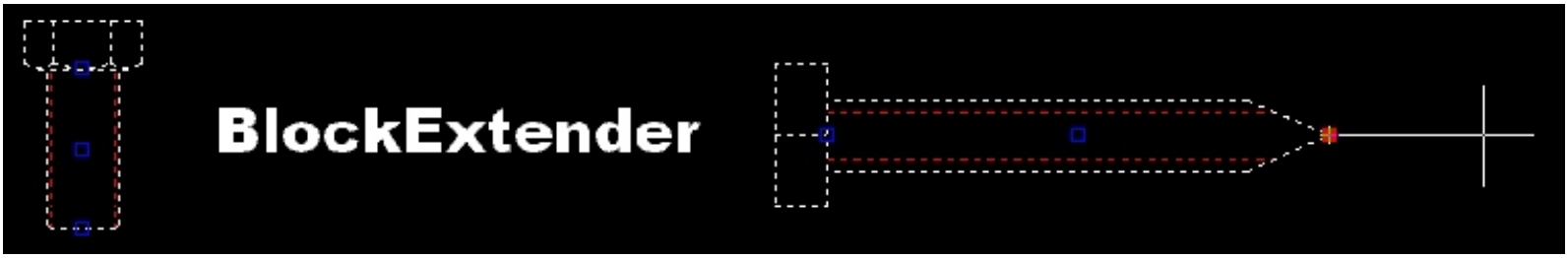
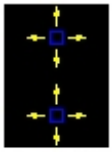


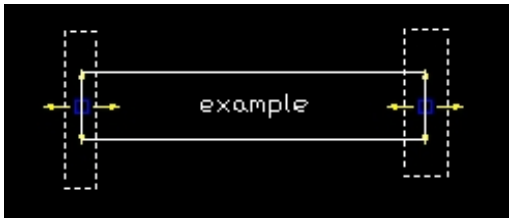
Intelligence Examples (Full Version Only)

Below I will demonstrate two different step by step examples that illustrate two different but common uses of intelligent blocks. These examples will show how to use the most common functions of BlockExtender's intelligent grip points.

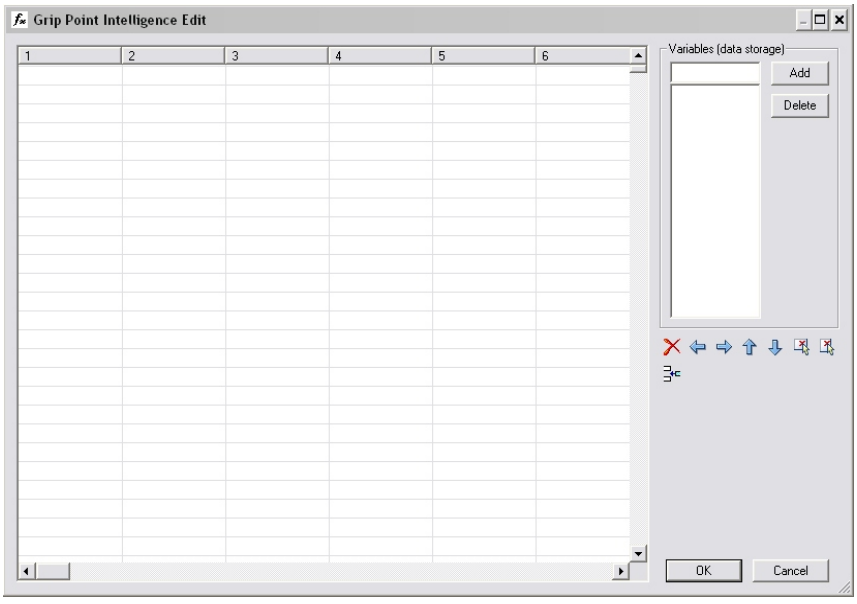
Example #1 - Intelligence1.dwg

This first example will test the distance between two known points of the drawing and show the results by changing the text of a text drawing object. Begin by drawing a simple rectangle, place text at the center, and add grips using the Line Multi Direction grips and pick each end, as shown below.


Line Multi Direction



Next type **iedit** at the command line and pick one of the grip points. The main grip point intelligence edit dialog box will now appear.



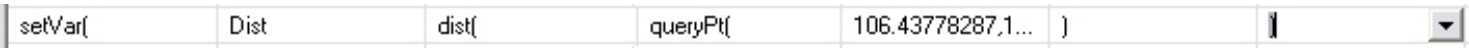
Now we are going to add the functions to make this drawing intelligent. Start by adding a variable named "Dist" to the variable list box in the upper right corner. Next select the top left cell of the spread sheet. Click on the drop down arrow and select "Set variable". The function "setVar(" will now be present in that cell. The function is used to store data in a variable.

You will notice that once the function name was inserted two blank spaces were inserted followed by a). This

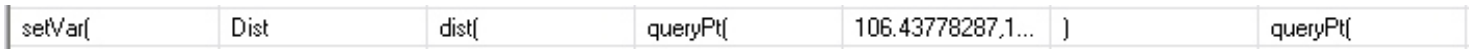
indicates that two input arguments are expected and the) indicates the end of the input arguments. In this case "setVar(" expects the name of the variable to be set, followed by the value to set the variable with.

Select the next cell to the right and type in the name of the variable "Dist". This defines variable to be set.

Select the next cell to the right and click on the drop down arrow. Select "Point Math Functions" then "distance between". The function "dist(" will be inserted into the cell with two more blank cells inserted for arguments.



The function "dist(" is used to extract the distance between two designated points in the block drawing. To set these designated points do the following steps in each of the new blank cells of the "dist(" function.
Select the drop down arrow in the first blank cell and select "Query Objects" then "Query point". The dialog box will then be hidden and you will prompted to select a point on screen. Pick the first grip. The function "queryPt(" will be inserted along with the point coordinates picked.



Now to select the second grip pick the second ")" (as shown above) after the point that was inserted. Repeat the selection from the drop down menu and select "Query Objects" then "Query point". The cells starting at the ")" will be automatically shifted over and the second "queryPt(" will be inserted.

Lets review what we have done so far. We have setup the function "setVar(" in cell 1 to assign a data value into the variable named "Dist" in cell 2. The value being passed in to cell 3 is the distance between the two grip points. The distance is extracted by using the function "dist(" in cell 3. The function "dist(" uses two input arguments to find and calculate those two grip points by using the function "getAtPoint(" in cells 4 and 7. This is the first step of our calculations.

Important Note: Ensure that all functions that accept input arguments have a matching ")" to correctly close the function so the calculation(s) can be made correctly.

Next we need to setup an "if" statement so we can test the value of "Dist" variable. In row two, cell 1 click on the drop down button and select "If Conditions" then "if".

The function "if" will be insert along with a "{" on the next line and a "}" three lines below it.

To add the conditional test select the cell to the right of "if" and click on the drop down button and select "If Conditions" then "<=".

Here we are going to test to see if the “Dist” value is equal to or less than 100.
In cell 3 place the variable “Dist” and cell 4 place the value to test it by, being 100. Now if the distance is 100 or less the functions place inside the “{“ and “}” will be calculated.

Now from between the “{“ and ”}” select the first cell. If there is not an empty row between the two “{“ and “}”, select the “}” cell and press the insert row button  to insert a new row.

Now click on the drop down arrow (as show above) and select “Actions” then “Set Text”. The dialog box will hide itself and you will be prompted to select a text, or mtext object. Select the text entity that you would have drawn in before as the screen capture of the block shows on the previous page. The row will now appear as below.

if	(<=	Dist	100	)		
{						
}						

Now we need to state what text to display in the text drawing entity. To do so type in the variable “Dist” in cell 5 to specify what text will be displayed by the text drawing entity. In this case it will display the value of “Dist” being the distance between the two specified points.

if	(<=	Dist	100	)		
{						
setText(	getEnt(	id:1074343336	)		)	
}						

When a number is passed into a function like “setText(“ that requires a text value, the numeric value will be translated into a text string using the parent drawing’s current units. So for example if the drawing that the block was inserted into has its units set to architectural with an accuracy of 1/16” the numeric value will be displayed in that format.

Lets review again what we have done. We have created an “if” statement that is used to test the value of “Dist” variable. Using “(<=” we are testing to see if that value is less than or equal to 100.

Below the “if” statement two brackets (“{“ and “}”) have been inserted. These indicate what functions to calculate if the value is calculated as 100 or less. In these brackets we have place a function called “setText(” used to modify the text displayed by a text entity. When we selected this function using the drop down menu it prompted us to select a text drawing entity to modify. It then filled in “getEnt(“ which is used to extract the drawing entity and it has only one input argument which is an identifying id number of the drawing entity and it will be display as “id:” followed by a long number.

Now we want to put in an “else do”statement after the “if” statement so that if the value of “Dist” is larger then100. Select the cell directly below the last “}”. Click on the drop down arrow and select “If Conditions” then “else do”.

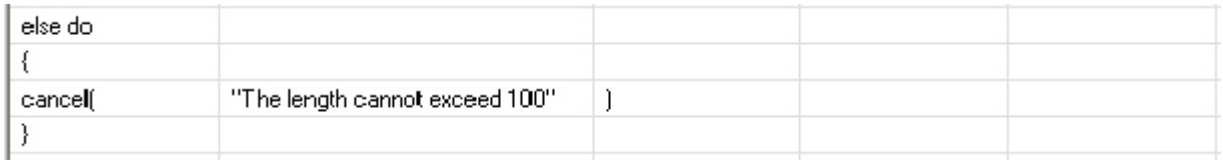
if	(<=	Dist	100	)		
{						
setText(	getEnt(	id:1074343336	)	Dist	)	
}						

The “else do” statement is similar to the “if” statement, except you cannot put any further conditions in. The “else do” statement is used to place in a function(s) that is to be calculated if the first “if” statement is not correct. In our case if the value of “Dist” is larger than 100.

Now we are going to set the text displayed of the text drawing entity to a text string that tells the user the length is too long. Follow the steps above to set the text. Select from the drop down menu “Actions” then “Set Text”. You will be prompted again to select the text drawing entity.

This time instead of placing “Dist” in, we are going to place the text string “too long” in it’s place. This will now display the text “too long” if the length between the two designated points exceeds 100.

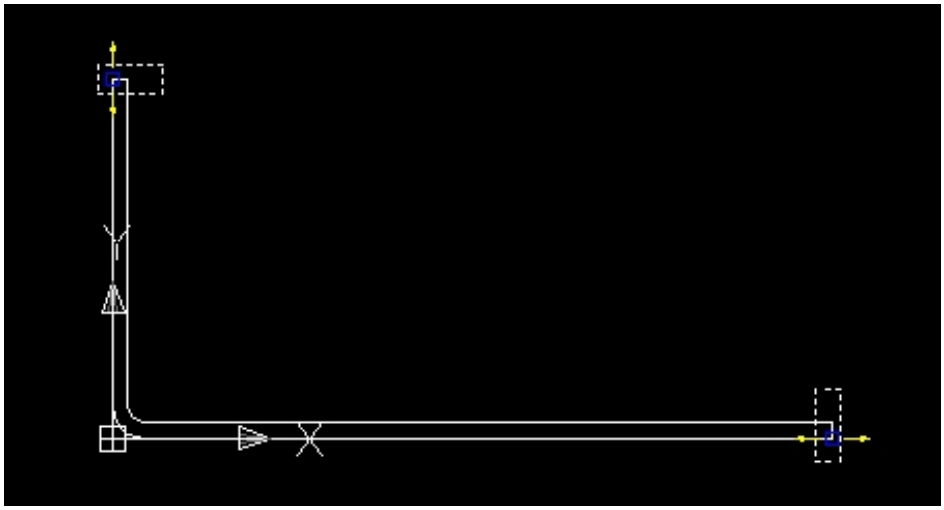
As an alternative to displaying text to tell the user the length is too long, we are going to call the “cancel(“ function. To undo the user’s stretch and inform them it’s was too long. Delete the “setText(“ row of the “else do” function and insert a new row. Then from the new row drop down button of cell 1, select “Cancel user’s action”.



Then from the empty cell after “cancel(“ place your cancellation text in that tells the user what has happened. In this case we are going to place the text “The length cannot exceed 100" (as shown above). Now if the user stretches the block too long they will be unable to get it to exceed 100 and a message will be printed to the command line explaining this. If not message is placed in the “cancel(“ function, no message will be displayed to the user.

Example #2 - AngleIron.dwg

This second example is designed to show how to create an engineering formula for a simple angle iron, as shown below.



In this example we will not follow the step by step procedures as shown in example #1, since we already learned how to work with the formula spread sheet. Instead in this example we will simply show you the formulas and explain how each row functions and what it does.

I will start by explaining the engineering rules we wish to apply to this block drawing. One leg of the angle iron will be ½ the length of the other. The longer leg can be stretched from 1" in length to 6" long, in ½" increments. The shorter leg will be ½" to 3" long in 1/8" increments.

To aid our formula we will rely on the fact that the insert section of the two arms of the angle iron will be at 0,0.

This first formula below is the formula applied to the lower right grip point. Notice that it is a horizontal stretching grip style.

setVar(	temp	getX(	queryThisPt()	)	)			
setVar(	temp	incCalc(	1	0.25	6	temp	)	)
setVar(	thisPoint	setPt(	temp	0	0	)	)	
setVar(	temp	(*	temp	0.5	)	)		
setVar(	otherPoint	setPt(	0	temp	0	)	)	
moveGripTo(	id:10744...	otherPoint	)					
moveThisGripTo(	thisPoint	)						

Now to explain row by row how the formula works. For further information on the individual functions, you can find a detailed explanation of each in the FormulaList.pdf.

The first row of the formula does three things. First the setVar(is used to set the data in a variable called “temp”. The functions getX(and queryThisPt(are used to set that variable.

setVar(	temp	getX(	queryThisPt()	)	)			

getX(is used to extract the X coordinate of the current position of the lower right grip point, which is retrieved by the function queryThisPt(.

The variable “temp” is now set the X coordinate or the length from the corner to the end of the larger arm of the angle iron.

The second row of the formula does two things. First the setVar(is used to reset the data in the variable called “temp”. The function incCalc(is used to set that variable.

setVar(	temp	incCalc(	1	0.25	6	temp	)	)

The reason we are resetting the variable “temp” is because we are passing in that variable as one of the arguments for the function incCalc(. You may find incCalc(useful to you in the future, it allows you to test a distance value against a known set of rules to determine the nearest incremental value. In this case we want to know the incremental value (min of 1" up to 6" with increments every ½").

What will happen when this row is processed is that the value of “temp” will be passed through incCalc(to get the nearest allowable incremental value and the variable “temp” will be set to that new value.

The third row of the formula does two things. It sets a variable called “thisPoint” to a point value.

setVar(	thisPoint	setPt(	temp	0	0	)	)	

Here “thisPoint” will be used to place the grip the user is stretching (the lower right grip) by passing in the value returned by te function setPt(. setPt(is passed in the variable “temp” in the X coordinate argument position followed by two 0's to set the Y and Z coordinates.

The fourth row of the formula does two things. It now sets the variable “temp” to the ½ of its original value.

setVar(	temp	(*	temp	0.5	)	)		

This is done to calculate the new length of the shorter arm. The variable “temp” now stores the length of the short arm of the angle iron

The fifth row of the formula does two thing. It now sets a variable called “otherPoint” to a point value so it can later be used to move the upper left grip of the angle iron.

setVar(	otherPoint	setPt(	0	temp	0	)	)	

Here setPt(is being passed a 0 for the X coordinate and the value of the variable “temp” is being passed into the Y coordinate position, followed by another 0 for the Z coordinate. This now effectively sets the variable “otherPoint” to the point that we want to move the upper left grip point.

The sixth row of the formula is moves the upper left grip to the coordinates stored by the variable “otherPoint”.

moveGripTo(	id:10744...	otherPoint	)					

The last row of the formula now moves the lower right grip point to the coordinates that are incrementally correct.

moveThisGripTo(	thisPoint	)						

This second formula below is the formula applied to the upper left grip point. Notice that it is a vertical stretching grip style.

setVar(	temp	getY(	queryThisPt()	)	)			
setVar(	temp	incCalc(	0.5	0.125	3	temp	)	)
setVar(	thisPoint	setPt(	0	temp	0	)	)	
setVar(	temp	(*	temp	2	)	)		
setVar(	otherPoint	setPt(	temp	0	0	)	)	
moveGripTo(	id:10744...	otherPoint	)					
moveThisGripTo(	thisPoint	)						

This second formula is almost identical to the first except the settings for the X and Y coordinates are reversed.

The first row of the formula does three things. First the setVar(is used to set the data in a variable called “temp”. The functions getY(and queryThisPt(are used to set that variable.

setVar(	temp	getY(	queryThisPt()	)	)			

getY(is used to extract the Y coordinate of the current position of the upper left grip point, which is retrieved by the function queryThisPt(.

The variable “temp” is now set the Y coordinate or the length from the corner to the end of the smaller arm of the angle iron.

The second row of the formula does two things. First the setVar(is used to reset the data in the variable called “temp”. The function incCalc(is used to set that variable.

setVar(	temp	incCalc(	0.5	0.125	3	temp	)	)

The reason we are resetting the variable “temp” is because we are passing in that variable as one of the arguments for the function incCalc(. You may find incCalc(useful to you in the future, it allows you to test a distance value against a known set of rules to determine the nearest incremental value. In this case we want to know the incremental value (min of ½" up to 3" with increments every 1/4").

What will happen when this row is processed is that the value of “temp” will be passed through incCalc(to get the nearest allowable incremental value and the variable “temp” will be set to that new value.

The third row of the formula does two things. It sets a variable called “thisPoint” to a point value.

setVar(	thisPoint	setPt(	0	temp	0	)	)	

Here “thisPoint” will be used to place the grip the user is stretching (the lower right grip) by passing in the value returned by te function setPt(. setPt(is passed in the variable “temp” in the Y coordinate argument position followed by two 0's to set the X and Z coordinates.

The fourth row of the formula does two things. It now sets the variable “temp” to the twice of its original value.

setVar(	temp	(*	temp	2	)	)		

This is done to calculate the new length of the longer arm. The variable “temp” now stores the length of the longer arm of the angle iron

The fifth row of the formula does two thing. It now sets a variable called “otherPoint” to a point value so it can later be used to move the upper left grip of the angle iron.

setVar(	otherPoint	setPt(	temp	0	0	)	)	

Here setPt(is being passed the value of the variable “temp” in the X coordinate position, followed by two 0's for the Y and Z coordinates. This now effectively sets the variable “otherPoint” to the point that we want to move the lower right grip point.

The sixth row of the formula is moves the lower right grip to the coordinates stored by the variable “otherPoint”.

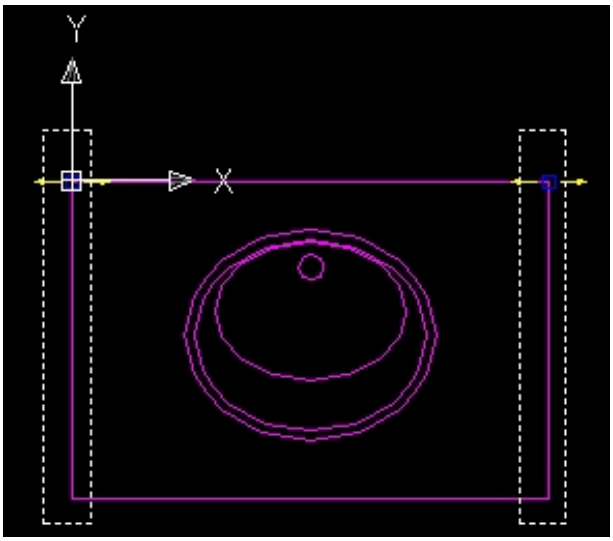
moveGripTo(	id:10744...	otherPoint	)					

The last row of the formula now moves the upper left grip point to the coordinates that are incrementally correct.

moveThisGripTo(	thisPoint	)						

Example #2 - AngleIron.dwg

This third example is designed to show how to center part of block between two points (grip points or otherwise).



I will start by explaining the engineering rules we wish to apply to this block drawing. The bathroom counter is to have two grip points. The user can stretch this counter out to the exact dimensions they need. The sink will always be display in the center or middle of the counter.

The insert point can be anywhere but we have chosen to place it the upper left corner of the counter.

The first grip point.

This first formula below is the formula applied to the left grip point. Notice that it is a horizontal stretching grip style.

setVar(	midPoint	mid(	queryPt(	0,0,0	)	queryPt(	36,0,0	)	)
movePtTo(	18,0,0	midPoint	)						

Now to explain row by row how the formula works. For further information on the individual functions, you can find a detailed explanation of each in the FormulaList.pdf.

The first row of the formula does two things. First the setVar(is used to set the data in a variable called "midPoint". This variable will store the calculated mid point between two designated points (our grip points). The functions queryPt(is called twice to get the coordinate of the first grip point and again for the second grip point.

setVar(	midPoint	mid(	queryPt(	0,0,0	)	queryPt(	36,0,0	)	)

Please note: The use of queryPt(allows to get the current coordinates of any point, even if there is no grip point located there.

The second row is used to relocate all drawing objects located at the current location of the indicated point to the new location stored in the variable “midPoint”.

movePtTo(	18,0,0	midPoint	)						

Please note: In the drawing counter.dwg we placed the sink in a block called “Sink” this way we only had to select the insert point of the sink which, by our design lies exactly on the mid point of the line between the two grip points.

The second grip point.
This second grip point formula (below) is exactly the same as the first grip point.

setVar(	midPoint	mid(	queryPt(	0,0,0	)	queryPt(	36,0,0	)	)
movePtTo(	18,0,0	midPoint	)						

Important:
Read each of the next few steps carefully.

Copying Grips, (Read Carefully)
First copy the first grip point over to the location where the second grip point is to go.

Now here is the tricky part, when you move, stretch, mirror, copy, etc. a grip point, any points stored in the formulas ***are recalculated as well***. This is useful for when you are copying drawing objects with the grip, so the grip will still work correctly with the drawing objects. But ***in this case*** we are not copying any drawing objects. So we must go into the formula using **iedit** again and reset the coordinates.

One at a time, select each of the 3 points/coordinates and reset them using the button: 

The formula dialog box will be hidden each time and you will be prompted to select a new point/coordinate. When all 3 points are reset, save the drawing and close it. You are now ready to insert it into another drawing for testing.

3rd Day Software
P.O. Box #808
Grande Cache, Alberta, Canada
T0E 0Y0
www.objectdcl.com/BlockExtender.html
sales@objectdcl.com