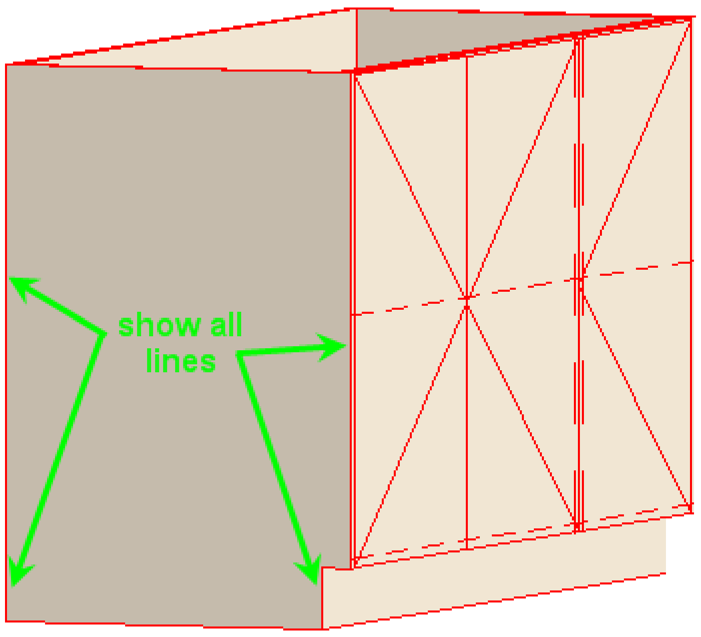
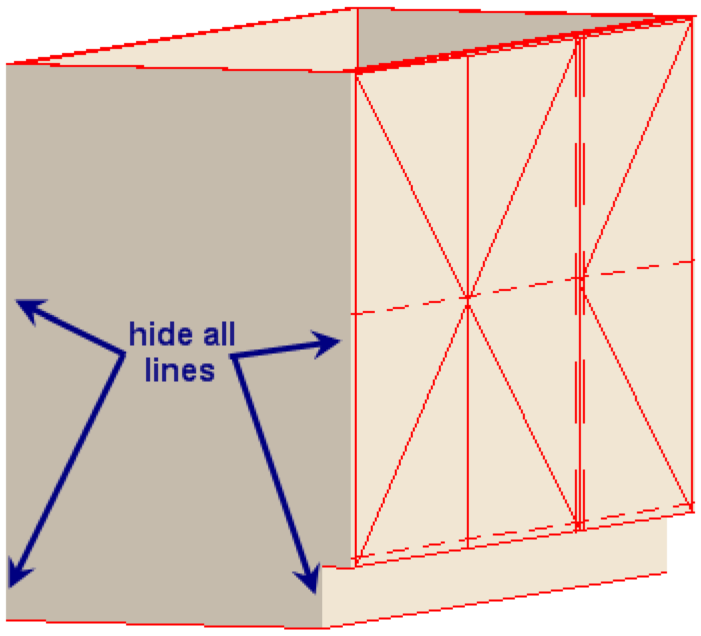
I am not sure that the lpxmlconverter is the answer to my question – although I must admit I really don't know what it does.
I am not interested in converting the actual object.

Let me see if I can explain my problem in more detail and you can tell me if the lpxmlconverter can do this.
Sorry in advance if this post get a little long.

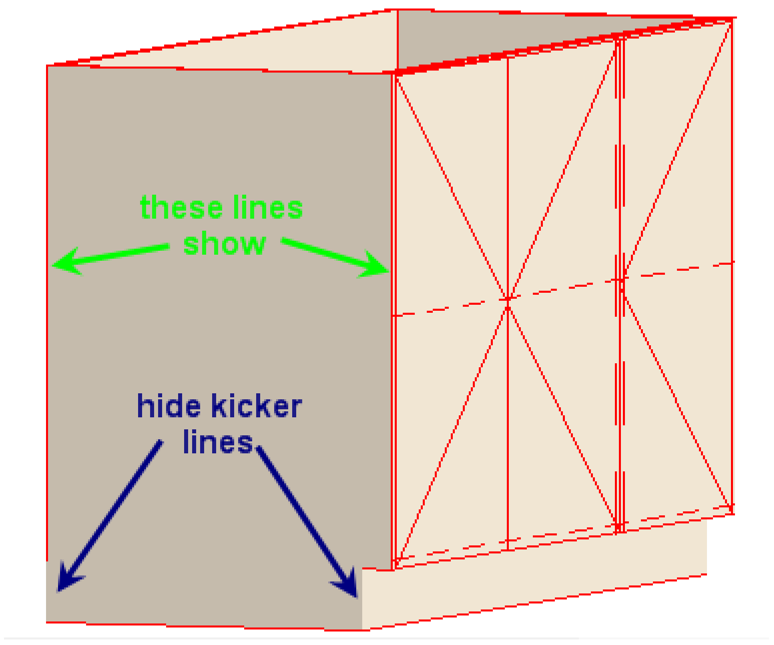
I have an object for a cabinet.
It has parameters that allow me to SHOW ALL the lines on the left side.



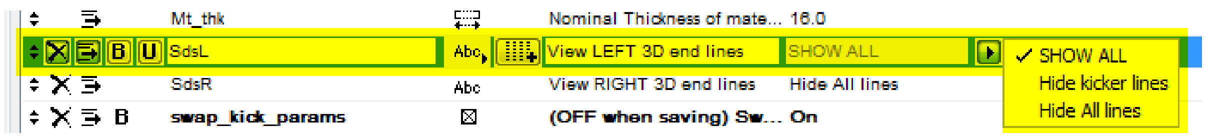
HIDE ALL lines



Or HIDE KICKER LINES

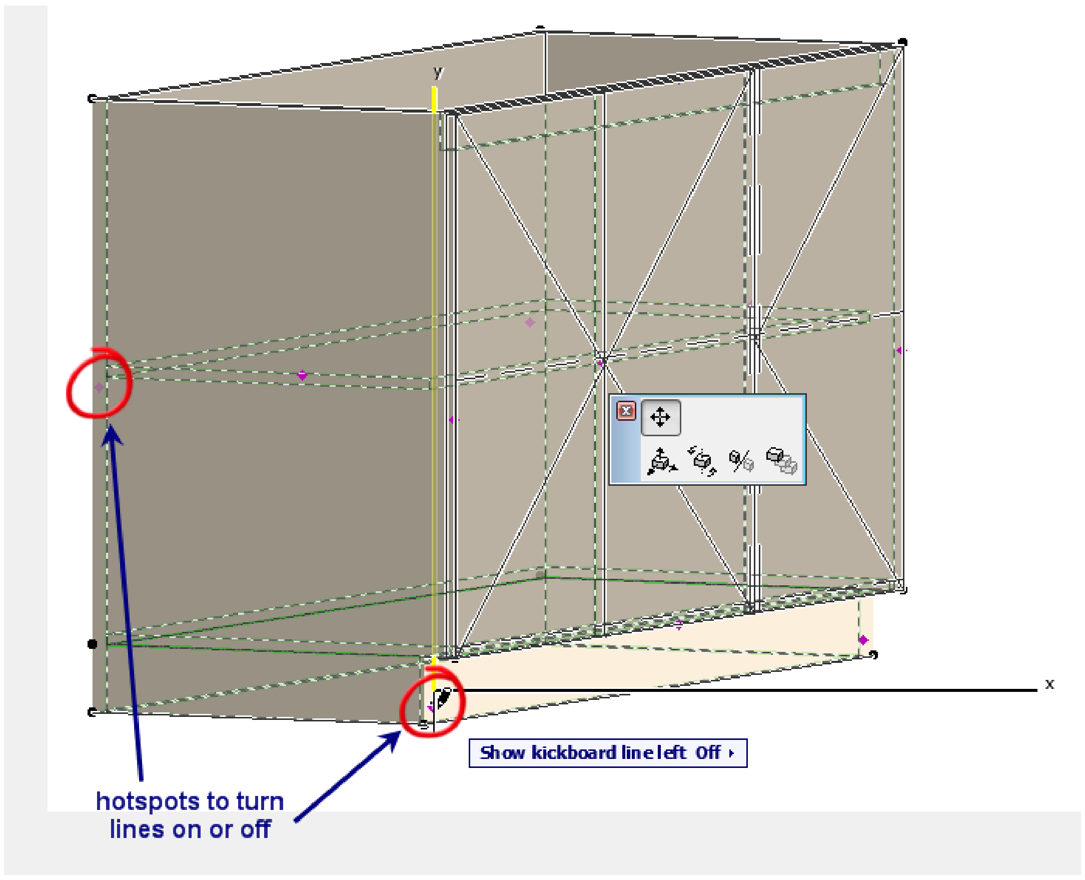


This is done using a text based parameter.



	Mt_thk		Nominal Thickness of mate...	16.0
B U	SdsL	Abc	View LEFT 3D end lines	SHOW ALL
	SdsR	Abc	View RIGHT 3D end lines	Hide All lines
B	swap_kick_params	☒	(OFF when saving) Sw...	On

What I have done now is add stretchy hotspots to the side so that by moving them up or down you can turn the side and kicker line on or off independently from the elevation or 3D view – no need to open the settings dialogue.



Now because I have dozens of these cabinets in thousands of existing jobs (and standard model plans for future jobs), I didn't simply want to create a new object as these hotspots would not be available in my existing jobs.

I could have just saved the existing object with these new hotspots and disabled the old parameter but that would mean that every job would have the wrong settings for the lines because some are all on some all off and some hide only the kickboard.
I can only save the object with one default setting for the new hotspots so in many jobs they would be wrong and would have to be reset individually.
So I said to myself if only I could determine the value of the original text variable when that object is used in an existing plan and then tell the new hotspots what to do (on or off) based on that text variable.
Then I thought that is easy to do with a simple IF/THEN statement – which I did.

Now the problem is that I have two parameters that control the visibility of the line so I hide the original text parameter so the user cannot see it.
But this creates another problem.
Because I can now switch the lines on and off independently I now have four combinations (all on, all off, kickers on/top off and kickers off/top on) which won't always match the original 3 text combinations.
If I leave both parameters in the object they will fight each other and I won't be able to control the status of the lines properly – i.e. I could never set kickers on/top off as it is not one of the text parameter options.

To solve this I use a boolean 'flag' variable called 'swap_kick_params'.
If it is '0' then the hotspot variables get set to match the text parameter.
Then this 'swap_kick_params' gets set to '1' so the script does not try to match the parameters ever again.

All this is fine and relatively simple until now.
I can't just have this in the script because even in the GDL editor the script runs, sets the hotspot parameters and switches 'swap_kick_params' to '1'.
Enter GLOB_CONTEXT

By doing this the parameters are not reset in the GDL editor.

```
if GLOB_CONTEXT <> 1 then
        swap_kick_params = 1
        PARAMETERS swap_kick_params = swap_kick_params
endif
```

When a job is opened with a cabinet object in it, regardless of the settings for the text parameter, as soon as that object is viewed in 2D or 3D, the hotspot parameters get set and the flag is tripped so they are never checked again.
It all worked so beautifully until version 19/20 came along.
This is just one of I don't know how many objects I have done similar things to.

If only you could keep GLOB_CONTEXT