab of the Block Authoring Palettes window, click the Array Action tool.
 - Click on the Length parameter to associate the action with the wall length
 - Select the wall stud on the right for the object to copy.
 - Enter the stud spacing, 16", as the column spacing.

Tip: When associated with a linear parameter, array actions array objects in only one dimension so they only use the column spacing property to determine how many objects to copy. When associated with XY parameters, array actions use both a column spacing and a row spacing property to determine how many objects to copy.



- Click to place the array action icon.
7. Save the block and return to the drawing.
 - Click Close Block Editor on the toolbar and save your changes.
 8. Experiment with the new wall block.

- Select the wall block in the drawing.
- Notice how changing the wall length stretches the drywall and hatch. Notice how a new stud rectangle appears every 16" as you stretch the wall.



Using a Polar Parameter and a Polar Stretch Action

You create geometry that rotates while it stretches using polar parameters and polar stretch actions.

Polar parameters define two custom properties measuring the distance and angle between the two points of the parameter. One of these points (the first one you define when placing the parameter) is called the base point; the other is called the endpoint.



When you change the distance property of a Polar Parameter, the base point remains fixed while the endpoint adjusts so that the distance between the two points equals the new distance value. When you change the angle property of a polar parameter the distance between the points remains the same, but the angle formed by the line connecting the points and the block's X axis changes to become the new angle.

Polar stretch actions combine the effects of a stretch action with those of a rotate action. Like a stretch action, a polar stretch action is associated with the movement of either the start point or the endpoint of the parameter with which the action is associated. When you define a polar stretch action you must also define the stretch frame and the objects in the block that the action acts upon. You can also define a separate set of objects that are rotated but never stretched.



Similar to the STRETCH command, a stretch action moves objects completely within the stretch frame, while stretching those which cross the stretch frame. However, because polar parameters have angle properties, objects in the polar stretch action's selection set also rotate as the associated parameter angle changes. The base point of rotation is always the opposite endpoint of the parameter.

Lesson 6 Using a Polar Parameter and a Polar Stretch Action

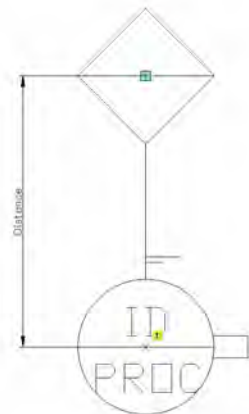
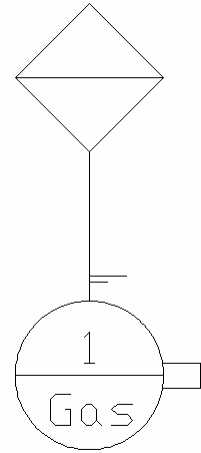
Objective: Learn how to use a polar parameter. Learn how to combine a polar parameter with a polar stretch action to stretch, move, and rotate geometry at the same time.

Consider an existing block representing an instrument callout in a process diagram. We want to turn this block into a dynamic block so that the diamond in the block can be repositioned relative to the circle. When the diamond is repositioned the line connecting the diamond with the circle needs to be stretched, while the “flags” hanging from the connector need to be rotated with the line.

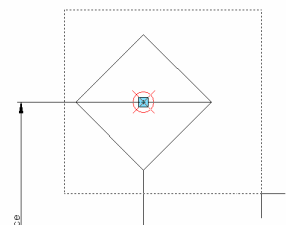
1. Create a new drawing.
2. Insert the *Instrument1.dwg* drawing as a block.
 - Enter **insert** on the command line.
 - Click Browse..., and then in the Select Drawing File dialog box, select the *Instrument1.dwg file*. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
3. Open the “Instrument1” block in the Block Editor.
 - Enter **bedit** on the command line to open the Block Editor.

Tip: Double-clicking on blocks with attributes opens the Enhanced Attribute Editor dialog box instead of the Block Editor. Because the “Instrument1” block contains attributes, we must use the BEDIT command to edit the block definition.

- Ensure that the “Instrument1” block is highlighted in the list.
 - Click OK to close the dialog box and open the Block Editor.
4. Place a polar parameter in the block.
 - On the Parameters tab of the Block Authoring Palettes window, click the Polar Parameter tool.
 - Click the center of the instrument circle for the start point.
 - Click the midpoint of the line bisecting the diamond for the endpoint.
 - Place the parameter dimension line to the left of the instrument.
 - Select the parameter. In the Properties palette, change:
 - a. Show Properties to No.
 - b. Number of Grips to 1.



5. Place a polar stretch action in the block.
 - On the Actions tab of the Block Authoring Palettes window, click the Polar Stretch Action tool.
 - Click near the top of the Distance parameter. Click a second time on the grip to select the key point for the parameter.
 - Define a stretch frame that encompasses all of the diamond and part of the connecting line.



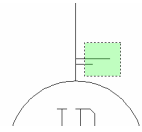
- Select the connecting line, diamond, and line bisecting the diamond for the objects to stretch.

Tip: Polar stretch actions affect objects differently depending on the relationship between the objects and the stretch frame. Objects *not crossing* the stretch frame are *moved and rotated*. Objects crossing the stretch frame are *stretched and rotated*.

- Select the “flags” hanging from the connecting line for the objects to rotate.

Tip: Polar Stretch actions maintain two selection sets. The second set contains objects that are rotated *only*, regardless of their relationship to the stretch frame.

- Place the polar stretch icon.

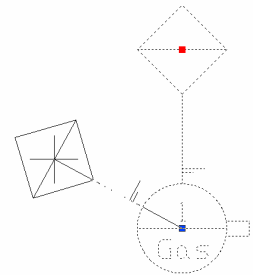


9. Save the block and return to the drawing.

- Click Close Block Editor on the toolbar and save your changes.

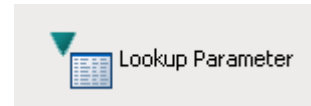
10. Experiment with the new instrument block.

- Select the instrument block in the drawing.
- Notice how you can move the polar grip in any direction. Notice how the diamond rotates and moves with the grip, while the line stretches (to get longer and shorter) and rotates where it meets the circle. Notice how the flags on the line just rotate.



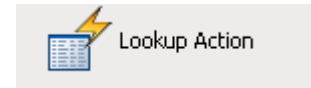
Computing Custom Properties With Lookup Tables

You can add read-only properties to blocks that are computed from other block properties by adding lookup parameters and lookup tables to your dynamic block.



Each lookup parameter added to a block adds a single read-only custom property and a triangular grip to the block definition. The Lookup property is a text value that is computed from the values of other properties in the block through a lookup table. The single, triangular grip that is added to the block definition, does not display in a block reference when the underlying custom property is read-only. (We will cover read-write lookup parameters later).

Lookup actions define spreadsheet-like tables that AutoCAD uses to compute the values of one or more lookup parameter properties from other parameter properties in the block. Lookup tables consist of input properties and lookup properties. When the value of an input property in the block changes AutoCAD “looks up” the new value in the table and finds the first row that matches the new property value. A row is only considered a match if all of the current values of the input properties match. When AutoCAD finds a match, it changes the Lookup property value to the corresponding value in the table. There is always a default row at the bottom of the table that AutoCAD uses if no match can be made.



For example, suppose you had a window block that is only available in wood when the window height and width are less than 6', and is only available in aluminum if either the height or width are greater than 6'. If you had two linear parameters in your block (one called “Width” and one called “Height”), you could add a lookup parameter called “Material” to your block to display the material for the window as a custom property. The table would look something like this:

Width	Height	Material
< 6'	< 6'	Wood
*	*	Aluminum

The row with the “*” values for Width and Height are used when no other matching row exists. Lookup tables in AutoCAD can match ranges of values for a particular property to one row, much like the “less than 6” mathematical notation above. The following rules apply when you specify lookup values in lookup tables:¹

- Use a comma as the delimiter between values.
- You can specify any number of unique values separated by commas. For example: 5,6,7 5.5,6.25
- To specify a range, use brackets [] to specify that the range includes the values separated by a comma, or use parentheses () to specify that the range does not include the values separated by a comma.
- For a continuous range, use a pair of values separated by a comma, enclosed in brackets or parentheses. For example: [3,10] specifies any value between 3 and 10, including 3 and 10 (3,10) specifies any value between 3 and 10, not including 3 and 10.
- For an open-ended range, use one value with a comma, enclosed in brackets or parentheses. For example: [,5] specifies less than or equal to 5; (5,) specifies greater than 5.

Using these rules the table above could also be written:

Width	Height	Material
[,6)	[,6)	Wood
*	*	Aluminum

¹ From the topic *Specify Values for Lookup Tables* in the *AutoCAD 2006 User's Guide*

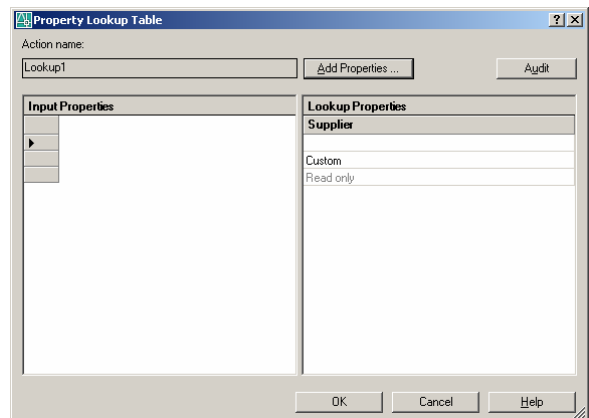
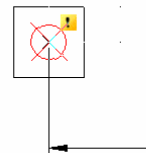
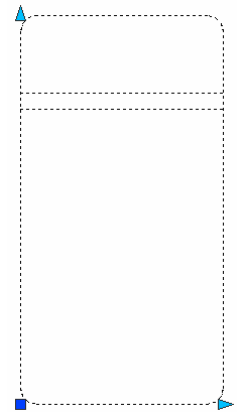
Lesson 7 Computing Custom Properties Using a Lookup Parameters and Actions

Objective: Learn how to use a lookup table to compute the value of a parameter from the values of other parameters in the block by “looking up” the value in a table.

Consider an existing dynamic block representing a simple bed. The block has linear parameters for controlling the bed ;length and width. We want to add a read-only, custom property to the block that describes the supplier. In our project we have two suppliers for our beds: “Sandman Enterprises L.L.C” supplies full- and twin-sized beds. “Morpheus Mattresses Inc.” supplies Queen-sized and larger beds.

In this lesson, you will enhance the existing dynamic block to have a new “Supplier” custom property. The property is calculated based on the Width dimension of the bed.

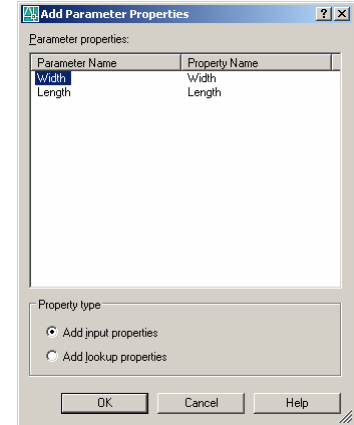
1. Open the sample drawing *Wall Base.dwg*.
2. Insert the *bed.dwg* drawing as a block.
 - Play with the block grips. Note how you can change the length and width of the bed. Notice how the block width must remain between 39” (twin) and 76” (king). Notice how the length must remain between 75” (twin) and 84” (California king).
3. Edit the “bed” block in the Block Editor.
 - Enter **bedit** on the command line to display the Edit Block Definition dialog box.
 - Select the “Bed” block from the list.
 - Click OK to close the dialog box and open the Block Editor.
4. Add a lookup parameter called “Supplier” to the block.
 - On the Parameters tab of the Block Authoring Palettes window, click the Lookup Parameter tool.
 - Place the parameter below and to the right of the bed.
 - Select the parameter and change the Lookup Label property to “Supplier”.
 - Change the Parameter Name property to “Supplier”.
 - Change the Number of Grips to 0.
5. Add a lookup table to the Supplier parameter.
 - On the Actions tab of the Block Authoring Palettes window, click the Lookup Action tool.
 - Select the Supplier parameter.
 - Place the action to the right of the parameter. Notice that the Property Lookup Table dialog is displayed.
 - Notice that the dialog box automatically added the Supplier lookup parameter to the table.



Tip: Lookup parameters can be read-only or read-write, but unlike other parameters this cannot be directly controlled by the block author but is

determined by the contents of the lookup table itself. We discuss read-write lookup parameters, also called reverse lookup parameters in a subsequent lesson.

6. Add the Width parameter as an input property.
 - Click Add Properties...
 - In the Add Parameter Properties dialog box, select Width from the list.
 - Click OK to close the dialog box and return to the Property Lookup Table dialog box.
 - Notice how the Width column appears in the Input Properties list.
7. Add a row to the table for the “Sandman Enterprises L.L.C” supplier for beds between 39 and 60 inches in width.



- Click in the top row in the Width column to edit the cell.
- Enter **[39,60]** into the cell. Notice the use of “[” and “)” characters to indicate the range of valid values. The “[” preceding the first value indicates any values greater than *or equal to* 39” are allowed, while the “)” following the last value indicates any values less than *but not equal to* 60”. Taken together this means that any beds equal to or greater than 39”, and less than 60”, match this row of the table.
- Notice that a new empty row appears as soon as you make changes to the empty row.

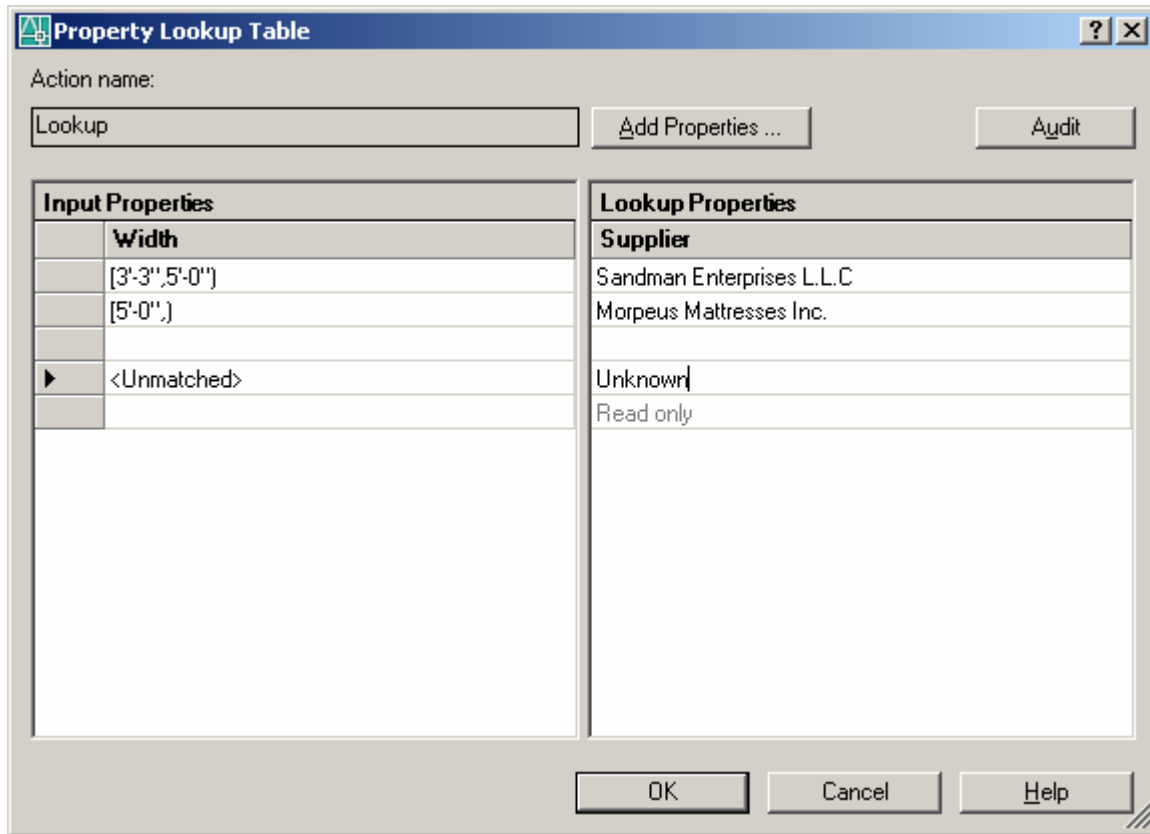
Tip: The Property Lookup Table dialog box always shows two extra rows in the table. The next-to-last row isn’t actually added to the table until you enter values in one of the cells. This is how you add new rows to a lookup table. The last row always exists, and lists all of the input properties as “Unmatched”. When no other row can be found to match all of the current values for the input properties, the lookup action uses this “unmatched” row to determine the values to assign to the lookup properties.

- Click in the top cell under the Supplier column.
 - Type “Sandman Enterprises L.L.C.” into the cell.
8. Add a row to the table for the “Morpheus Mattresses Inc.” for beds between larger than 60 inches in width.
 - Click in the second row in the Width column to edit the cell.
 - Enter **[60,]** into the cell. Notice there is no upper end to the range. This string causes all bed widths greater than *or equal to* 60” to match this row.
 - Click in the second row cell under the Supplier column.
 - Type “Morpheus Mattresses Inc.” into the cell.

Tip: We could have used a similar unbounded range for the first supplier as well, since they supply all mattresses less than 60” wide. The syntax in that case would have been “[,60)”.

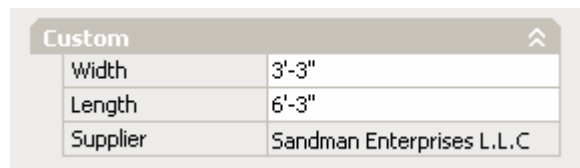
9. Make the supplier property Unknown for unexpected bed sizes.
 - Click on the cell containing the word “Custom” under the Supplier column. Change the value to “Unknown”.

- The figure below shows the final content of the lookup table. Click OK to close the dialog box and return to the Block Editor.



- Save the block and return to the drawing.
 - Click Close Block Editor on the toolbar. Notice a dialog appears prompting you to save changes and update the existing block(s) in the drawing.
 - Click Yes to save changes and return to the floor plan.
- Experiment with the new Supplier property on the block.

- Select the bed in the drawing. Notice the new read-only Supplier property now appears in the Properties palette for the block. Notice how the property automatically switches between the two suppliers depending on whether the bed width is less than 60".



- Notice that it is not possible to get the Supplier property to display the Unknown value. This is not due to an error in your lookup table, but because of the way the lookup table works in harmony with the constraints on the Width property, which prevents the user from assigning any width value to the block that does not match one of the two rows we added earlier.

Reverse (Read-Write) Lookup Parameters

Normally, lookup parameters are read-only block properties whose values are computed from other custom properties in the block, called input properties, by looking up the current values of the input properties and finding the matching row in the table.

If all of the rows for input properties in a lookup table are unique (each cell for an input property is unique and no input row cell contains a range or list) then any lookup parameter in the table which also has unique rows can be made “read-write” by changing the Read Only setting for the property to Allow Reverse Lookup.

You can change the values of read-write lookup parameters on a block reference through the Properties palette or by clicking on the triangular grip that displays when the block reference is selected in a drawing.

Lookup Properties
Lookup
a
b
Custom
Allow reverse lookup

Lesson 8 Reverse Lookup Parameters

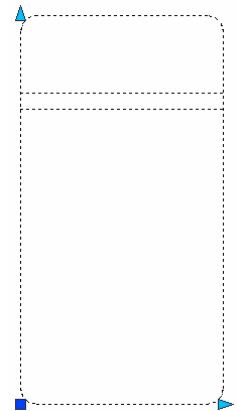
Objective: Learn how to use a reverse lookup table to drive other parameters from changes in the lookup table.

Consider our earlier example of a dynamic bed block that used a lookup parameter to compute the supplier. In that example the Bed Width and Bed Length properties were constrained to be within the range of standard bed sizes. However, it was possible to create combinations of bed length and width that are not standard sizes, for example a bed as wide as a twin but as long as a California King.

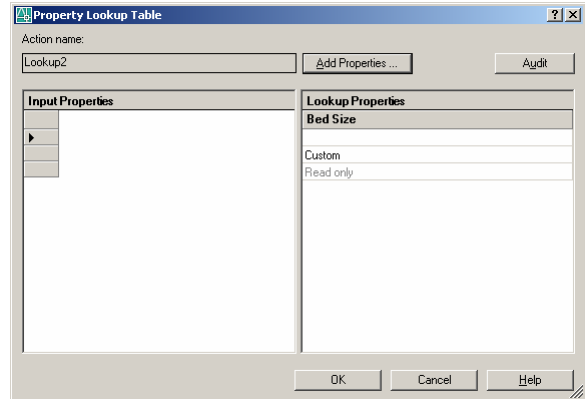
In this lesson, you will enhance the existing dynamic block to have a new Bed Size custom property. This property is calculated based on the combination of the current width and length of the bed as described in the following table. Because the table we create defines rows that are mutually exclusive (any combination of width and length matches exactly one row in the table), this property can be either read-only or read-write (allows reverse lookup).

Bed Width	Bed Length	Bed Size
3'-3"	6'-3"	Twin
3'-3"	6'-8"	Twin Extra Long
4'-6"	6'-3"	Full
5'	6'-8"	Queen
6'-4"	6'-8"	King
6'	7'	California King

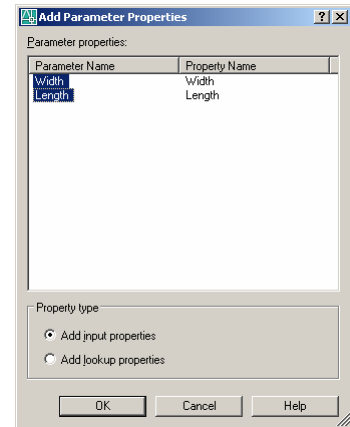
1. Open the sample drawing *Wall Base.dwg*.
2. Insert the *bed2.dwg* drawing as a block.
 - Play with the block grips.
 - Notice how you can change the length and width of the bed. Notice how the block width must remain between 39" (twin) and 76" (king). Notice how the length must remain between 75" (twin) and 84" (California king).
3. Edit the "bed2" block in the Block Editor.
 - Enter **bedit** on the command line to display the Edit Block Definition dialog box.
 - In the Edit Block Definition dialog box, select the "Bed2" block from the list.
 - Click OK to close the dialog box and open the Block Editor.
4. Change the Dist Type property of the Width parameter so that only bed widths from the table above are allowed.
 - Select the Width parameter.
 - In the Properties palette, change the Dist Type property to List.
 - Note how the list matching the allowed bed widths is already filled in for this block.
5. Change the Dist Type property of the Length parameter so that only bed lengths from the table above are allowed.
 - Select the Length parameter.
 - In the Properties palette, change the Dist Type property to List.
 - Note how the list matching the allowed bed lengths is already filled in for this block.
6. Add a Bed Size lookup parameter for the bed size.
 - On the Parameters tab of the Block Authoring Palettes window, click the Lookup Parameter tool.
 - Place the parameter below the bottom-left corner of the bed.



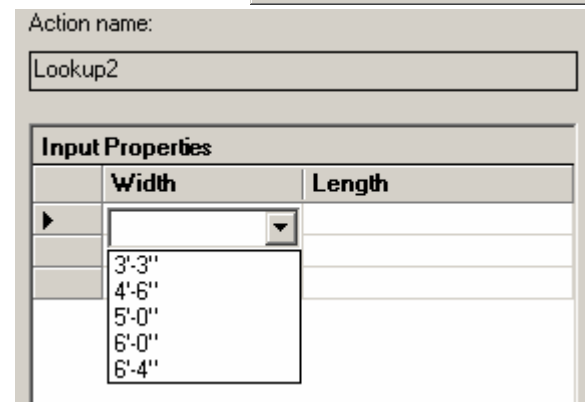
- Select the parameter and change the Lookup Label property to “Bed Size”.
 - Change the Lookup Name property to “Bed Size”.
7. Add a lookup table to the Bed Size parameter.
- On the Actions tab of the Block Authoring Palettes window, click the Lookup Action tool.
 - Select the Bed Size parameter.
 - Place the action to the right of the parameter. Notice that the Property Lookup Table dialog is displayed.
 - Notice the dialog automatically added the Bed Size lookup parameter to the table (under Lookup Properties).



8. Add the Width and Length parameters as input properties.
- Click Add Properties...
 - Select Width and Length from the list at the same time by holding down the SHIFT key as you select each item.
 - Click OK to close the dialog box and return to the Property Lookup Table dialog box.
 - Notice how the Width and Length columns appear in the Input Properties list.



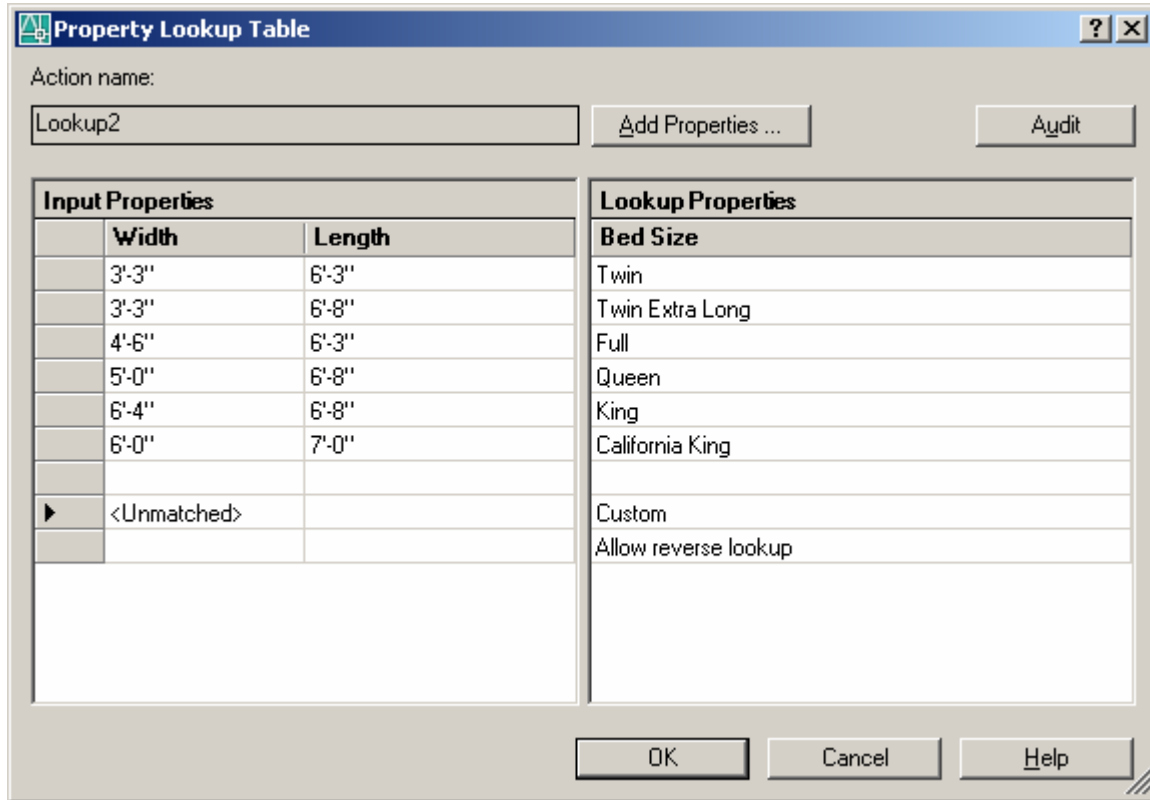
9. Add a row to the table for the Twin bed size.
- Click in the top row in the Width column to edit the cell. Since this property value has been restricted to a list of allowed values, the Property Lookup Table dialog box displays a combo box from which to select the cell value.
 - Select 3'-3" from the list, the width for a twin bed.
 - Click on the top row in the Length column to edit the cell.
 - Select 6'-3" from the list, the length of a twin bed.
 - Click on the top row in the Bed Size column.
 - Type "Twin" for the bed size.



10. Add rows for each of the remaining bed sizes.
11. Change the table to allow the Bed Size lookup parameter to be used as a reverse lookup parameter:
- Select the bottom cell labeled “Read only” in the Bed Size column.
 - Change the value to Allow Reverse Lookup.

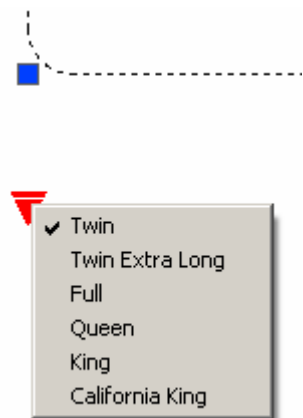
Tip: Because any given combination of input properties uniquely identifies one and only one row in the table (including the “<Unmatched>” row), lookup properties in the table can be made read-write properties if they too have unique values in each row. For these properties the Read only property can be changed to Allow Reverse Lookup to make them writable.

- When you are finished, your table should look like the one in the following figure:



- Click OK to close the dialog box and save your changes to the table.
8. Save the block and return to the drawing.
 - Click Close Block Editor on the toolbar. Notice a dialog appears prompting you to save changes and update the existing block(s) in the drawing.
 - Click Yes to save changes and return to the floor plan.
 9. Experiment with the new Bed Size property on the block.

- Select the bed in the drawing. Notice the new read-write Bed Size property now appears in the Properties palette for the block. Notice how changing the property automatically changes both the width and length of the block properties.
- Notice how you can also select the Bed Size property from the drop-down list displayed when you click on the new lookup grip.



Tip: Turning off shortcut menus in AutoCAD's drawing area (Options/User Preferences/Shortcut Menus in Drawing Area) turns off the display of lookup grip menus, but not the lookup grip itself!

- Notice that it is still possible to individually change the Width and Length properties of the block. Notice how selecting a combination of Width and Length values that do not match any standard bed sizes causes the Bed Size property to list the <Unmatched> value string "Custom".

Parameter Key Points and Action Base Points

All parameters have one or more *key points*. When you associate some actions with a parameter you must specify which key point on the parameter the action is associated. A change in the associated key point from a grip moving or a property changing triggers the action.

Rotation and point parameters have only a single key point, the center of rotation or the point position. AutoCAD never prompts for the key point when associating actions with these parameters.

Linear and polar parameters have two key points. The base (or start) point is the first key point you pick when defining these parameters, the endpoint is the second key point. AutoCAD draws parameter base points with a small X marker in the Block Editor, while it draws Parameter endpoints with a small plus (+) sign.

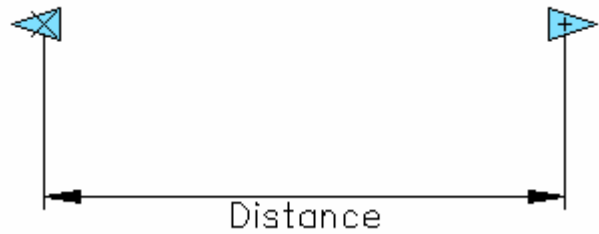
XY parameters have four key points, one at each corner. The base point is the first key point you pick when defining these parameters, the endpoint is the second key point. The bottom right corner is called the X Corner and the top left corner is called the Y Corner.

Scale and rotate actions require a base point about which to scale and rotate geometry. The default base point is the associated key point on the parameter. Actions can also have base points that are independent of the location of the associated key points.

Whether an action uses the key point for its base point is controlled by the Base Type property. Base points that use a key point are called *dependent*, while those that do not are called *independent*.

AutoCAD does not draw dependent base points in the Block Editor. Dependent base points can be offset from the actual key point location by changing the Base Offset property of the action in the Properties palette.

AutoCAD draws independent base points in the Block Editor with a small X marker and provides a grip for changing the base point location. (The location of the action lightning bolt icon has no relation to the action base point or how the action behaves). You can also change independent base point locations through the Properties palette.



Base type	Dependent
Base offset X	0.0000
Base offset Y	0.0000

Base type	Independent
Base X	0.0000
Base Y	2.0000

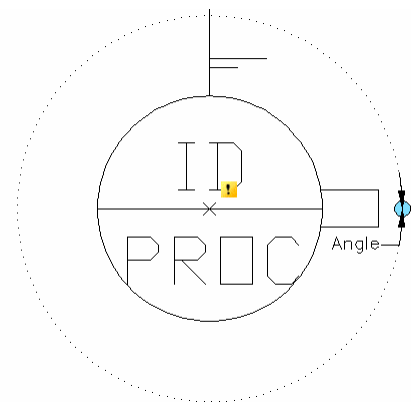
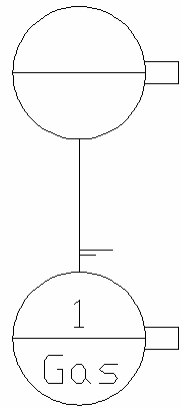


Lesson 9 Action Base Points

Objective: Learn about actions that have base points. Learn about the difference between dependent and independent base points, and how dependent base points use key points on parameters.

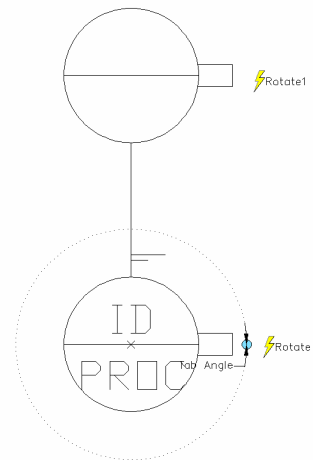
Consider a variation of our earlier instrument callout block. This block has two circles with “tabs” hanging from them. We want to turn this block into one where the tabs can be rotated about the circles, but where the two tabs are always located at the same position on their respective circles.

1. Create a new drawing
2. Insert the *Instrument2.dwg* drawing as a block.
 - Enter **insert** on the command line.
 - Click Browse...
 - In the Select Drawing File dialog box, select the *Instrument2.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
3. Open the “Instrument2” block in the Block Editor.
 - Enter **bedit** on the command line to open the Block Editor.
 - Ensure that the “Instrument2” block is highlighted in the list.
 - Click OK to close the dialog box and open the Block Editor.
4. Add a rotation parameter centered on the bottom circle.
 - On the Parameters tab of the Block Authoring Palettes window, click the Rotation Parameter tool.
 - Click a point so that the rotation radius is larger than the circle and tab.
 - Enter 0 for the base angle.
 - Select the rotation parameter. In the Properties palette change Show Property to No.
5. Add a rotate action for the tab on the bottom circle.
 - On the Actions tab of the Block Authoring Palettes window, click the Rotate Action tool.
 - Click the rotate parameter.
 - Select the lines making up the tab on the bottom circle for the action set.
 - Click to place the action icon.



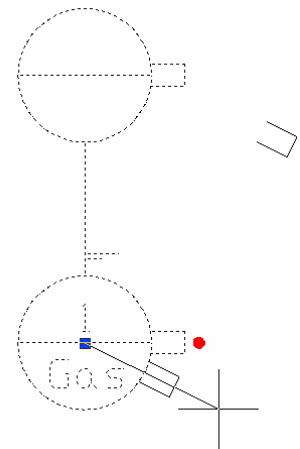
You will add a second rotation action for the tab on the top circle. To illustrate the need for independent base points, you will add this parameter the *wrong way* first, then you will go back and correct the problem to get the desired behavior.

6. Add a second rotate action for the tab on the top circle.
 - On the Actions tab of the Block Authoring Palettes window, click the Rotate Action tool.
 - Click the rotation parameter.
 - Select the lines making up the tab on the top circle for the action set.
 - Click to place the action icon.



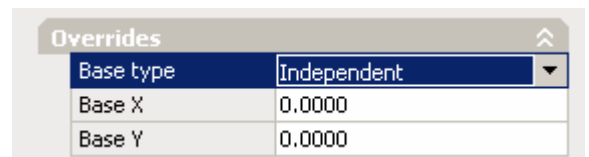
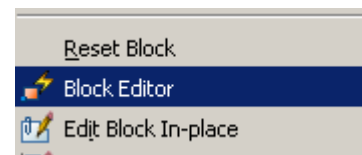
7. Save the block and return to the editor.
 - Click Close Block Editor on the toolbar.
 - Click Yes to save changes and return to the main drawing.

8. Experiment with the rotate grip.
 - Disaster! Notice that the tab along the bottom circle is moving around the circumference of the circle, but the tab along the top circle is separating.
 - Also, notice that both tabs are rotating about the center of the rotation parameter.

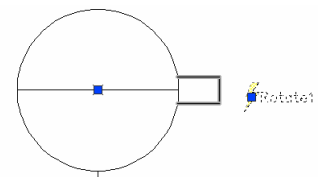


Tip: Scale and rotation parameters have a Base Point property that can be set to Dependent or Independent. Dependent base points use the associated key point of their associated parameter. Independent base points use a fixed point in the block. Action base points are displayed as a small X marker in the Block Editor.

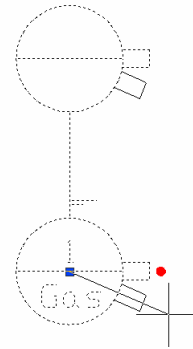
9. Edit the block to fix the problem.
 - Select the block.
 - Right-click and select Block Editor from the menu.
10. Change the top rotation action's base point type to Independent.
 - Select the top rotate action.
 - Notice how it shows a small X marker at the center of the rotation parameter.
 - Notice how it only has one grip--the one for the action icon.
 - In the Properties palette, change the Base Type to Independent.
 - Notice how a second grip appears.



11. Move the base point for the rotate action to the center of the top circle.
 - Click on the grip for the action's base point.



- Drag the grip to the center of the top circle.
12. Save the block and return to the drawing.
- Click Close Block Editor on the toolbar.
 - Click Yes to save changes and return to the main drawing.
13. Experiment with the rotate grip.
- Success! Notice that the tabs along both circles move around the circumference of the circle.



Moving Parameters and Actions with Actions: Order Dependencies

You can move parameters and other actions in a dynamic block just like other geometry by including them in the selection set of objects when you define an action. Parameters can move, rotate, stretch, scale and flip when they are in an action's selection set. Independent base points on actions can move, rotate, stretch, scale and flip when the actions are in a selection set. Angle offsets on actions can rotate and flip when the actions are in a selection set.

For example, suppose you have a linear parameter that stretches some geometry in your block. Suppose you have a rotation parameter and a rotate action that rotates the geometry about some fixed point. By including the linear parameter and its stretch action in the rotate action's selection set, you can rotate the linear parameter and its grips right along with the rest of the geometry.

Associated actions *are not* triggered when a parameter's key point or other properties (e.g., distance properties) change as a result of being in an action's selection set.

You may not get the expected behavior when an action operates on other actions or parameters, depending on the objects in an action's selection set. Unexpected behavior typically occurs when two or more actions operate on the same geometry, but you did not include the other actions and parameters in the selection set.

In many cases, this leads to blocks whose appearance depends on the order in which they are modified. We call blocks that have the same properties but different appearances *order dependent blocks*.

For example, you might insert two instances of a block that has a Point and a Rotation parameter, acting on the same geometry. On the first block you set the position of the point parameter, then the rotation. On the second block you set the same values but in the opposite order. When you are done you find that while the blocks have the same property values, the results are very different.

The figures below demonstrate the concept of order dependence graphically. The same block has had the same values applied to the Position and Rotation properties, but in opposite order.



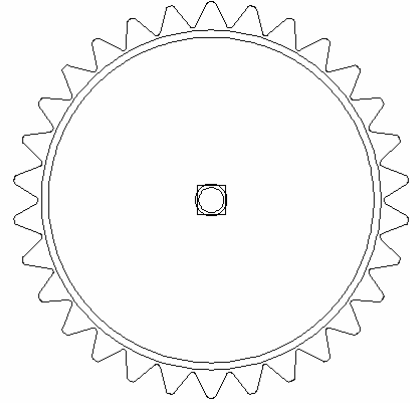
Position X	0.5000
Position Y	0.5000
Angle	45

You are most likely to get the expected behavior when including a parameter or action in an action's selection set while also including all of the geometry controlled by the parameter or action.

Lesson 10 Moving Parameters and Actions with Actions: Order Dependencies

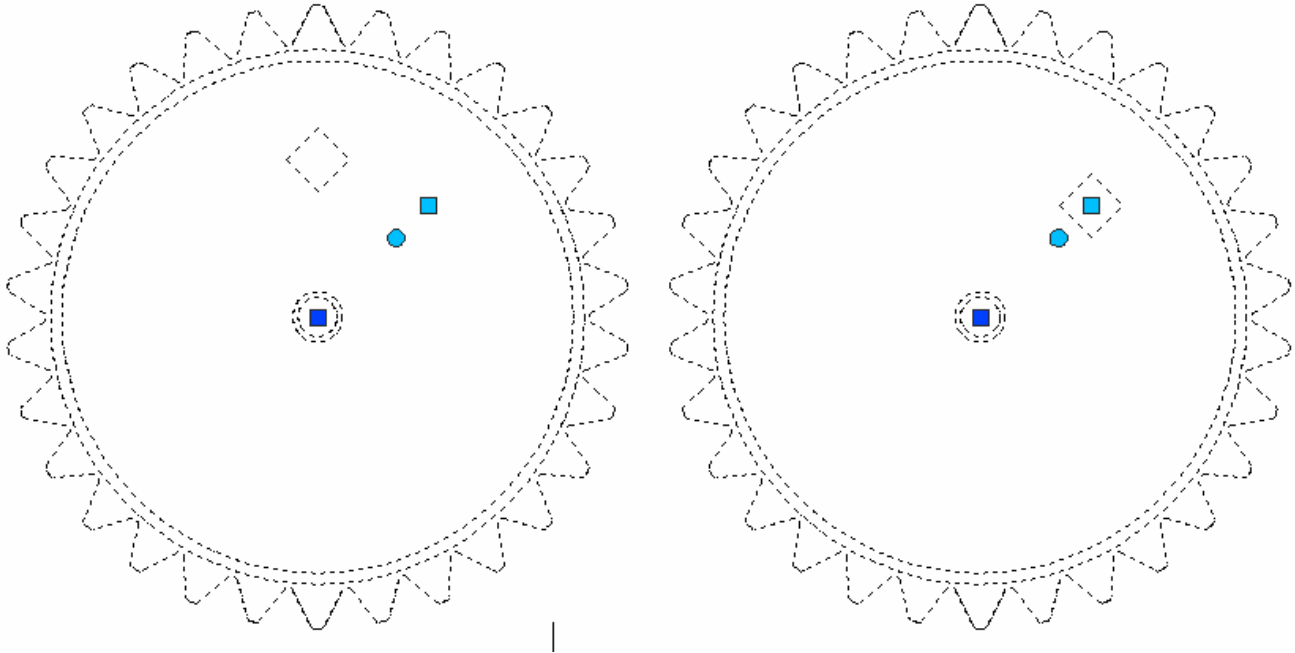
Objective: Learn how to transform actions or parameters with actions by putting them in the selection set of an action. Learn about order dependency in the way actions are applied.

In this lesson we start with a dynamic block of a gear with a key hole. The block uses a rotate action and a move action to reposition and rotate the key hole. The block does not function very well because both parameters are acting directly on the object, ultimately fighting for control over how the object is positioned. You will modify this block so that one parameter acts on the geometry, and the other parameter acts on both the geometry and the other parameter.

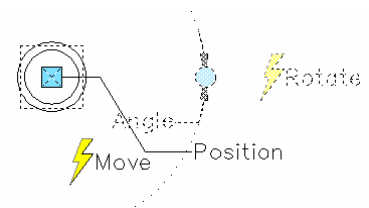


1. Create a new drawing.
2. Insert two copies of the *Gear.dwg* drawing as a block
 - Enter **insert** on the command line.
 - Click Browse....
 - In the Select Drawing File dialog box, select the *Gear.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
 - Repeat to insert a second copy.
3. Change the Position and Angle properties of the two blocks to the same value *but apply them in a different order*.
 - Select one of the blocks. In the Properties palette:
 - a. Set the Position X and Y properties to 5.
 - b. Set the Angle property to 45.
 - Select the other block. In the Properties palette:
 - a. Set the Angle property to 45.
 - b. Set the Position X and Y properties to 5.

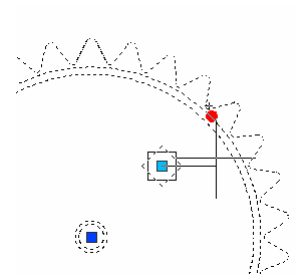
- Notice that while the grip positions and property values for the two gears are now the same, they are *not* the same gear block!



- Since both actions are acting on the square, the order in which the properties change affects the location of the final geometry in the block.
4. Fix the problem by making the move action in the block move the rotation parameter along with the box, so that the center of rotation remains centered in the box.
 - Double-click on the block, or select it then right-click and select Block Editor from the menu.
 - Select the “Gear” block from the list.
 - Click OK to close the dialog box and open the Block Editor.
 5. Add the rotation parameter and action to the Move action’s selection set
 - Enter **bactionset** on the command line.
 - Select the move action.
 - Select the Modify option to add or remove objects from the existing set. Notice how the box is automatically selected because it is already in the action’s selection set.
 - Pick the rotate parameter to add it to the selection set.
 - End the command by pressing ENTER or the space bar.
 - Now when the Position property changes and triggers the move action, both the rotation parameter and the rectangle will be moved.



6. Save the block and return to the drawing.
 - Click Close Block Editor on the toolbar.
 - Click Yes to save changes and return to the main drawing.
7. Test the modified blocks.
 - Notice how the existing blocks now match. Because the changes to the definition were propagated to the blocks they have both been updated to use the new action sets.
 - Select one of the blocks.
 - Move the square position grip. Notice how the rotation grip moves along with it.
 - Move the round rotation grip. Notice how the center of rotation for the grip, and for rotating the box, is always the center of the box.



Triggering Actions on Other Parameters: Chaining Actions

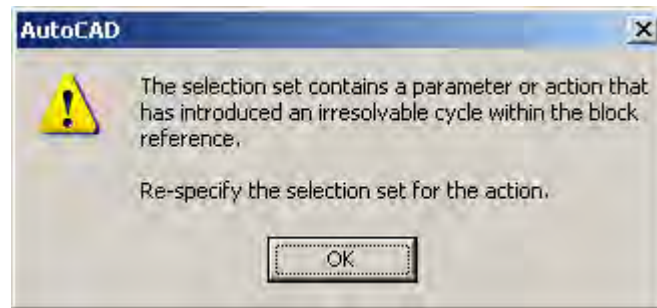
Normally, actions are not triggered by actions that move, rotate, scale, or stretch other parameters. There are times when it is desirable to trigger actions dependent on a parameter when the parameter moves, stretches, or rotates as the result of another action.

AutoCAD uses the term *chaining* to describe whether AutoCAD triggers actions on a parameter when the parameter changes as the result of another action. AutoCAD disables chaining by default, but you can set the Chain Actions property by selecting a Parameter in the Block Editor and changing the property in the Properties palette. Chaining applies to all actions associated with the parameter.

Show Properties	Yes
Chain Actions	No
Number of Grips	1

When you use chaining, it is possible to introduce cyclical dependencies between parameters in a block. For example, a cycle would exist if an action acts on a parameter with chaining enabled, and one of the actions on that parameter operates on the original action or its parameter. AutoCAD cannot perform the action because action A triggers action B which triggers action A which

AutoCAD issues an alert when you try to create a selection set for an action that would introduce cycles between actions:



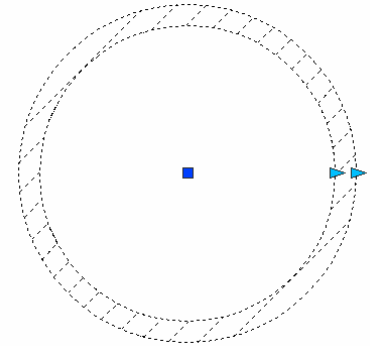
You may find it difficult to decide whether to use chaining when authoring a block. Many problems that can be solved with chaining can also be solved without chaining by simply having both actions act on the same geometry. You should consider avoiding chaining if possible because of the hidden complexities and potential for cycles. However, there are some cases where chaining is particularly useful:

- When a block has common actions that would have to be duplicated between parameters. In this case, manipulating the common parameter with another action and enabling chaining avoids duplicating actions.
- When an action has geometric dependencies that cannot be satisfied by one parameter alone. For example, the Scale action computes the scale factor as the ratio of the new parameter length to the original distance. If you want one grip or property to scale objects by different amounts for the same change in the grip or property you can only achieve this by having two separate linear parameters tied together with chaining.

Lesson 11 Direct Parameter Relationships: Chaining Actions

Objective: Learn how to use action chaining to cause a change in one parameter to trigger another parameter's actions.

In this example, you will modify a simple dynamic block representing the cross-section of a pipe. The block has two linear grips. The outer grip changes the pipe diameter by scaling both the inner and outer circles as well as the parameter associated with the inner grip, and as a result changing the pipe diameter does not maintain the wall thickness. The inner grip changes the wall thickness. You will modify this block so that the outer grip changes the pipe diameter while maintaining a constant wall thickness.



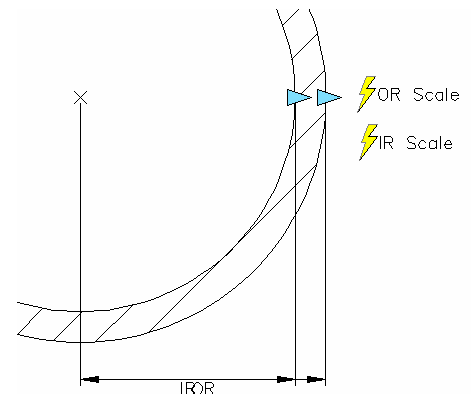
1. Create a new drawing.
2. Insert the *Pipe.dwg* drawing as a block.
 - Enter **insert** on the command line.
 - Click Browse...
 - In the Select Drawing File dialog box, select the *Pipe.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
3. Examine the block properties
 - Select the block and examine the inner radius (IR) and outer radius (OR) properties in the Properties palette. Notice that the outer radius is 1.0 and the inner radius is 0.875. The wall thickness is $1.0 - 0.875 = 0.125$.

Custom	
OR	1.0000
IR	0.8750

4. Click on the outer grip and move it so that the outer radius is 2.0
5. Examine the block properties.
 - Select the block and examine the inner and outer radii again. Notice how the outer radius is double (2.0), as is the inner radius. (1.75). Notice how the wall thickness scaled with the pipe instead of maintaining a constant value ($2.0 - 1.75 = 0.25$, compared to 0.125 before moving the grip).

Custom	
OR	2.0000
IR	1.7500

6. Open the block in the Block Editor.
 - Double-click on the block, or select it then right-click and select Block Editor from the menu.
 - Select the "Pipe" block from the list.
 - Click OK to close the dialog box and open the Block Editor.
7. Examine the existing block in the drawing.
 - Notice that the block has two parameters IR (inner radius) and OR (outer radius).
 - Notice that each parameter has an associated scale action: IR Scale and OR Scale.
 - Notice that the IR Scale action scales the inner circle.
 - Notice that the OR scale action scales both circles as well as the IR parameter.



In this block, the parameters manipulate the geometry independently. The inner radius parameter updates all of the required geometry when it changes, and the outer radius parameter updates all of the required geometry *plus* the inner radius parameter to keep its radius coincident with the inner circle. Because the OR Scale action applies only a single scale factor to all of the geometry, the relative radii of the inner and outer circles maintain their proportion when the OR Scale action operates on them. This *does not* maintain a constant wall thickness, but a constant relative scale.

For this block, you want the outer parameter to stretch the inner parameter, and for the inner parameter's IR Scale action to be triggered by the IR parameter being stretched by the outer parameter. Normally, when a parameter (or grip) changes directly from another action, the changed parameter's actions are not triggered.

Dynamic blocks use the term *chaining* to describe whether to perform actions driven by a parameter when the parameter changes directly as a result of another action.

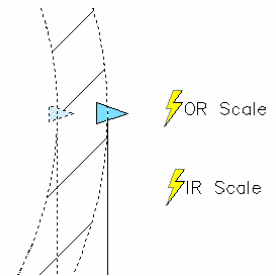
8. Change the IR parameter Chain Actions property to Yes.

- Select the IR parameter.
- Change the Chain Actions property to Yes in the Properties palette.

Misc	
Base location	Startpoint
Show Properties	Yes
Chain Actions	Yes
Number of Grips	1

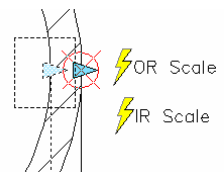
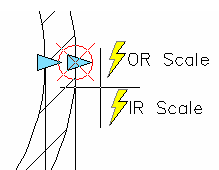
9. Change the OR Scale action so that it only scales the outer circle.

- Double-click on the OR Scale action. This starts the BACTIONSET command, which can be used to modify the selection set for an action.
- Notice that the command highlights the objects that are currently in the action's selection set.
- Remove the inner circle, IR parameter, and IR grip from the selection set.
 - a. Enter r to remove objects from the selection set.
 - b. Select the IR Parameter.
 - c. Select the IR grip.
 - d. Select the inner circle.
 - e. Leave the outer circle selected. Press ENTER or the space bar to end the command.



10. Add a stretch action to stretch the IR parameter when the OR grip moves.

- On the Actions tab of the Block Authoring Palettes window, select the Stretch Action tool.
- Select the OR parameter, then click on the OR grip to confirm the parameter point for the action.
- Select a stretch frame that includes the IR parameter grip.
- Select the IR parameter and grip for the selection set.
- Place the stretch action.

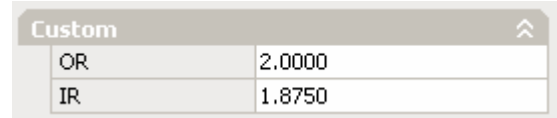


11. Save the block and return to the editor.

- Click Close Block Editor on the toolbar.
- Click Yes to save changes and return to the main drawing.

12. Examine the block properties

- Select the block and examine the inner and outer Radii again. Notice that the OR property is still 2.0, but the IR property is now 1.875. Notice how the wall thickness no longer scales with the pipe. ($2.0 - 1.875 = 0.125$).

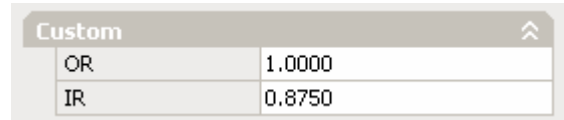


Custom	
OR	2.0000
IR	1.8750

13. Click on the outer grip and move it to 1,0.

14. Examine the block properties.

- Select the block and examine the inner and outer Radii again. Notice how the outer radius is half (1.0), and that the inner radius (0.875) stretched along with the circle to maintain the wall thickness ($1.0 - 0.875 = 0.125$).



Custom	
OR	1.0000
IR	0.8750

Legacy Application Compatibility and Dynamic Blocks

Many custom applications extract information from block references in drawings, for example to count blocks or to extract detailed information stored in block attributes. Many of these applications use the name of the referenced block to limit the search or otherwise filter and organize the results.

When you modify a dynamic block reference's grips or properties so that its geometry differs from the original block definition, AutoCAD creates an anonymous block containing the correct geometry and adds it to the drawing. Anonymous block names start with "*"U" followed by a unique number. Anonymous blocks are not visible in the user interface and you cannot normally insert or edit them². AutoCAD optimizes the use of anonymous blocks: if two block references in your drawing need the same geometry, AutoCAD creates one anonymous block and shares it between the references. AutoCAD purges unused anonymous blocks automatically when you save and re-open a drawing, or you can purge them yourself using the PURGE command.

Many of these applications need to be updated to work with dynamic blocks because of the way AutoCAD manages the anonymous blocks used to draw dynamic blocks. To existing applications, a modified dynamic block which references an anonymous block looks like an anonymous block, not like the block you inserted to create the reference.

AutoCAD 2006 introduced several new programming interfaces that applications can use to interact with dynamic block definitions and dynamic block references which are listed in Appendix E.

The most notable change is the introduction of tests for whether a block reference is a dynamic reference, and of a new EffectiveName property on block references. The EffectiveName property augments the existing Name property. Where the old Name property returns the name of the block definition *containing the geometry to draw* for the reference, the new EffectiveName property returns the name of the block *used to insert the block*.

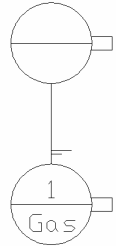
Both properties can be examined at any time for any kind of block reference. For old blocks and for unmodified dynamic blocks, the two properties return the same block name. For modified dynamic blocks the Name property returns the name of the anonymous block generated for the reference, while the EffectiveName property returns the name of the original dynamic block definition.

² AutoCAD exposes anonymous blocks through various programming interfaces to applications. Custom applications may expose anonymous blocks directly to the user.

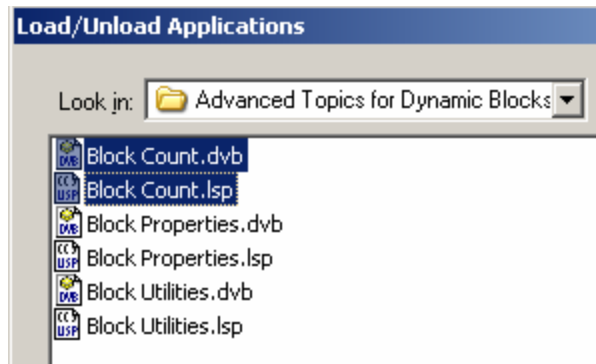
Lesson 12 Legacy Application Compatibility

Objective: Learn how dynamic blocks are represented in the database. Learn about the IAcadBlockReference2 interface and how to determine if a block is a dynamic block, how to elaborate the custom properties and change them from a script.

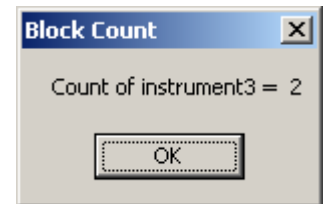
1. Create a new drawing.
2. Insert two copies of the *Instrument3.dwg* drawing as a block.
 - Enter **insert** on the command line.
 - Click Browse...
 - In the Select Drawing File dialog box, select the *Instrument3.dwg* file. Click OK to close the dialog box.
 - Click OK to close the Insert dialog box and insert the block.
 - Repeat these steps to insert a second copy of the block.



3. Load the Block Count VBA and VisualLisp macros that define commands for listing information about block definitions and block references.
 - Enter **apload** on the command line.
 - Select the *Block Count.dvb* and *Block Count.lsp* files.
 - Click Load to load the files.
 - Click Close to close the Load/Unload Applications dialog box.



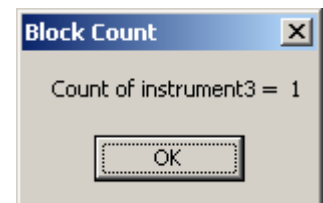
4. Use the CountBlockRefs command to count the number of Instrument3 blocks in the drawing.
 - Run the CountBlockRefs command.
 - Type "Instrument3" for the block name. Note the dialog counts 2 Instrument3 blocks in the drawing.
 - Click OK to close the dialog box.



5. Change the Tab Angle on one of the instruments to 45.
 - Select the instrument.
 - In the Properties palette change the Tab Angle property to 45.



6. Use the CountBlockRefs command to count the number of Instrument3 blocks in the drawing.
 - Run the CountBlockRefs command.



- Type “Instrument3” for the block name. Note the dialog counts 1 “Instrument3” block in the drawing. Yikes!
- Click OK to close the dialog box.

7. Examine the CountBlockRefs macro in the VBA editor.

- From the menu click on Tools->Macro->Visual Basic Editor.
- This is an “old-style” VBA macro for counting block references. For each block reference in the current space (paper/model), it compares the AcadBlockReference::Name property to the string entered by the user:

```

32 For Each blockref In activeSpace
33
34     ' In AutoCAD 2006, the IAcadBlockReference2::EffectiveName property
35     ' returns the name of the block definition used to insert the block
36     ' reference.
37
38     'Set dynblockref = blockref
39     'If StrComp(dynblockref.EffectiveName, blockname, vbTextCompare) = 0 Then
40
41     ' Prior to AutoCAD 2006, the AcadBlockReference::Name property
42     ' returned the name of the block definition used to draw the block.
43     ' This is still true, but it may not be the name of the block definition
44     ' used to insert the block reference.
45
46     If StrComp(blockref.Name, blockname, vbTextCompare) = 0 Then
47         count = count + 1
48     End If
49
50 Next

```

The problem with the script is that the AcadBlockReference::Name property always returns the *name of the block used to draw the reference*. Dynamic blocks sometimes use the original block to draw themselves, but may also use anonymous block names. This macro needs to be updated to work with dynamic blocks.

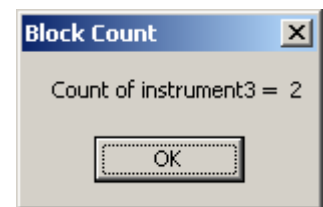
Tip: Use the AcadBlockReference::Name when you need the block definition name used to draw a block. Use IAcadBlockReference2::EffectiveName when you need to know the name of the block that was inserted. Both properties work on dynamic and non-dynamic blocks.

8. Update the script to use the new IAcadBlockReference2 interface and compare the IAcadBlockReference2::EffectiveName property to the desired block name

- Delete the existing “If StrComp(blockref...” line of code at line 46, and the comment immediately preceding it.
- Uncomment the lines beginning with “Set dynblockref = ...” and “If StrComp(dynblockref...” at line 38. When you are done the loop should look like this:

```
32 For Each blockref In activeSpace
33
34     ' In AutoCAD 2006, the IAcadBlockReference2::EffectiveName property
35     ' returns the name of the block definition used to insert the block
36     ' reference.
37
38     Set dynblockref = blockref
39     If StrComp(dynblockref.EffectiveName, blockname, vbTextCompare) = 0 Then
40         count = count + 1
41     End If
42
43 Next
```

9. Use the CountBlockRefs command to count the number of Instrument3 blocks in the drawing.
 - Return to the drawing.
 - Run the CountBlockRefs command.
 - Type "Instrument3" for the block name. Note the dialog counts 2 "Instrument3" block in the drawing.
 - Click OK to close the dialog box.



Appendix A: Parameter and Action Combinations³

Parameter Type	Description	Supported Actions
Point	Defines an X and Y location in the drawing. In the Block Editor, looks similar to an ordinate dimension.	Move, Stretch
Linear	Shows the distance between two anchor points. Constrains grip movement along a preset angle. In the Block Editor, looks similar to an aligned dimension.	Move, Scale, Stretch, Array
Polar	Shows the distance between two anchor points and displays an angle value. You can use both grips and the Properties palette to change both the distance value and the angle. In the Block Editor, looks similar to an aligned dimension	Move, Scale, Stretch, Polar Stretch, Array,
XY	Shows the X and Y distances from the base point of the parameter. In the Block Editor, displays as a pair of dimensions (horizontal and vertical).	Move, Scale, Stretch, Array
Rotation	Defines an angle. In the Block Editor, displays as a circle.	Rotate
Flip	Flips objects. In the Block Editor, displays as a reflection line. Objects can be flipped about this reflection line. Displays a value that shows if the block reference has been flipped or not.	Flip
Alignment	Defines an X and Y location and an angle. An alignment parameter always applies to the entire block and needs no action associated with it. An alignment parameter allows the block reference to automatically rotate around a point to align with another object in the drawing. An alignment parameter affects the rotation property of the block reference. In the Block Editor, looks like an alignment line.	None (The action is implied and contained within parameter.)
Visibility	Controls the visibility of objects in the block. A visibility parameter always applies to the entire block and needs no action associated with it. In a drawing, you click the grip to display a list of visibility states available for the block reference. In the Block Editor, displays as text with an associated grip.	None (The action is implied and controlled by visibility states.)
Lookup	Defines a custom property that you can specify or set to evaluate a value from a list or table you define. It can be associated with a single lookup grip. In the block reference, you click the grip to display a list of available values. In the Block Editor, displays as text with an associated grip.	Lookup
Base	Defines a base point for the dynamic block reference relative to the geometry in the block. Cannot be associated with any actions, but can belong to an action's selection set. In the Block Editor, displays as a circle with crosshairs.	None

³ From the topic *Use Parameters in Dynamic Blocks* in the *AutoCAD 2006 User's Guide*

Appendix B: Dynamic Block Commands

BACTION	BCLOSE	BSAVEAS
BACTIONSET	BCYCLEORDER	BVHIDE
BACTIONTOOL	BEDIT	BVSHOW
BASSOCIATE	BGRIPSET	BVSTATE
BATTORDER	BLOOKUPTABLE	RESETBLOCK
BAUTHORPALETTE	BPARAMETER	
BAUTHORPALETTECLOSE	BSAVE	

Appendix C: Dynamic Block SYSVARS

BACTIONCOLOR	BLOCKEDITLOCK	BPARAMETERSIZE
BDEPENDENCYHIGHLIGHT	BLOCKEDITOR	BTMARKDISPLAY
BGRIPCOLOR	BPARAMETERCOLOR	BVMODE
BGRIPOBJSIZE	BPARAMETERFONT	GRIPDYNCOLOR

Appendix D: Commands and Sysvars disabled in the Block Editor

3DSOUT	LAYMRG	SUPERHATCH
3DARRAY	MVIEW	SHOWURLS
ATTEXT	MVSETUP	TXTEXP
BASE	PAGESETUP	UCS
BLOCK	PLOT	UCSMAN
BLOCKREPLACE	PSETUPIN	VIEW
DXFOUT	PREVIEW	VPORTS
DDVIEW	PSOUT	VIEWPORTS
DDVPOINTS	PURGE	WBLOCK
DDUCS	QSAVE	WMFOUT
EDGE	RENAME	WSSAVE
EXPORT	RTUCS	WORKSPACE
HIDE	RENDER	
LAYOUT	SAVE	
LAYOUTWIZARD	SAVEAS	
LAYDEL	SHADE	

Appendix E: Dynamic Block Programming Interface Summary

Object	Programming Interface	Capabilities
Dynamic Block Reference	C++ AcDbDynBlockReference VBA/VisualLisp IAcadBlockReference2 .Net BlockReference	Expose test for whether a block reference is a dynamic block. Has method for resetting dynamic blocks, accessing collections of dynamic block properties, and getting the original (effective) block name.
Dynamic Block Property	C++ AcDbDynBlockReferenceProperty VBA/VisualLisp IAcadDynamicBlockReferenceProperty .Net DynamicBlockReferenceProperty DynamicBlockReferencePropertyCollection DynamicBlockReferencePropertyCollectionEnumerator	Expose information about a dynamic block custom property, including the property name, current value, and range of allowed values. Also has a method for setting the current property value.
Block Definition	C++ AcDbDynBlockReference AcDbBlockTableRecord VBA/VisualLisp IAcadBlock3 .Net BlockTableRecord	Exposes methods for testing whether a block definition contains dynamic block information. Also exposes the block units and whether references to the block can be exploded or scaled.