

ARCHIFRAME

ArchiFrame instructions 8th September 2023

Pdf-version here

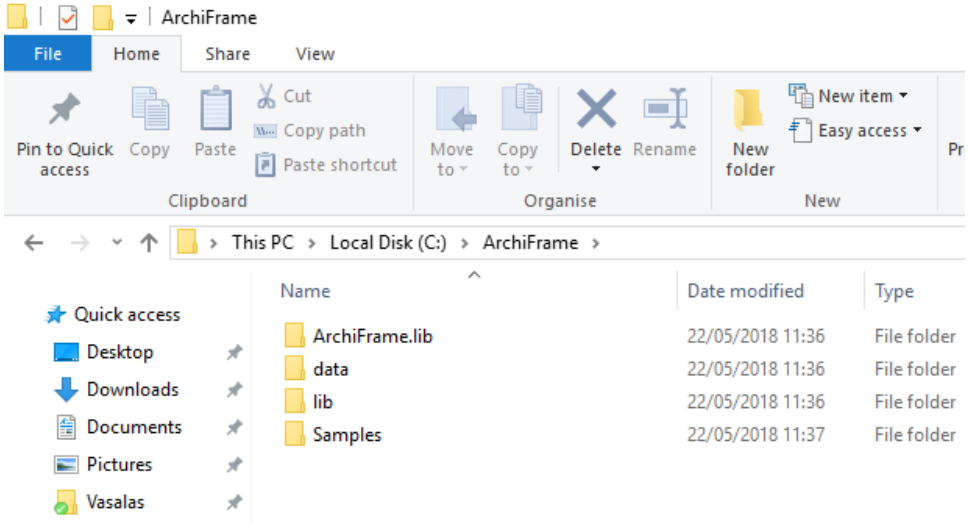
1 Windows Installation 3
2 Mac Installation 4
3 Getting started: entering main settings and setting up the material list 5
4 Selection tool 11
5 RenderLights Click N' Go 13
6 Free tools 13
7 Individual planks 13
8 Plank tools 17
9 Extended tool palette 40
10 Elements 53
11 Add & Edit element tool 56
12 Fascia tool 113
13 Modal dialogs 114
14 Examples of using log structures 201
15 Examples of using frame structures 203
16 Listings 204
17 Material list ArchiFrameBlocks.xml 206
18 Element definition file ArchiFrameElements.xml 219
19 Examples of using elements and trusses 258
20 Trusses 271
21 Weatherboards 278
22 Foundation drawing 288
23 XML-settings for an Archicad element 293
24 Lua scripts 297
25 Tips and tricks 332

1 Windows Installation

The installer installs the add-on to all supported and user-selected Archicad-versions. By default, the library is installed into the folder C:\ArchiFrame\ArchiFrame.lib.

Next to the *ArchiFrame.lib*-folder is the *data*-folder which contains main ArchiFrame settings including an Excel template and cnc-file producers. This folder or its contents should not be edited. Please read chapter 3 for more instructions. All Archicad-versions share the same library and *data*-folder.

If you would like to check some sample projects, these can be found in C:\ArchiFrame\Samples, another folder which will be installed together with the *ArchiFrame.lib* and *data* folders. In that folder you can find a few sample files.



The installer does not include an uninstaller. ArchiFrame needs to be removed manually by deleting:

- Main ArchiFrame folder, default C:\ArchiFrame.
- Add-on inside Archicad installation folders, for example C:\Program Files\Graphisoft\Archicad 21\Add-Ons\ArchiFrame.

After installation ArchiFrame can be found in the Archicad *Options*-menu. Inside ArchiLogs the tools are in menu *ArchiLogs / Joints*.

ArchiFrame and ArchiLogs cannot co-exist. If you are an ArchiLogs user, a license update will add Archiframe functionality to ArchiLogs.

2 Mac Installation

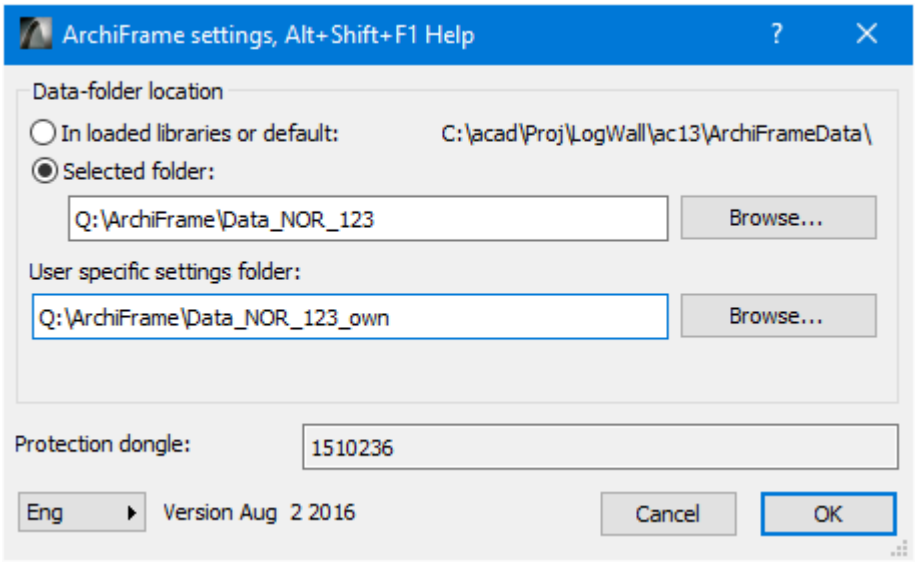
The zip-package has the following structure:

- ArchiFrameXxAddOn, the add-on for the specific Archicad-version.
- ArchiFrame:
    - Data, all configuration files (described in the following section)
    - ArchiFrame.LibEng, English library
    - ArchiFrame.LibFin, Finnish library
    - Samples, sample projects

The add-on must be manually copied into the corresponding Archicad Add-Ons folder (for example, Programs/Graphisoft/Archicad 21/Add-Ons).

The recommended location for the *ArchiFrame*-folder is on the desktop. This means that the ArchiFrame-library folder (ArchiFrame.LibEng) must be added to every project where it is needed. It is also possible to place the library inside Archicad-libraries, so it is always loaded.

ArchiFrame will find the *data*-folder from the default location (Desktop/ArchiFrame/Data). If using a different location, the folder must be selected from the settings:



Please read the following topic *Getting started: entering main settings and setting up the material list* about the data folders.

The samples contain PC-library references which lead to missing libraries in Mac – please edit the library list so that the ArchiFrame-library will be properly loaded when opening samples.

ArchiFrame and ArchiLogs cannot co-exist. If you are an ArchiLogs user, a license update will add Archiframe functionality to ArchiLogs.

3 Getting started: entering main settings and setting up the material list

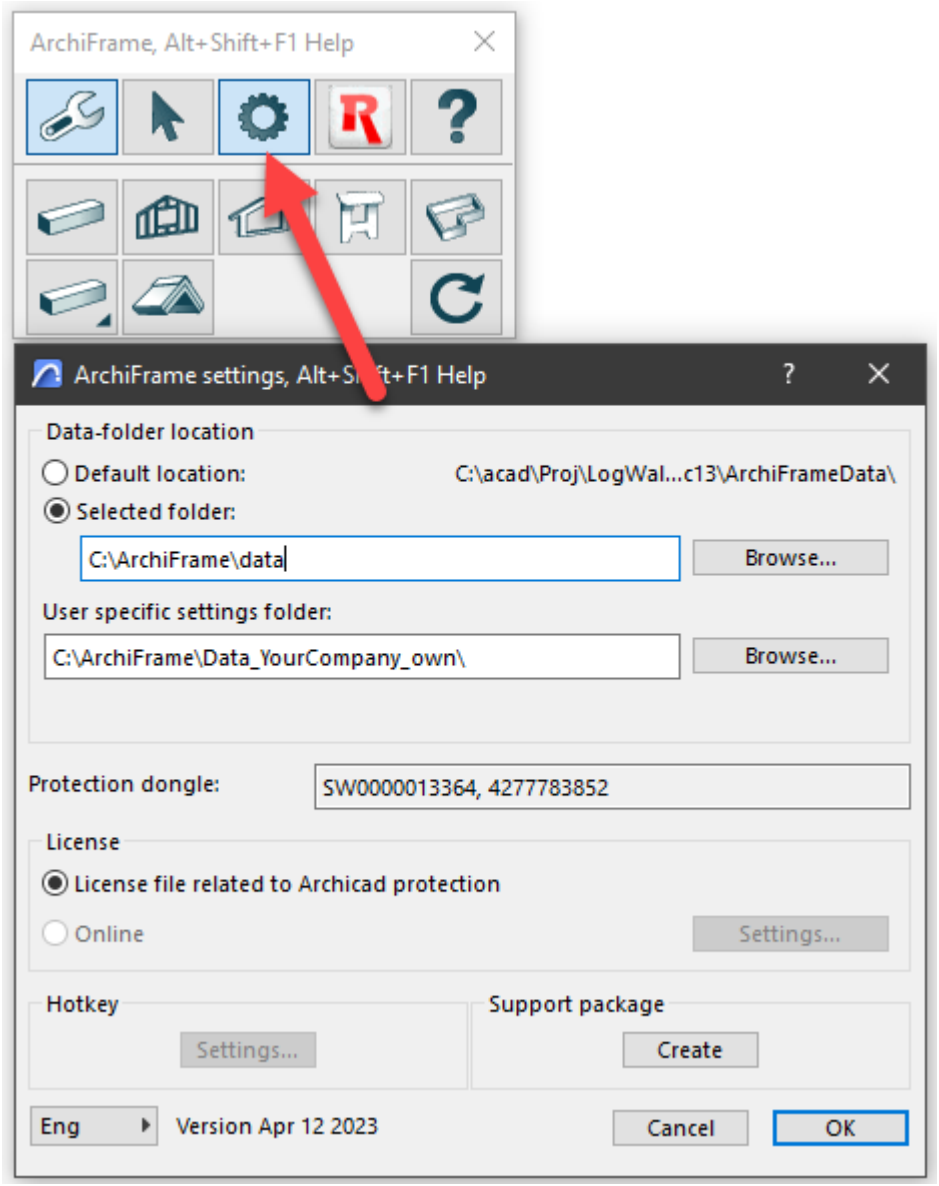
3.1 Setting up the data folder and the user-specific settings folder

Video in English here and in Finnish here.

Before starting your own project, it is important to set up two folders: the main data folder which is changed with ArchiFrame updates and the user- or company-specific settings folder. This is to ensure that your own settings and element definitions are not lost when ArchiFrame is updated. In addition, having your own settings in a separate folder makes it easy to back up your settings and to share them with others.

The main data folder exists in the C:\ArchiFrame directory after ArchiFrame installation. To create user-specific settings folder, follow these steps:

- Copy folder Data_Yours_BASE_own from Samples-folder to C:\ArchiFrame
- Rename that folder according to your company name so that you have C:\ArchiFrame\Data_MyCompany_own
- Update the new folder location through the ArchiFrame user interface according to the picture below:



With this setup folder C:\ArchiFrame contains all settings and can be shared to other users or other computers or just backed up easily.

Help for Support package/Create here.

3.2 Contents of the main data folder and the user-specific settings folder

The main data-folder contains the following files, not to be edited by the user:

- ArchiFrameBlocks.xml, the default material list
- ArchiFrameCnc.lua, cnc-producer. Built in script supports Hundegger bvn-files. There may be many files like ArchiFrameCncBtl.lua and ArchiFrameCncWup.lua to write different cnc-formats.
- ArchiFrameCoveringBoards.xml, weatherboard settings.
- ArchiFrameElements.xml, element related settings.
- ArchiFrameFoundation.xml, foundation related settings.
- ArchiFrameFoundationFin.xml, foundation related settings in Finnish.
- ArchiFrameListing.lua, listings producer. Default script produces listings either in Excel or in text format.
- ArchiFrameTrusses.xml, truss settings.

- FrameListingEng.xls, Excel-template in English.
- FrameListingFin.xls, Excel-template in Finnish.
- FrameListingElemEng.xls, Excel-template for element listing.
- FrameListingSummaryEng.xls, Excel-template for summary listing.

All measurements in xml-setting files are given in meters and the decimal separator used is a full stop, for example, 57 mm is 0.057.

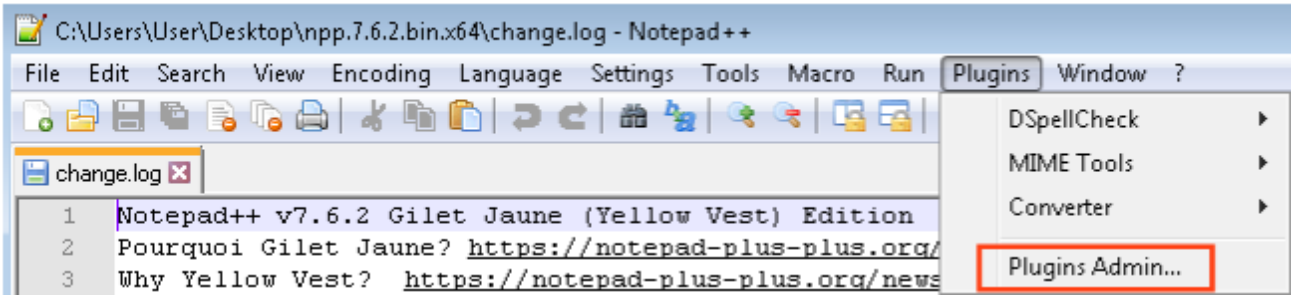
Usually all updates contain new versions for ArchiFrame library, add-on and the data-folder. To maintain your own settings when installing updates, it is possible to copy some of the standard data-files listed above into user-specific settings folder. If the file exists in user-specific settings folder, it is loaded from there rather than from the main data-folder. There is a mechanism to replace/add/remove parts of data-folder xml-file with information from user-specific settings-folder – more of that later. User specific settings-folder contains files (not all have to be present):

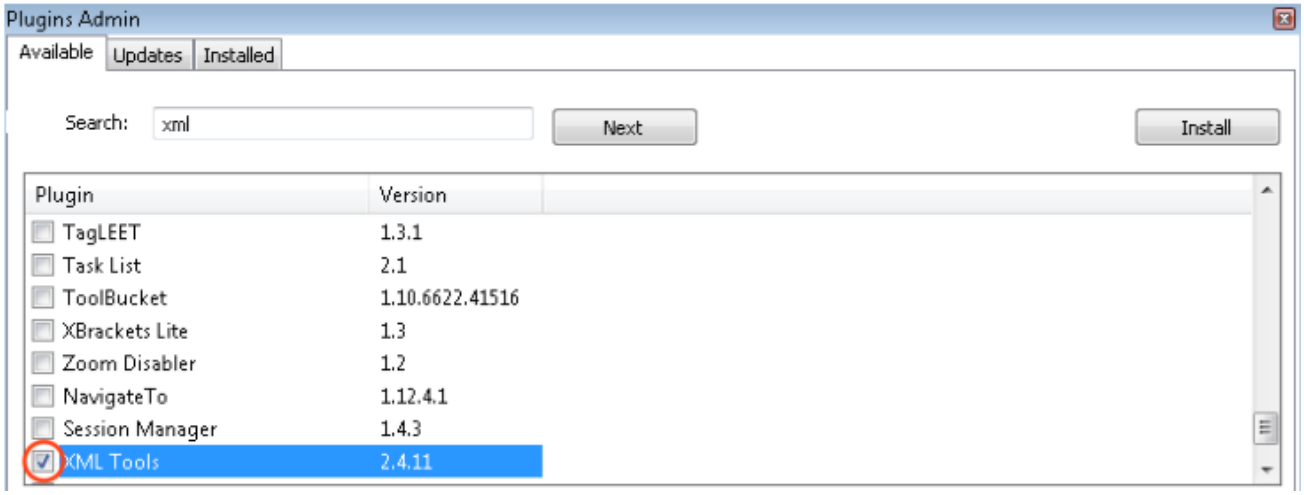
- ArchiFrameOwnElems.xml, own element types defined in the main element tools maintained by ArchiFrame. Please see section Own element types/Custom element tools.
- ArchiFrameOffsetsEdge.xml, edge offset settings from the main element tool palette maintained by ArchiFrame. Please see section Custom layer edge/opening offsets.
- ArchiFrameOffsetsOpening.xml, opening offset settings from the main element tool palette. maintained by ArchiFrame Please see section Custom layer edge/opening offsets.
- ArchiFrameCorners.xml, custom corner types from element tool palette maintained by ArchiFrame. Please see here.
- ArchiFrameNailings.xml, nailing presets maintained by ArchiFrame.
- ArchiFrameLic.txt, the license file may be located here or in the *data*-folder. This is the recommended place since it allows seamless operation even if the *data*-folder is updated with program update.
- ArchiFrameBlocks.xml, to use this file instead of the file in the *data*-folder. The recommended way is to use ArchiFrameBlocksChanges.xml file to change parts of the base *data*-folder's ArchiFrameBlocks.xml -file.
- ArchiFrameElements.xml/ArchiFrameElementsChanges.xml as previous but settings for elements.
- ArchiFrameCoveringBoardsChanges.xml/ArchiFrameCoveringBoardsChanges.xml for weatherboards.
- ArchiFrameTrusses.xmlArchiFrameTrussesChanges.xml for trusses.
- ArchiFrameTenonMortPresets.xml, presets for extended tool palette's tenon & mortise etc tools.
- ArchiFrameCltPresets.xml, used by CLT-types dialog.
- ArchiFrameEdgeConnPresets.xml, used by board edge connections dialog.
- ArchiFrameArchiCornersPresets.xml, used by corner planks dialog.
- ArchiFrameEdgeIBeamGro.xml, used by I Beam grooves dialog.

3.3 Editing the xml settings files

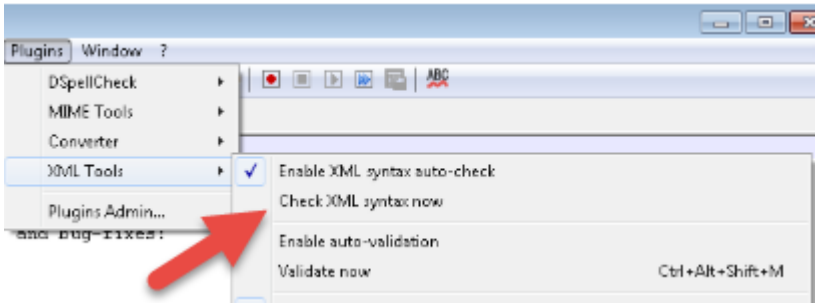
To edit raw plank material list, please check this video.

Settings files in xml format are edited with a text editor, rather than Word or any other word processor. Notepad++ is recommended instead of plain Notepad, although plain Notepad also works (http://notepad-plus-plus.org/). The benefit of Notepad++ is that it gives warnings about xml-markup errors and formats the file, making it easier to read. To get the xml-support follow these steps:





Then to check if the syntax is ok:



The best tool in Windows for editing xml is Microsoft Visual Studio. A free version of it can be downloaded from this link:
https://visualstudio.microsoft.com/free-developer-offers/

3.4 The ArchiFrame*Changes.xml files

It is possible to add, remove or replace information from base *data*-folder's xml-file for easier maintenance. That allows new features to come from the base *data*-folder and still have customizations in certain parts of the file. ArchiFrame reads first the file from base *data*-folder and then applies operations from user specific *Changes.xml-file. For example, to add some materials before default materials, there is file ArchiFrameBlocksChanges.xml in *user specific* settings folder with following content:

```
<?xml version="1.0" encoding="utf-8"?>
<archiframe>
  <materials merge="addpre" mergechildren="1">
    <material id="25x38" name="25x38" thickness="0.025" height="0.038" shape="block">
    </material>

    <material id="45x75" name="45x75" thickness="0.045" height="0.075" shape="block">
    </material>
```

```
  </materials>

  <materials>
    <material merge="addpost" mergepath="[@id='45x170']" id="45x170 C24" name="45x170 C24"
thickness="0.045" height="0.170" shape="block" maxlen="5.1">
    </material>

    <material merge="replace" mergepath="[@id='45x45']" id="45x45" name="45x45"
thickness="0.045" height="0.045" shape="block" maxlen="999">
    </material>

    <material merge="remove" mergepath="[@id='45x220']">
    </material>
  </materials>
</archiframe>
```

ArchiFrame processes only nodes having attribute merge = “xxx”. It may have values:

- addpre, add current node before existing children.
- addpost, add current node after existing children.
- remove, remove current node with all its children.
- replace, replace base data xml with current node.

If attribute mergechildren is given with value 1 like in the example, the current node’s children are processed instead of the current node. It is also possible to give attribute mergepath to specify the target with attributes using the xpath-like syntax like in the example.

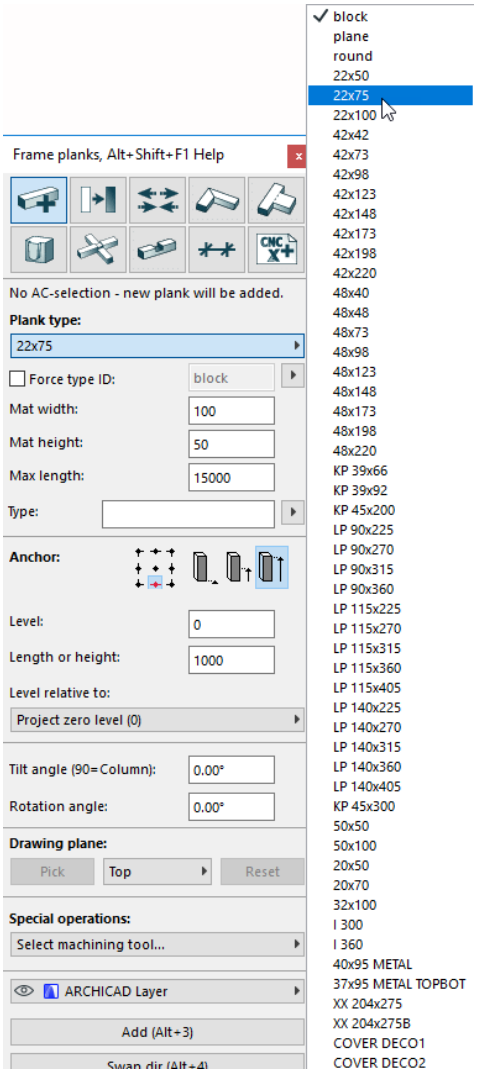
To debug the xml-settings, ArchiFrame copies the resulting texts to clipboard if *Shift*-key is pressed when closing the settings dialog with OK.

Another option to make changes is to copy the file from default data folder into the user specific settings folder and edit that file. For example, adding material 22x75, original:

```
  <materials>
    <!--shape must be specified block/plane/round even if custom profile is given m3factor is
the coefficient factor to get m3-quantity from the material. If you omit it, default formula
will be used. -->
    <material id="block" name="Block" thickness="0.000" height="0.000" shape="block">
</material>
    <material id="plane" name="Plane" thickness="0.000" height="0.000" shape="plane">
</material>
    <material id="round" name="Round" thickness="0.000" height="0.000" shape="round">
</material>
    <material id="22x50" name="22x50" thickness="0.022" height="0.050" shape="block"
maxlen="99" m3factor="0.000"> </material>
    <material id="22x100" name="22x100" thickness="0.022" height="0.100" shape="block"
maxlen="99" m3factor="0.000"> </material>
    <material id="42x42" name="42x42" thickness="0.042" height="0.042" shape="block"
maxlen="99" m3factor="0.000"> </material>
```

This file can be freely edited and to reload the new settings into ArchiFrame, please open ArchiFrame settings and close the settings dialog with OK.

```
    <material id="22x50" name="22x50" thickness="0.022" height="0.050" shape="block"
maxlen="99" m3factor="0.000"> </material>
    <material id="22x75" name="22x75" thickness="0.022" height="0.075" shape="block"
maxlen="99" m3factor="0.000"> </material>
    <material id="22x100" name="22x100" thickness="0.022" height="0.100" shape="block"
maxlen="99" m3factor="0.000"> </material>
    <material id="42x42" name="42x42" thickness="0.042" height="0.042" shape="block"
maxlen="99" m3factor="0.000"> </material>
```



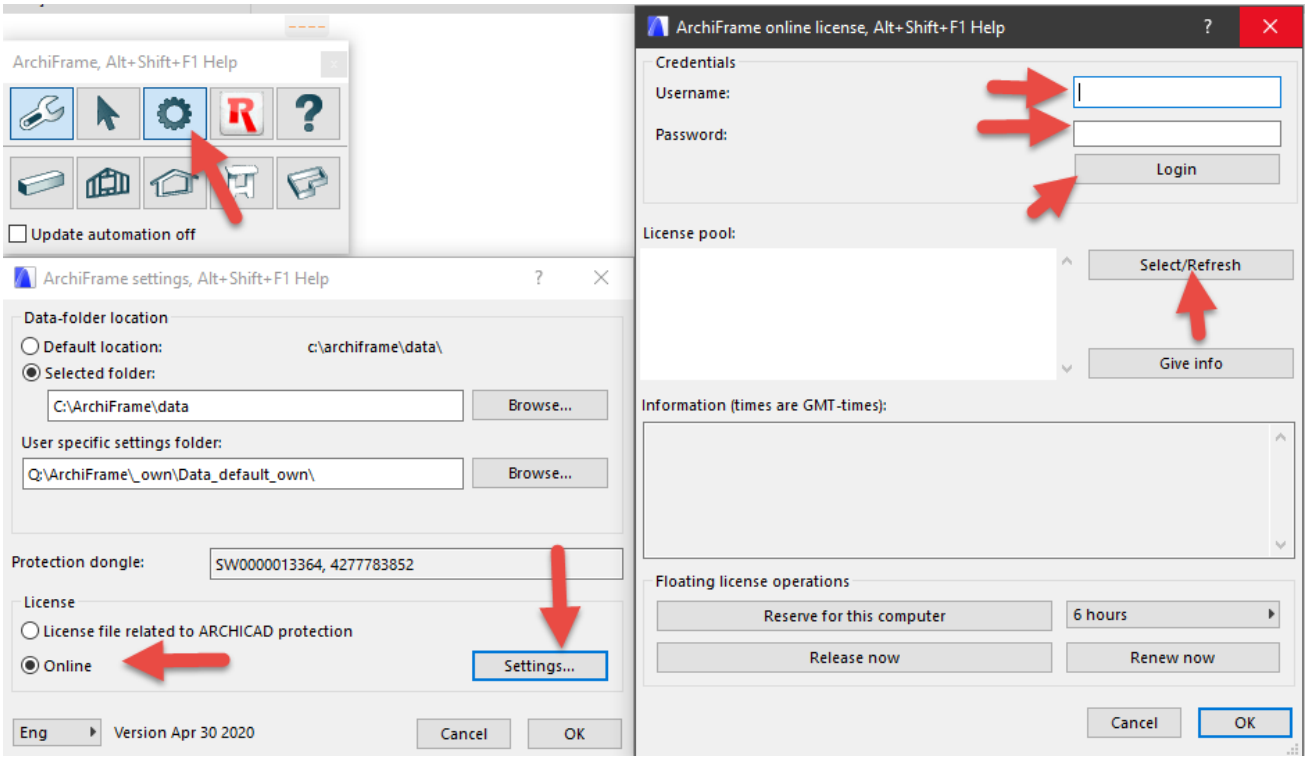
This way the custom settings are saved in the user specific settings folder and installing ArchiFrame updates will not change them.

3.5 ArchiFrame's own settings location

The settings of different tool palettes and the online license are saved into *system's user- specific*-folder. In Windows, this folder is C:\Users\[username]\AppData\Local, for example, C:\Users\petteri\AppData\Local. In Mac, the folder is [user]/Library/Preferences. ArchiFrame related files start with the name ArchiFrame. There is no need to edit these files.

3.6 Entering online license information

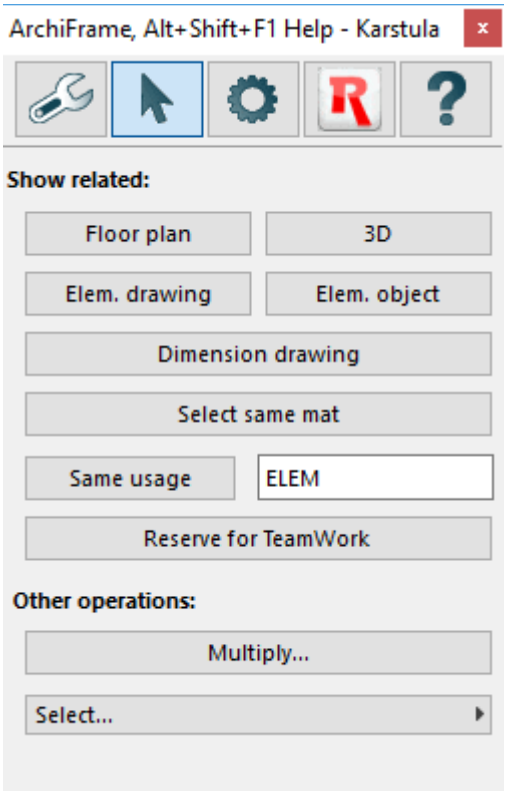
The online license information is entered here – the login information is given to you by the reseller:



The license pool list shows all available licenses for current login. There may be for example licenses with and without cnc-support, so it is important to select the correct one.

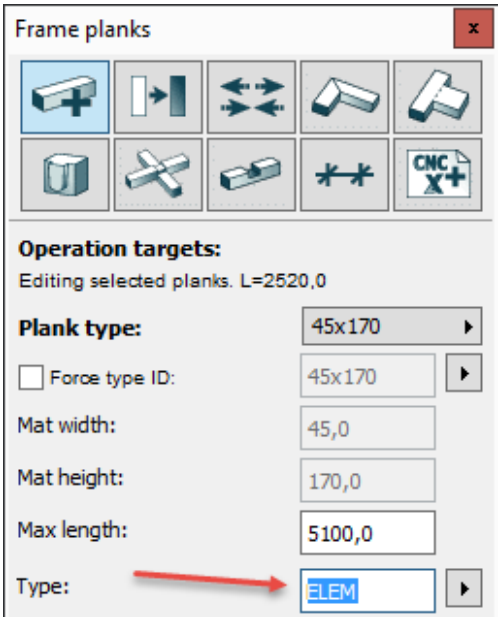
For more information, please see here.

4 Selection tool



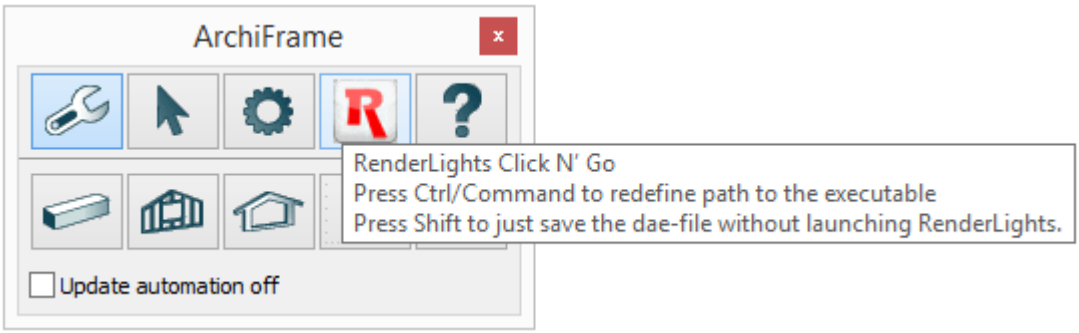
The selection tool makes it easy to switch between Floor plan, 3D and element drawing. By clicking one of the *Show related*-buttons it will open a view showing currently selected elements in the selected view. For example, if you have a plank selected in the floor plan and you click on *Elem. Drawing,* it will open the element drawing with the originally selected planks zoomed into view. The selection tool works with any Archicad element type.

- *Floor plan* opens the floor plan, showing the selected elements.
- *3D,* the same in the 3D window.
- *Element drawing* opens the element elevation and selects the ArchiFrame planks/boards from all projections. Please note that projection pieces are special 2D objects maintained by ArchiFrame.
- *Element object* selects the ArchiFrame element object owning the current selection. If the selection contains just one or more *ArchiFrameElement*-objects, the button is changed into *AC element*. Then it will show the Archicad elements where the geometry and *door & window* weights are read from.
- *Dimension drawing* opens the dimension drawing made with *Create and update dimension drawings*-tool.
- *Select same mat* selects planks with the same material ID as the selection.
- *Same usage* selects all planks having the similar type/usage text as in the selection. The type/usage is given here:



- *Reserve for TeamWork* takes the selection and extends it to contain everything in the element including other layers, elevation drawings, floor plan pieces, dimension drawings and related ArchiFrameElement-objects. Then it reserves these elements for editing. After the editing is done, all changes are sent to the TeamServer with standard *Send & Receive* and finally released with *Release All*-button.
- *Multiply* opens ArchiFrame's multiply dialog that has some extra features compared to Archicad's built-in Multiply tool.
- *Dump profile from fill* is used to define the plank's profile. Pressing shift allows the origin to be outside of the profile.

5 RenderLights Click N' Go



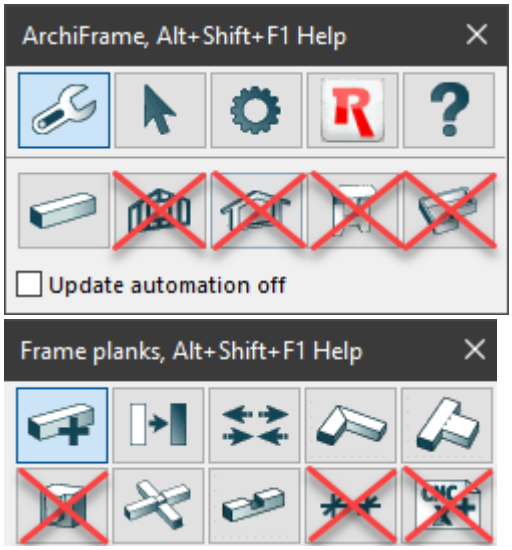
ArchiFrame contains integration to RenderLights visualization and virtual reality software. Please see http://www.renderlights.com/ for more information. RenderLights must be purchased separately and there is also a free trial version available.

ArchiFrame integration does following:

- When *RenderLights*-icon is clicked in the 3D window the current view is saved with current Archicad file name extension changed to .dae. The textures are saved in a separate folder next to the dae-file. Then ArchiFrame launches RenderLights to continue working with the 3D-model.
- If the user presses *Control*-key or *Command*-key in Mac while clicking *RenderLights*-icon, it may point to where the RenderLights application is installed (RLPower.exe in Windows and RL.app in Mac).
- If the user presses *Shift*-key just the dae-file is saved, RenderLights is not launched.

6 Free tools

Without a license these tools are available:

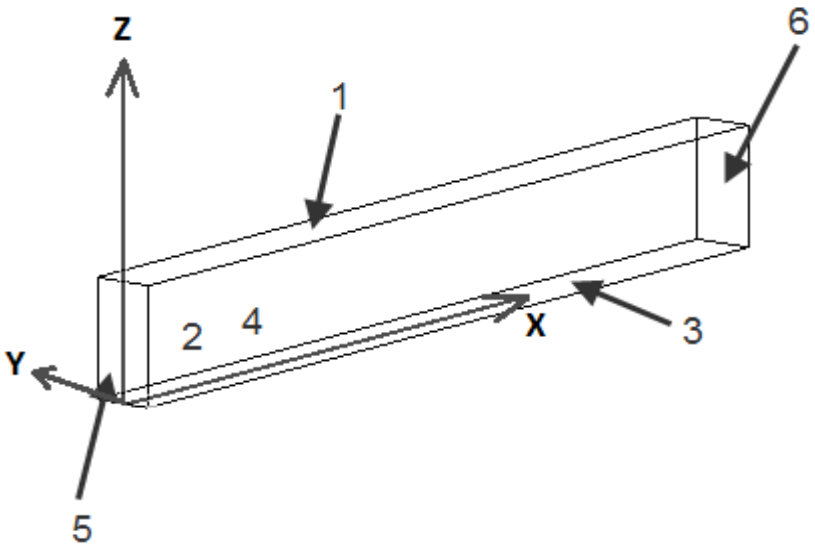


7 Individual planks

7.1 About the planks

Planks can be copied with the standard *Archicad* drag/rotate/mirror etc. tools. ArchiFrame observes changes in planks and unmirrors planks automatically. Because of this, planks should not be mirrored or edited if ArchiFrame is not installed.

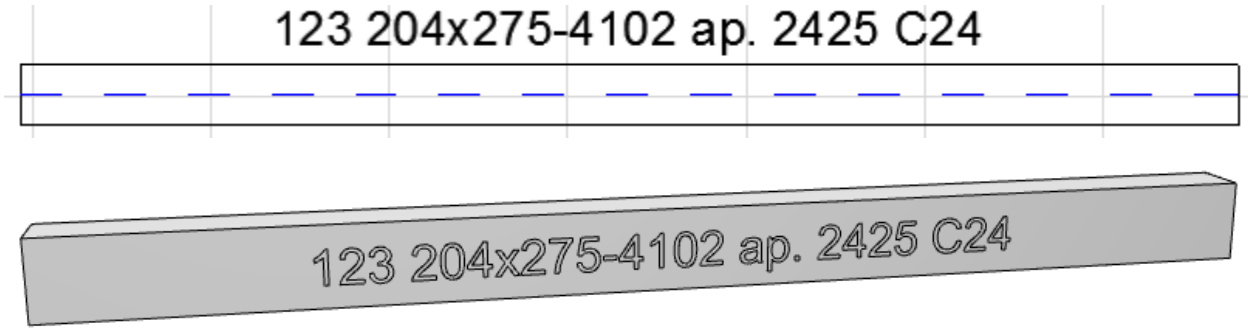
Plank reference line is at the bottom middle and the surfaces are numbered as:



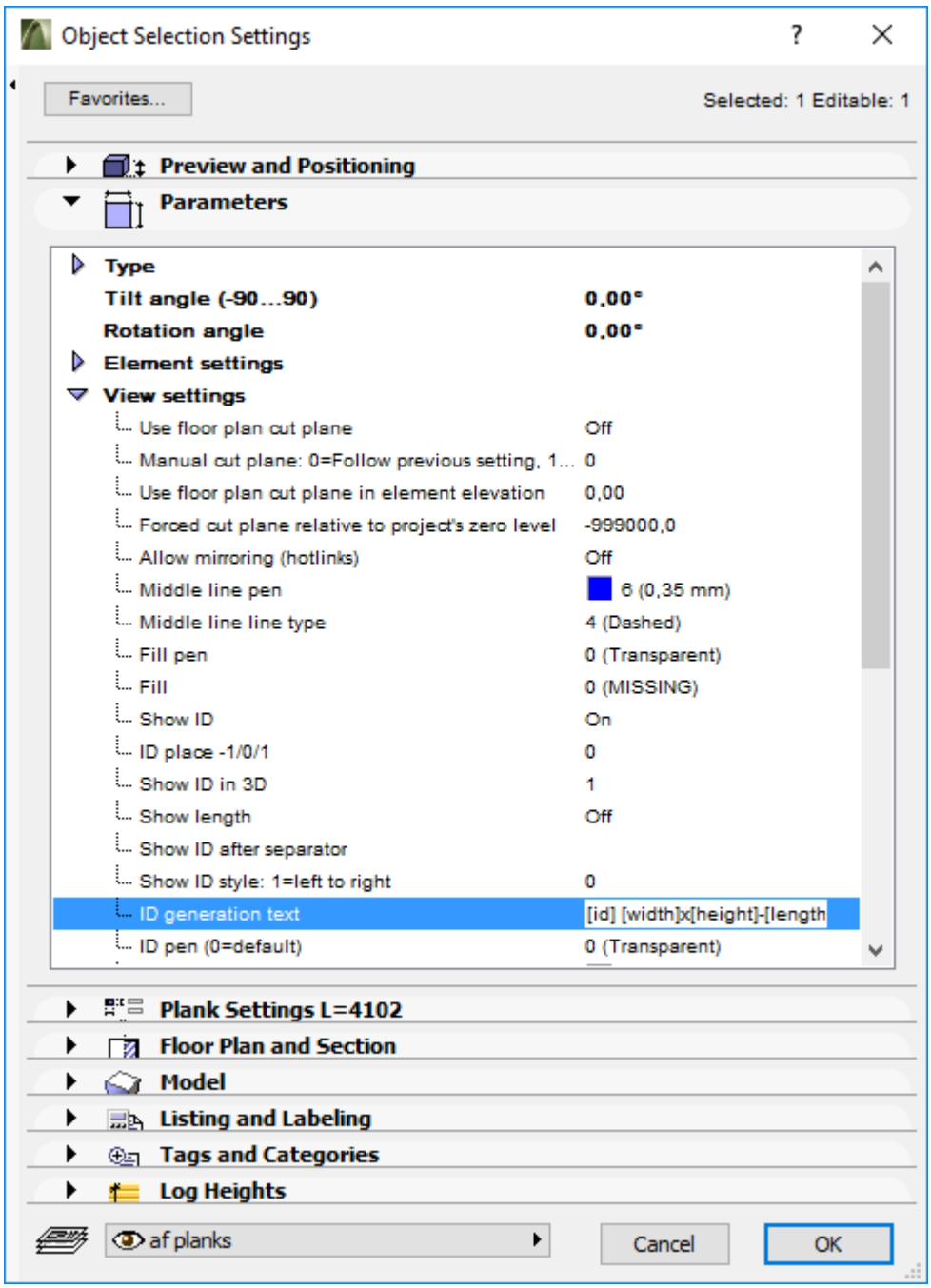
1. Top
2. Front
3. Bottom
4. Back
5. Begin
6. End

The reference line is the *X*-axis shown in the image above. The origin of the front (2) and back (4) sides is at the lower left corner in the illustration above. Bottom, back and end side coordinates (3, 4 and 6) are similar to the corresponding front surfaces. In other words, those surfaces are viewed through the plank.

The plank can show additional information instead of just ID:



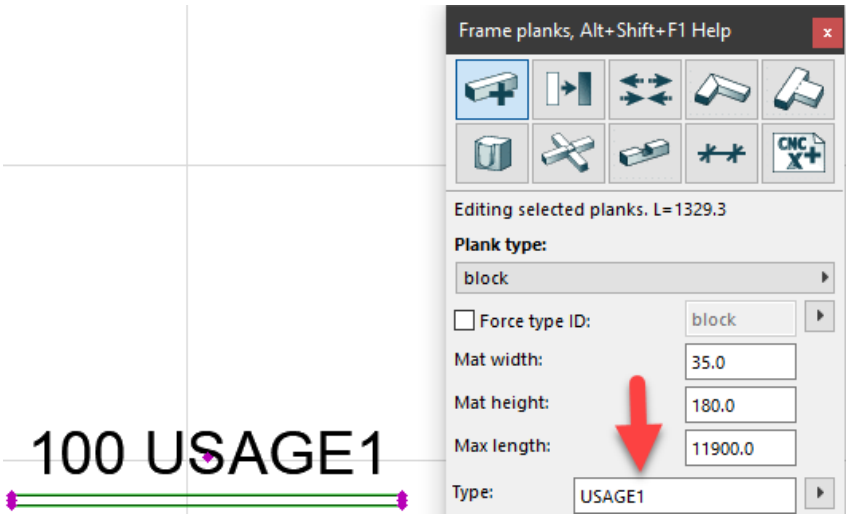
Special IDs are generated according to plank object's ID generation text parameter (example value is: [id] [width]x[height]-[length] ap. [level] [quality]):



It can contain following tags:

- [id], plank's id after separator character
- [orgid], original full id
- [hotid], full id including hotlink's master ID as prefix if used in hotlink (currently does not work in 3D window, waiting for Archicad updates)
- [hotmasterid], just the hotloink's master ID or empty if not part of a hotlink (currently does not work in 3D window, waiting for Archicad updates)
- [:hotmasterid], just the hotloink's master ID prefixed with : or empty if not part of a hotlink (currently does not work in 3D window, waiting for Archicad updates)
- [elemid], id part before separator character
- [matid], material id
- [width]
- [height]

- [length]
- [level], global level at reference line
- [quality], value of parameter Type/Quality class
- [module], value of parameter cnc-Module (iElemModule)
- [usageid], value of parameter iUsageID given here:



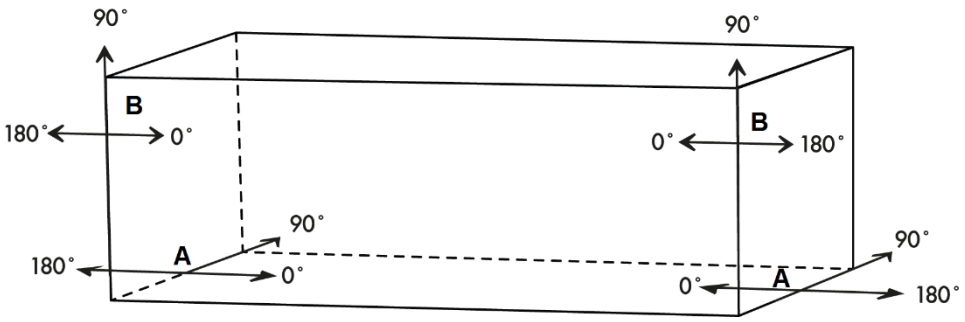
To be able to use different IDs in hotlink source with element elevations and hotlinked 3D-model the parameter *ID generation text* (iIDContent) is no longer synchronized between 3D and elevations allowing to have different values in 3D and elevations. In elevations typical use is to have *ID Generation text* empty and in 3D model the ID is made using the *ID Generation text*-parameter.

The same mechanism is available for the planks, boards, and ArchiFrameElement-objects.

7.2 Angled endings

The angled endings are specified with values:

- Angle viewed from top A.
- Bevel angle B.
- The plane is anchored with point defined by three coordinates:
    - X, distance from plank end, positive inside the plank and negative outside the plank.
    - Y, offset in plank's Y-axis.
    - Z, offset in plank's Z-axis.



For example, cut with 20 degrees bevel angle starting from half of the plank's height is defined with values:

- Angle 90 (perpendicular).

- Bevel angle 160.
- X = 0, Y = 0, Z = height/2.

Plank may be cut with multiple angled cuts. ArchiFrame calculates the resulting plank length and adjusts the cut values as necessary.

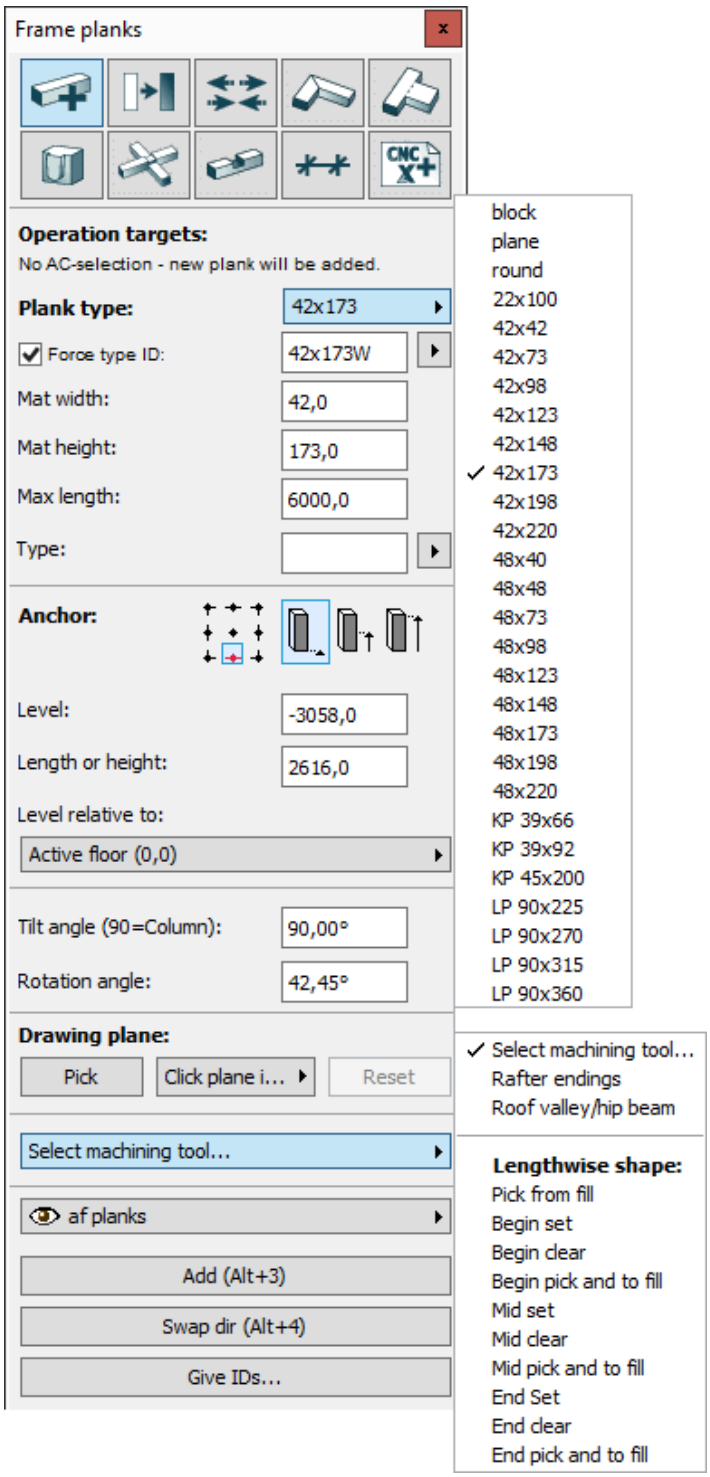
8 Plank tools

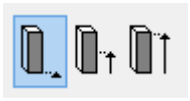
8.1 Add and edit planks

First, the settings in this palette and Archicad object settings are adjusted. If something is selected, then changes are saved to the selection. If nothing is selected, then a new plank is created for use at the chosen location. If placing a new plank, it will be selected by pressing *Alt*-key when adding it.

Related videos:

- Creating a roof structure
- Tidying up the Walls and Roof


Frame planks
Operation targets:
No AC-selection - new plank will be added.
Plank type:
42x173
Force type ID:
42x173W
Mat width:
42,0
Mat height:
173,0
Max length:
6000,0
Type:
Anchor:
Level:
-3058,0
Length or height:
2616,0
Level relative to:
Active floor (0,0)
Tilt angle (90=Column):
90,00°
Rotation angle:
42,45°
Drawing plane:
Pick
Click plane i...
Reset
Select machining tool...
af planks
Add (Alt+3)
Swap dir (Alt+4)
Give IDs...
block
plane
round
22x100
42x42
42x73
42x98
42x123
42x148
42x173
42x198
42x220
48x40
48x48
48x73
48x98
48x123
48x148
48x173
48x198
48x220
KP 39x66
KP 39x92
KP 45x200
LP 90x225
LP 90x270
LP 90x315
LP 90x360
Select machining tool...
Rafter endings
Roof valley/hip beam
Lengthwise shape:
Pick from fill
Begin set
Begin clear
Begin pick and to fill
Mid set
Mid clear
Mid pick and to fill
End Set
End clear
End pick and to fill

- Plank type can be either the full type or the base type if *Force type ID* is checked.
- *Force type ID* can be used if the plank type is seldom used and not worth adding to ArchiFrameBlocks.xml. Also, can be used for example to distinguish weatherproofed materials from normal materials. Dropdown list next to the forced type ID contains last saved forced types to quickly edit the planks. The dropdown list saves the base plank type, material size, maximum length and type text.
- Type is the usage id. It is used to distinguish, for example, weatherboards from other planks.
-  Anchor point defines the anchor in plank's cross section. This anchor affects adding the plank, changing its size (anchor point remains fixed) and the level.
-  Define anchor for the level in plank's direction: begin, middle and end.
- The tilt angle is lengthwise tilted from plank begin to endpoint. Positive tilt makes the plank rise and negative lower.
- The rotation angle is plank rotation counter clockwise viewed at the beginning of the plank.
- Drawing plane is useful for example to draw roof structures. After picking the drawing plane, the plank positions in floor plan (XY-coordinates) are kept and the level is adjusted to remain relative to the drawing plane when adding and editing the planks. It can be picked from:
    - Frame-plank.
    - Roof.
    - Beam.
    - Slab.
    - Tilted wall.
- *Swap dir* swaps the direction of the plank or board. All machinings remain in place. For cladding boards it swaps the front/back sides of the target.
- *Give IDs*, re-numbers the planks. The operation can be started with or without selection. More information here.
- Rafter endings, more information here.
- Roof valley/hip beam, more information here.

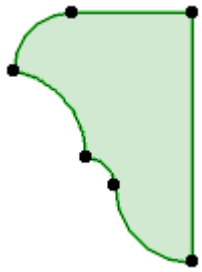
8.1.1 Lengthwise shape

These tools allow to make for example a piece like below (note that using a lot of arcs will make 3D window slow):

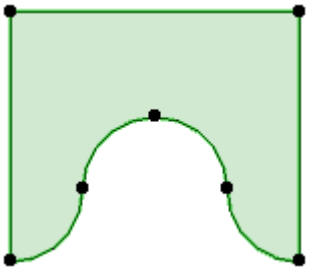


The shape is picked from an Archicad fill and then injected to target plank's begin/middle/end. In the middle the fill is automatically repeated from begin to end excluding possible begin and end shapes. End shape is drawn as begin shape – it is mirrored when used at the end automatically. The fill is automatically adjusted to the plank height if either top or bottom side is straight.

An example of the fill used at the ends:



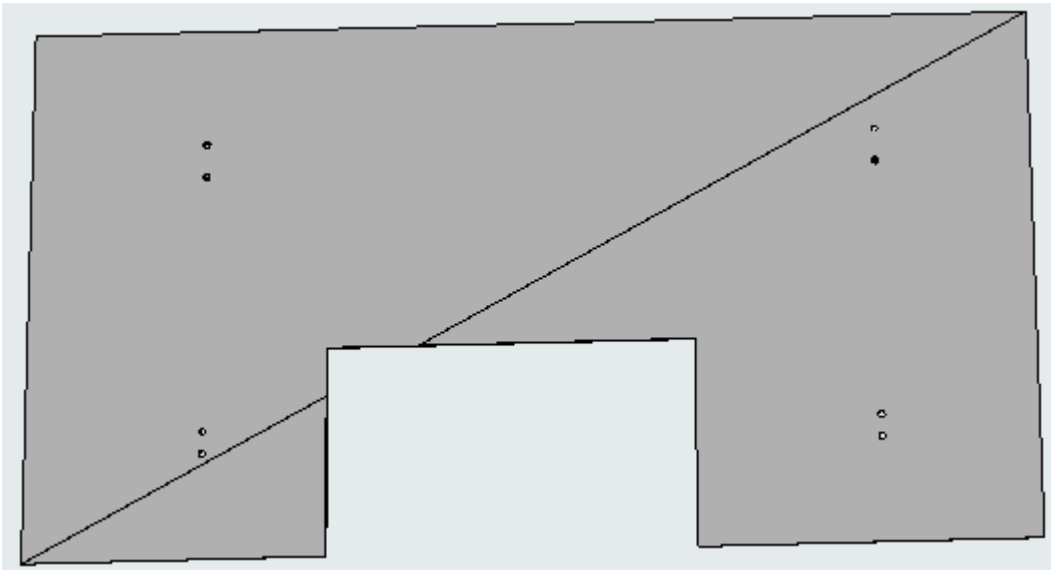
And the repeating middle part:

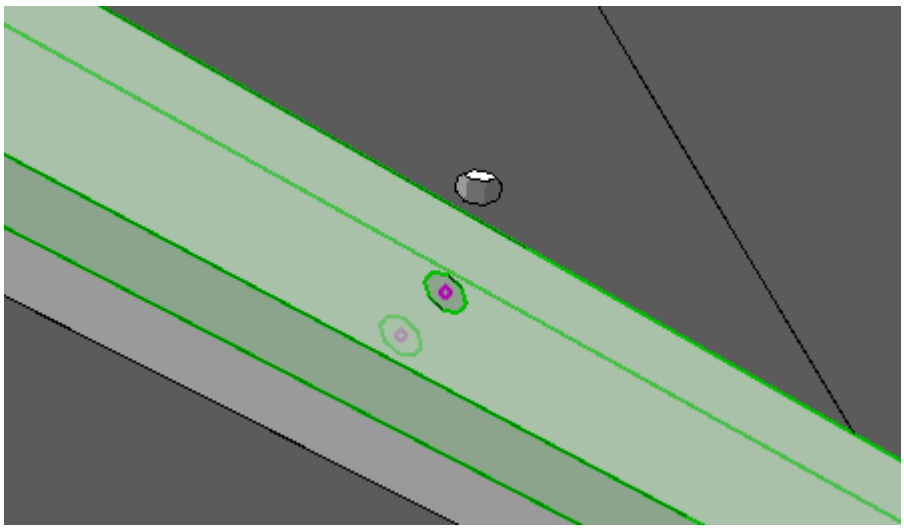


It is also possible to pick the shape from an existing plank with the commands Begin/Mid/end pick and to fill. In this command Esc can be pressed to avoid placing the fill into floor plan.

8.1.2 Element lifting drillints floor

This automation tool is used to add drillings to the floorboard and the related beams:

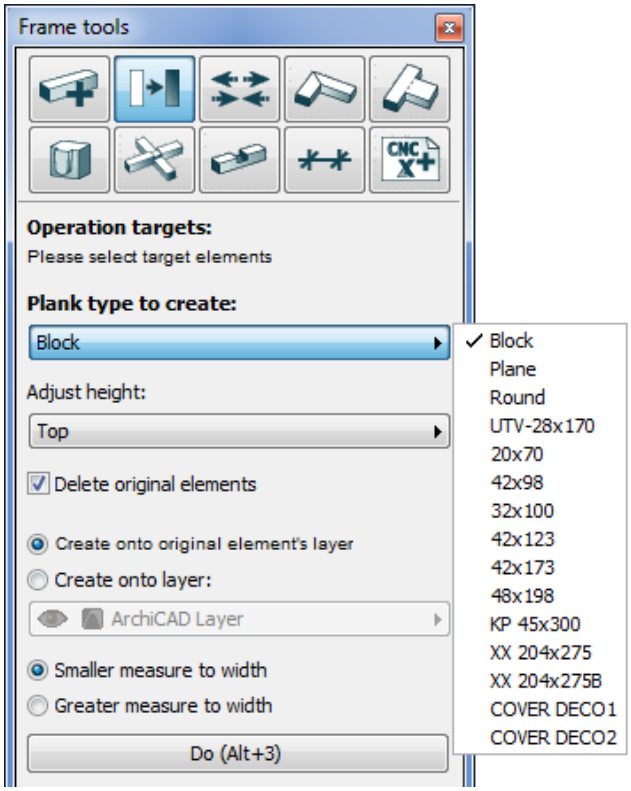




To use, the target boards and beams are selected. Then the drilling position is clicked either in the element elevation, floor plan, 3D or section/elevation window. The tool calculates the drilling position perpendicular to the beams. Multiple beams are processed as a group producing just two holes to the board.

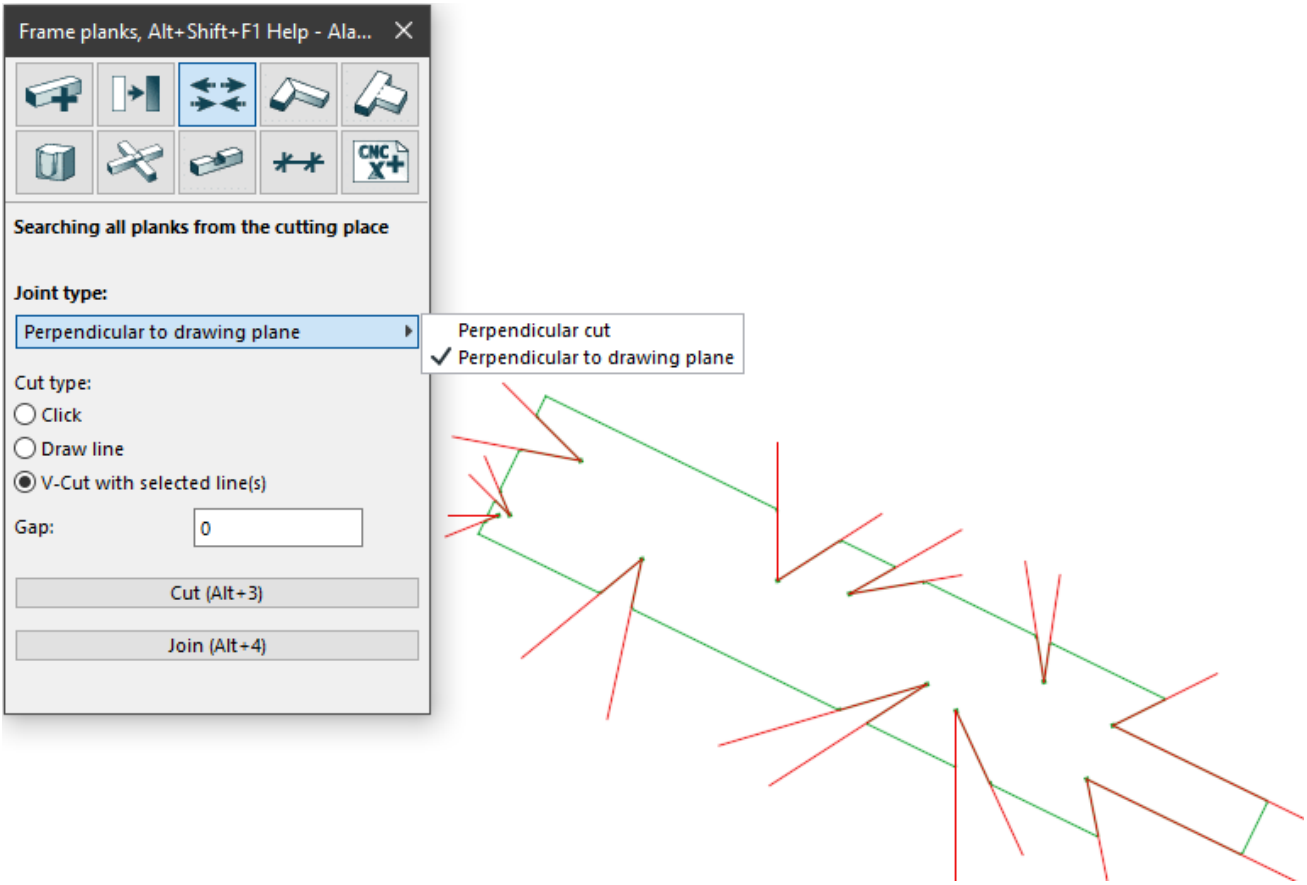
8.2 Convert to planks

Converts select Archicad walls, beams or columns into ArchiFrame planks. First, the target elements are selected and the conversion settings adjusted. The first three plank types are generic and will result in exactly the same size as the original element. For example, a glued laminated beam with any size is converted to generic block since there is no matching size stock material.



Conversion groups elements into single group. Text (Alt+3) refers to the keyboard shortcut for the command.

8.3 Cut and join



Related videos:

- Using V-Cut

Target planks can be selected with Archicad selection tool before cutting. ArchiFrame cuts the clicked plank. Alternatively, you can draw a line that will cut all planks intersecting that line. Joint type setting *Cut from watching direction* cuts planks vertically in 2D and at element watching direction if cut from element elevation.

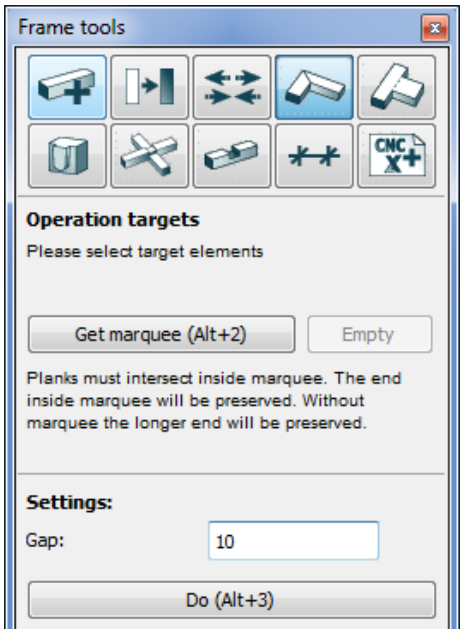
V-Cut with selected lines makes less than 180-degree angle cut according to two connected Archicad lines. Please see this video. Cutting works in floor plan, sections/elevations and in projections.

To join planks the target planks are first selected and then these will combine planks at the same line, combining all machinings.

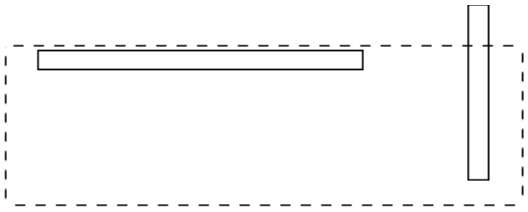
Press shift if you want to keep small pieces left from the operation.

8.4 Adjust endings

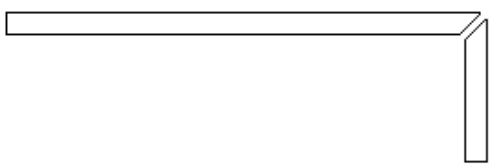
This tool combines planks from ends halving the angle and leaving a given gap between the planks.



Limit the working area by clicking on *Get marquee*-command, so that right hand side plank's lower end will be preserved:



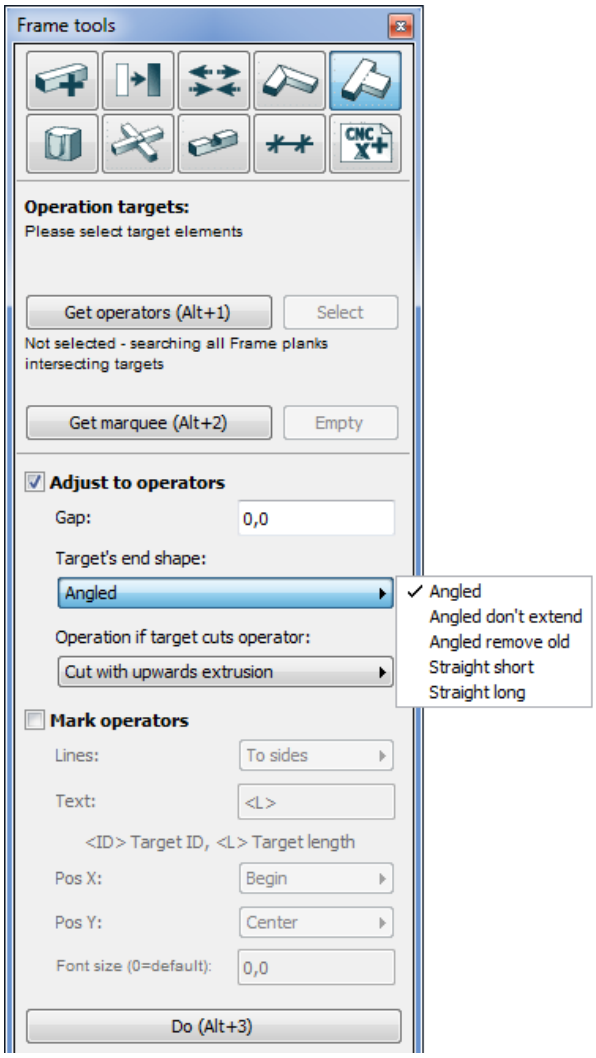
Select the planks and click *Do*. The result is here:



Without picked marquee ArchiFrame preserves the furthest end of the plank. In other words, the unchanged end is selected to produce the longest possible plank.

8.5 Adjust to operators and markings

This tool joins target planks to operators, or trims the targets to a roof for example. In addition, the tool allows the addition of markings to joining places. Please see this video.



Target's end shapes are:

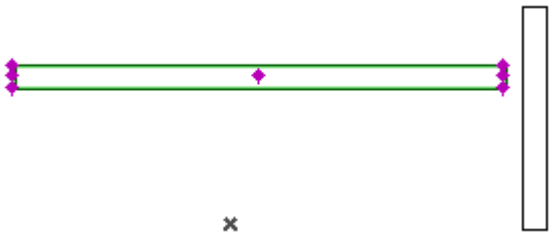
- Angled, extend and join to operator.
- Angled don't extend, cut with operator.
- Angled remove old, like angled but all existing angled cuts are removed.
- Straight short, straight ending to first intersection position.
- Long straight, straight ending to furthest intersection position.

For example, joining a horizontal plank to a vertical plank (in floor plan) with 100 mm gap.

1. Select vertical plank as operator (if operators are not set, ArchiFrame searches the closest intersecting plank for target plank).

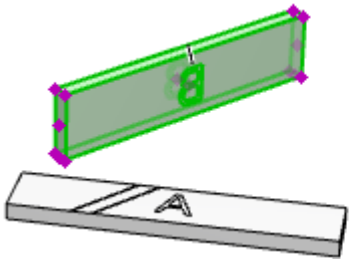


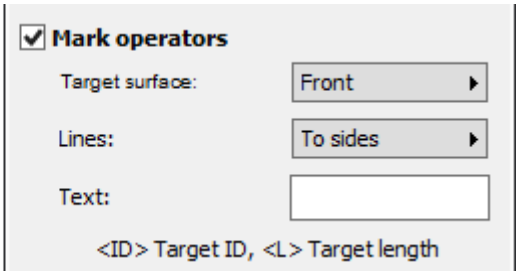
2. Select horizontal plank and click *Do*:



Press shift if you want to keep small pieces left from the operation.

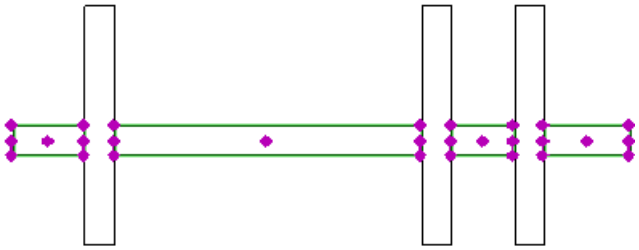
Select target surface to mark planks even if there is no intersection:





Operations:

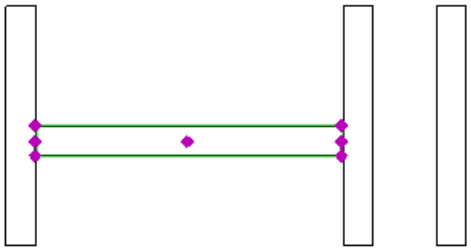
- *Cut to the operator*:



- *Cut between begin and first operator*:



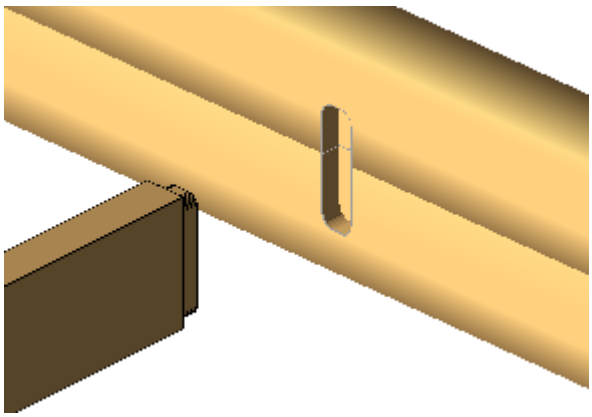
- *Cut keeping just the longer part* to always select the longer part of the piece and stop processing when end is cut away:



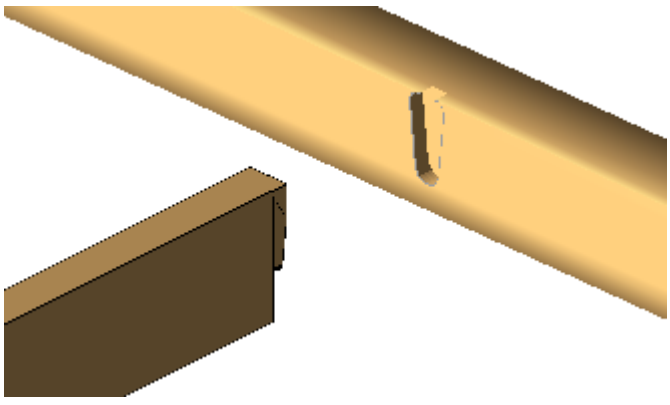
8.6 Dovetail, tenon and beam shoe joints

This tool is used to make mortise & tenon, dovetail and beam shoe joints. All joint types are done in the same way. Hidden shoe does not make any mortise/pocket machining and balk wedge does not make any tenon kind of machining.

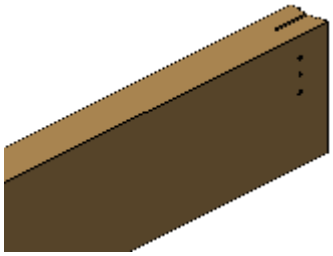
Mortise and tenon:



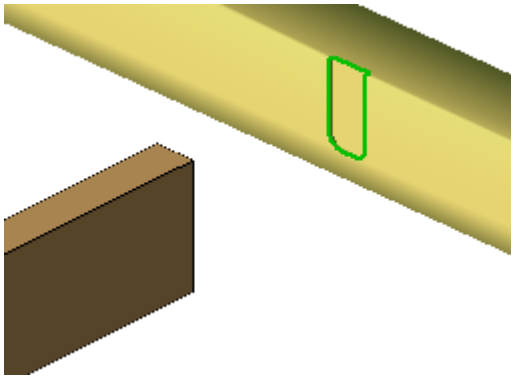
Dovetail:



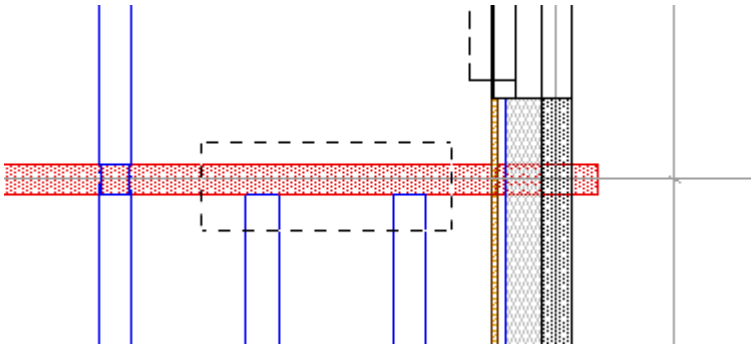
Hidden shoe, the accessory is placed next to the plank (operator) and the beam (target) is attached with dowels to the accessory:



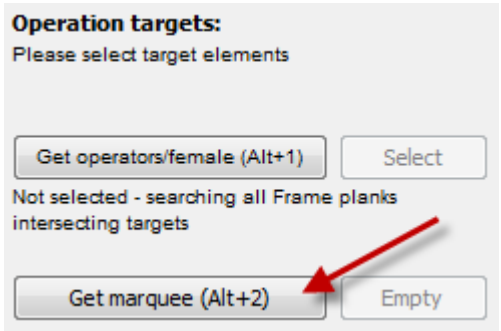
Balk wedge, U-shaped pocked to the operator and join accessory to the end of the beam:



For example, mortise and tenon joint to two intermediate floor beams connected to ArchiLogs log objects:

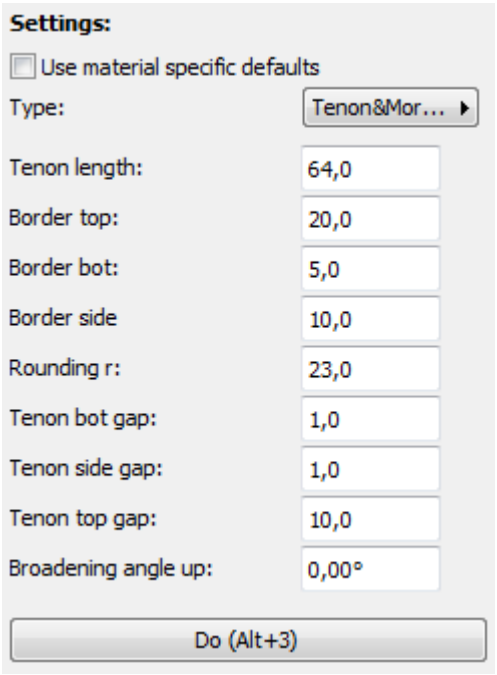


Pick the marquee as example above to limit the operation only to the upper ends of the beams. Then select two target beams. The mortise/pocket parts can be picked with *Get mortise/pocket* if needed. If the operators are not picked, ArchiFrame will search for every plank (or log) intersecting with targets that are inside the picked marquee, if there are any.



The default tenon length is half of the plank thickness. This value can be edited to match the target part. In this case the mortise will be placed at 128 mm thick log and the mortise depth will be 64

mm + 1 mm bottom gap. Rounding radius has two different values that affect the tenon: Zero makes rectangular tenon and any other value, a rounded tenon. With rounded shapes it is best to use the value that is close to the cnc-machine's cutter tool radius, for example 23 mm.



The result is like this in the mortise/pocket parts:

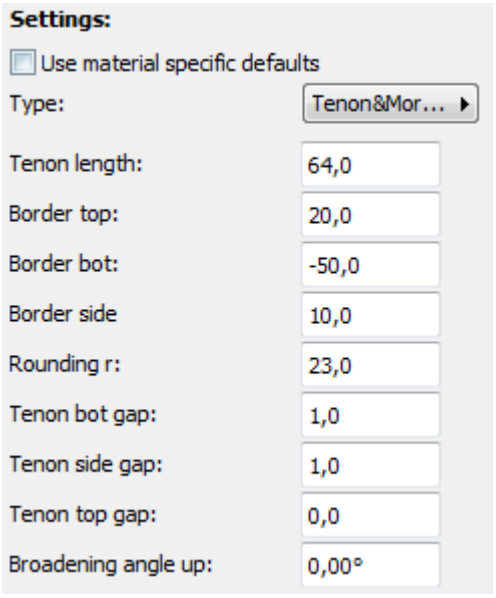


This is not satisfactory. Instead make tenons that are rectangular from bottom and that do not extend to the lower log. To do that, first manually remove the mortises from the logs.

Then select the upper log as mortise/pocket part:



Measure the level difference between beam bottom and mortise/pocket part bottom. In this case, it is 10 mm. Set negative value to *Border bottom* to be sure that cutter goes far enough to make rectangular bottom:

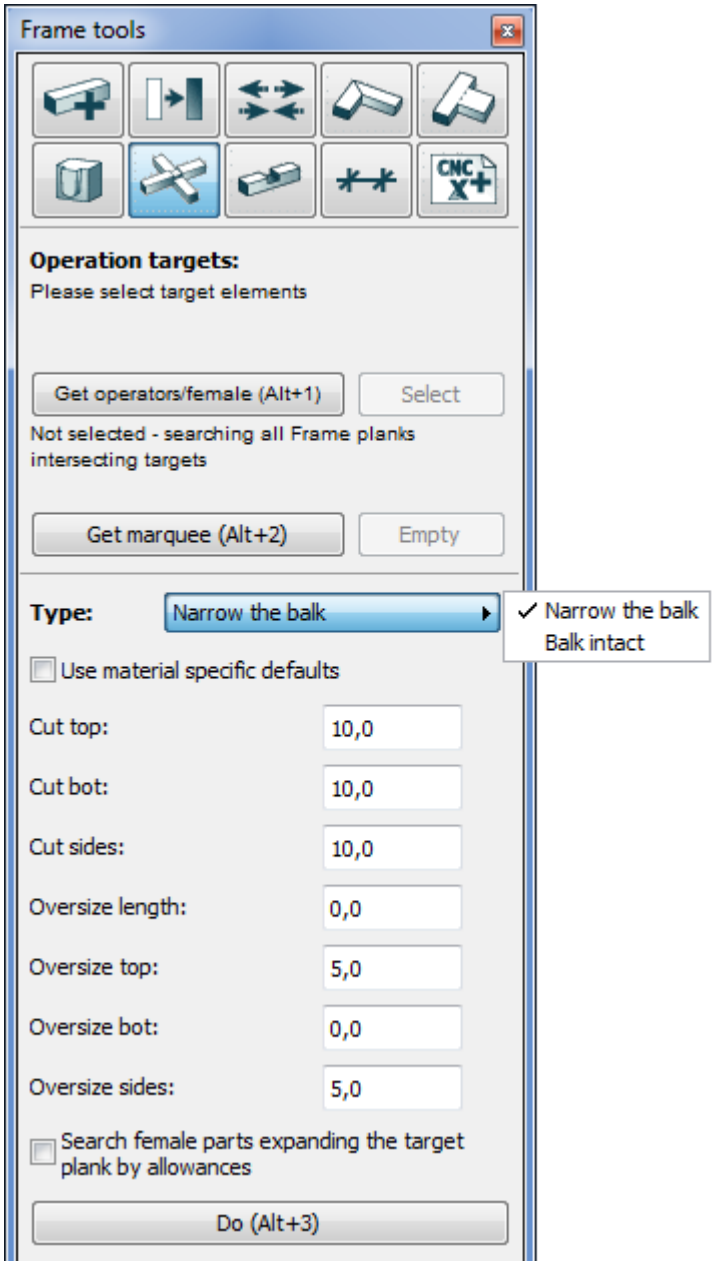


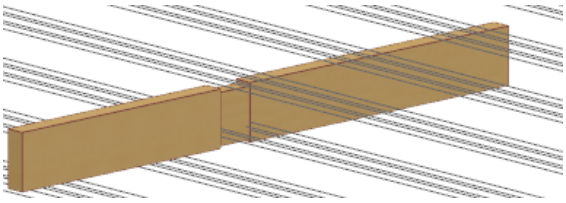
Then select the two beams and click *Do*. Note that the negative value in the bottom border causes a square shape at bottom of the joint.



8.7 Balk joints

This tool is used to make balk joints which have the balk either intact or narrowed. It also allows setting oversize for the target parts.

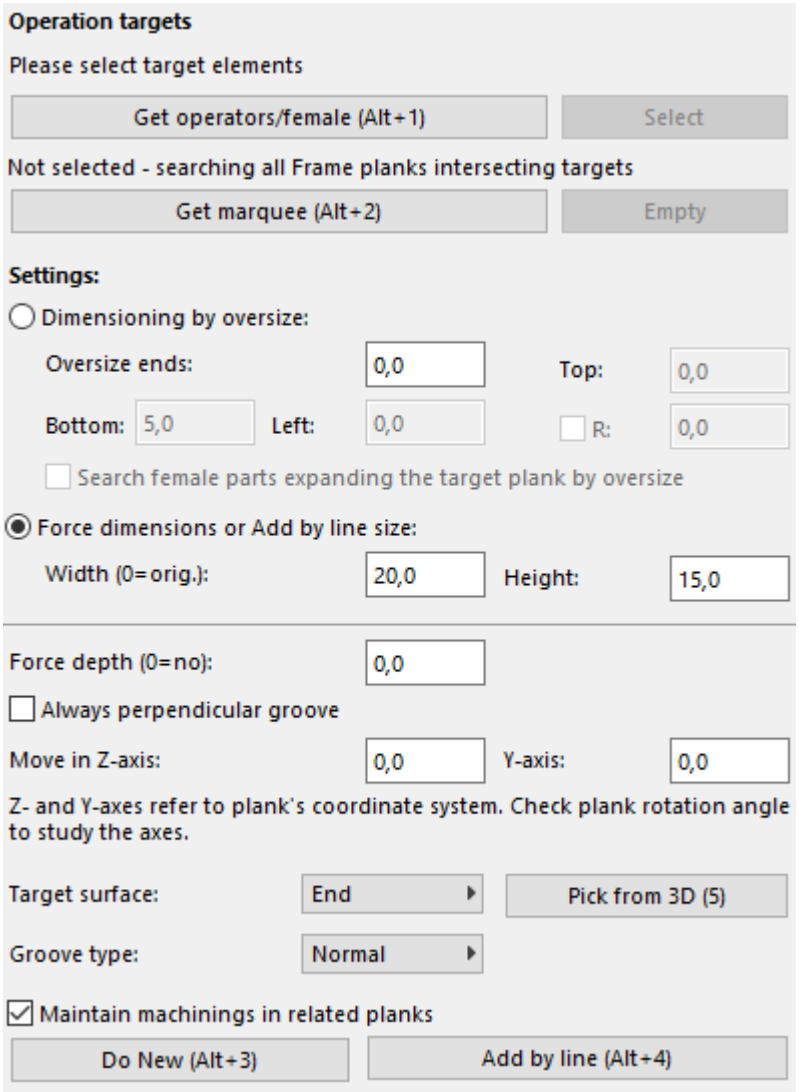




Narrowing values must be carefully adjusted for each situation. Oversize top can be used to set settlement space. Oversize length defines how much longer the narrowed part is compared to the target parts.

8.8 Grooves

This tool is used to make solid subtraction-like operations with selected elements (male parts). Male parts can be Frame-plank, log object, Archicad wall or beam. Roof is also acceptable if its slope is rectangular and parallel to the reference line. Target parts must be either ArchiFramePlanks or ArchiLogs' log objects.



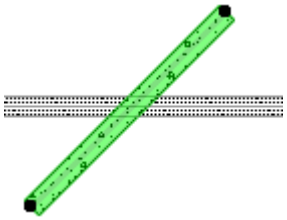
The tool resembles with the balk tool except for the following differences:

- Selected operator elements are not changed – those are used to make grooves to target parts.
- More adjustments to create grooves.

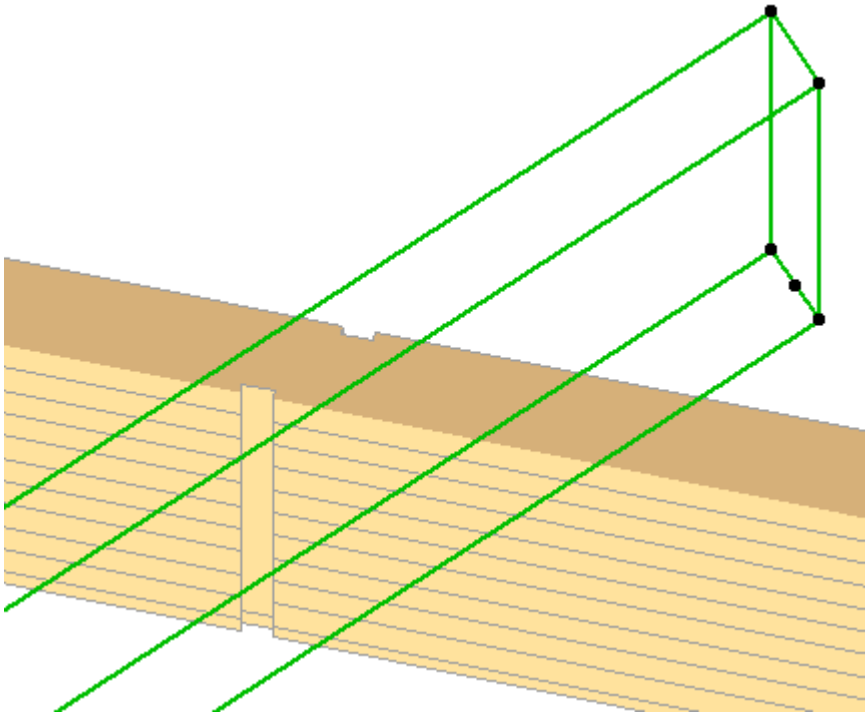
When working with *Dimensioning by oversize,* given values are added to male part sides. If option *Search taget parts expanding the target plank by oversize* is not checked, ArchiFrame searches the intersecting target parts with original operator part size. This can be used to ensure the target part will be cut, leaving a piece above or below untouched.

Force dimensions option changes male parts to a given size, keeping the reference line unmoved at the bottom middle. In other words, the bottom side and middle line remain at the same place and top side and sides are moved according to forced values.

Force depth makes the groove a given depth. If the operator part intersects the target part, the target part remains continuous but with forced depth grooves to both sides. Forcing depth works only for the male parts X-axis. For example, let's make 8 mm deep and 77 mm wide grooves in following situation to a log wall:



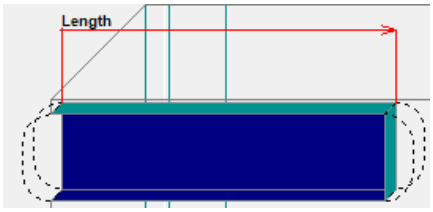
This is the result the male part on wireframe layer:

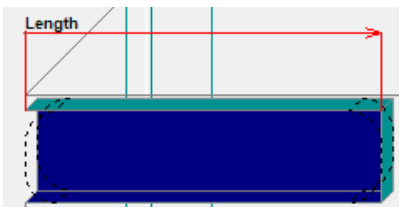


See also I-beam grooves.

Groove type affects cnc-creation. Possible values are:

- *Round*, rounded groove for visualization only. Radius is width/2.
- *Drip groove* will make the whole groove to move when dragged from hotspots.
- *Saw groove* may affect cnc-production (currently unused in standard cnc-writers)
- *Hun 1500* makes Hundegger cnc-writer to use code 1500/Dado.
- *Hun 1501* makes Hundegger cnc-writer to use code 1501/Rabbet. Rabbet can only be used at the edge of the part.
- Inside means that the groove dimensions define the fully cut groove, outside defines the total cut area in target (tool roundings inside the groove):

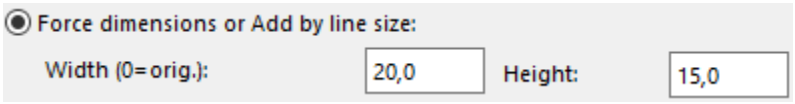
Inside: 

Outside: 

The extended tool palette gives these extra options:

- *Maintain machinings in related planks* to make solid subract type of connection between the pieces: When cutting piece is edited, the chante is reflected to the target piece.

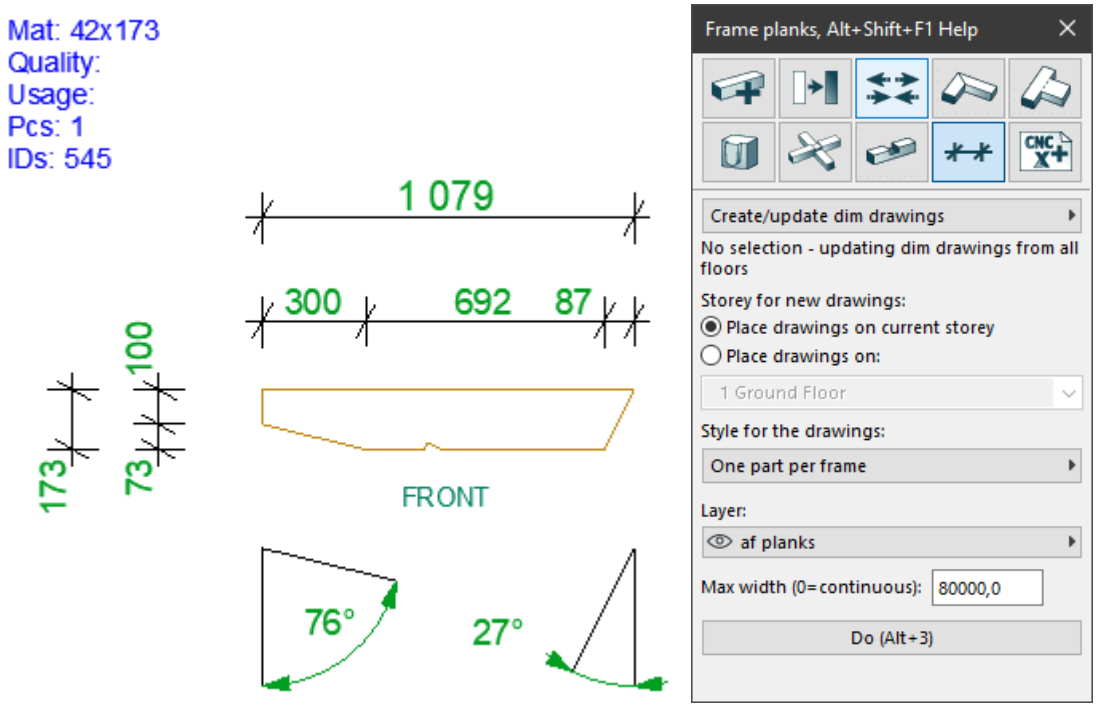
- *Pick from 3D* allows user to point target surface in 3D window.
- *Add by line* is used to point the groove's right-hand side. The groove extends to the left from the line. In this case these values define the groove size (height is the groove depth from target surface) and *Groove type*-settings sets the type:



8.9 Dimension tools

This palette contains two sub-parts: *Create and update dimension drawings* and *Dimensioning tool.*

8.9.1 Create and update dimension drawings



This tool is used to make detailed dimension drawings for selected planks. If the operation is started without any Archicad selection, ArchiFrame will update all existing changed dimension drawings. To force updating all dimension drawings, the *Ctrl*-key needs to be pressed while clicking *Update*.

Dimension drawings are not updated automatically and changes to dimension drawings will not be reflected in the original planks.

Parts of the palette are:

- *Storey for new drawing* will define where to place the dimension drawing. You can change the storey and then the old one will be deleted and the new one created to the pointed storey.
- *Style for the drawing* defines what kind of style to use. There are built in ones and the user can add own ones. The documentation is in the file ArchiFrameBlocks.xml under tag <dimdrawings>. New custom styles should be placed to the customer specific file ArchiFrameBlocksChanges.xml, please see its documentation.
- *Layer* sets layer for all other parts but the planks/boards. Planks/boards will be placed onto the same layer where the 3D-piece is placed onto.

- *Do* will ask for the place for the drawings and then create new dimension drawings for selection or update existing dimension drawings. If there is no selection, just parts having dimension drawings are updated if the dimension drawing is not up to date. If there are changes to the dimension drawing, old items are deleted, and new ones created.

8.9.1.1 Information for creating own custom styles

A dimension drawing contains some dimensioned projections of one plank or board, and additional data in text form.

A projection contains one side-view of the object, some dimensions around that side view, and a label (telling which side is this).

Dimensions can be horizontal, vertical, angled and diameter of holes. They all come from two data-sources: the basic sizes of the master object and various machinings applied to the object.

Frames are special objects to group dimension drawings. Different frames can be applied to each dimension drawing, or there can be one common frame for all dimension drawings of the selected objects.

ArchiFrame checks the selected elements (if any) and existing dimension drawings, then it decides if new dimension drawing creation will occur, or an update of existing dim drawing is needed.

Dimension lines will be drawn if their "showalways" parameter is set to 1. Otherwise, they are drawn only if they contain new information. All dimension lines around all projections will be checked in the dimension drawing, then ArchiFrame will decide, which ones are the most important. If a dimension line does not contain any new data compared to other dimlines, it will be skipped.

Similarly, projections can be skipped, too. If a projection has the parameter "showalways" not set to 1, and it has no dimension lines (for example they were skipped because the previos reason), it will not be drawn.

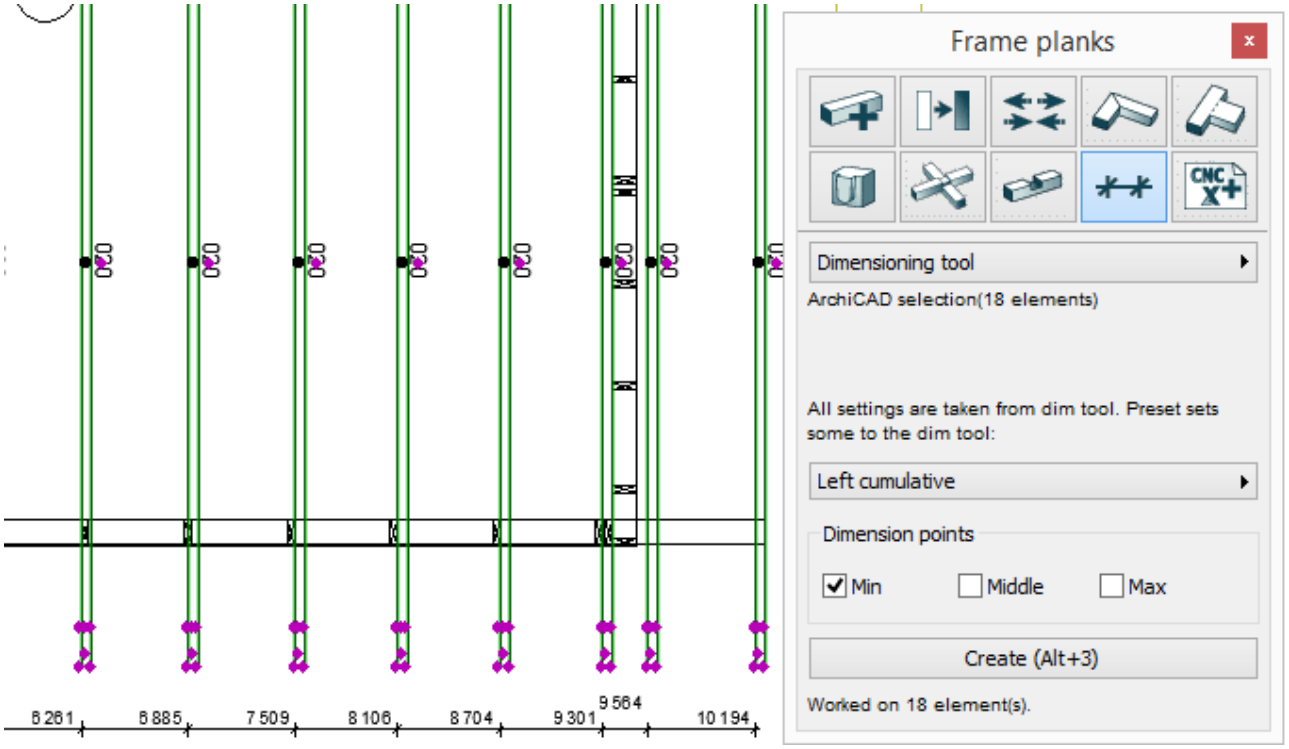
If a projection has parameter "showalways" set to 1, all its dimlines will be drawn for sure, too. It is like the dimlines automatically get showalways=1.

There are 7 types of dimension lines, as follows.

- main_horizontal: it is horizontally positioned, it has only two points showing one size of the element (only for main projection),
- main_vertical: it is vertically positioned, it has only two points showing another size of the element (only for main projection),
- machining_horizontal: it is horizontally positioned, beside the size of element it contains measures of machinings, if they can be represented commonly (available for all projections),
- machining_vertical: it is vertically positioned, beside the size of element it contains measures of machinings, if they can be represented commonly (available for all projections),
- machining_detail: all machining measure, which cannot be commonly drawn, usually has two points, for example showing drilling center point (available for all projections),
- machining_detangle: angled dimension, usually for cuttings (available for all projections),
- machining_dettext: used for displaying the diameter of drilled holes (available for all projections).

8.9.2 Dimensioning tool

This tool is used to make a dimension line for selected elements. Pre-settings are loaded from current *data*-folder's file ArchiFrameBlocks.xml. For example, cumulative dimension line anchored to the left sides of the rafters:



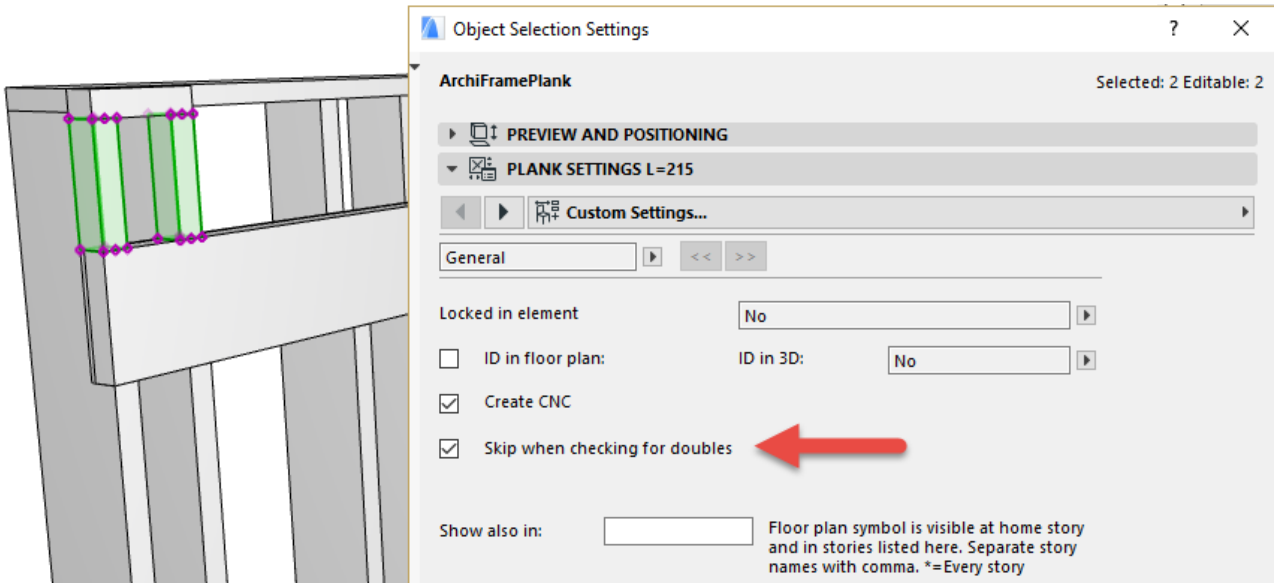
Dimension points define which places of the elements are added to the dimension line in dimension line's direction. Due to technical limitations, it is possible to add automatically updated dimensions only for objects. For other types, the dimension line will be static even if not selected from the dimension tool. This tool can be used with following Archicad element types:

- Walls.
- Columns.
- Beams.
- Roofs.
- Objects.

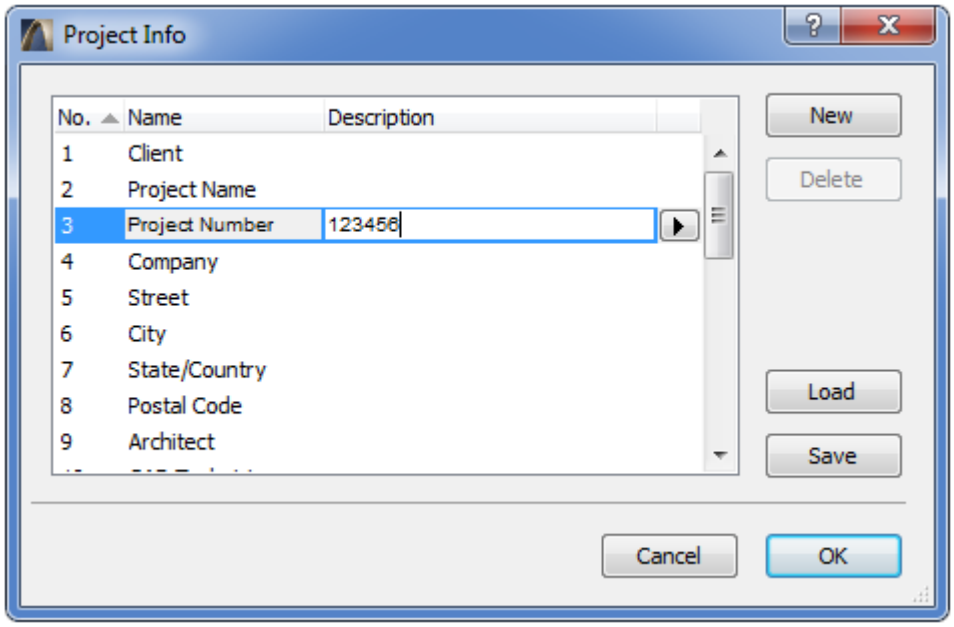
8.10 Excel- and cnc-files

These tools are used to produce cnc-file and quantity takeoffs either in txt or Excel format. Saving as an Excel file only works in Windows if there is Microsoft Office 2003 or newer installed.

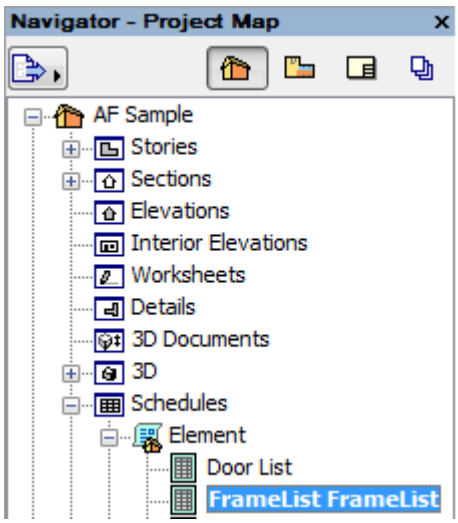
Check for doubles will scan either active storey or the whole model based on the selection. It will select all found doubles. This check is automatically made when writing any listing or cnc-file. ArchiFrame does just full plank collision checking skipping possible cuts and grooves. It is possible to skip false reported planks from plank's settings:



Excel´s default file name is the same as the current pln-file name with _excel.xls added. Cnc-file default name is taken from *File / Info / Project info / Project Number*:



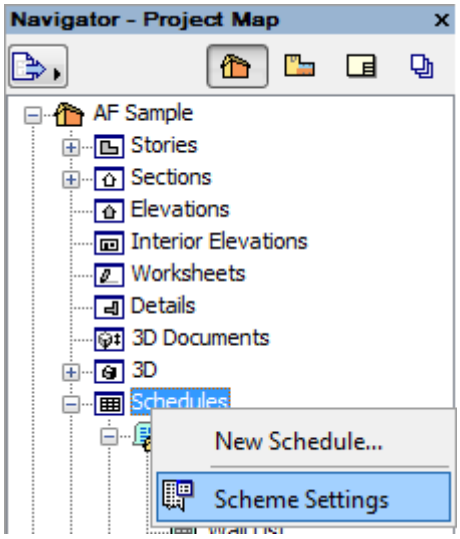
It is possible to use Archicad *schedule*-tool also. Please note that grouping similar planks must be left to ArchiFrame's *Give IDs* feature (or automatic for elements). With Archicad schedule it is impossible to distinguish similar planks with different machinings.



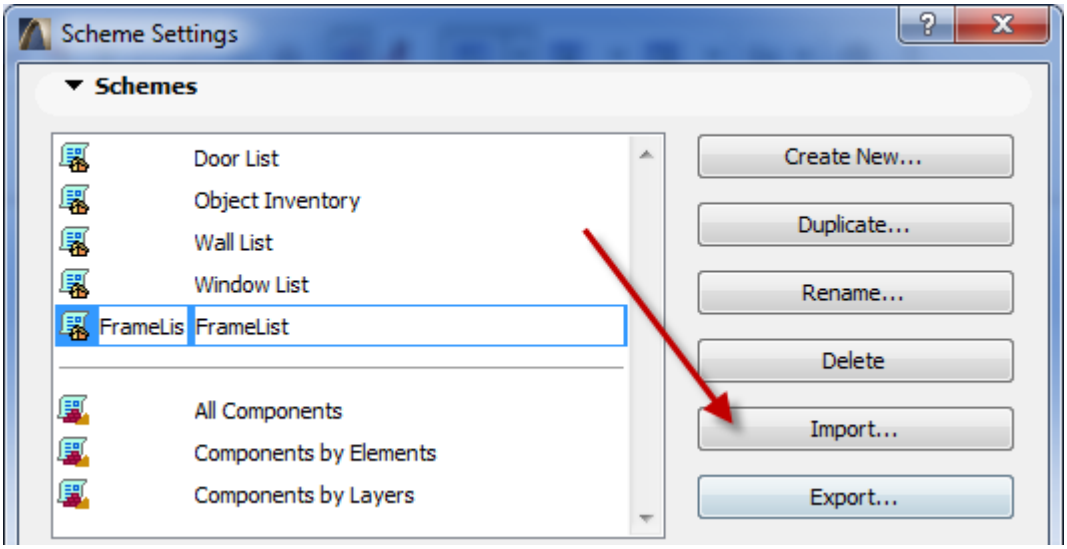
FrameList					
ID	Mat ID	Width	Height	Length	3D Front Axonometry
	I 300	47	300	4 744	
A	block	50	200	2 050	
B-001	45x45	45	45	2 509	
B-002	45x45	45	45	2 415	
B-003	45x45	45	45	6 077	
B-004	45x45	45	45	1 210	

The sample schedule is in *samples*-folder with name FrameList.xml. It can be imported to the actual model with these steps:

- Right-click Schedules in Navigator and select *Scheme settings*:

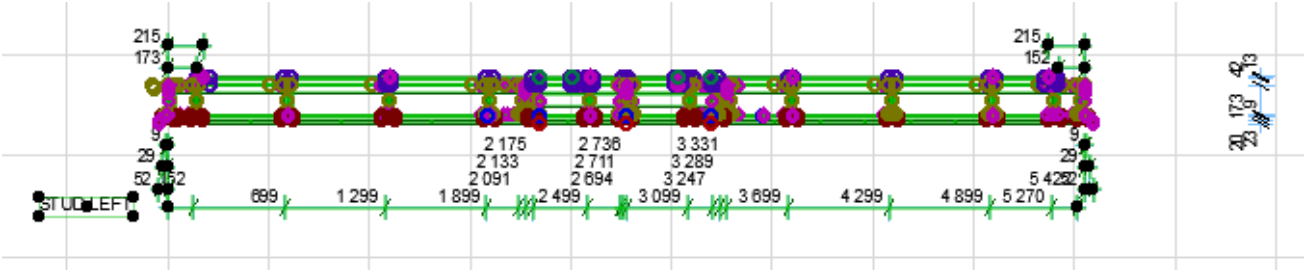


- Import FrameList:

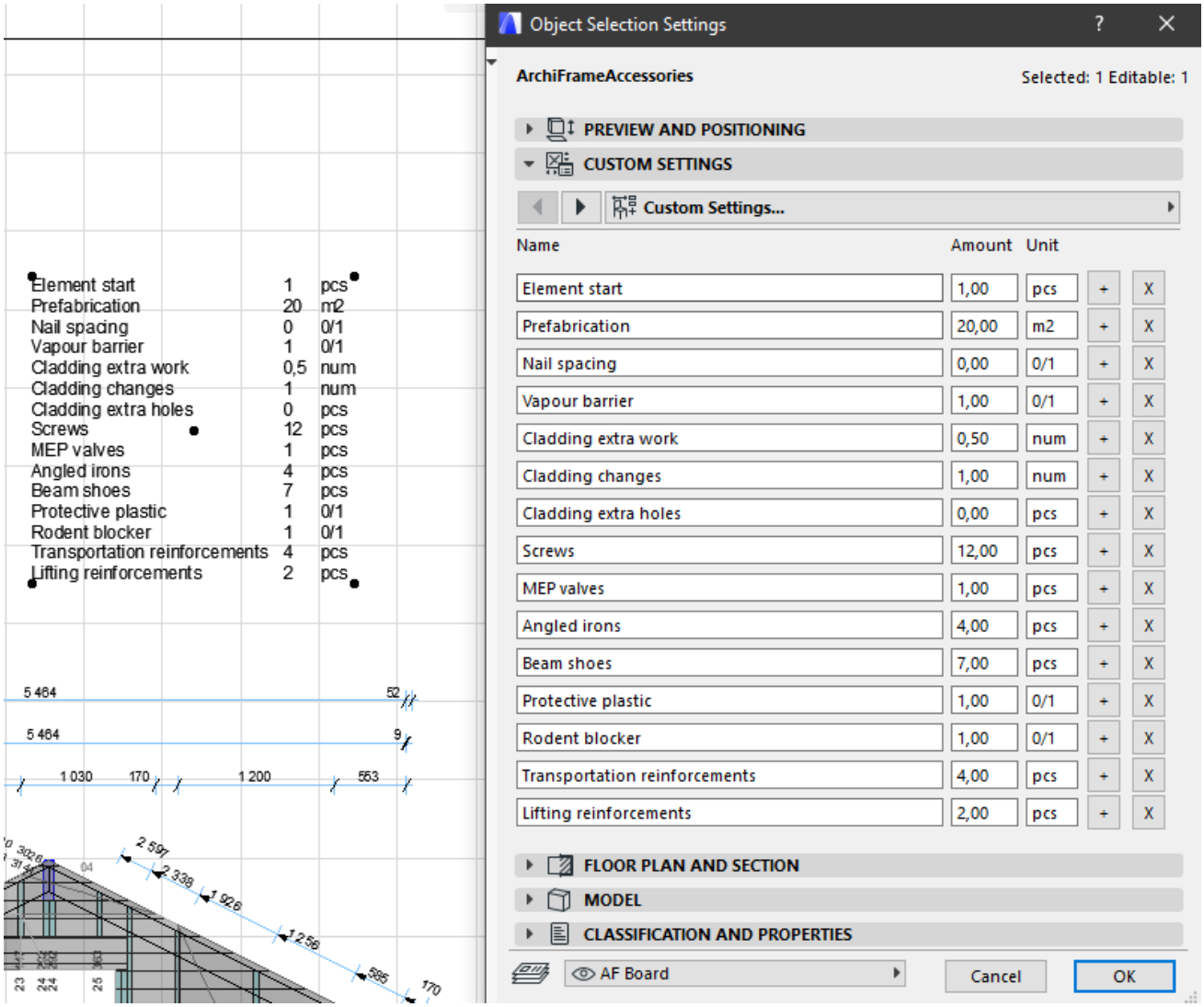


8.10.1 Special listing Element amount of work listing

To see how it works, please open sample file *ArchiFrameSample2015.pla*, switch to layer combination *AF 3D*, navigate to the storey *4 AF Elevations* and locate element *Y*. Then select everything from top view:

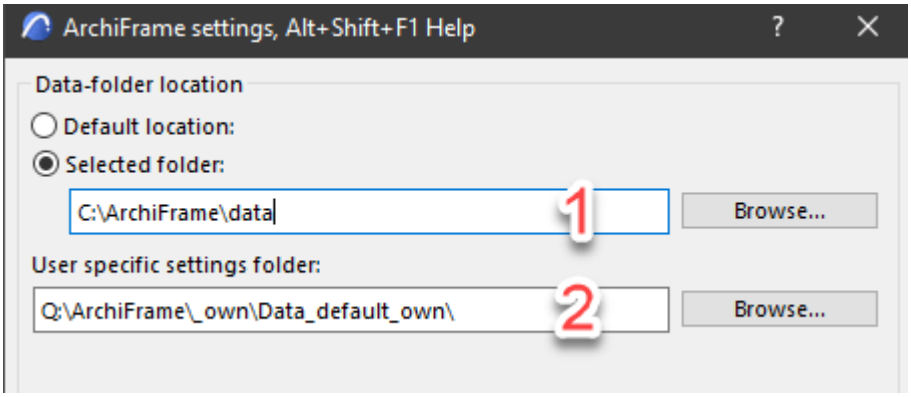


And write the xlsx-file. From the resulting file you can see the result and there are more instructions in sheet *Definitions*. Please note that ArchiFrameAccessories-object gives this listing more options:

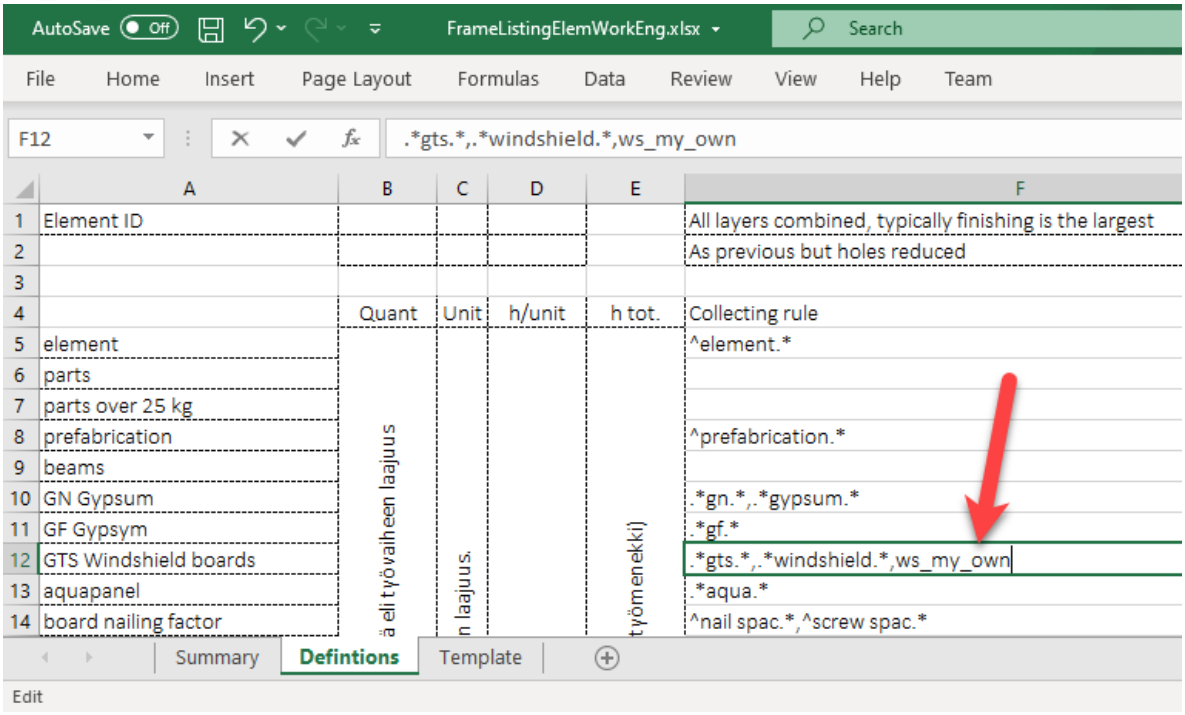


To change for example board collection rules, the following steps should be done:

- Copy the default listing template file FrameListingElemWorkXxx.xlsx to from folder 1 to the user specific folder 2:



- Edit the needed rows of the Definitions sheet, for example to collect also board type ws_my_own, add it here:



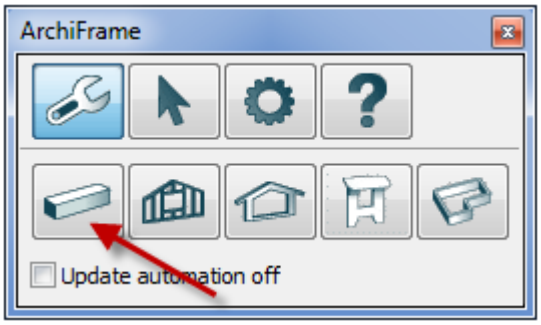
The board names use Lua patterns. Most useful is .* (dot+asterisk) that matches anything. ^ means beginning of the text.

Please see a video showing the steps to add the needed definition here.

9 Extended tool palette

9.1 General

tool palette opens by holding *Ctrl*-key down when clicking the plank tools icon (in Mac press *Command*-key):



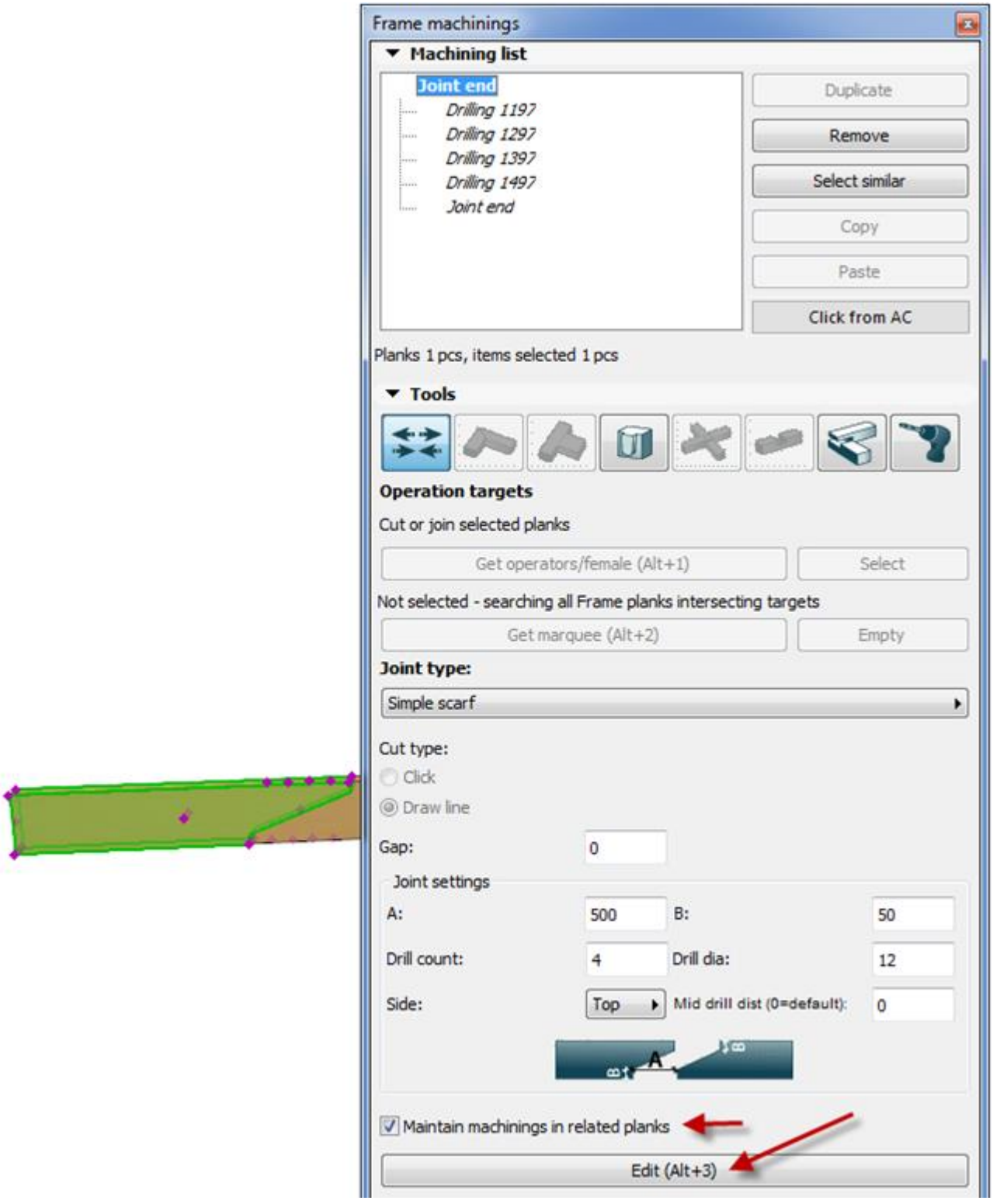
Extended tools have the option to update the machinings automatically. For example, in mortise and tenon joints moving the tenon part moves automatically the related mortise. Using the cut tool moving plank's end moves automatically also the related end.

When copying planks having automatic machinings, these rules are valid:

- If both parts of the machining are copied, the operation is copied to the new planks. For example, if a pair of lap jointed rafter is copied, the lap joint is also copied.

- If only other of the parts is copied, the operation is copied only if it is valid for the new plank. For example, copying groove target part will have the groove if new part intersects with operator part.

Existing machinings are edited by first selecting the machining to be edited, adjusting the values and finally clicking *Edit*:



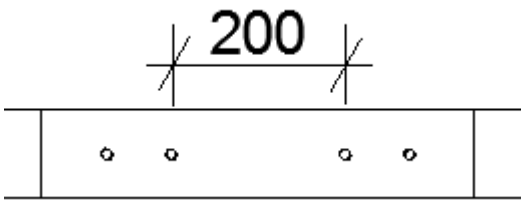
Greyed tools will be implemented later. Buttons related to machining list are:

- *Duplicate*, duplicates selected machining. The machining is edited at the bottom of the palette and finally added with *Add*-button.
- *Remove* removes selected machining(s).
- *Select similar* allows user to quickly expand the selection for example to all grooves.
- *Copy* saves selected machining(s) to be later pasted into other parts.

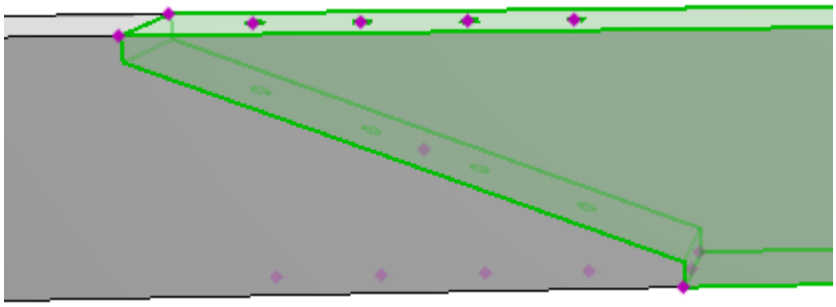
- *Click from AC* allows user to click the machining's hotspot in floor plan or ArchiFrame element elevation and then selects the clicked machining from the list. Works in AC 22 and newer versions.

9.2 Cut and join

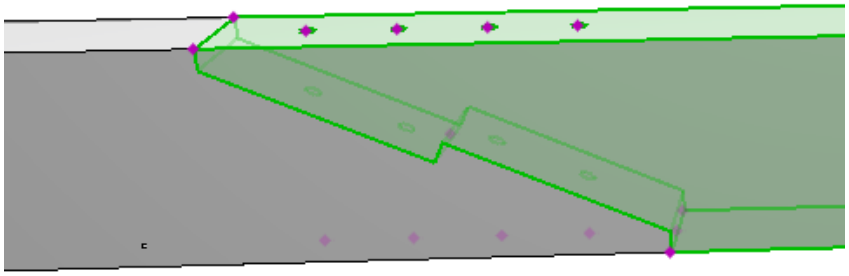
Please see Cut and join. In the simple scarf it is possible to give middle drill distance (for example with 200 mm):



Simple scarf is like this:



Scarf joint is like this:



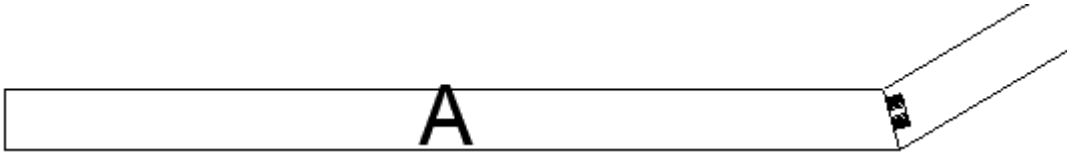
9.3 Dovetail, tenon, beam shoe and hobafix joints

Video clips:

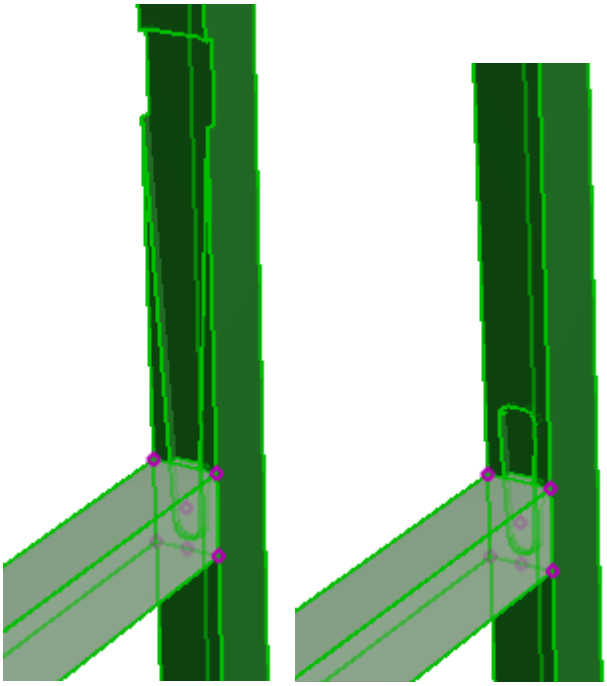
Video: https://vimeo.com/379999926
https://player.vimeo.com/video/379999926

Settings:

- Join to Side: Makes the joint always to operators' sides.
- End or side: Makes the joint to the end, if male plank's middle line intersects with operator's end.
- End and join: Joins the pieces together from ends and adds selected joint:

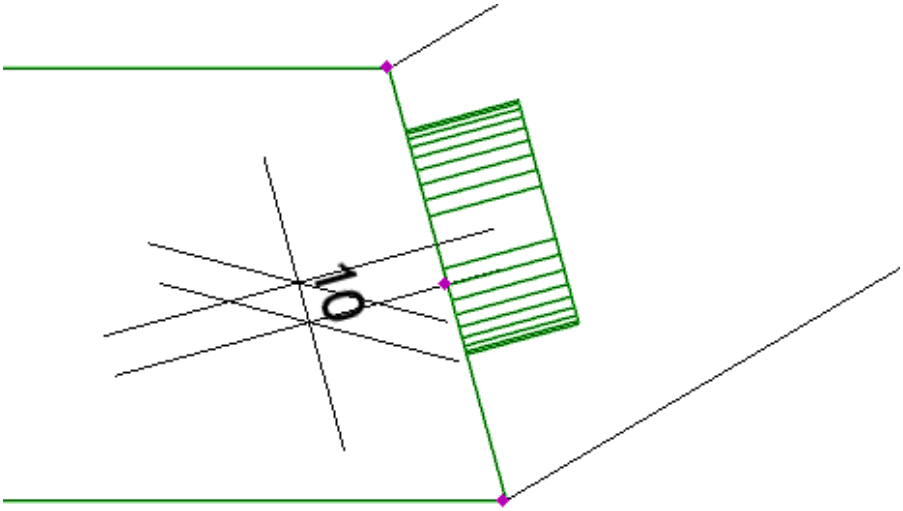


- Tenon length: Tenon total length also if double tenon is checked.
- Border bottom: Border for the tenon anchor side.
- Border top or Tenon height: Border for the opposite side to the anchor side or tenon height directly. Negative tenon height is used to limit the dovetail pocket height in case like below (left is positive value, right is negative value):

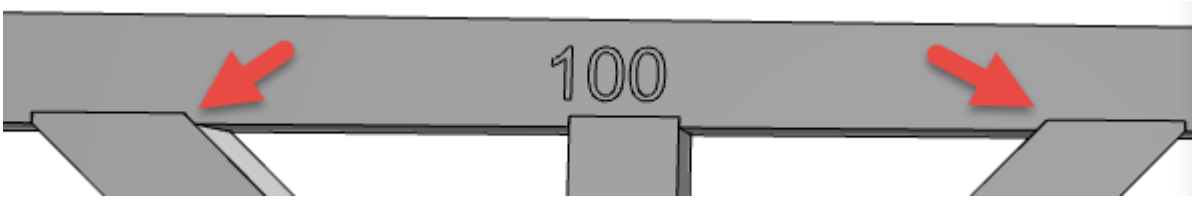


With negative value ArchiFrame visualizes the required extra cut by drawing the DT pocket doubled the height. The final result depends on the CNC-machine used.

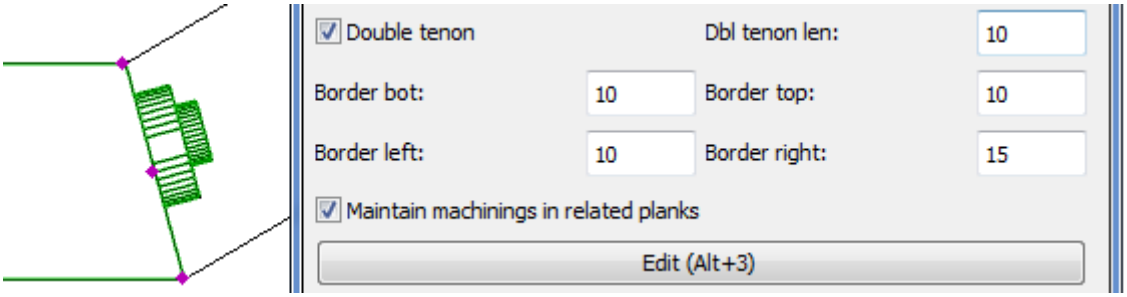
- Border sides or tenon width: Tenon width defined by border or by direct value.
- Rounding r: Zero makes rectangular tenon.
- Tenon bot gap: How much deeper mortise is compared to tenon.
- Tenon side gap: On both sides.
- Tenon top gap: Gap at side opposite to the anchor side.
- Broadening angle up: Mostly used with dovetail.
- Move tenon right: Offset from middle to right, negative value moves the tenon to left:



- Tenon anchor/dir: For example, upwards makes dovetail to widen upwards. For special cases the bottom side can be anchored to given plank side.
- Tenon rot angle: Counterclockwise angle looking in tenon direction.
- Housing depth: To sink mortise joint. Oversize for the housing sides is taken from setting *Tenon side gap*. Joining pieces must be similar sized.



- Double tenon: The borders for the end tenon. Mortise is bottom depth deeper than the tenon also in both the base and end tenon. Watching direction is the tenon direction also in this case:

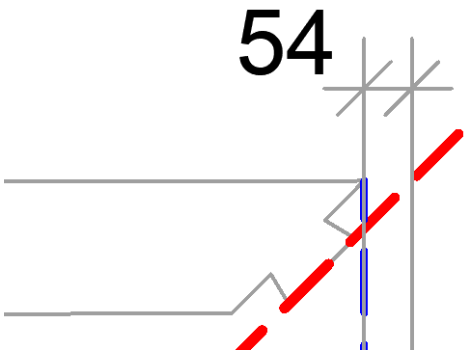


There is a related setting in ArchiFrameBlocks(Changes).xml attribute dtangle:

```
<cnclimits gromaxdepth="0.160" nailmindistplank="0.0031" nailmindistboard="0.0031"
elemgeo="1" dtangle="0"></cnclimits>
```

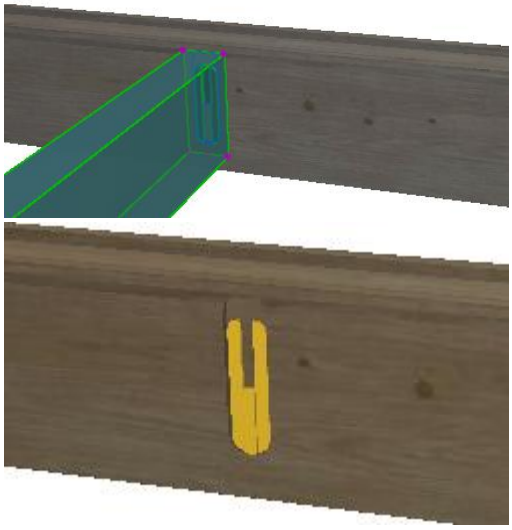
To optimize length of the planks having angled tenons and dt-tenons it is possible to specify the angle of dt-tool to dtangle-attribute. With value 0 no optimization is done to either tenons or dt-tenons. Nonzero value enables ArchiFrame to produce shorter planks – old way was to extend plank with the red line, new version checks the real needed length and uses blue line instead in

this case:



9.3.1 Hobafix

Hobafix joint is like U-shaped but ArchiFrame allows putting the mortise to the side part, beam end or both. Also, the metal accessory is modelled and calculated:



ArchiFrame sets the hobafix accessory ID just to the side connection piece to avoid calculating it twice.

Changes to input fields when using hobafix:

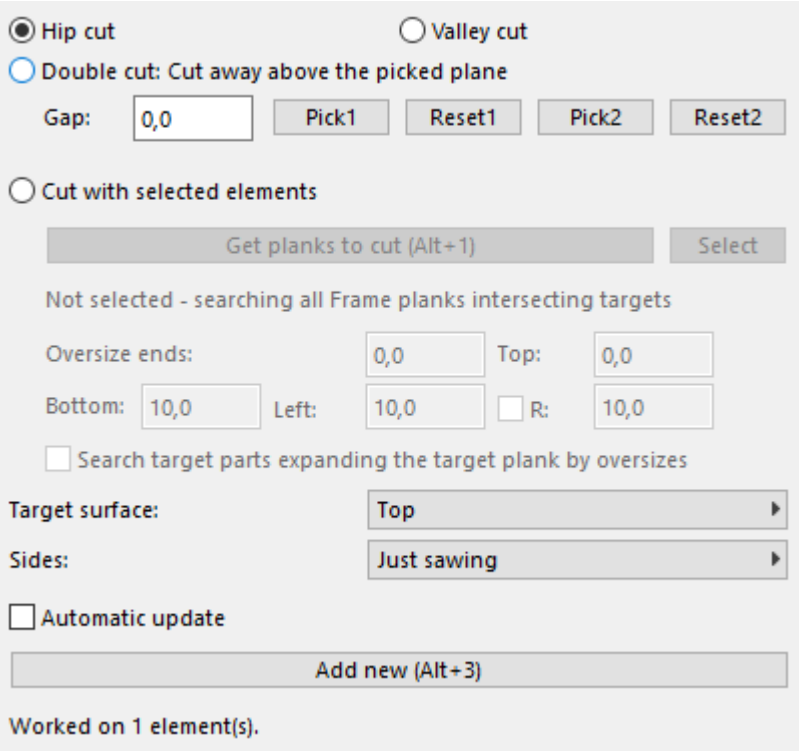
- *Tenon length* defines the pocket depth in the side connection piece. Setting this value to zero causes the pocket to be entirely at the beam end. If tenon length is not zero or hobafix thickness, both parts will get a pocket.
- *Tenon side gap* is the pocket oversize for all sides.

Hobafix-connection types are defined in ArchiFrameBlocks(Changes).xml:

```
<!-- List of hobafix-joints -->
<hobafixes merge="replace">
  <hobafix name="HF70" id="HF70" thickness="0.010" width="0.030" height="0.100"
roundingr="0.010" acmat="" len="0.010">
  </hobafix>
  <hobafix name="HF135" id="HF135" thickness="0.010" width="0.050" height="0.135"
roundingr="0.015" acmat="" len="0.010">
  </hobafix>
  <hobafix name="HF170" id="HF170" thickness="0.010" width="0.050" height="0.170"
roundingr="0.015" acmat="RAL 1018" len="0.010">
  </hobafix>
</hobafixes>
```

Xlm-attribute	Description
name	Shown in the hobafix type-list
id	Saved as connector type id
thickness	Thickness of the metal part
width	Width of the metal part
height	Height of the metal part
roundingr	Rounding radius of the metal part
acmat	If given, sets the surface material used in the metal part
len	Default value for tenon length

9.4 Lengthwise cut



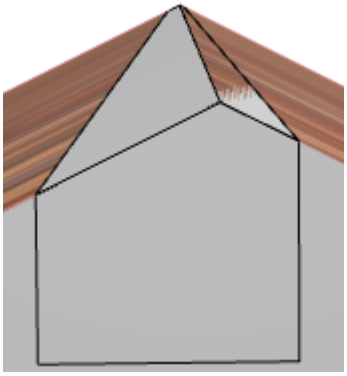
This tool is meant to produce lengthwise cuts made with a saw in resulting cnc-file. The cut must be parallel to the plank and it may be single or double cut. Use *groove*-tool to make free form cuts (made with a cutter in resulting cnc-file).

The tool can be used in two ways:

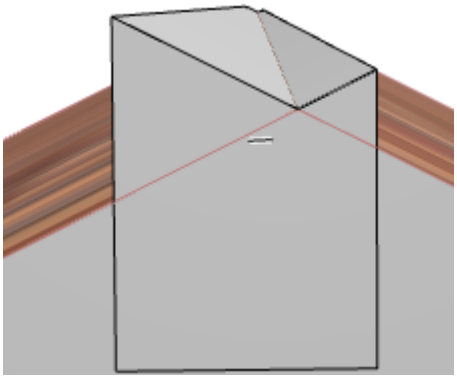
1. Pick static cutting plane or two planes from existing elements to make single or double cut. In this case the cut can be automatically updated if target plank is moved.
2. Cut target plank with another plank or Archicad wall, roof, slab, column or beam. If cutting piece is a plank, the operation can be automatically updated.

Different cut modes are like this:

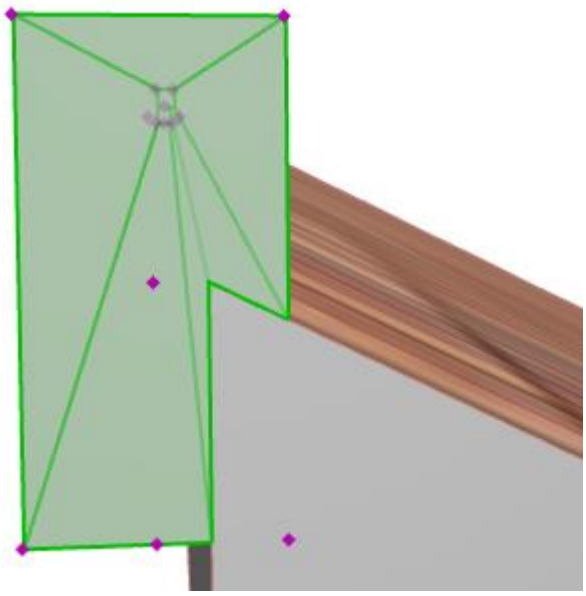
- Hip cut to the top of the piece:



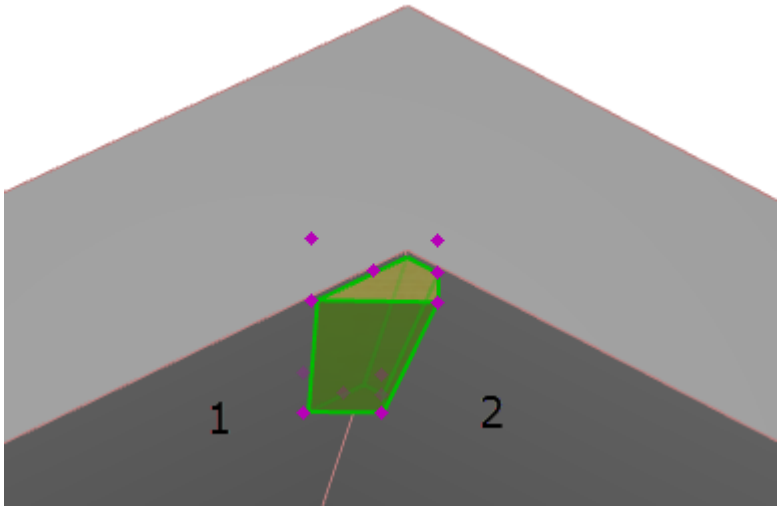
- Valley cut to the top of the piece:



- Double cut to the bottom of the piece allows result like this:

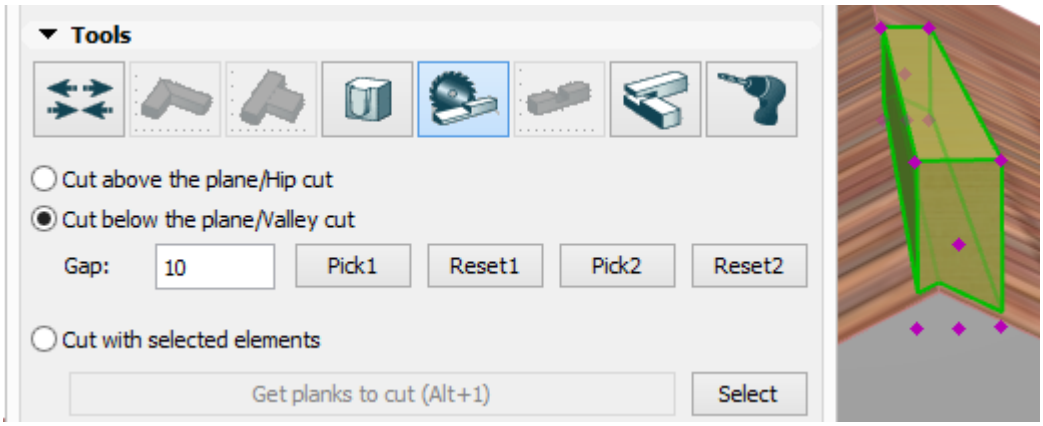


To produce cutting like this:



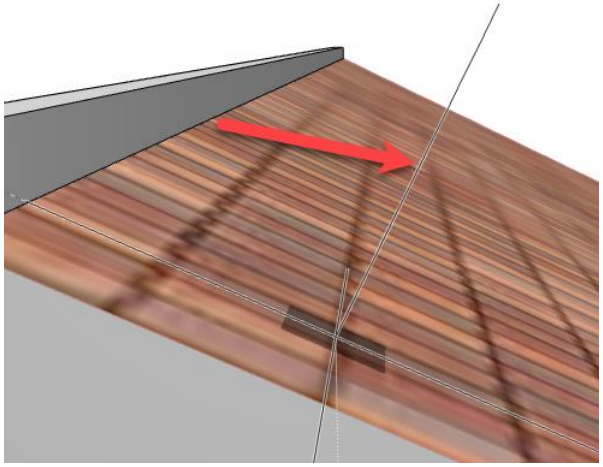
- Use settings above.
- Click on the button *Pick1* to pick cutting plane 1 (left hand side roof lower surface).
- Click on the button *Pick2* to pick cutting plane 2 (left hand side roof lower surface).
- Select the target plank and set target surface to front in this case (plank has been rotated 90 degrees counter clockwise, the reference line has pink hotspot).
- Click *Add new.*

V-cut is made the same way picking the cutting planes from roof upper surface and changing cut type and target surface to the bottom (please note the three dots showing the bottom surface):

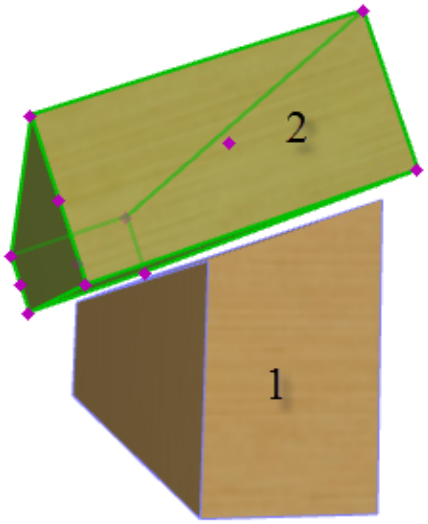


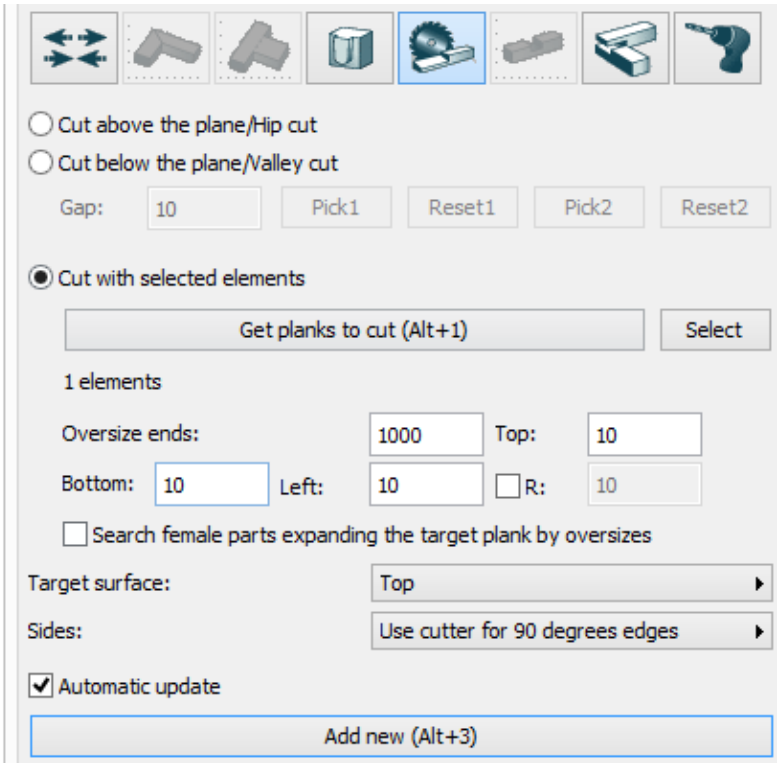
The *Double cut*-mode requires the user to pick cutting planes so that the plane is pointing towards the cutting direction and looked at the begin of the plank, the left hand side cutting must be picked as *Pick1* and right hand side as *Pick2*. If the cut is wrong, typically it is fixed by paying attention to the cutting plane's direction and order of picked planes. Please note that cutting plane direction can

be changed by pressing Ctrl/Cmd-key when pointing the plane. This is the direction pointing line:



To cut with planks like below:

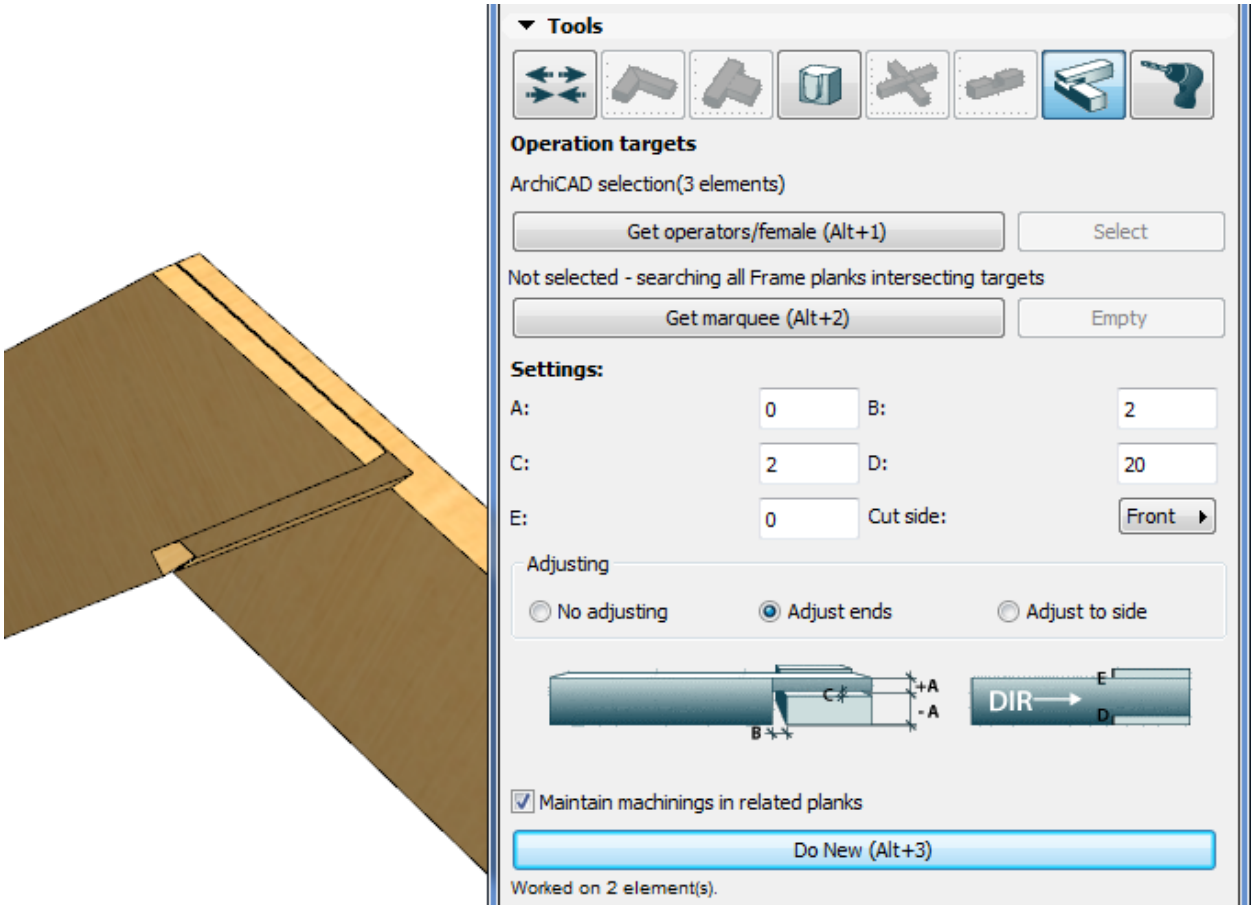




- Select plank 1 and click *Get planks to cut.*
- Adjust oversizes and other settings.
- Select cutting piece (2).
- Click *Add new.*

Sides-setting controls resulting cnc-: If the cut is not full plank's length, the cutter may be used to remove the piece and to make sharp corners. Final result depends on the actual cnc-writer.

9.5 Lap joint



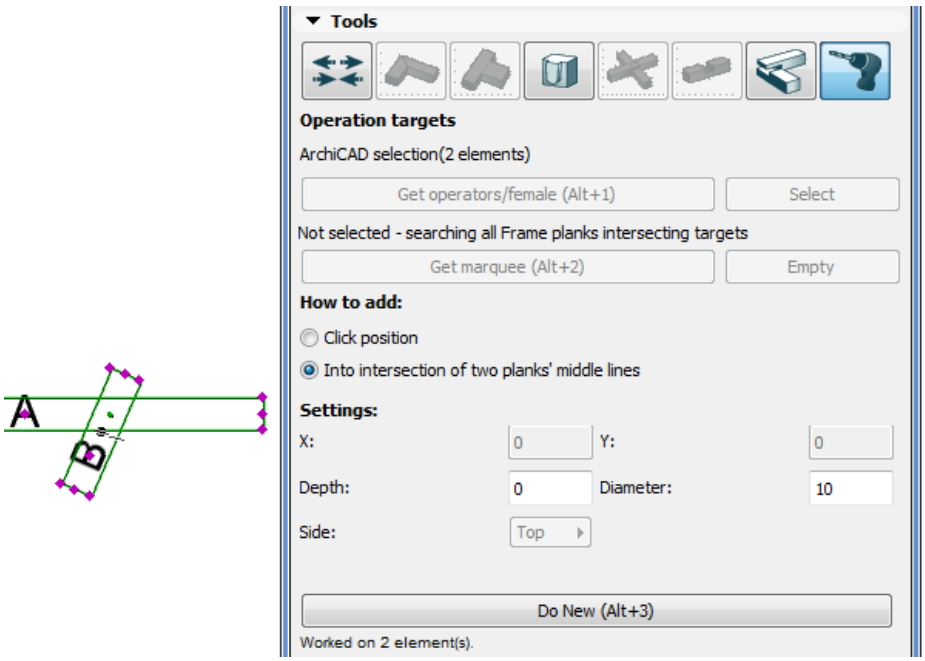
For the lap joint the plank pairs are selected (for example one or more rafter pairs), the settings are adjusted and readjusted if any specified. Then the joint is done with *Do New*.

9.6 Drillings

Related video:

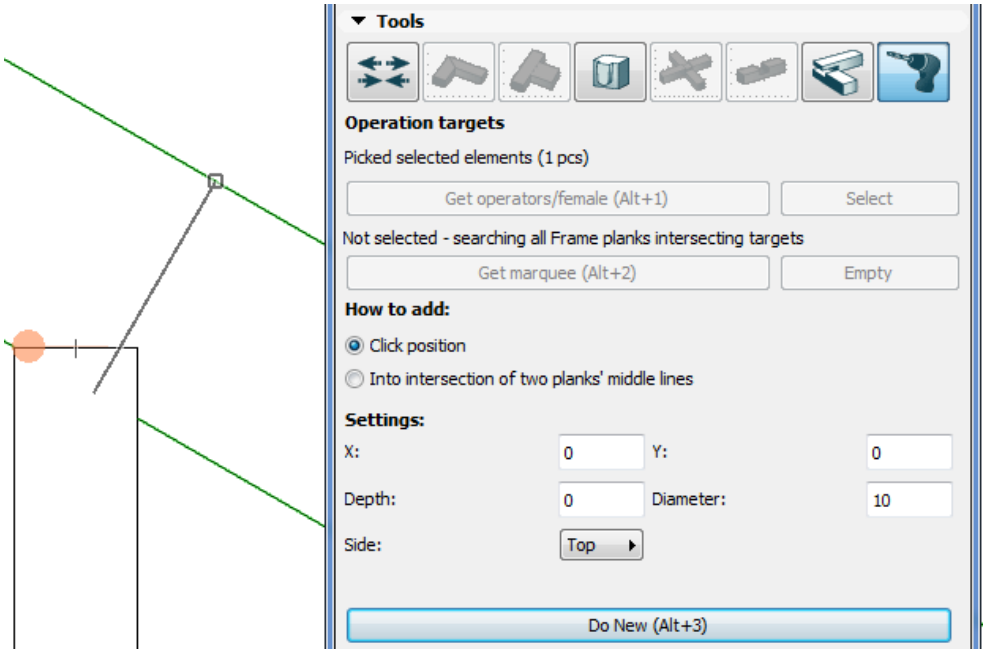
- Drillings explained

The simplest way to add a drilling is to place it to the intersection of middle plane of two pieces:



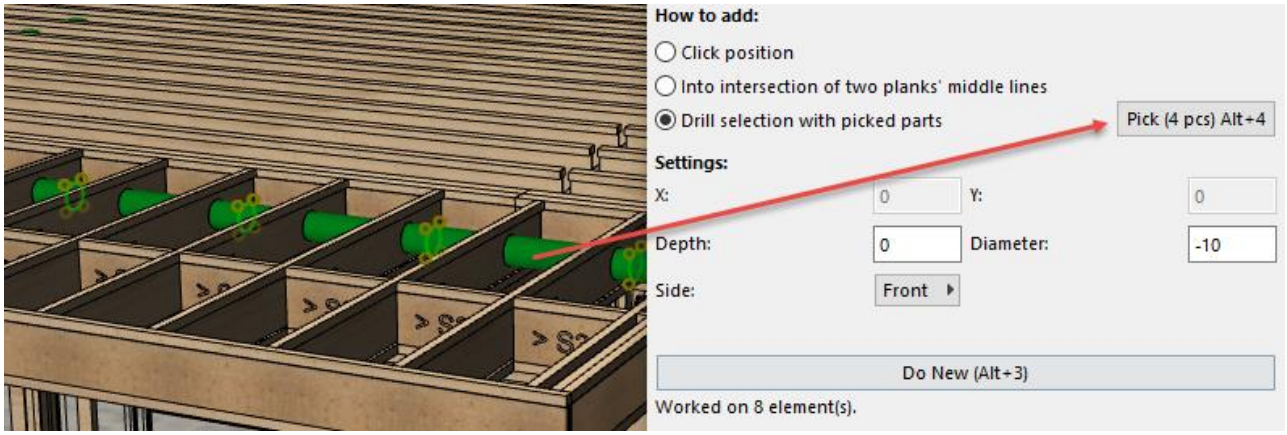
Zero depth means drill through.

Using *Click position* in elevation is the easiest way to add a drilling to a rafter based on supporting beam placement:



- X: Drilling position on target surface X-axis. With value zero the position is at the middle of the axis if the current view does not allow pointing the X-axis.
- Y: Similar for Y-axis.
- Depth: Value zero makes the drill through.
- Diameter: Drill diameter.
- Side: Target surface which must be given when using *Click position.*

Third option is to work in 3D and use any 3D element as drilling piece and drill the selected pieces. For example, here the green tubes are picked to make the drillings, then target pieces are selected and by clicking *Do New* it will add the drillings based on the 3D-information:



In this case negative value in *Diameter* defines the oversize used in the drill (diameter is 10 mm bigger than the green tube). 3D representation of round pieces is inexact, so it may be better to force the diameter. ArchiFrame takes the section of picked parts on the surface defined in Side-setting.

10Elements

10.1 Element object

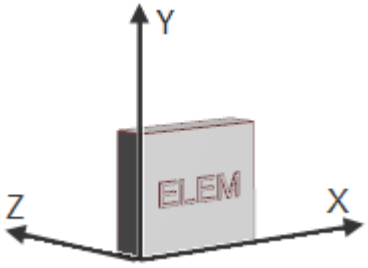
Element object represents the orientation and shape of a single structural area in a building. There is one element drawing per element object. The single plank can be visible in none or many element drawings. If the house is built on-site it is best to set the element object to contain the whole wall. If the elements are manufactured at the factory, then element object models one element. Element object IDs are set to planks that the element owns. The plank ID will have the form: [element id]-001 where 001 is number inside the single element. Similar planks will have a similar ID.

The process to make framed walls (and other structural elements) is:

1. Place element object and edit those to give correct corners and general shape.
2. Create planks to elements.
3. Attach required beams and supporting columns to individual elements.
4. Check and edit the element elevations. Lock the key planks and recreate planks if necessary.
5. If the element geometry changes, the planks must be recreated with *Create planks* command.
6. Use *Update* without any AC selection to recreate all elevations (recreates every dimension line).

Changes to the original AC-elements (walls, roofs, slabs) are not reflected in the element objects or from those to framed elements, so updating must be done manually.

Element coordinate system is:



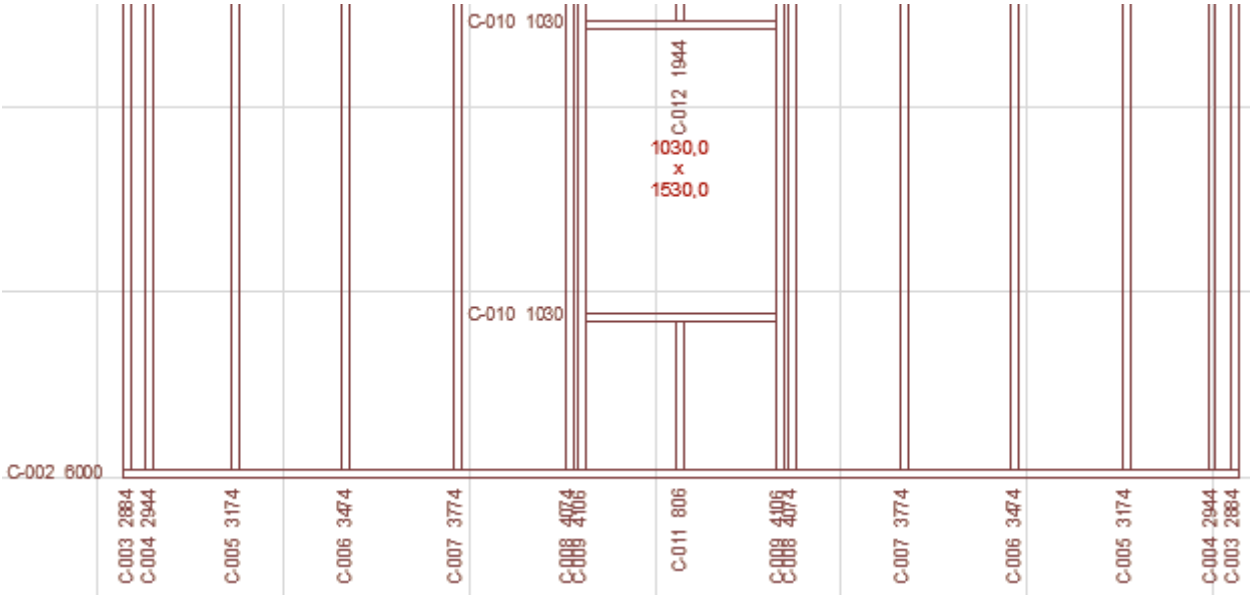
The main elevation viewing direction is the Z-axis. The element is always viewed from its right side.

10.2 Planks and element elevations

Elevations contain projections of the original planks. These original planks are called master planks. ArchiFrame keeps the master planks and projected planks synchronized. In the projections it is possible to move, drag plank ends and edit the planks. Also mirroring, mirroring a copy and dragging a copy are supported in projections. The master planks will move on the projection plane. Projection planks do not have 3D – it is not possible to select these planks and show those in 3D. Instead *ArchiFrame selection* tool must be used to select related master planks and to see those in 3D.

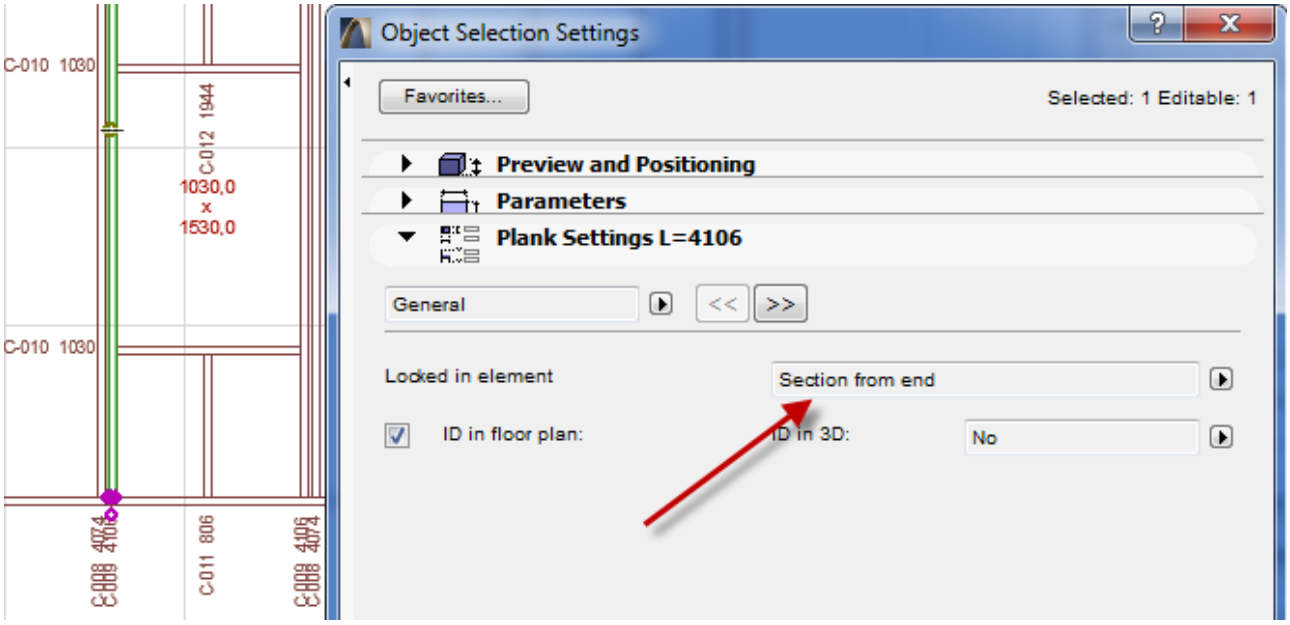
10.2.1 Locking planks in element elevations

Here is an example of an element created with default rules:

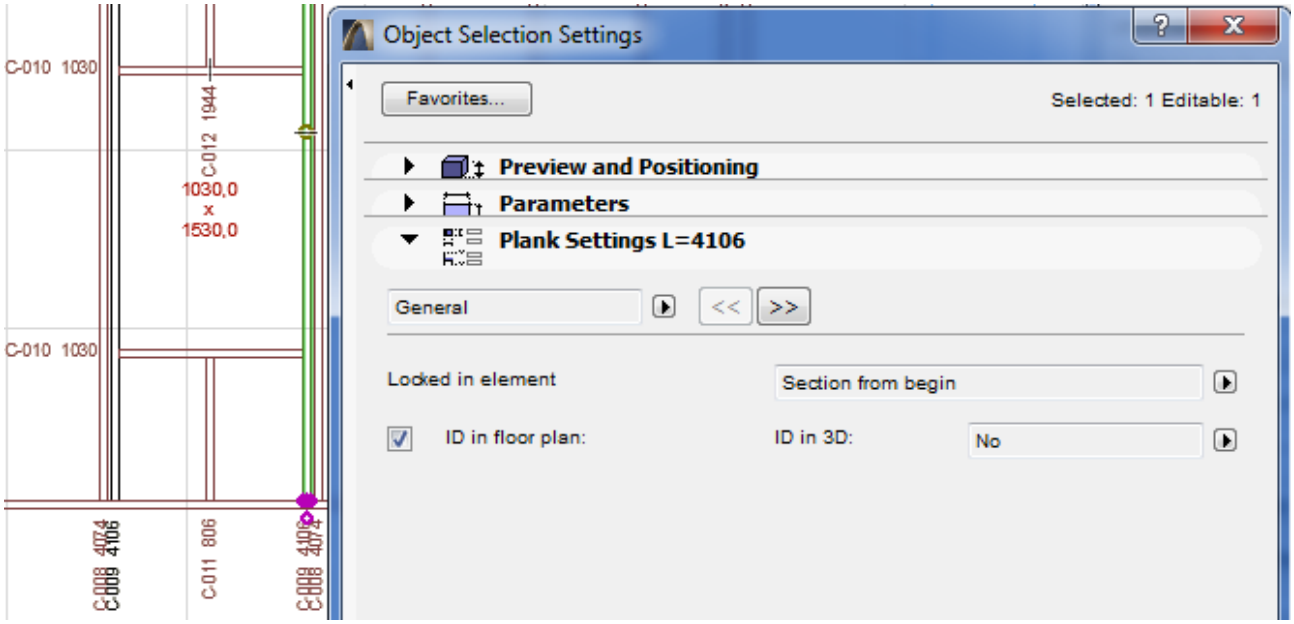


To force starting division from both sides of the opening we do these steps:

1. Set locking to *Section from end* (origin for the section is at the end of element, will continue to left side).

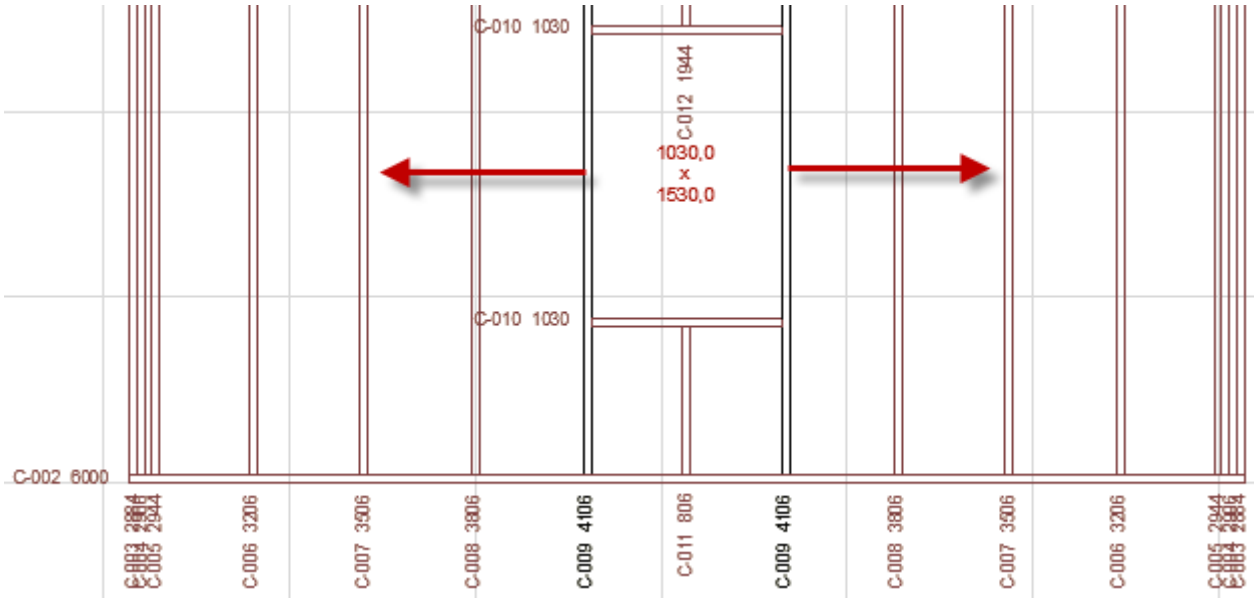


2. Do the same for other side setting *Section from begin*:



3. Recreate the element planks with element tool button *Create planks* having one plank selected (it defines the target element for the operation).

Result is like this (the arrows show the direction of guided stud placement):



It is possible to define multiple sections by having one or more pairs of locked studs.

10.3 Boards

Boards are like planks, but they have polygonal outline. In other words, the boards can be edited like planks and it is possible to apply machinings to the boards.

11 Add & Edit element tool

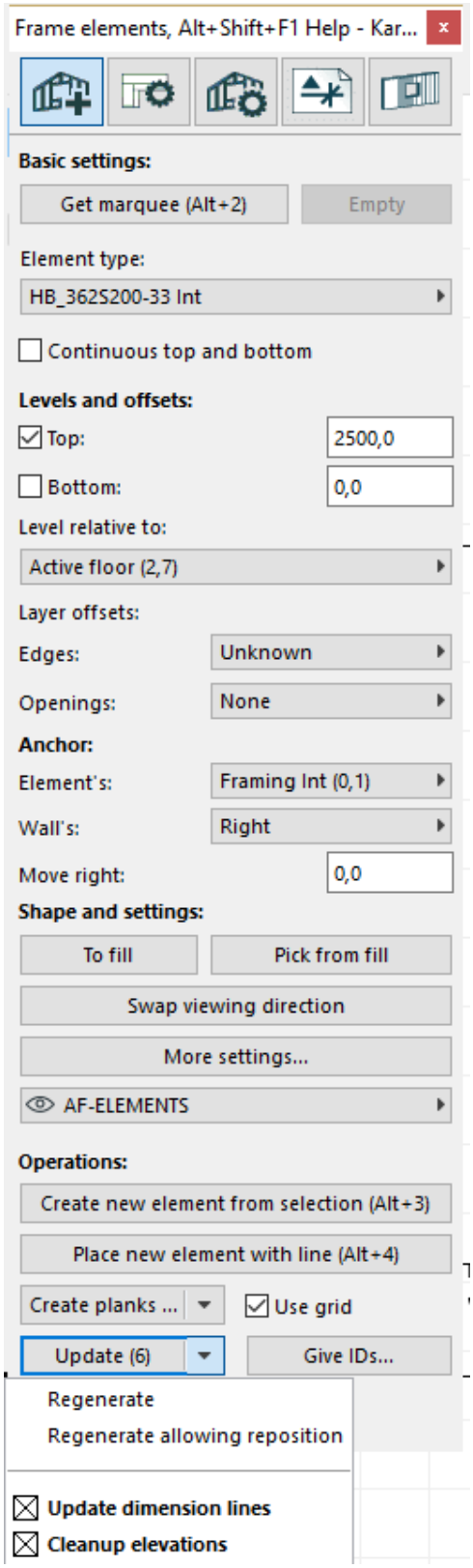
Video clips:

ArchiFrame elements: an overview, https://vimeo.com/178327076.
https://player.vimeo.com/video/178327076

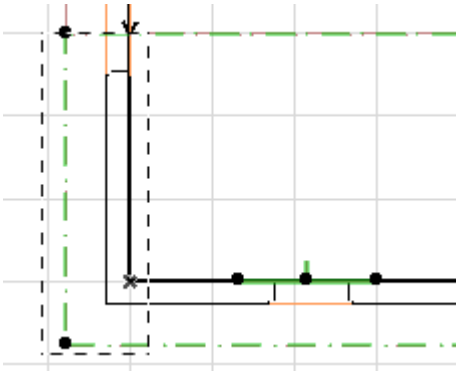
Placing rectangular wall elements, https://vimeo.com/180035580.
https://player.vimeo.com/video/180035580

Placing complex wall elements, https://vimeo.com/181255910.
https://player.vimeo.com/video/181255910

Adding a wall element with steel framing, https://vimeo.com/173754604.
https://player.vimeo.com/video/173754604



- *Get marquee* create an element with marquee boundaries instead of the original element, using Archicad marquee. It can be also extend original Archicad roof or slab.



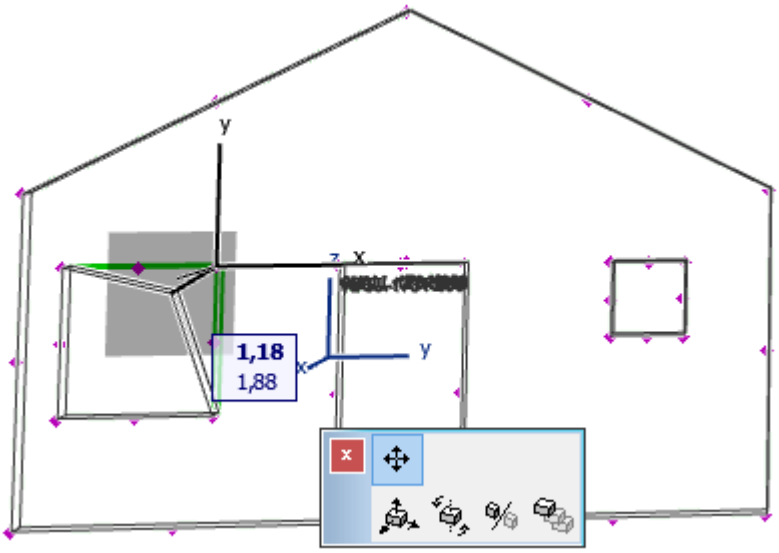
- Marquee can be picked from the AC marquee or any polygonal element (hatch, slab etc).
- The element types are defined in *data*-folder's file ArchiFrameElements.xml. It will set type for new elements or currently selected elements. Command *Apply element type's new definitions to selected target(s)* will apply changed settings from custom element type's composite settings to selected structure. This command is unavailable if there are no changes to apply. *Create planks* will apply rules defined in the layer level.
- *Continuous top and bottom* results as continuous top & bottom sills even if an opening goes all the way up or down, for example, in this case:



- *Force new/set old level* sets the element's bottom and top levels. Please see this video. For existing elements, first select the element object and then enter the value. Rules for searching the walls when placing the element with line:
    - If there are any selected walls when drawing the line, only those walls will define the element's geometry.
    - Without selection, all walls intersecting given bottom and top levels and drawn line will be taken. If no bottom and top level is forced, levels used for search are current storey's bottom and top levels. The wall placement storey does not have any effect.
- *Layer offsets edges/openings,* please see Custom layer edge/opening offsets. Please note that an area to work with openings can be limited by earlier picked marquee (picked with button *Pick marquee*). ArchiFrame will apply opening offsets to any opening that intersects with the picked marquee. The marquee can be set either on floor plan or on any projection.
- *Anchor* specifies the anchor point used for original AC-element and new element object. Selected points will be matched. For example for walls, it is useful to match the element's inner surface (plasterboard in for example) and wall's interior side.
- *Move right/up* is offset value for matching the anchors.
- *To fill / Pick from fill* allows you to edit the shape of ArchiFrame elements and boards with the help of fills.
    - *To fill* creates a 2D fill from the selected element. You can then edit the fill's shape with standard Archicad tools.
    - With *Pick from fill,* you can inject the shape of the edited fill to the ArchiFrame element. If the fill origin is moved after placing it, the movement is reflected to original ArchiFrame object. To intentionally avoid this, make a duplicate of the fill,

move it and pick the shape from the duplicate fill. In this case, you can have only a single fill since ArchiFrame has saved the target ArchiFrame-element when the fill was initially created with *To fill*-button.

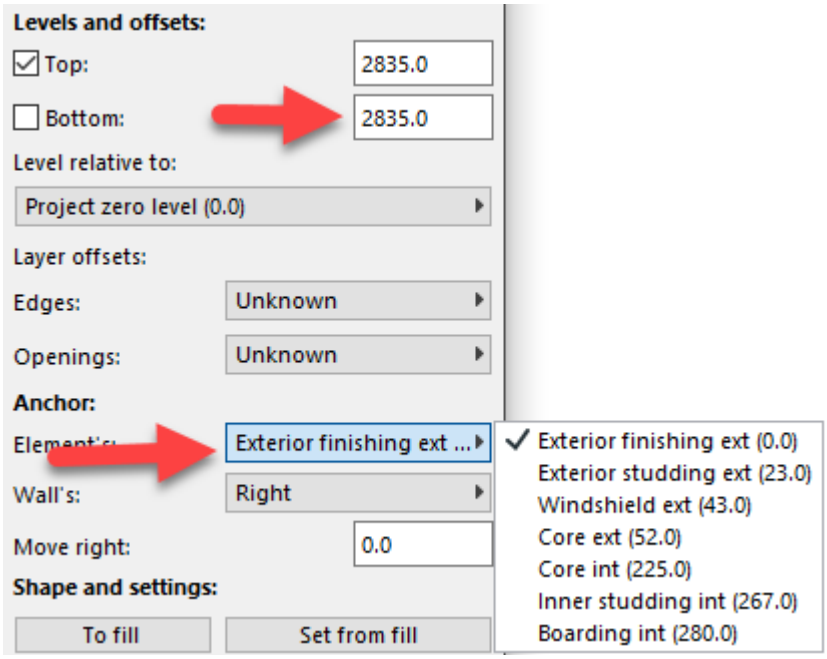
- For multilayer elements *To fill* creates fills for each layer unless *Suspend element groups* is checked on. *To fill* can be used also to edit board shape in element drawing.
- Element and board shapes can be edited in 3D/sections with moving hotspots (converting to fill is still needed to add new points to the polygon):



- *Swap viewing direction* rotates the element by 180 degrees and mirrors the element polygon. In short, it swaps the viewing direction.
- *More settings* opens Element settings dialog.
- *Create new element from selection* uses the AC-selection to create a new element object. The selection may contain many source elements. Every element is projected to the first element's plane and only one element object is created from the selection. This design allows constructing element object from many Archicad elements and on the other hand forces user to create each element object separately. ArchiFrame will ask for direction of the rafters/joists before creating the new structure. The direction is given by a line.
- *Place new element with line* places new element based on entered line. Note that element will be viewed from the right side of the line and that is the standard side for exterior side. The tool always uses the tilt value from Force tilt. If there is a wall where the line is entered, its geometry is taken into element projection. If there is no wall, the level values from Bottom and Top fields are used:

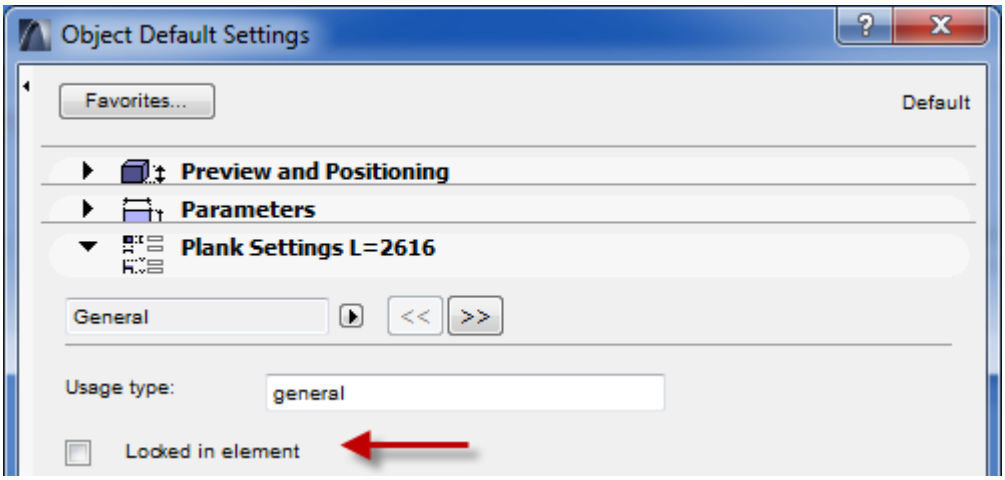


- *Place new horizontal item* will place a slab kind of element. The user shows the rectangle of the element in the floor plan and the level from the *Bottom*-input of selected anchor is used for the newly created element:

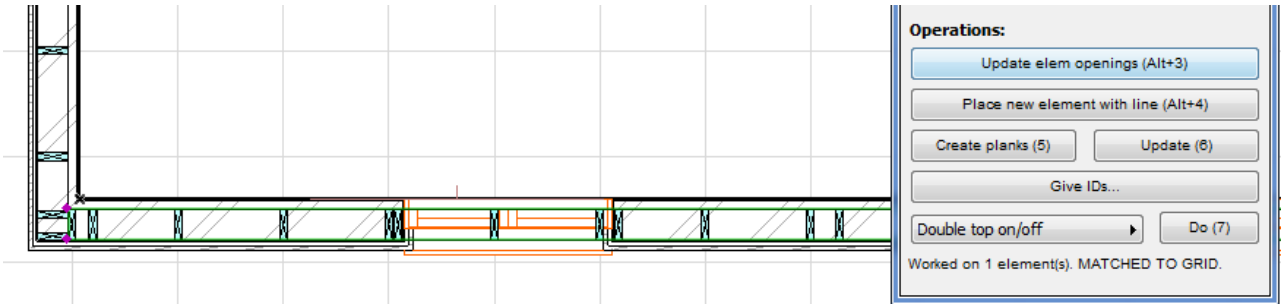


Please see this video.

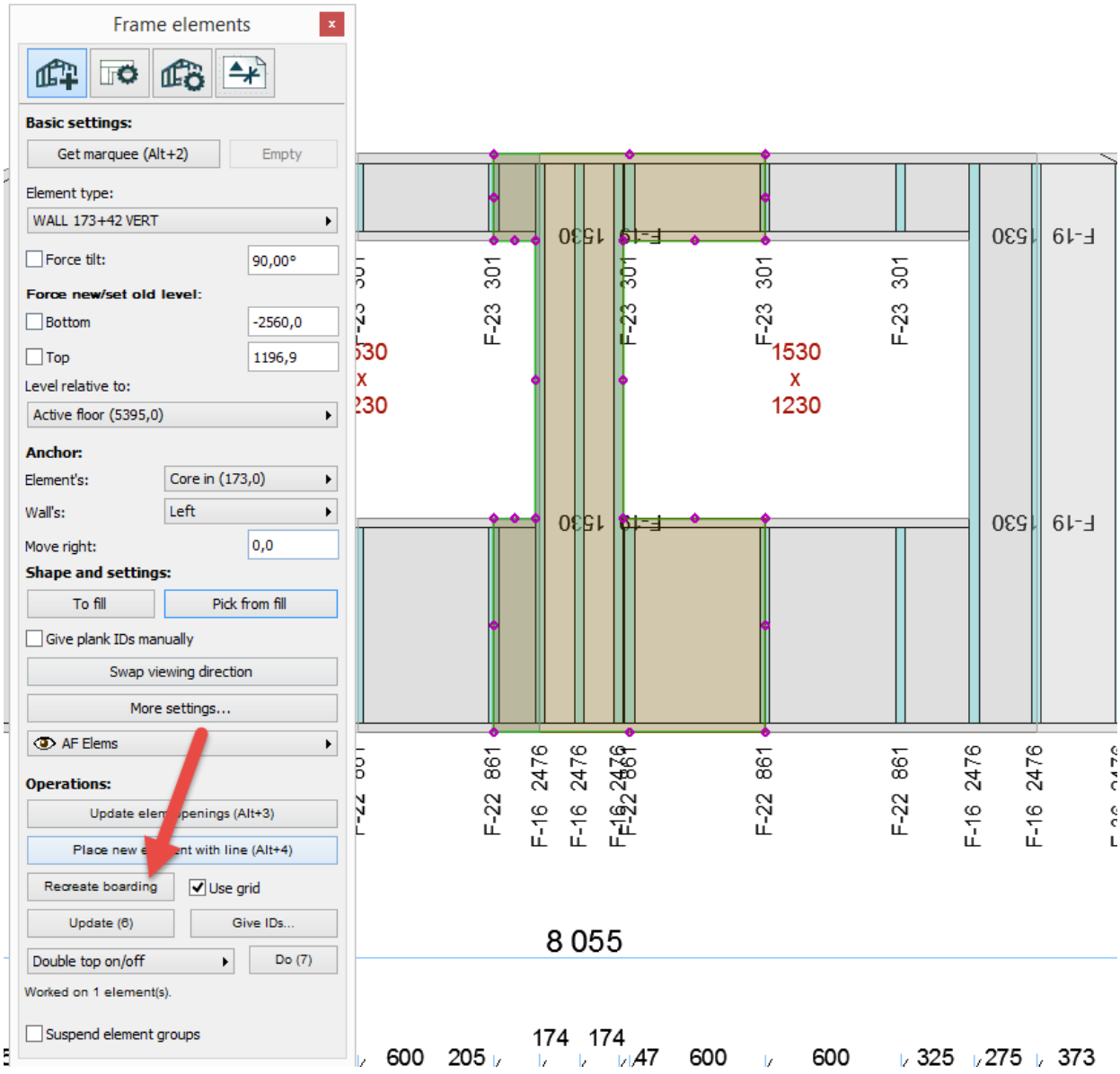
- *Create planks* deletes all existing planks except locked ones and recreates the framing taking the locked planks into account. The plank can be locked from element plank tools or from ArchiFramePlank object settings:



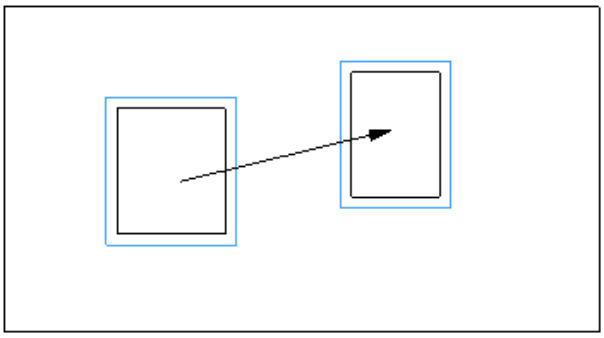
If there is a construction grid bigger or equal to stud spacing and *Use grid* is checked when creating the planks, ArchiFrame will place the studs to the grid if possible:



Just a single boarding layer is recreated by selecting a board and then clicking *Recreate boarding*. In the picture below the board shape has been edited, the board locked from object settings dialog and then the boarding is recreated to be compatible with the locked board:



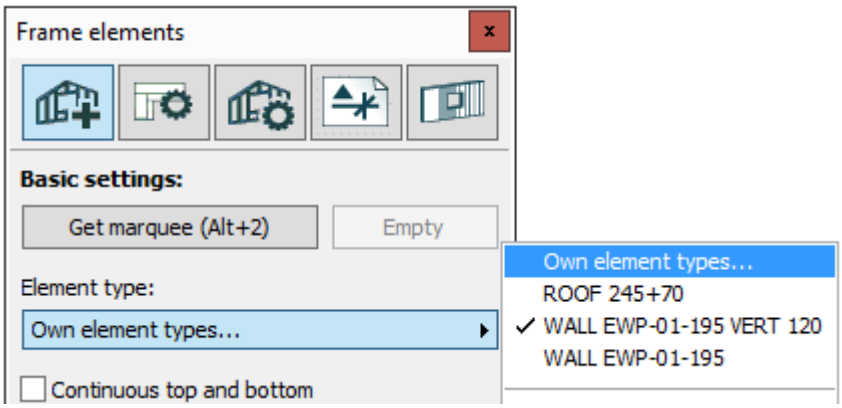
- *Crate planks* is changed to *Update elem openings* if there is an element or any piece related to an element selected. It will do nothing if there are no changes in the openings including any changes in force oversizes settings. In case of changes, ArchiFrame updates only the area of changed openings extended by two times the stud material width. The update area is marked with a blue line for the moved opening. Any plank having reference line intersecting with update area is recreated:



- *Update* updates elevation, its dimension lines and cut lists to ensure the projections match original planks. The 3D-model remains untouched except new IDs can be assigned.
- *Regenerate* forces ArchiFrame to re-create all projection items with current settings from xml-file. Planks and boards are kept in current positions. The 3D-model remains untouched except new IDs can be assigned.
- *Regenerate allowing reposition*, ArchiFrame resets the element elevation layout. The 3D-model remains untouched except new IDs can be assigned.
- *Check framing for opening dimensions*, if checked on, ArchiFrame will subtract framing polygons from ArchiFrameElement's polygons when creating opening dimension lines. Unchecked it will work like version 2017 and earlier and take openings directly from related ArchiFrameElement-object.
- *Update dimension lines* allows keeping the manual changes to the dimension lines when updating the elevations. If not checked, the dimension lines are kept as they are when updating. Regenerating or createing planks again will always update dimension lines.
- *Cleanup elevations* cleans up:
    - Overlapping plank IDs.
    - Board IDs overlapping with planks and other board IDs.
    - Cross dimension overlapping with any ID or planks.

- *Update opening text markings* defines whether the text related to openings are updated during the update. Setting it off allows manual editing of those texts.
- *Give IDs* opens renumber dialog. More information here.
- The last button is an operation that affects the whole element. For example, it can add a top beam either to inner/outer side of the element or create grooves to the bottom/top wood. To execute the operation, an element or any plank related to the element is selected.
- *Suspend element groups* affects multi-layer elements. Unchecked changes will affect every layer and when checked, changes apply only to selected element objects. The setting affects these operations:
    - Changing *Bottom* and *Top* levels will affect only the selected layers instead of every element layer. To limit, for example, inner boarding top level at gable walls it is useful to check *Suspend element groups* and edit *Top*-value.
    - *To fill* creates fills from single element only when checked and from every layer if unchecked.

11.1 Own element types/Custom element tools



The first item in the element type list opens *Custom Elements*-dialog. Below it ArchiFrame shows the 10 last used custom element types. To select one that is not in this most recently used list the *Custom Elements*-dialog must be opened – it will show all added types.

11.1.1 Custom Elements dialog

Define wood structures 1: https://vimeo.com/179027834
https://player.vimeo.com/video/179027834

Define wood structures 2: https://vimeo.com/173878793
https://player.vimeo.com/video/173878793

Define metal structures: https://vimeo.com/173751192
https://player.vimeo.com/video/173751192

This dialog is used to define a single or a multilayer element types. It resembles Archicad's composite structures dialog. Please note that also single layer types are defined here – those have just a single layer.

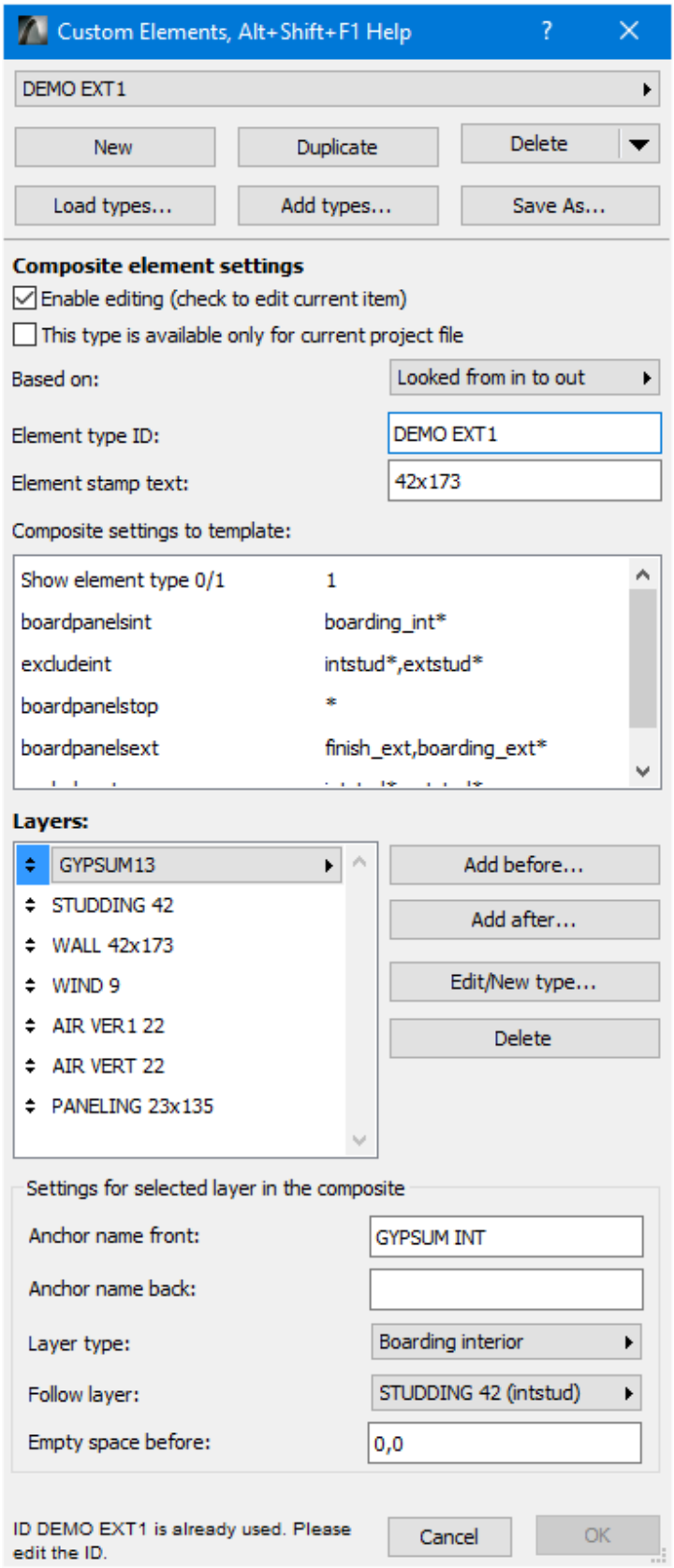
Editing the type here does not change all ArchiFrame element objects as changing Archicad composite structure does. This limitation is due to performance reasons. The changes are applied only for selected ArchiFrame element objects.

The main watching direction of the layers go from top to bottom. For example, if there is an element containing framing and single board layer it should be defined the board layer first and then framing. Then having just single projection from front will show the boards in front of the framing as it is assembled in the factory or at building site.

The global custom element types are saved into user specific folder into file ArchiFrameTemplates.xml (in Windows 10 the folder is C:\Users\[user_name]\AppData\Local). Current project file specific types are saved only into the pln-file.

Before editing custom element types, it is recommended to save a backup of all definitions using *Save As*-button. The backup file can be taken into use again by clicking *Load types*-button. Canceling the dialog reverts all changes.

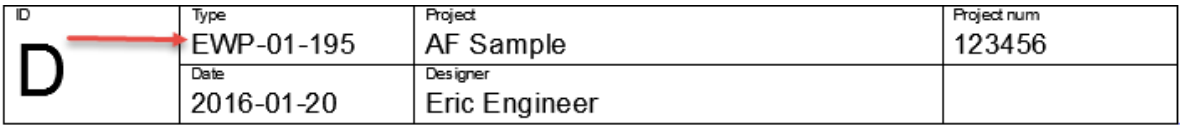
The template source file from the *data*-folder's file ArchiFrameElements.xml contains *replace*-tags in syntax [setting_id; prompt=Shown to the user; default=default value; type=text/length/real/matid/angle/layer/panelid]. Type angle is a special case: It is saved in the template and perhaps applied to the objects as radians but shown to user as degrees. All unknown types are treated as text.



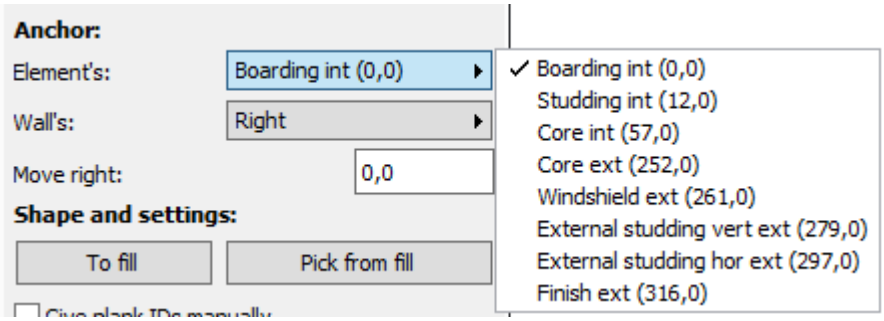
Parts of the dialog from top to bottom:

- *Element type* dropdown list, this list contains all the defined types. When closing the dialog the type here is set to any selected element object.
- *New* creates new composite structure without any layers.
- *Duplicate* duplicates current composite structure for further editing.

- *Delete* deletes current composite structure. Be very careful not to delete any structure that is in use. The custom element types are shared among all projects as default. So delete a type only if you are absolutely sure that it is not needed any more.
- *Load types* deletes all the custom composite types and single layer types and loads everything from an external file. This is very useful for example if there is a single person in the company taking care of the structures and shares the types to other users.
- *Add types* is as *Load types* except it does not delete current types. It merges the types from the file using any type having the same id from the given file.
- *Save as* saves all custom types to an xml-file to be used for *Load types* or *Add types*. The file can be edited using text editor for example to limit types to import to another user.
- *Enable editing* must be checked on before making any changes. Its purpose is only to avoid editing existing types accidentaly.
- Check box *This type is available only for current project file* defines that the type is available only in current Archicad project file (pln-file).
- *Based on* defines the template to use for the composite structure. The templates are defined in the data folder's file ArchiFrameElements.xml and they define mostly the projections in composite structures. Layer templates define how the framing is done. Your supplier can help you with the element templates if new ones are needed or some custom framing is needed.
- *Element type ID* is the unique ID for the composite custom element type. The ID should never be changed if the element type is in use. It is advisable to design the ID types carefully before actually adding elements. For example, it could be WALL layer1+layer2+layerN (WALL GYP13+45+195+WIND9) or shorter form.
- *Element stamp text* is shown in element elevations here:

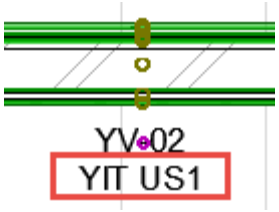
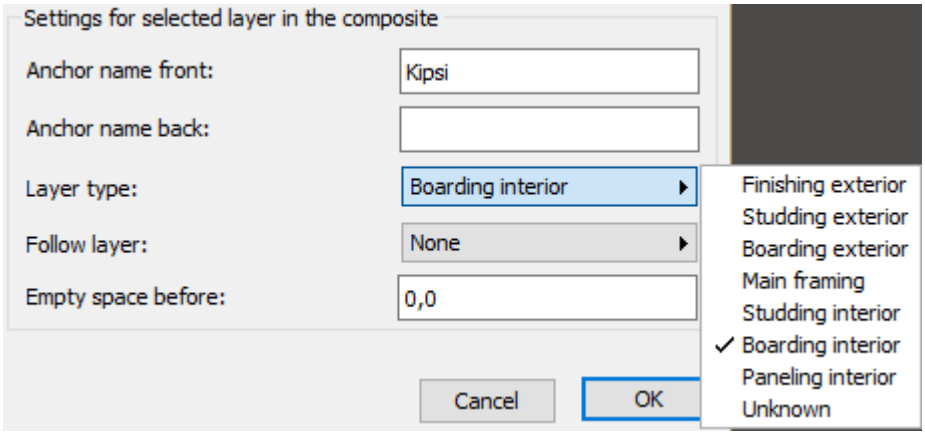


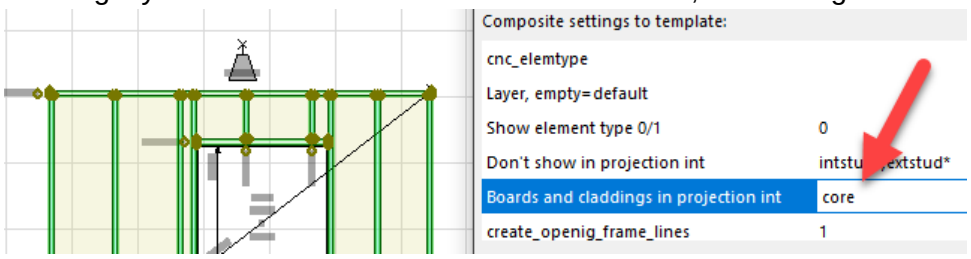
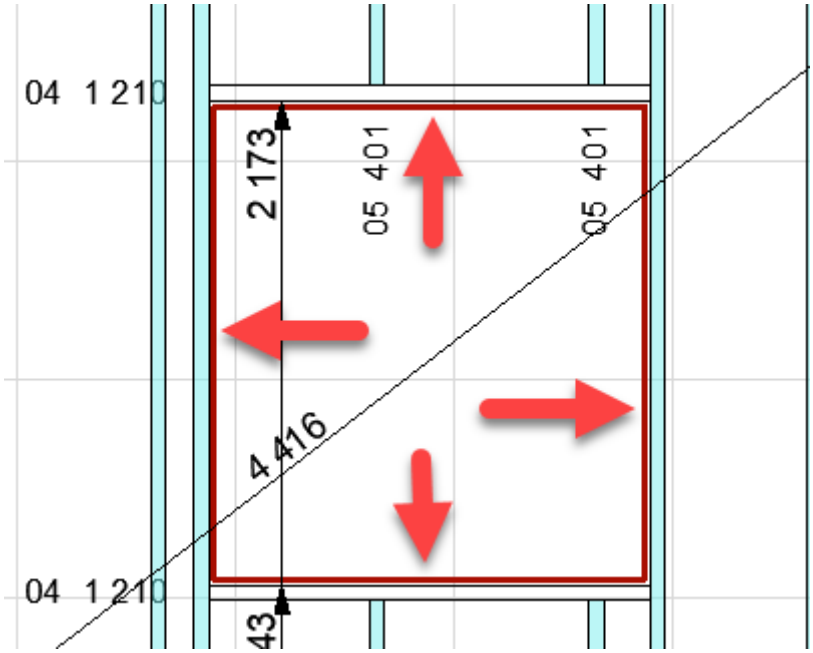
- *Composite settings to template* lists all editable parts of the template file. For composites, it contains mainly settings of which layers are visible in the different projections. By default all framing layers are visible and board/panel layers are not.
- *Layer list* shows the layers making up the composite structures. The order of the layers can be edited by dragging the item with mouse from the list's left hand side dragging icons. The best way to make a composite structure is to first add all the layers and after that set the dialog's bottom part settings per layer.
- *Add before* adds new layer before the selected one. *Add after* adds after the selected one.
- *Edit/New type* opens the single layer editing dialog. If there is no correct layer type defined already it can be added with this button.
- *Delete* deletes the selected layer from the composite structure.
- *Anchor name front* and *back* tells the anchor points to show here. The watching direction goes from the top of the layer list towards the bottom and *Based on* template defines if the watching direction is from inside or outside:

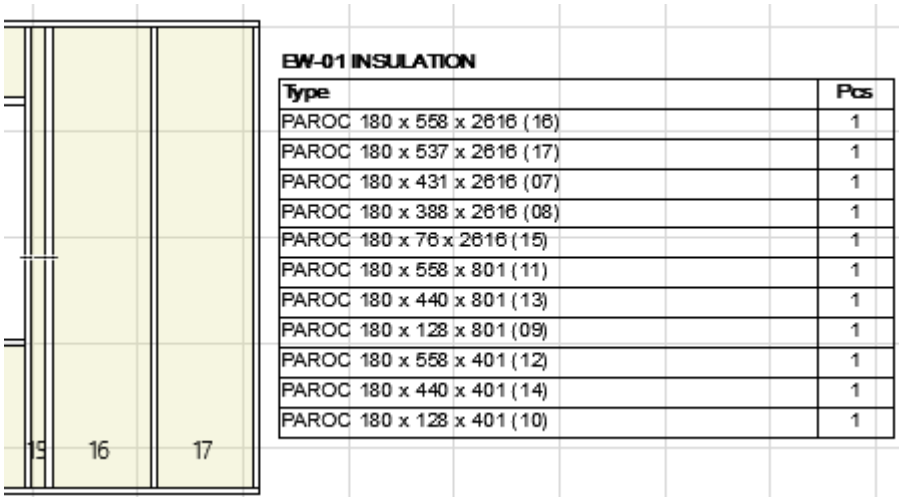


- *Layer type* tells ArchiFrame the usage of the layer. There must be exactly one *Main framing* layer. For projections, usually all layers before *Main framing* are defined as interiror types and after main framing as exterior. This rule is usually more important than interior/exterior classification.
- *Follow layer* is used for example for inner side additional studding. It should usually follow the main framing layer. Also for the boards this must be set to get the board edges on a stud.
- *Empty space before* can be used if some layers are omitted from the ArchiFrame composite structure (for example asphaltic felt in roof structures could be 3 mm thick).

11.1.2 Settings to element template for composites

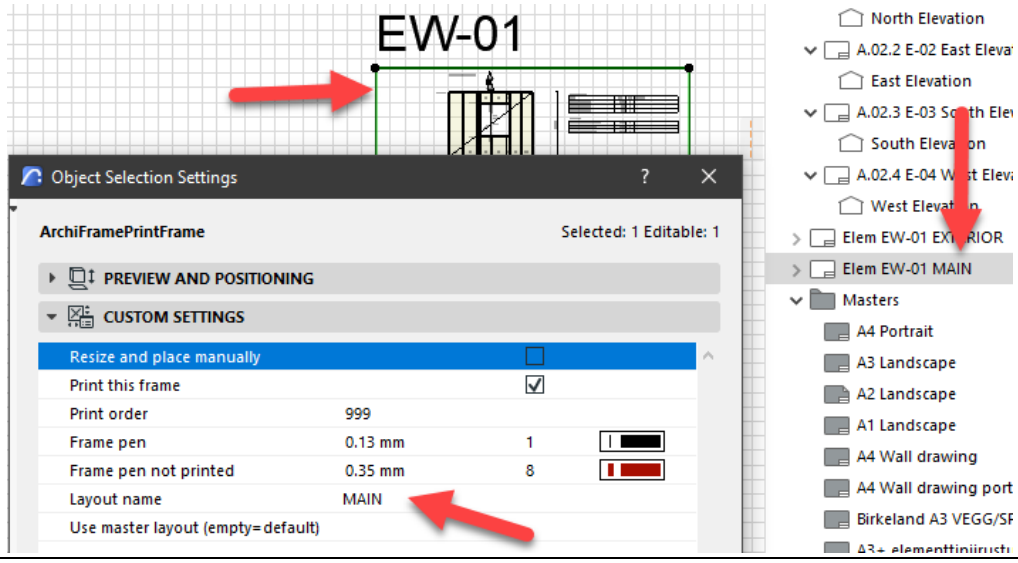
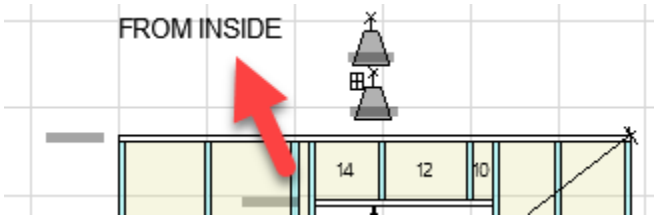
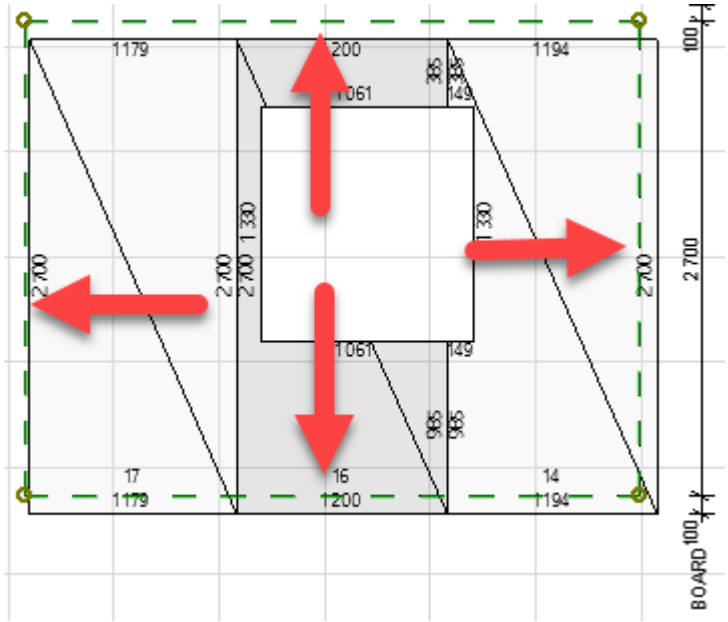
Name	Description
Show element type	With value 1 the element type (YIT US1) is shown in floor plan and 3D in addition with the element ID (YV-02): 
Layer	Defines the Archicad layer for the ArchiFrameElement-objects – not planks & boards.
Boards and claddings in projection ext/int and Don't show in projection int/ext	Layer type names to be shown or excluded in the related projections. Names refer to the layer type defined here:  The internal names are: • Finishing exterior, finish_ext.

	• Studding exterior, extstud. • Boarding exterior, boarding_ext. • Main framing, core. • Studding interior, intstud. • Boarding interior, boarding_int. • Paneling interior, finish_int. If there are many layers of the same type, there will be a sequence number added to the end of the name. For historical reasons, the numbering starts from 2. For example: intstud2, intstud3. Wildcard * can be used here (intstud*). Insulation is modelled with ArchiFrameBoardPanel-objects and therefore to get them visible in the elevation, the insulation layer name must be included in the boarding layers to show. To show insulation with core, this setting is needed:  By default, all planks are shown, and all boards are hidden. In other words, for planks typically don't show definition is used and for boards show definition instead.
Create opening frame lines	With value 1 ArchiFrame will show door&window frame edges as lines in the element projections:  For non-square shapes the window shape must be taught to the weather boards module file ArchiFrameCoveringBoards.xml.
Show center of gravity	Possible values are: • 0, do not calculate. • 1, show the real position of the center of gravity. • 2, show above the elevation.

	• 3, show above the elevation and show separate value calculated. Without doors&windows if different.
Weight text, unit, and layer	Text to set into the weight marker, (weight) is replaced with the calculated weight:  Weight unit can be kg or lb and ArchiFrame will show the value using given unit. ArchiFrame will use default plank layer for weight marking unless an Archicad layer is selected from the list.
Insulation create cutlist	With value 1 ArchiFrame will place an insulation cut list to the element elevation:  **EW-01 INSULATION** Type / Pos PAROC 180 x 558 x 2616 (16) / 1 PAROC 180 x 537 x 2616 (17) / 1 PAROC 180 x 431 x 2616 (07) / 1 PAROC 180 x 388 x 2616 (08) / 1 PAROC 180 x 76 x 2616 (15) / 1 PAROC 180 x 558 x 801 (11) / 1 PAROC 180 x 440 x 801 (13) / 1 PAROC 180 x 128 x 801 (09) / 1 PAROC 180 x 558 x 401 (12) / 1 PAROC 180 x 440 x 401 (14) / 1 PAROC 180 x 128 x 401 (10) / 1

cnc_elemtype	Wup cnc-output: Sets the element type line in wup-file:
	`VERSION 3.3;`
	`ANR 123456;`
	`ELB EXTERIOR;`
	Default is EXTERIOR if looking from outside and INTERIOR if looking from inside.

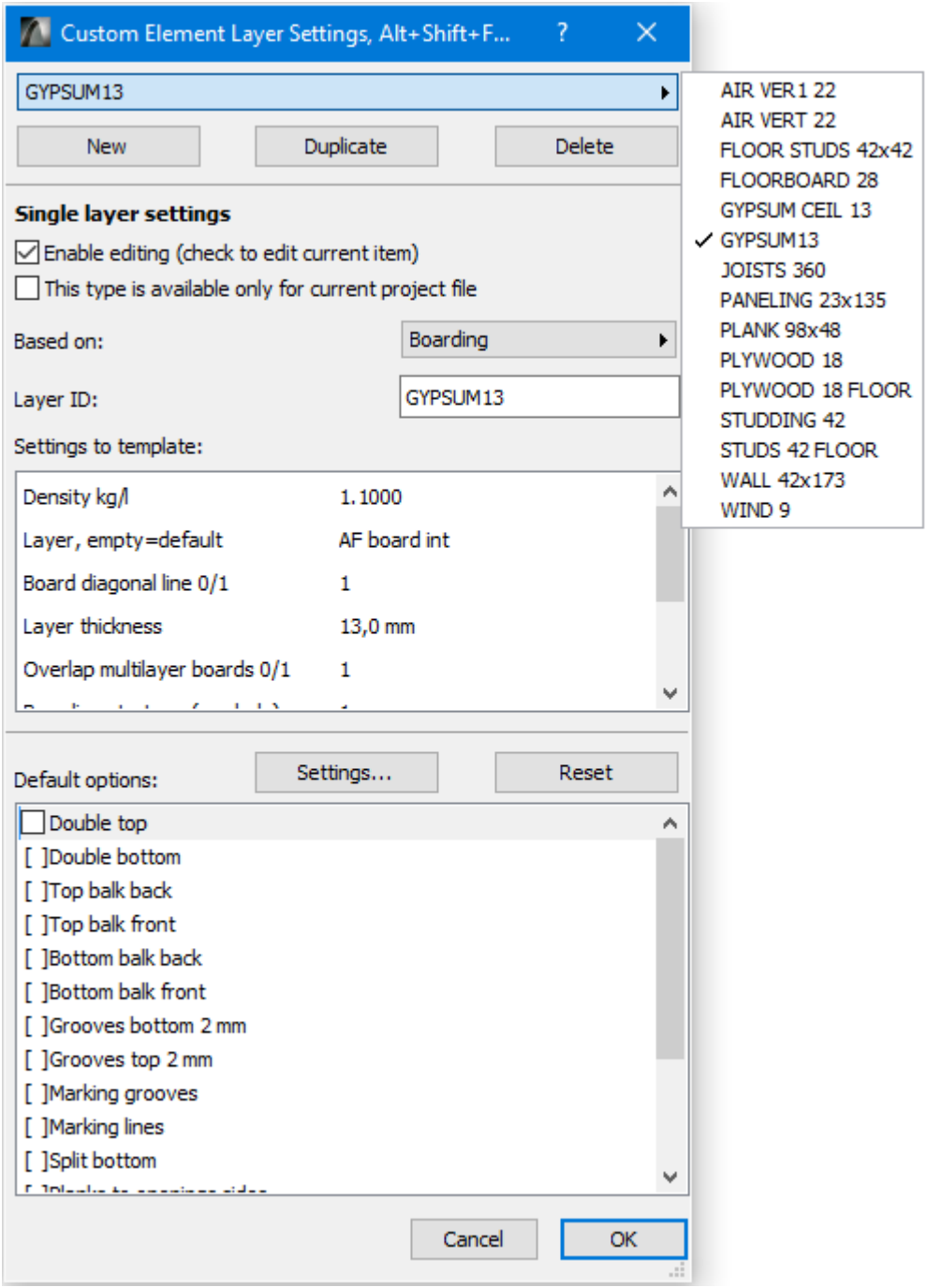
cnc_wupsettings	If given, adjusts the settings in wup cnc-writing dialog. Has key=value pairs separated with comma (for example: inout=0,rot=180,dirconv=1). Possible values are:
	• inout=0/1
	• writelayers=core/coreint/coreext/all
	• writepla2:
	  ○ 0 no
	  ○ 1 write pla2 having all markings
	  ○ 2 write new plaX just before the related layer
	• rot=0/90/-90/180
	• mirrory=0/1/2 (2=swap y and mirror watching direction)
	• dirconv=0/1 (Setting *Plank directions etc* from wup settings dialog)
	• ela=1/2/3 1=automatic/2=INTERIOR/3=EXTERIOR (Setting *Value of ELA* from wup settings dialog)

Layout name	Sets the related ArchiFramePrintFrame-object's name that will be taken into the *Layout book*: 
Projection name	Sets the static text title related to a projection: 
Framing as lines	When used, typically the main framing is shown with lines instead of projecting every related layer's plank into a projection. It is possible to set the pen, line type and layer for the lines: 

11.1.3 Custom Element Layer Settings dialog

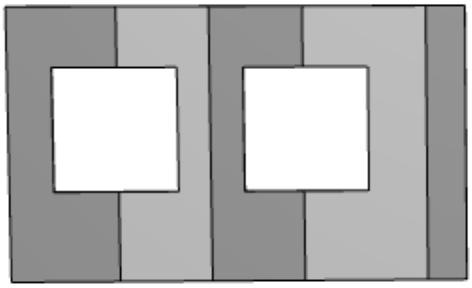
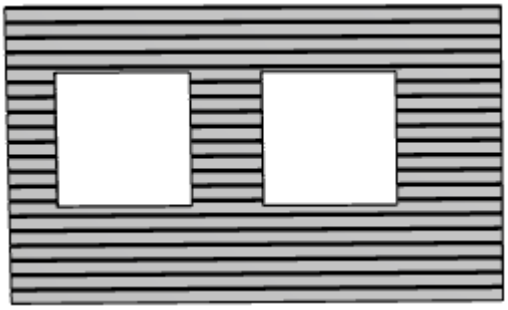
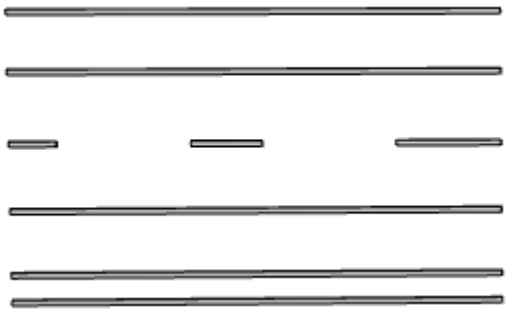
This dialog is used to define framing parameters or board types for a single layer. It resembles Archicad's building materials dialog.

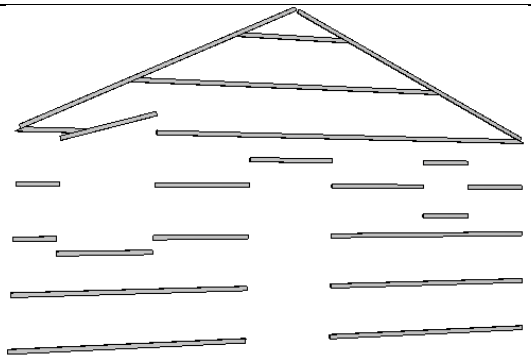
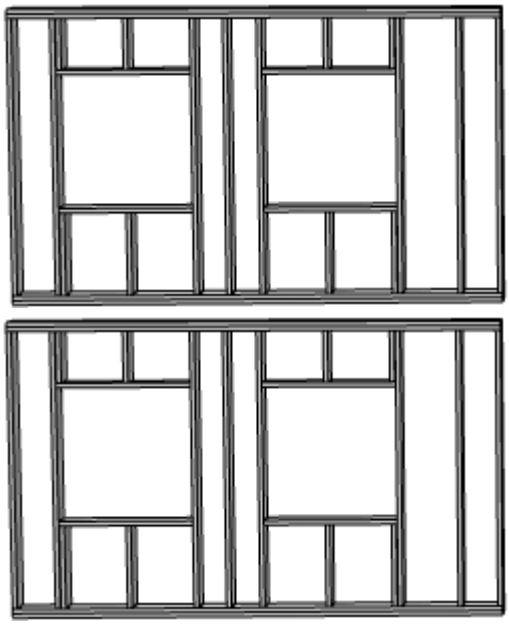
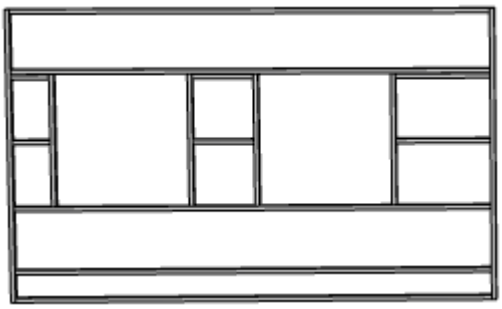
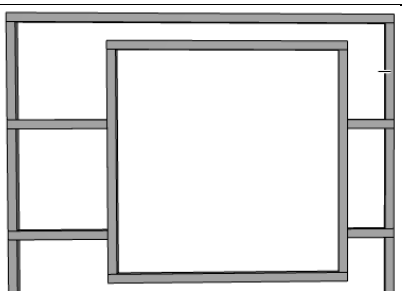
The panel profiles must be defined into *data*-folder file ArchiFrameElements.xml and are referred from a custom layer by its ID.

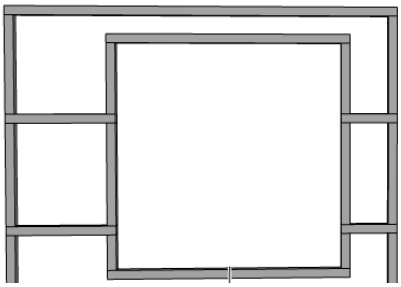
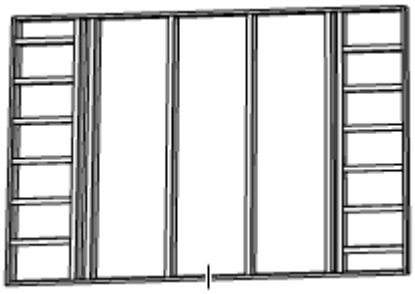
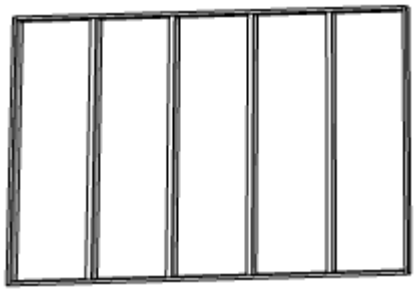


- At the top of the dialog there is the custom layer list like in the composite dialog.
- *New* creates new empty layer definition.
- *Duplicate* duplicates existing one for further editing.
- *Delete* deletes the current layer type.
- *Enable editing* must be checked on before making any changes. Its purpose is only to avoid editing existing types accidentaly.

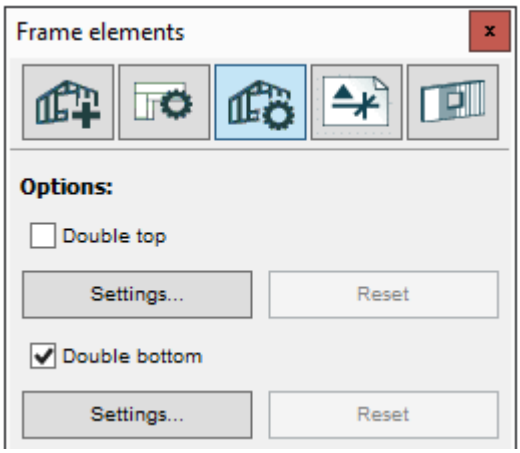
- Check box *This type is available only for current project file* defines that the type is available only in current Archicad project file (pln-file).
- *Based on* defines the template to use for the layer structure. The templates are defined in the data folder's file ArchiFrameElements.xml and for the layers they define the framing rules. Your supplier can help you with the element templates if new ones are needed or some custom framing is needed. The default types are:

Boarding	 See boarding strategies.
Paneling, paneling itself must be defined in ArchiFrameElements.xml manually.	
Air space strips vertical	
Air space strips horizontal	

Air space strips horizontal with angled parts	 Useful settings: • opening_square • Planks to opening sides=2 • skipcorners=1
Framing	 Please see Main framing rule/attribute framingrule.
Framing horizontal	
Framing horizontal long	

Framing horizontal long opening sides cut	
Floor/roof with sides	
Floor/roof no sides	

- *Layer ID* is the unique ID for the composite custom layer type. If the ID is edited and other composites refer to that layer, ArchiFrame will update those references to the new edited ID. Anyway, it is advisable to design the ID types carefully before actually adding elements. For example, it could be WALL 45x195 telling the intended use and the main material for the framing.
- *Weight* opens Weight calculation settings.
- *Settings to template* lists all the editable parts of the template file. The settings include the layer thickness, material id for the framing, stud spacing and more. If the value begins with character @, it will point to another key in the template. For example, if there is common default material with key matid it can be pointed with value @matid. See Settings to element template.
- *Default options* is a list of all available options from the Element tools/Element options:

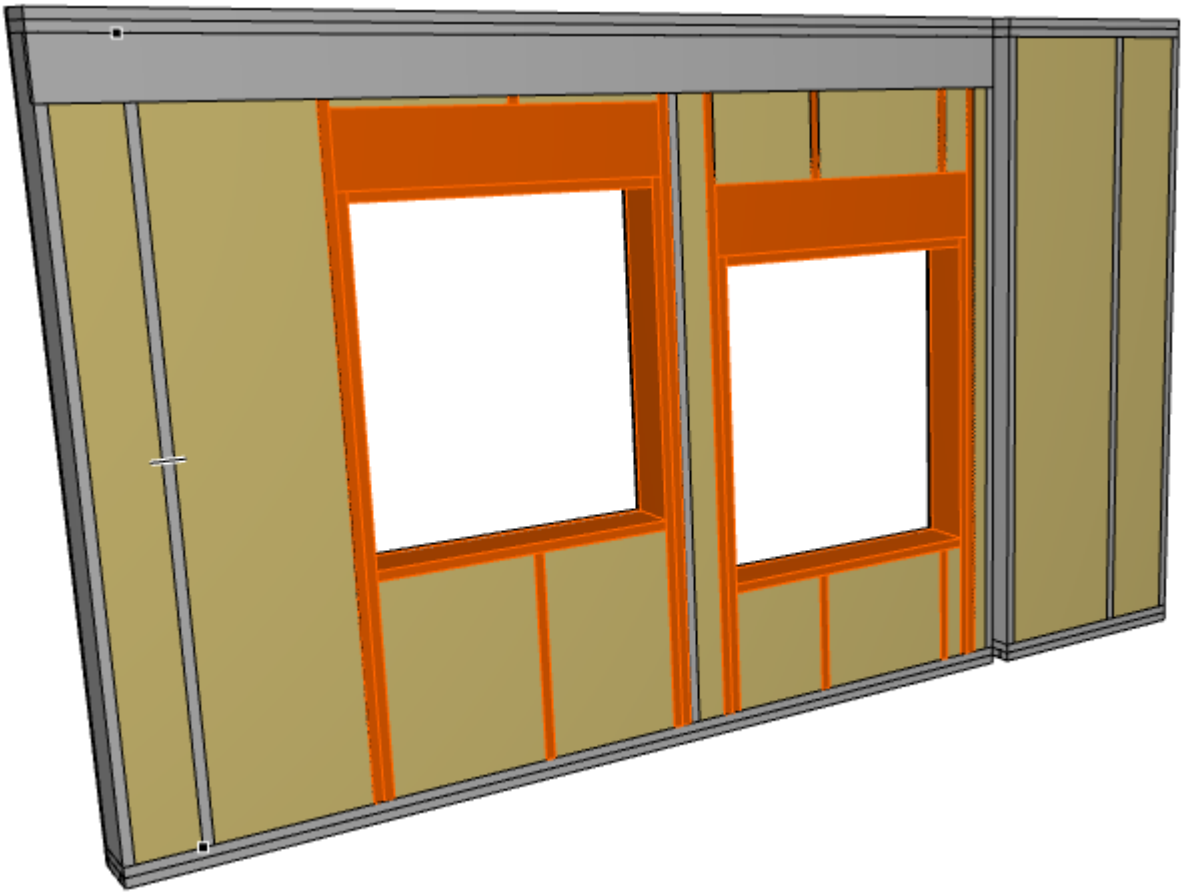


For example it may be useful to have double bottom as default on for the main framing but not for additional studding.

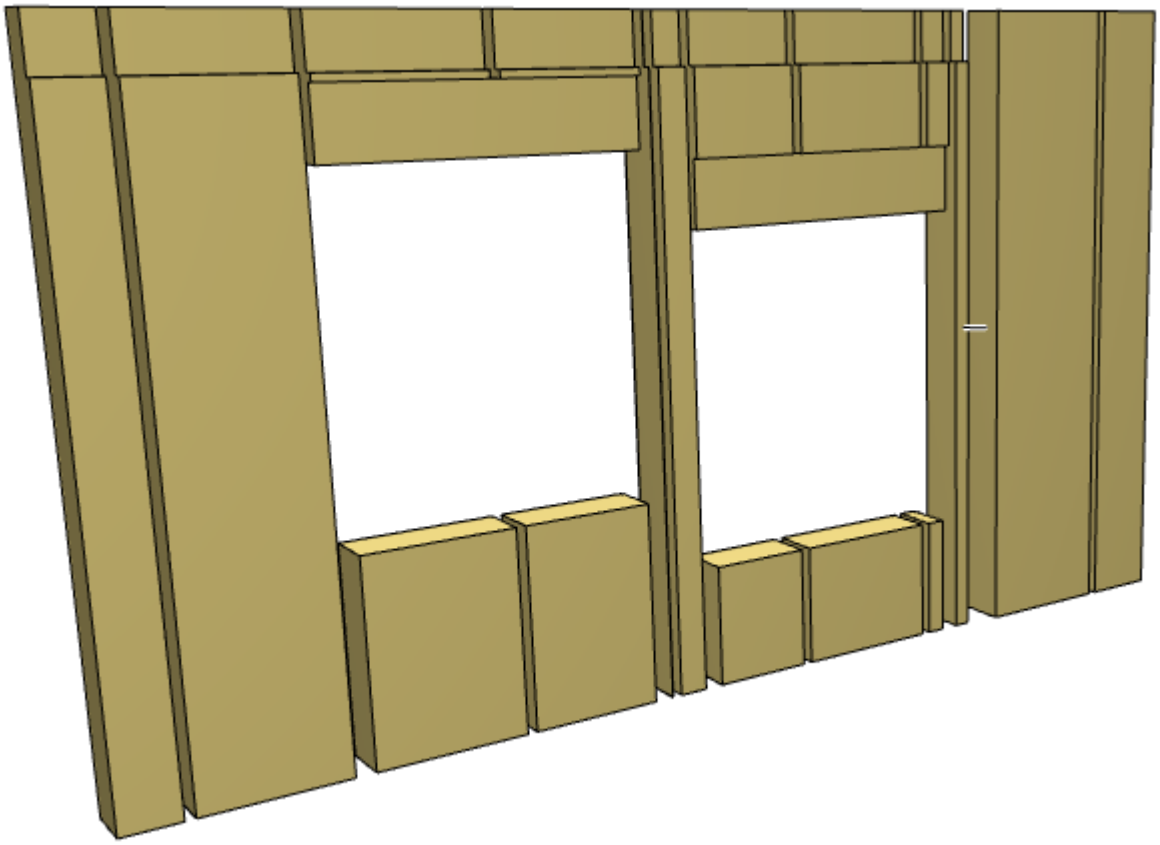
- *Settings* opens selected option's settings if available.
- *Reset* clears all given settings and uses option with default settings.

11.1.3.1 Insulation settings

Insulation is modelled with ArchiFrameBoardPanel-object like below:



And without framing:

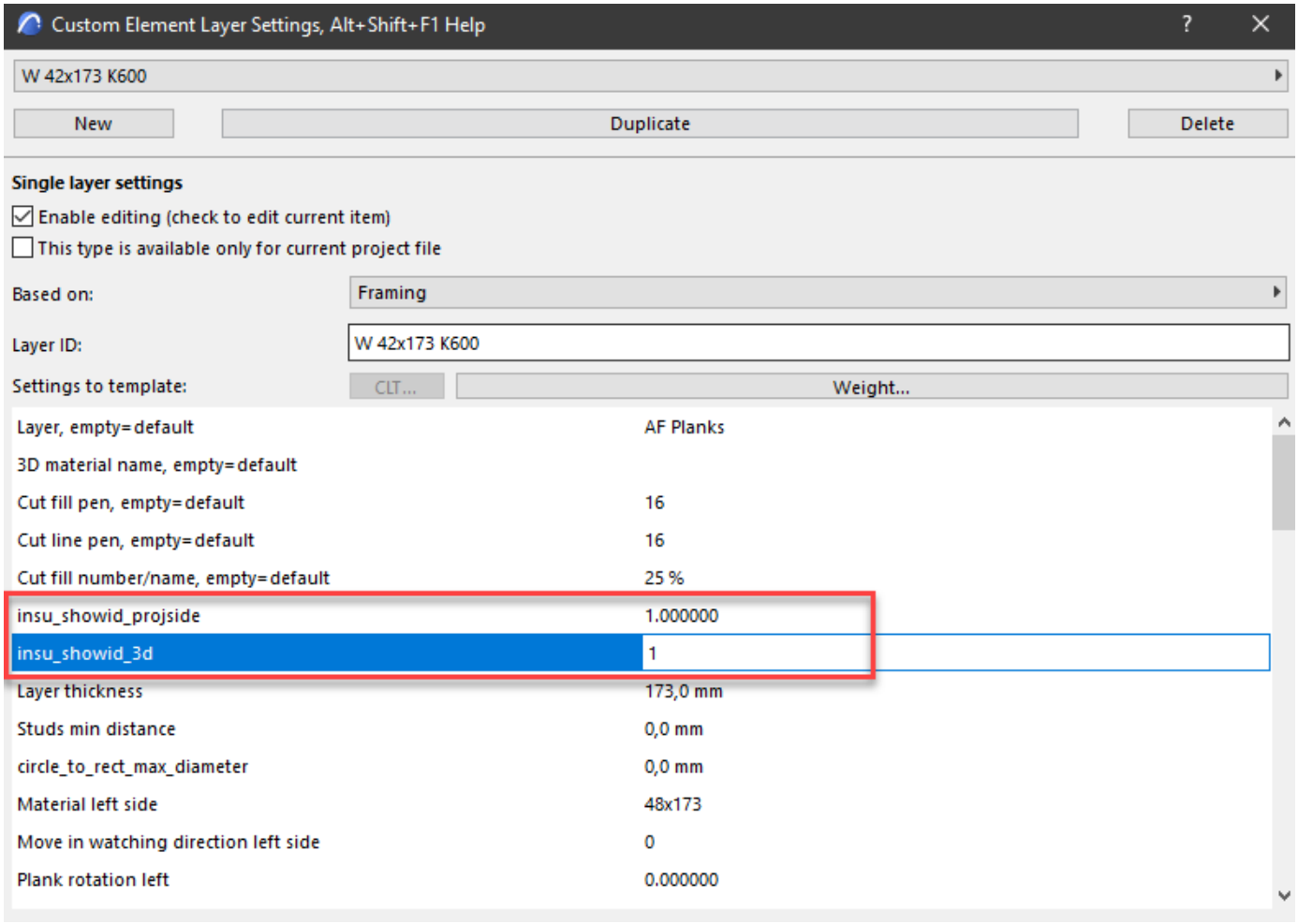


ArchiFrame automatically cuts away the framing related to the same composite layer. Insulation settings are:

- *insu_create*, with value 1 the insulation is created. If not, weight and amount of insulation is calculated the old way using the layer weight settings.
- *insu_thickness*, value 0 uses the layer thickness. This value can be set to produce thinner insulation.
- *insu_zoff*, distance to begin of the insulation in the element's watching direction from the front surface.
- *insu_xxxoff*, how much to cut off from the related side of the insulation. Recommended values are the thicknesses of related edge parts. If using double top/bottom, it is good to set the total value here. ArchiFrame will anyway cut away the framing but setting these gives the insulation edge hotspots to better places.
- *insu_layer*, the Archicad layer to use. If the layer does not exist, ArchiFrame will create it.
- *insu_fill_2d*, the fill to use in floor plan.
- *insu_matid*, the insulation ID in listings.
- *insu_weight_kg_per_m3*, typical value is between 20-60 kg per cubic meter.
- *insu_rgb_r/g/b*, insulation is modelled in 3D using this colour instead of any Archicad surface type.
- *insu_explode*, set to value 1 to explode insulation to separate pieces.
- *insu_minsize*, defines minimum width of an insulation piece. Smaller pieces are automatically removed.

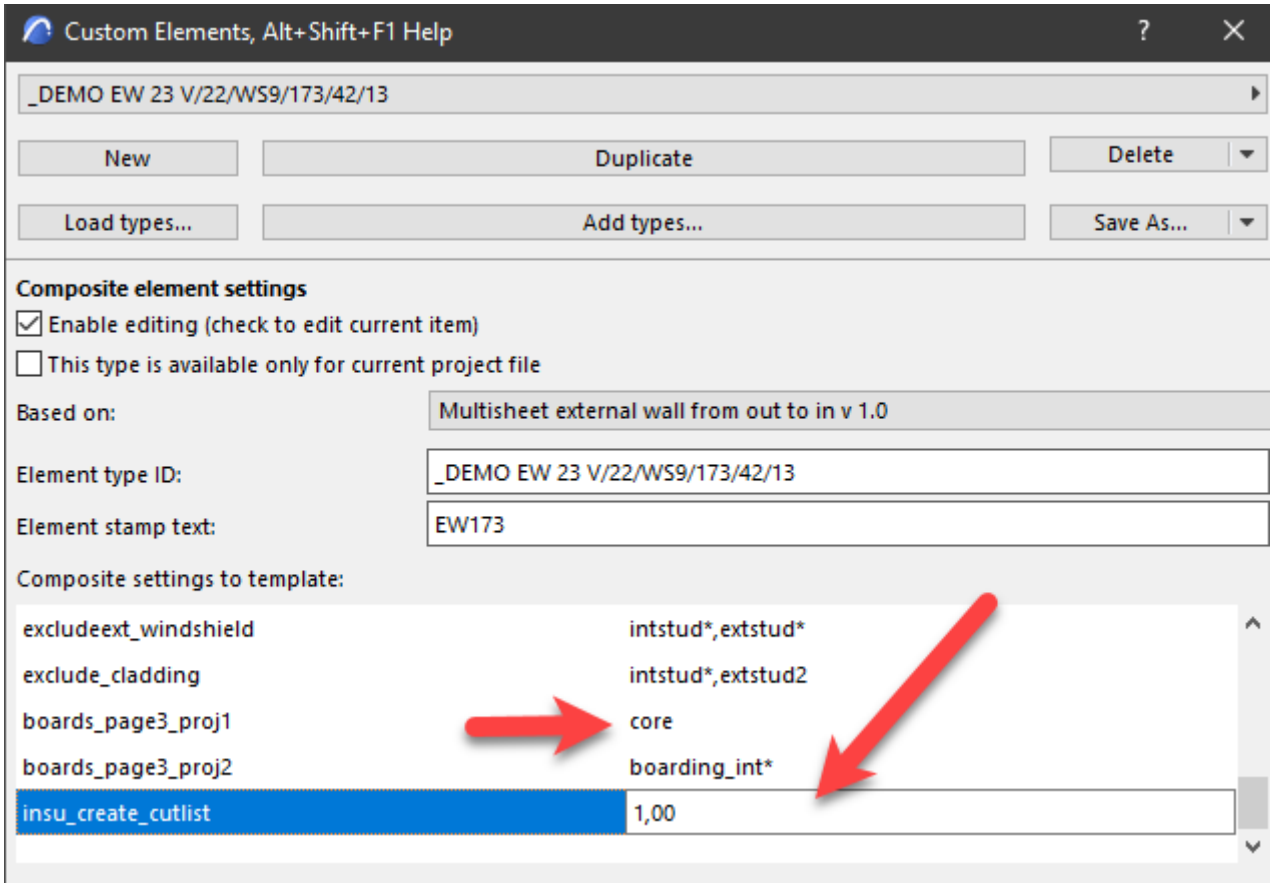
A video covering insulation settings here.

The visibility of insulation piece IDs is defined in the framing layer with settings *insu_showid_projside* and *insu_showid_3d*:



Also, to get the insulation and the related visible, the settings showing the insulation boards in core (or other) layer must be given in the composite settings. The actual setting name varies by *Based*

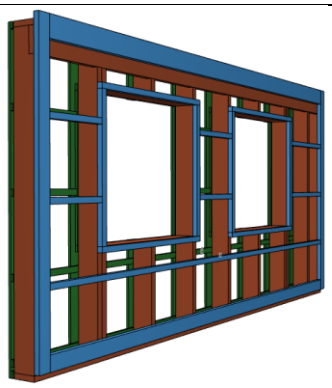
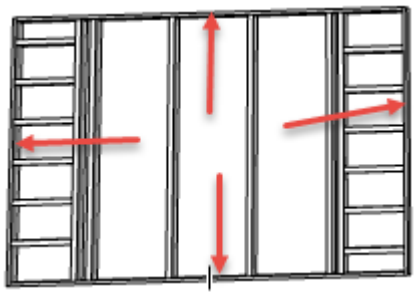
on-type:

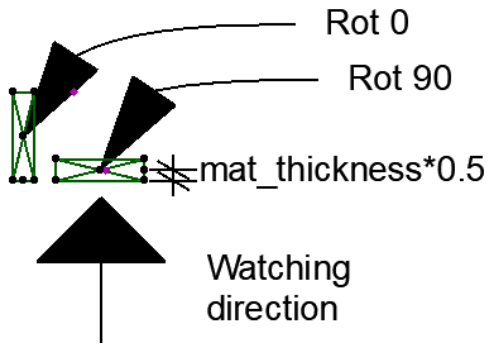
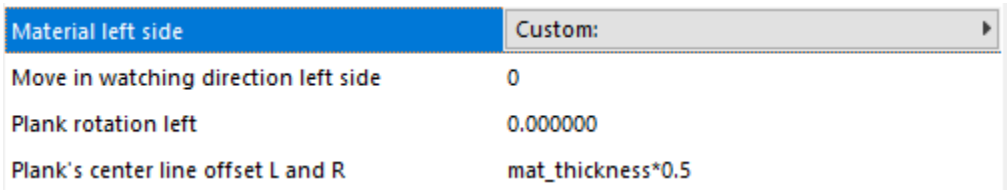


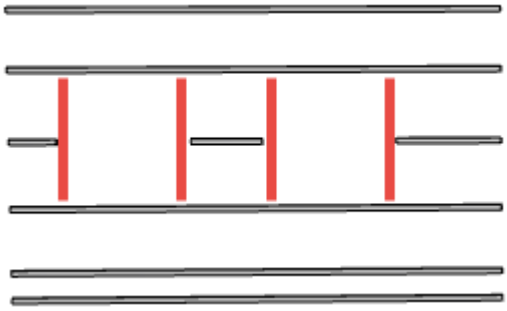
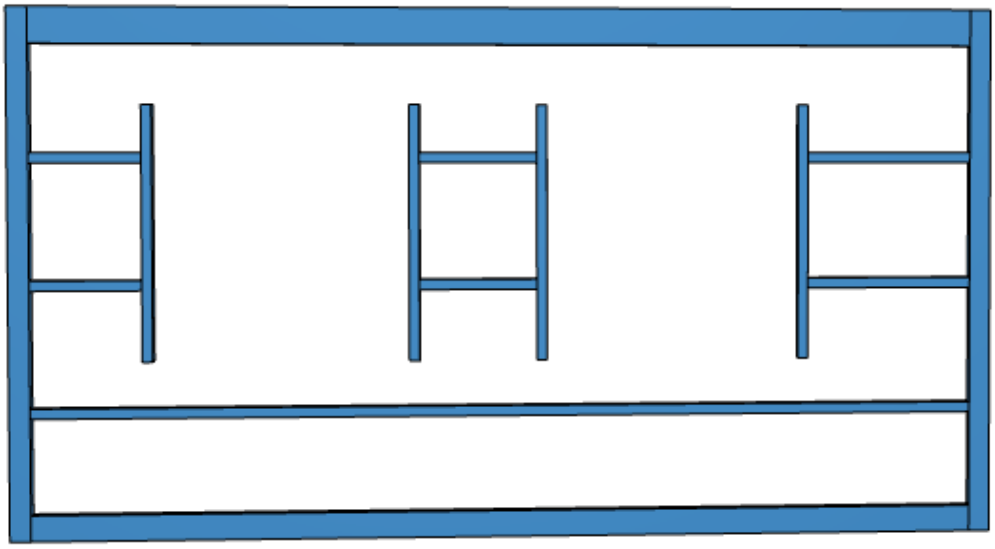
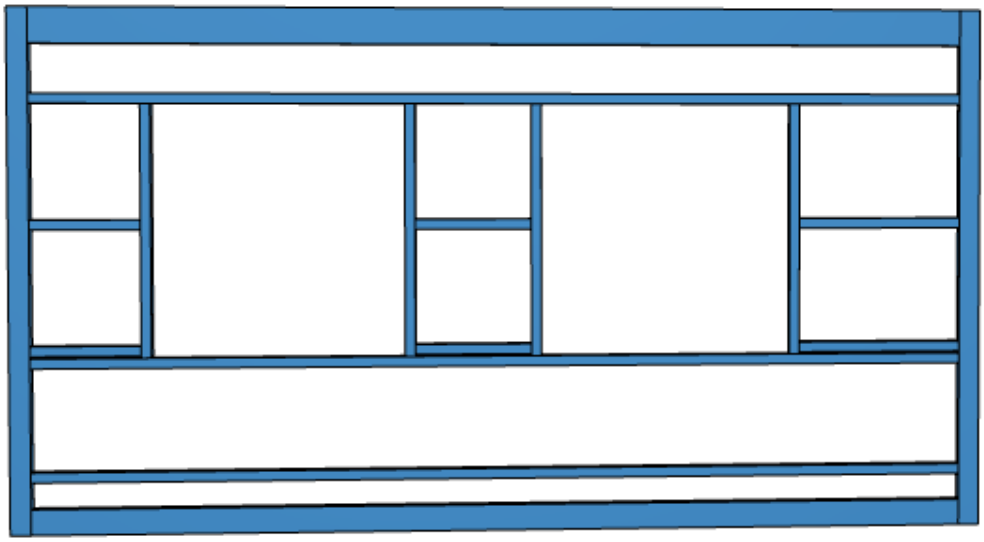
11.1.4 Settings to element template for layers

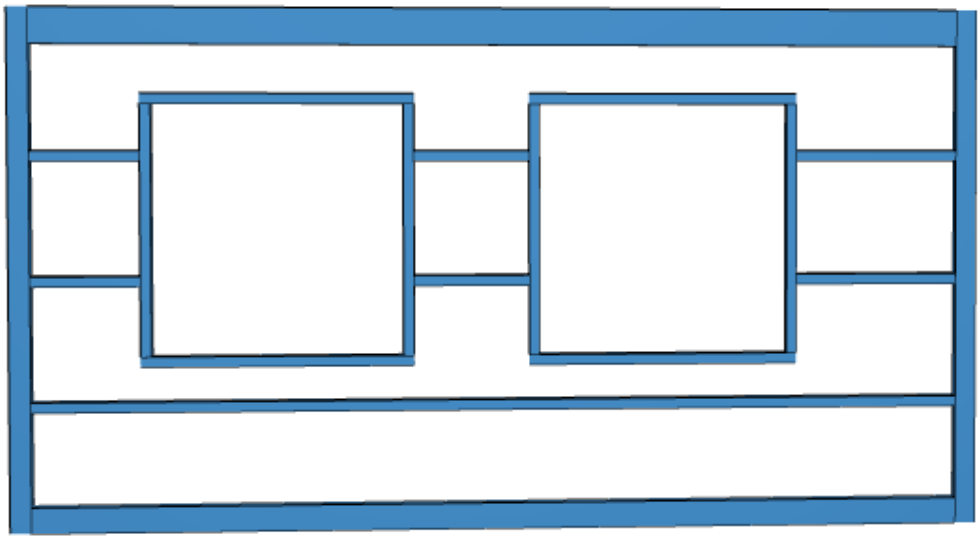
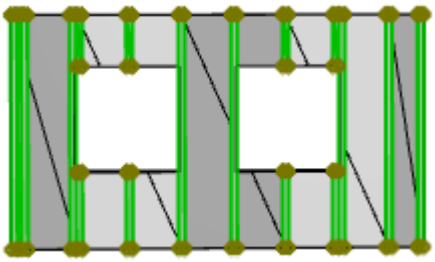
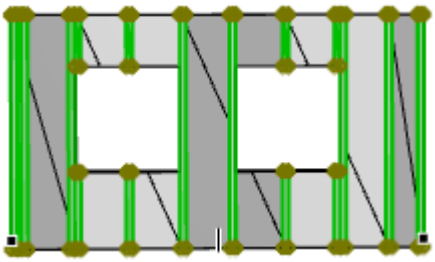
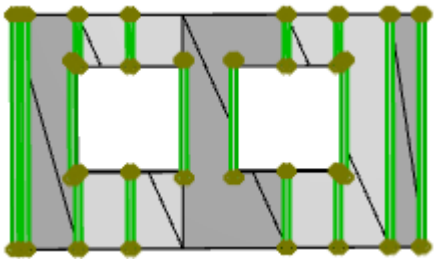
There may be any settings in the element template. Following table lists built-in tags:

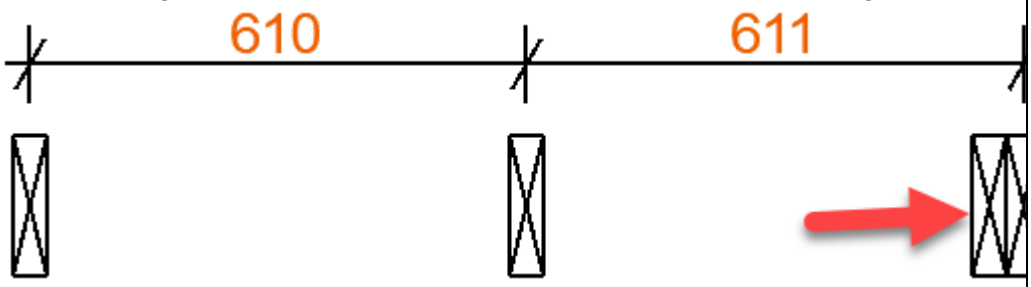
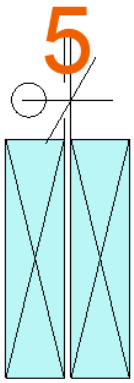
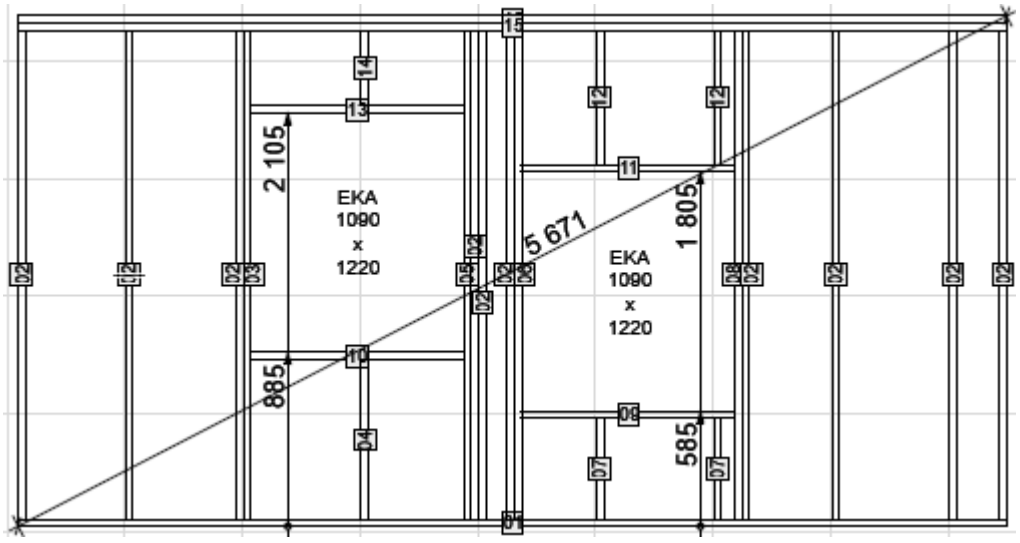
Name	Description
Layer	The Archicad layer where the target pieces are placed. You can choose a pre-existing layer from the drop-down menu, or else type in a new layer name. Typing the layer name creates a new Archicad layer.
3D material name	Allows setting different 3D material for each layer. The material must be written exactly as it is defined (faster) so be sure to copy&paste material names to text editor before opening own elements dialog. Wildcards * (any string) and ? (any character) may be used but then ArchiFrame will scan attributes until match is found. For example, having different colours for different layers – single layer's settings:

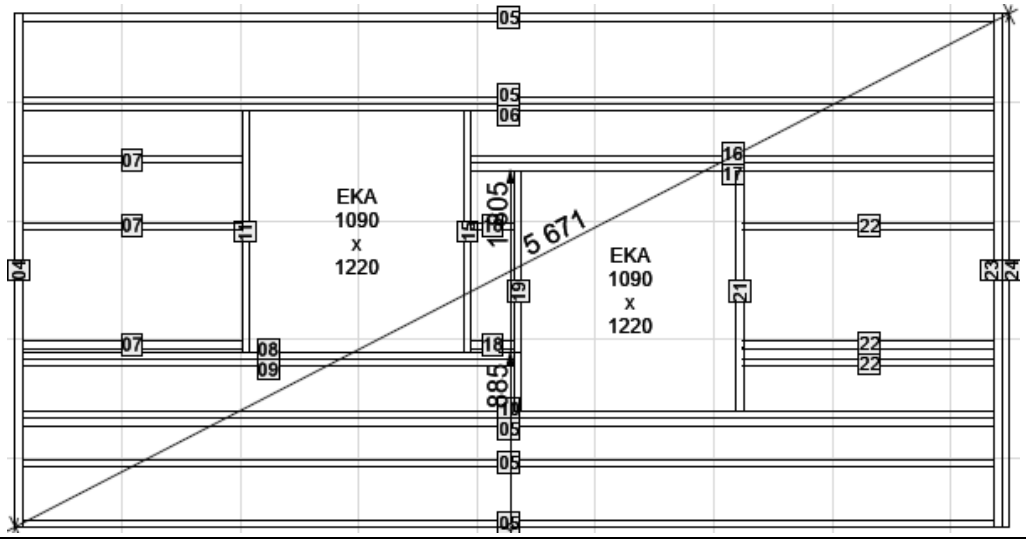
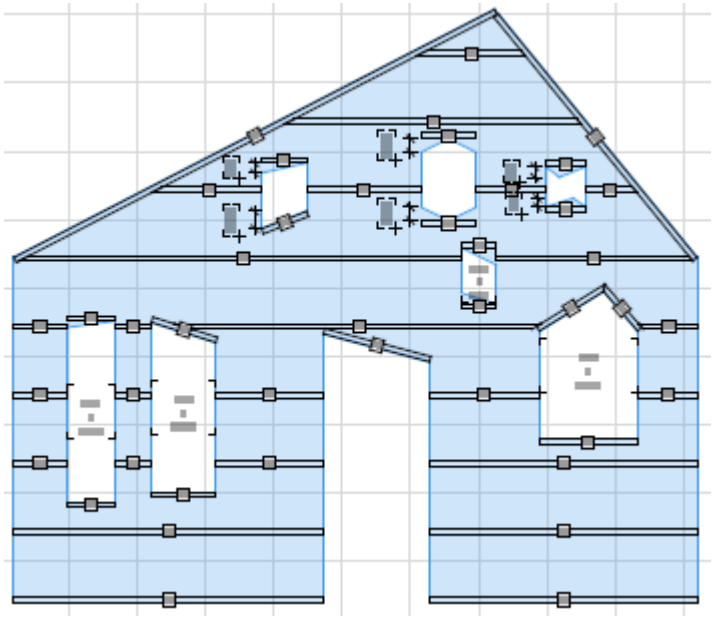
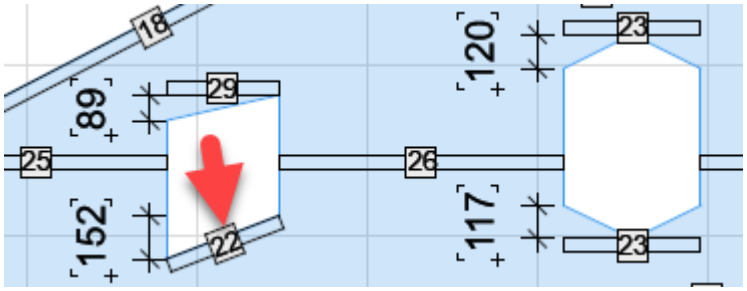
	 Settings to template: Weight settings... Density kg/l 0.0500 Layer, empty=default af planks 3D material name, empty=defa... Brick-red Cut fill pen, empty=default 3 Cut line type, empty=default Cut fill number/name, empty=... 50 % Layer thickness 198,0 mm
Cut fill pen, line type and fill type	*Cut fill pen* sets the pen used in sections for cut planks, must be given as number or empty=use default in this order (applies to all these cut attributes): • What was set into object tool default settings defined when planks were created. • Value set in ArchiFrameBlocks.xml. • (Value given here). *Cut line type* and *Cut fill* are also used in sections. These attributes can be entered as a number or text containing wild cards (*). For example: 50 %
Layer thickness	Layer thickness. Usually equals to material depth in element's watching direction but in special cases may be different.
Board type ID	Board type ID for listings.
Board width and height	Affects creating boards.
Create border	0/1, Used in floor structures: 
Density	Layer density for element listings.
Material ID, top/bottom / left/right	*Material ID* is the default material to use. The material must exist in the *data*-folder's ArchiFrameBlocks.xml file or rather in user specific folder's ArchiFrameBlocksChanges.xml.

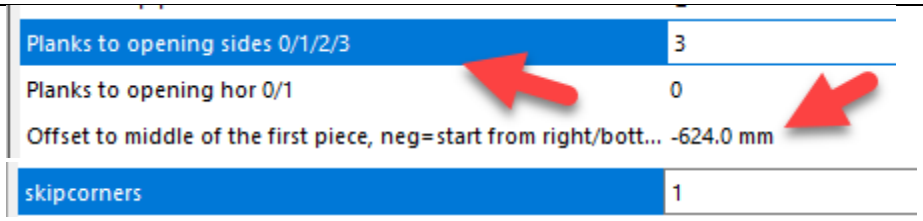
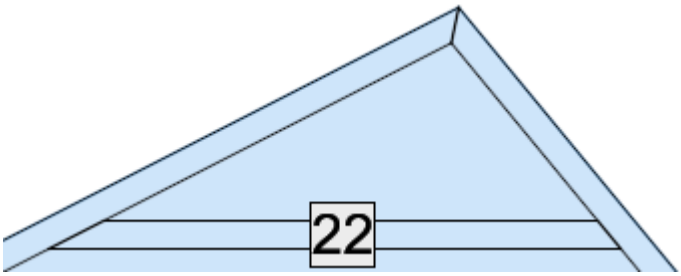
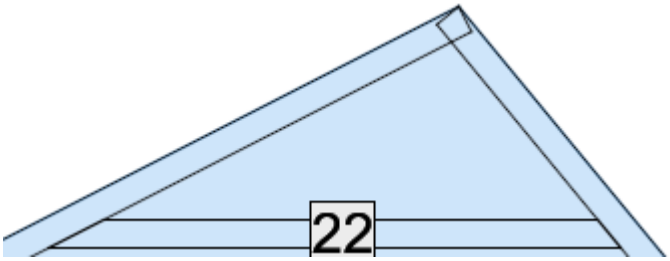
	Used in framing to define bottom or top plate material if different from the default. The default material can be referred by entering manually text @matid. This is the default value.
Rotation and z-offset	Rotation angle is given in degrees. ArchiFrame sets the rotation so that the plank extends towards the watching direction. Rotation angle value is added to that. For example, plank with size 50 x 100 mm is placed with default settings to the layer's front surface and the plank extends 100 mm in the watching direction. Z-offset is a text field that must be given in meters using dot as decimal separator (0.010). Common value with rotangle 90 is mat_thickness*0.5 which defines that, for example, with 20 x 50 mm piece the placement position for plank's anchor point is 10 mm begind the element's front surface. Variables that can be used here are (exmples for plank size 50x100 mm): • mat_thickness (0.050) • mat_height (0.100) Rotation and z-offset affect like this:  Please see information about the planks.
Plank's center line offset	For the side pieces this value is the placed part's middle line from the edge (can be placed outside the element also). If the part is rotated by 0 degrees, these settings are good (please see the above item also):  And the formula here for copy&paste: mat_thickness*0.5 or mat_height*0.5
cnc_top and cnc_bottom	These settings can be used to automatically exclude top&bottom plates from the cnc-file. If there is a framing station in element assembly, it typically provides the top and bottom plates.

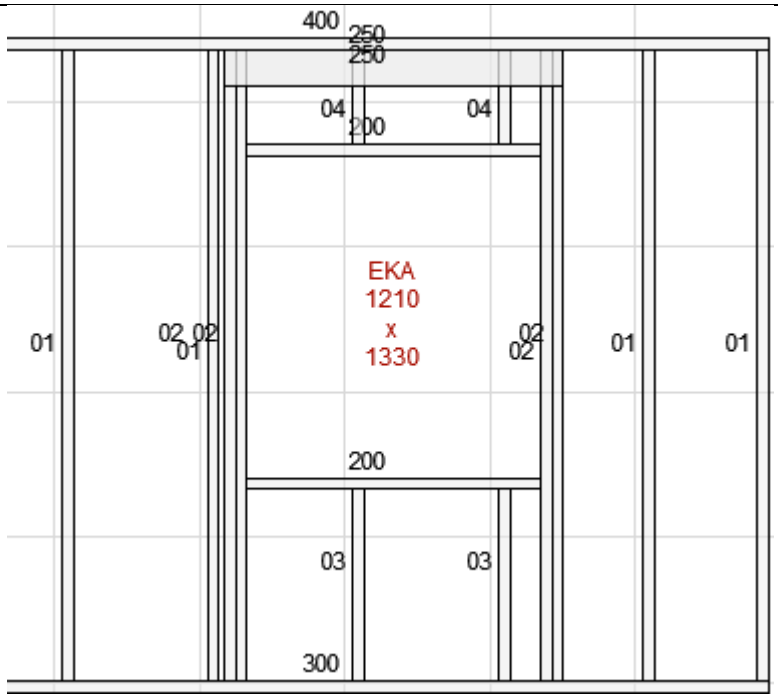
Planks to opening sides and opening hor	Used in horizontal air stripes. The setting *Planks to opening hor* affects the ref pieces and with the value 1 the red pieces will be created:  Setting *Planks to opening sides* works like here for framing horizontal, value 0:  1:  2:

	 For vertical air stripes the result is for value 0 (follows the main framing and that causes pieces to left and right hand sides of the openings):  Value 1:  Value 2 extends opening sides top & bottom by plank's width:  Value 3 is same as value 2 but planks ends exactly to opening line.
Spacing	Spacing between studs or similar pieces.
Spacing tolerance	How much for example a stud position may differ from the spacing rule before there will be a double stud. For example, value of 10 mm will result in a double

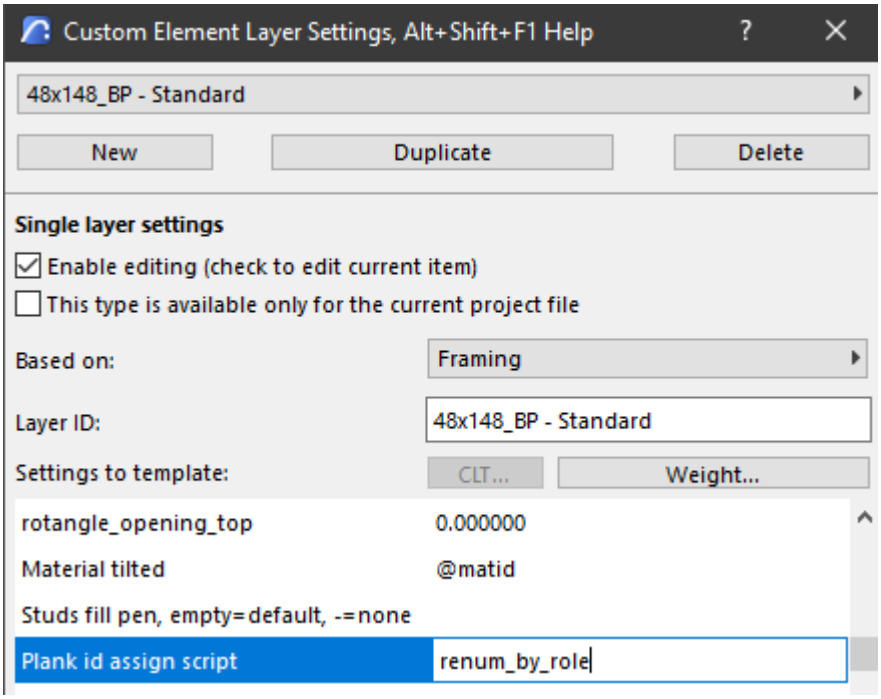
	stud if existing stud's position is further than 10 mm from the spacing rule. 
Min distance between studs	This value is used when studs are touching each other (or the be exact, the gap is less than this value). The new stud is moved further to allow putting insulation between the studs to avoid air leaks. For example: 
rotframing	Sets rotation for framing the element different to what is used to create the element drawings/projections. The value can be set only to framing layers. An example using wall, without rotation: 

	With 90 degrees rotation - please note the double top plate at the right and that the projection direction is unchanged: 
opening_square	This “openings to square” setting allows to change the angled opening tops and bottoms to straight. That can be very useful for horizontal air space strips like this:  Here the value is set to 150 mm to get the angled parts lower or equal to that converted to straight:  Other useful values for this type are:

	 Planks to opening sides 3 cause the horizontal pieces to match exactly the opening width – number 2 would add plank's height to the length. Offset to middle -setting was used to get the first spacing rule plank's center to be 624 mm from the bottom of the element. Negative value here starts spacing from the bottom overriding the *Main framing rule* setting. Setting *skipcorners* will skip placing a plank to each corner.
joinends_contour joinends_opening	With the default value of 1, the planks on tilted contours are connected automatically:  Value 0 (to be adjusted with element operations differently):  Attribute joinends_opening will affect openings instead of the contour line.
Plank id assign script, renum_script	Allows using ID-numbers based on plank's role in the element like this:

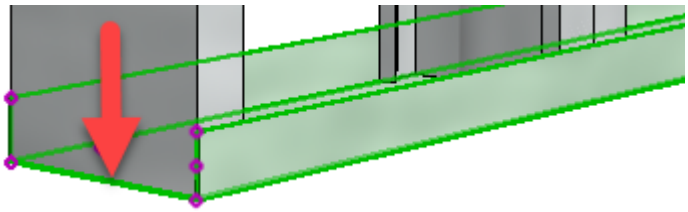
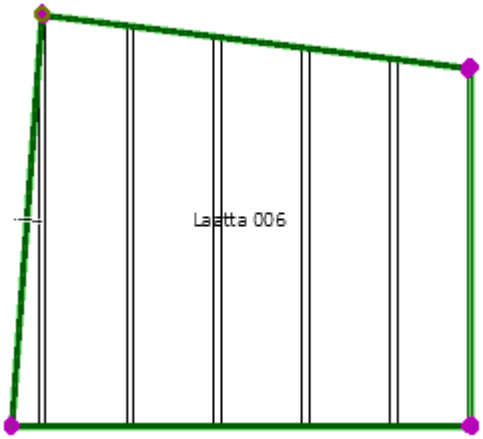
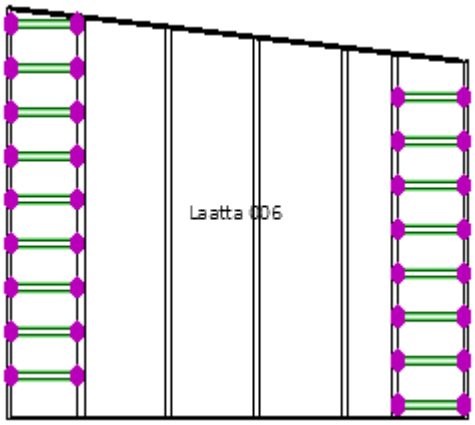
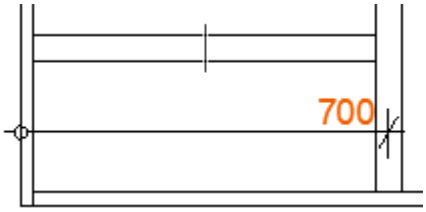


To use, a script is needed. There is a built in script *renum_by_role* in data/ArchiFrameElements.xml that can be referred like this:



More information: xml-tag renumscript.

CNC output settings	There are these settings used in various cnc-exporters: • cnc_outputline, reserved for future: To be used at btl-files telling the output slot. • cnc_nailgun, wup-cnc-export: Default nail gun index number to use for nailings. • cnc_matindex, wup-cnc-export: Material index number for the layer. • cnc_matname, wup-cnc-export: Material name for the layer.

	• cnc_isframing, wup-cnc-export: Is the layer framing layer, value 1 sets as framing and the planks in the layer are saved as QS, OG, UG, etc. Otherwise all pieces are saved as PLX. As default core and intstud* layers are saved with framing codes. Layers intstud* will be saved as PLx if this is set to value 0.
Extend joint	Used for metal framing telling how much the stud should extend into the top/bottom/opening top/bottom pieces: 
Create left, right, top, bottom and angled	By default, all element's contour line parts are created. Setting the related setting to zero will cause the piece to be missing.
Create border	Used in floor structures to define whether to create the border parts: 
Floor side piece settings	These settings affect the floor side pieces:  • Distance to side mid is this distance:  • Spacing side is the spacing for the side pieces

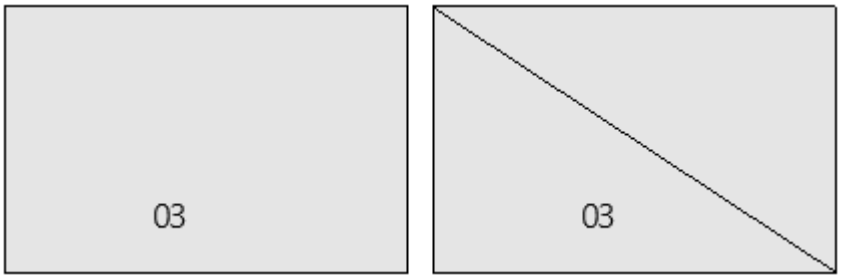
<table>
<tr><td></td><td>

- Spacing tolerance side, please see this.

</td></tr>
<tr><td>Board settings</td><td>

Various settings for newly created boards.

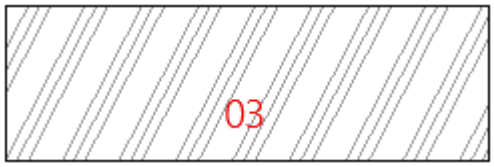
- Diagonal line with value 0 and 1:



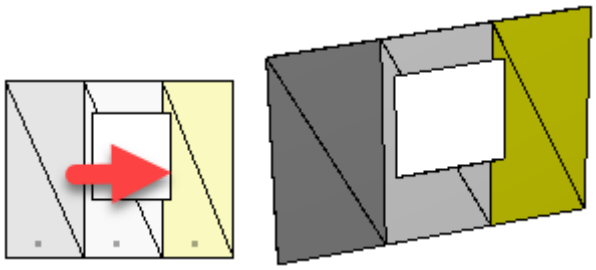
- ID pen:



- Board fill 2D



- board_fill_pen_from_3dmat makes the 2D fill color to be the best match to the 3D-material:



- Show ID in front projection:



</td></tr>
<tr><td>Board id and size</td><td>

Board size settings

- Board ID is shown to the user in the cut lists:

BOARD EXT

Type	Pcs
WS 9 x 1200 x 2700 (02)	1
WS 9 x 1200 x 2700 (03)	1
WS 9 x 1094 x 2700 (01)	1

ArchiFrame excpects the board ID to be in format NAME (thickness)x(width)x(height), separator can also be character *. For

</td></tr>
</table>

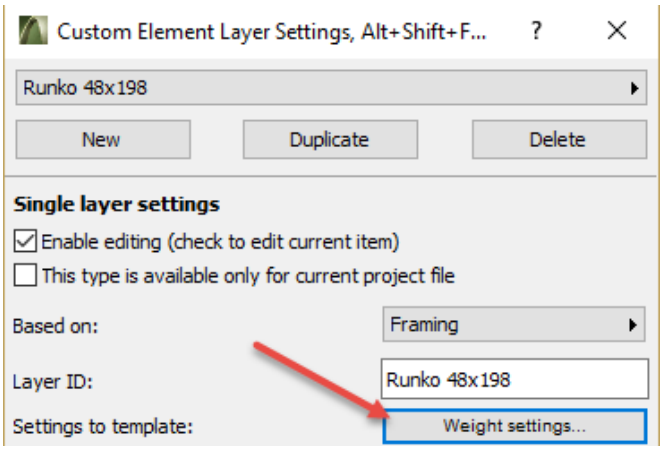
	example: WS 9x1200x2700 or WS 9*1200*2700 • Board width is the maximum width of the raw board material • Board height is the maximum height of the raw board material To model the whole layer with a single board, big width and height can be used.
Cladding settings	These are the available cladding settings: • Cladding rotation: 0 produces horizontal and 90 vertical cladding. Negative value -90 can be used to swap the panel profile. • Paneling anchor defines where ArchiFrame places the anchor point. • Pane offset can be used to start cladding with a cut piece and/or to get uncut piece above openings if there is a standard height for walls. • Cladding type is one of the types defined in ArchiFrameElements(Changes).xml. There are two types: inout and outin. The outin types are correctly oriented for composites looked from outside having the exterior layer defined first and the other way round. To use panelling in interior side, the inout/outin must be swapped: If looking from outside, inout type is the right one to interior side. Please see more information here.

11.2 Weight calculation

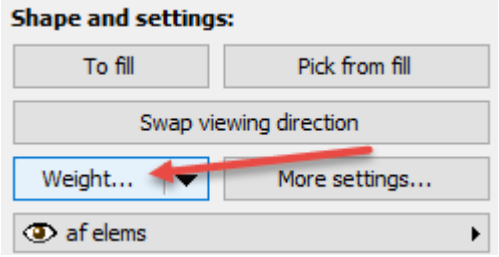
ArchiFrame contains weight definitions in own element settings per composite structure layer and in plank definition file ArchiFrameBlocks.xml in the *data*-folder. The order of getting a plank's or board's weight is (the latest definition will be used):

- Built in default which is:
    - 450 kg per cubic meter for planks (450 gr per liter), that is density for dry pinewood.
    - 1100 kg per cubic meter for boards (1,1 kg per liter) which is density of typical gypsum board.
    - 30 kg per cubic meter for insulation (30 gr per liter).

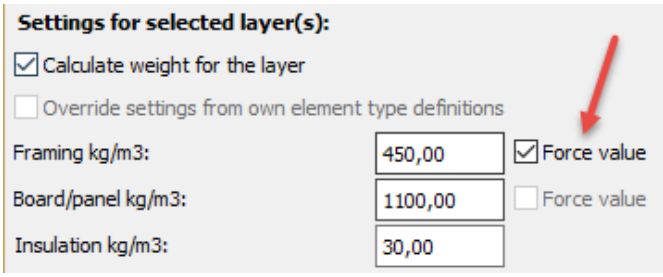
- Value set custom element layer settings:



- Overridden value from placed element settings:



- Value set for the plank type in ArchiFrameBlocks.xml unless Force value is checked in the weight settings dialog:



- Any Archicad door/window/lamp/object having parameter FM_ObjectWeight nonzero will use that value. Related parameter FM_ObjectWeightUnit can be kg or lb.

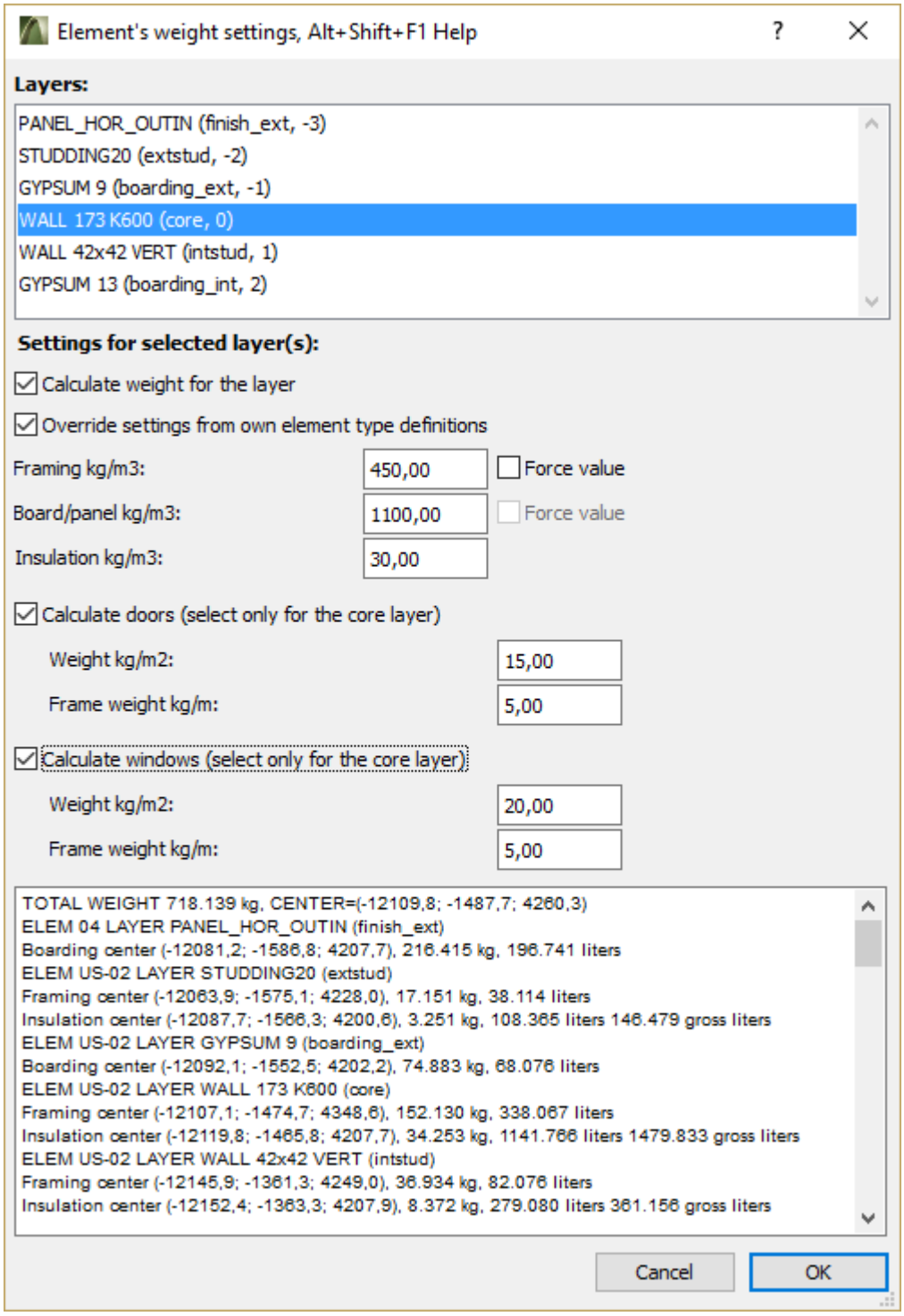
ArchiFrame can calculate weight for an element or for any Archicad selection. In latter case, it will also place a 3D center of gravity marker. The gravity marker is not linked to calculated elements and if the weight is calculated again, there will be a new marker object.

Weight for doors & windows are calculated based on linked Archicad-windows/doors. If ArchiFrameElement-object is not connected to any Archicad wall and is vertical, all holes in the element are treated as windows. In that case a warning is issued: Element EW-01 - Copy is not linked with any Archicad-element, weight calculating treats all element holes as windows.

If the cladding layer has any planks, all ArchiFrameBoardPanel-objects must be exploded to separate pieces to calculate the weight correctly. Shortly: either all or nothing should be exploded in cladding layer.

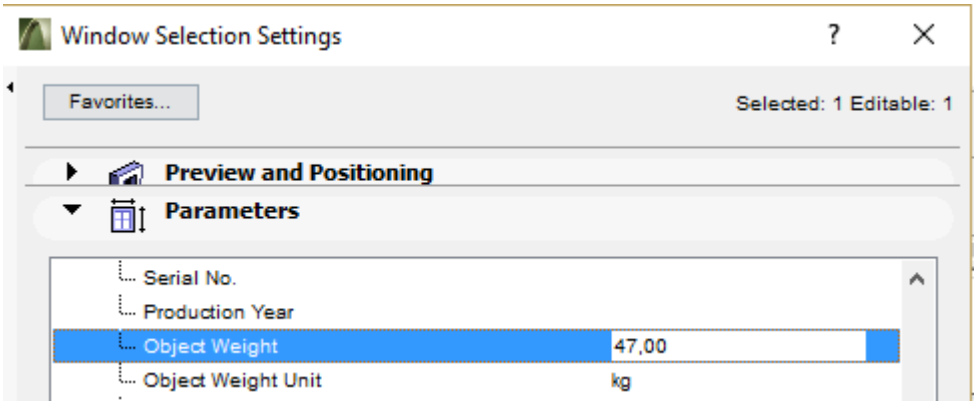
Insulation modelled with ArchiFrameBoardPanel-object is calculated always using net volume – in other words the volume of the GDL-object is used framing and any cuts removed from the volume.

11.2.1 The weight settings dialog

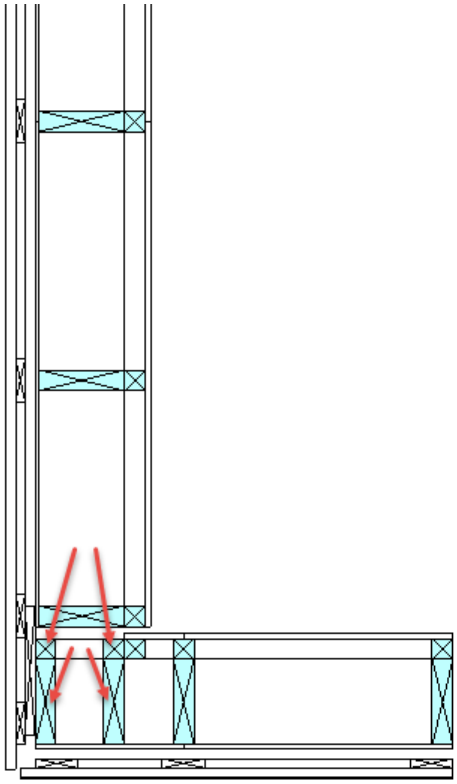


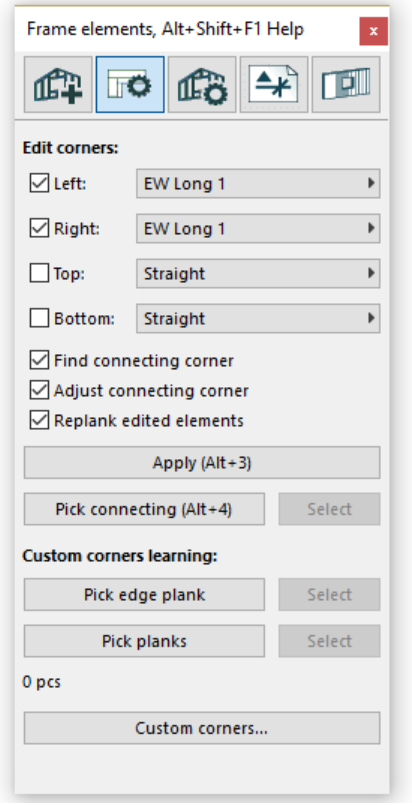
- *Layers* lists all different composite layers in selection. If opened from *Custom element layer settings*-dialog, it will contain just the current layer. It is possible to select multiple items from this list and edit the values to all selected composite layers.
- *Calculate weight for the layer* can be turned off, for example, if the related layer is built at the building site after installing the element.
- *Override settings from own element type definitions* enables setting values differing from the custom element layer settings for specific placed element. It is useful for example if a board type is changed in just a specific wall in the building.

- *Framing kg/m3* defines density of framing parts. For profile pieces like I-shaped beams the volume of the plank is calculated taking the profile into account. Please note that in ArchiFrameBlocks.xml the density for a plank type can be given as kg/liter, kg/m or lb/foot.
- *Force value* for framing will override the plank specific density given in ArchiFrameBlocks.xml.
- *Board/panel kg/m3* defines the density of the boarding layer. For non-exploded cladding this value contains also the air space caused by then planed profile.
- *Calculate doors/windows* enables calculating the weight of related Archicad doors with the element layer. This option should be checked only for single layer in the composite structure. Anyway, ArchiFrame will not calculate the same part twice.
- *Weight kg/m2* defines the weight of the door leaf/window glass. ArchiFrame calculates the area for the door leaf following the door/window frame so the area is probably slightly oversized.
- *Frame weight kg/m* defines the weight for the door/window frame. For door and windows, it is possible to set Archicad standard parameter FM_ObjectWeight nonzero value and it will be always used if set.



11.3 Element corner tools





The corner tool connects elements (both single and multi-layer) together and contains rules for placing special planks at corners. In the image above, the special planks, marked with arrows, are placed according to the corner rules. Corner rules also include layer offsets. The tool is split into two parts:

- Built in corners that are not editable from ArchiFrameElements.xml.
- Custom corners that user teaches to ArchiFrame.

Video: https://vimeo.com/277423174
https://player.vimeo.com/video/277423174

Parts of the corner tool palette are:

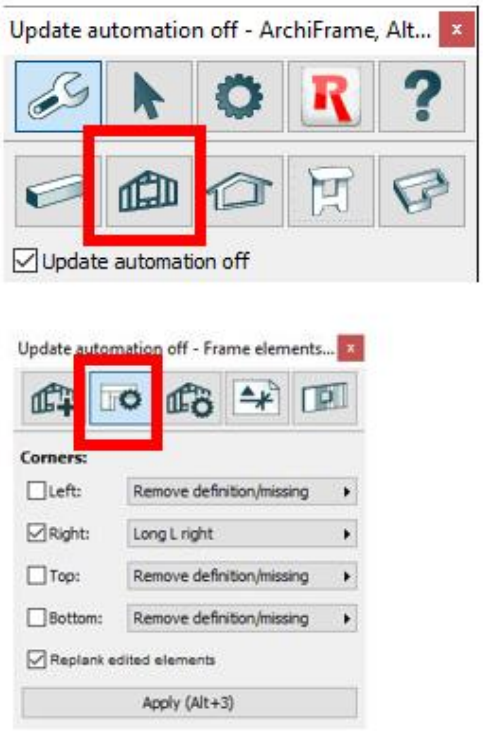
- *Left/Right/Top/Bottom* check boxes control which sides of the element are set. Unchecked, the setting will not be changed for that side. If the check box text is **bold**, the setting is set in selected element.
- *Find connecting corner* defines whether the corner tool should search for related elements. If unchecked the target element's edge will not be moved. Otherwise it is moved to given distance from connecting element's surface. If operating in the floor plan only elements in the current storey are scanned – in the 3D all elements are scanned.
- *Adjust connecting corner* enables setting both parts at the same time. It requires that the selected corner type has defined the connecting corner's type.
- *Replank edited elements* automatically creates planks again if target elements are changed.
- *Apply* applies any changes that have been made.

- *Pick connecting* will pick the selected element object to be used as connecting element. It also defines connecting corner when teaching new custom corner type. To clear definition, click this button without any selection in Archicad.
- *Pick edge plank* is only used when teaching new corner type to ArchiFrame. It must be a plank related to the main framing layer. ArchiFrame learns the closest element edge to this plank. To clear definition, click this button without any selection in Archicad.
- *Pick planks* is only used when teaching new corner type to ArchiFrame. The picked planks' types and positions are included in the corner rule. To clear definition, click this button without any selection in Archicad.
- *Select* next to any *pick*-button will select picked elements in Archicad.
- *Custom corners* opens custom corner settings dialog where it is possible to edit old and add new corner types.
- *ArchiCorners* is used to add corner covering boards.
- *Overlap double tops* crosses the top plates and is widely used in precut. Please see the related video.

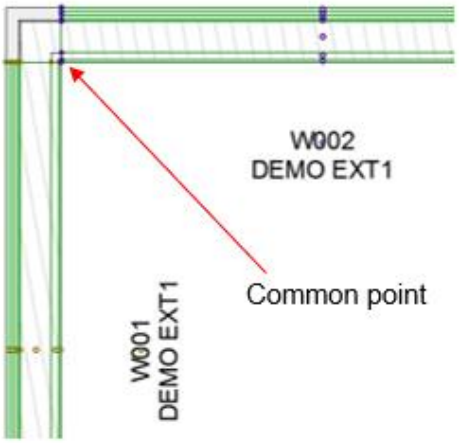
11.3.1 Follow these steps to model corners:

1. Open the corner tool by going to Element Tools → Corners and joints.
2. You must have two elements that share at least one point.

1.



2.

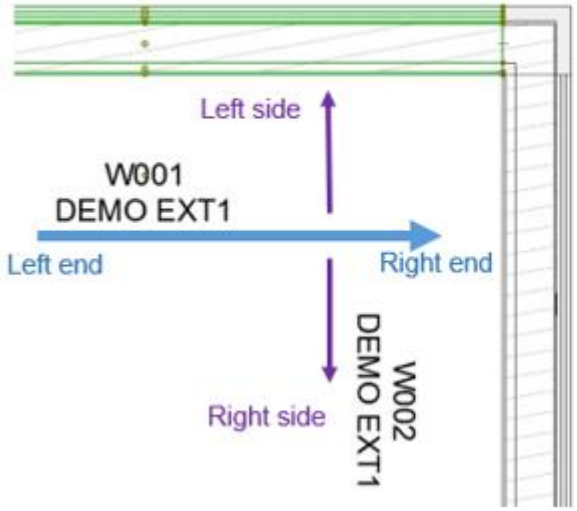


3. Select one element, choose appropriate corner settings and click "Apply".

- In Picture 3, the selected element is W001 (highlighted in green).
- The direction of W001 is shown by the blue arrow, and can also be determined from the direction of its text (W001 Demo Ext1).
- W002 is therefore at the right end of W001 and on its right side.
- Choose "Long L right" as the type of the element's right corner:
    - Right corner because the connecting element is at the right end of W001.

- “Long L right” because we want W001 to make up the longer part of the corner and because the connecting element is on the right side of W001.

3.





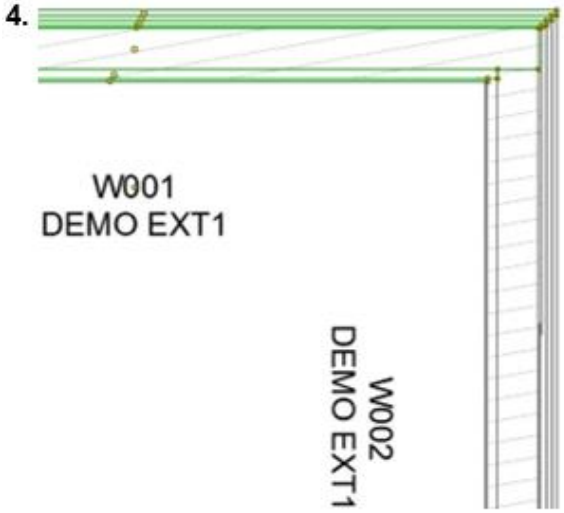
4. The result.

- The wall element W001 now makes up the longer part of the corner.
- Archiframe automatically sets W002’s end to type “short”.

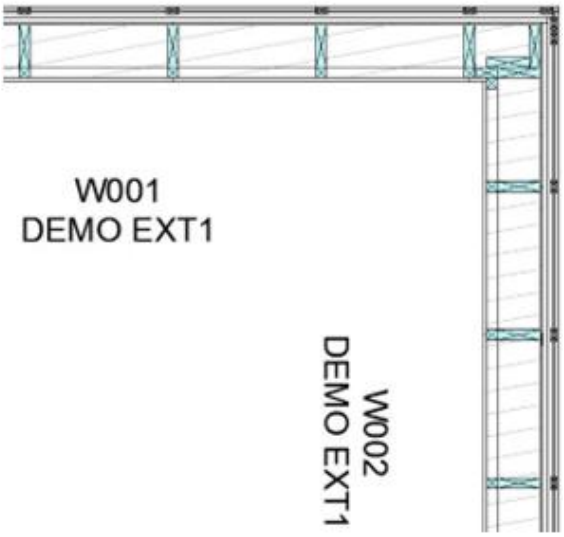
5. Select both elements and click “Create planks” to see how the corner planks are placed.

- Create planks can be found in the “Add Element” window.

4.

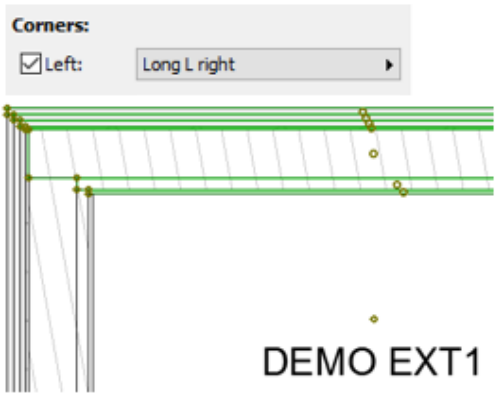


5.

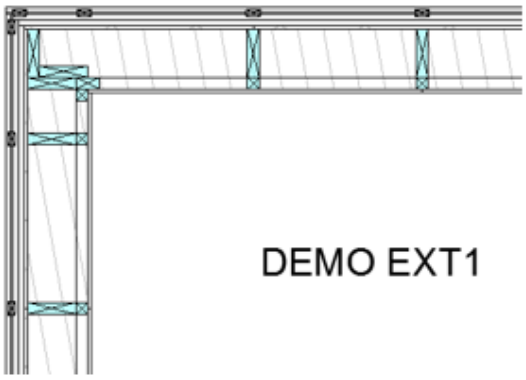


11.3.2 Examples of different corner types

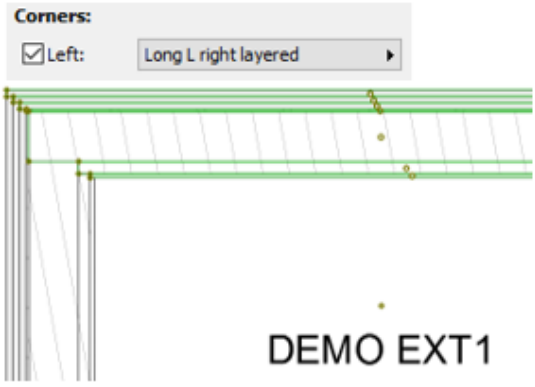
- Sometimes different corner types look the same when set for elements. In such cases the corner types' differences become apparent after planking is created.
- For example, styles *Long L right* and *Long L right layered* only differ after planking:



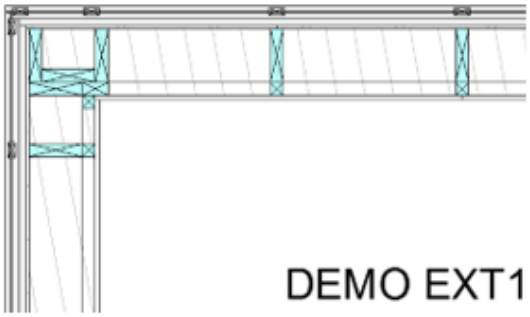
The selected wall element's left corner has corner style *Long L right*.



Result after planks have been created.



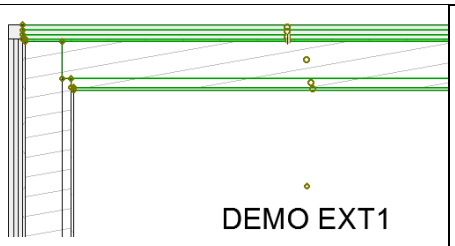
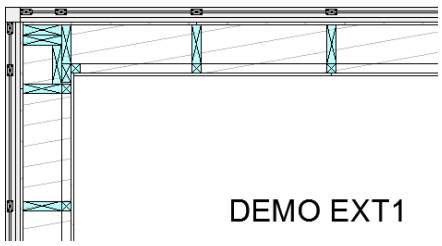
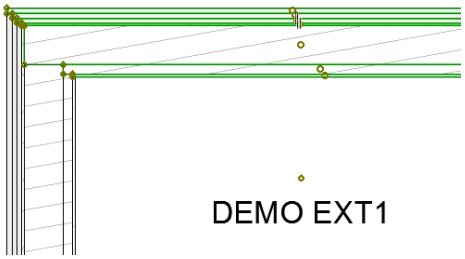
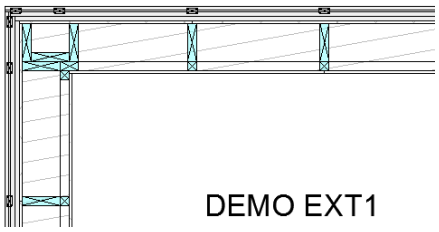
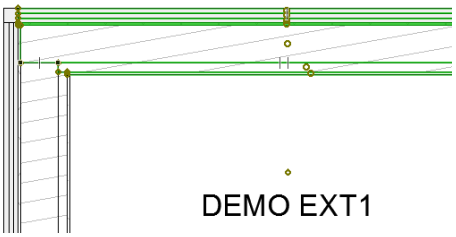
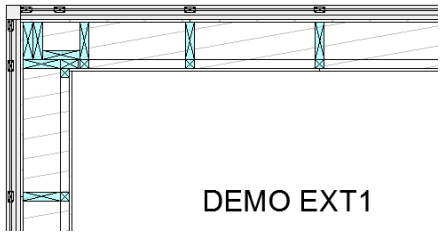
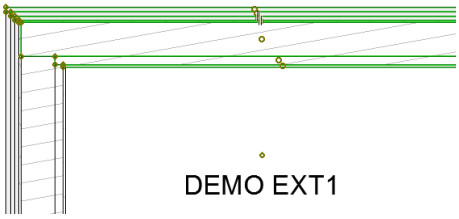
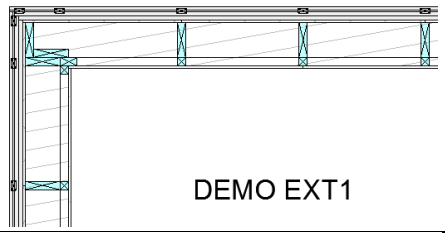
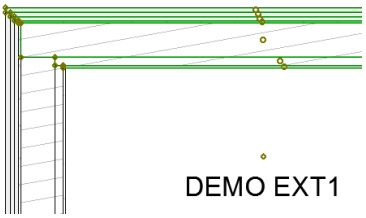
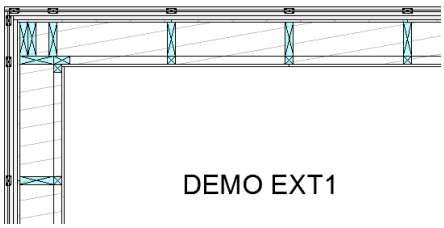
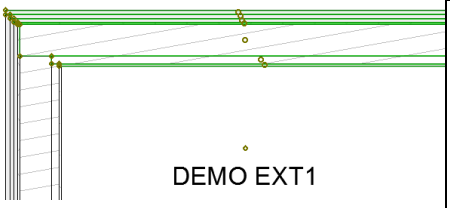
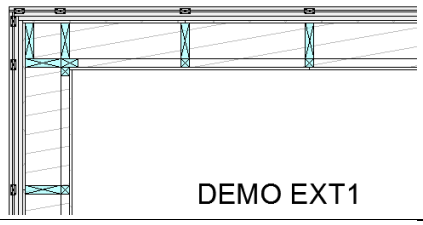
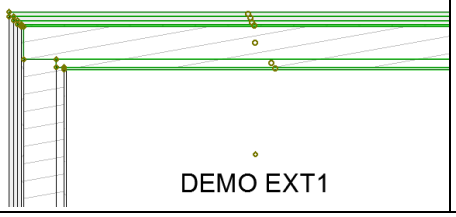
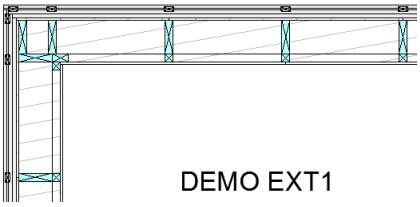
The selected wall element's left corner has corner style *Long L right layered*.

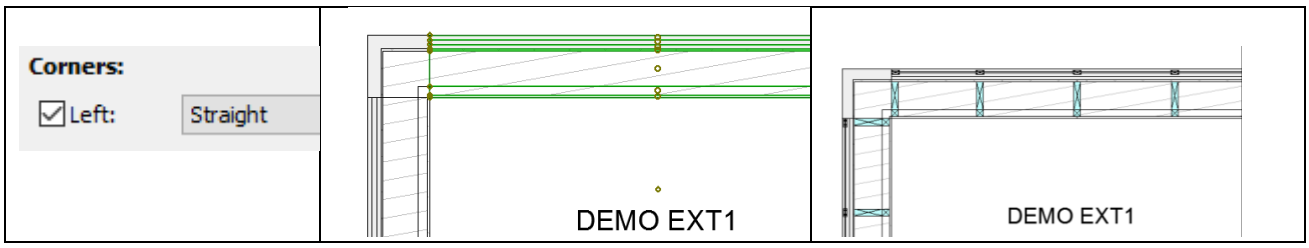


Result after planks have been created. The planking is different from the one above.

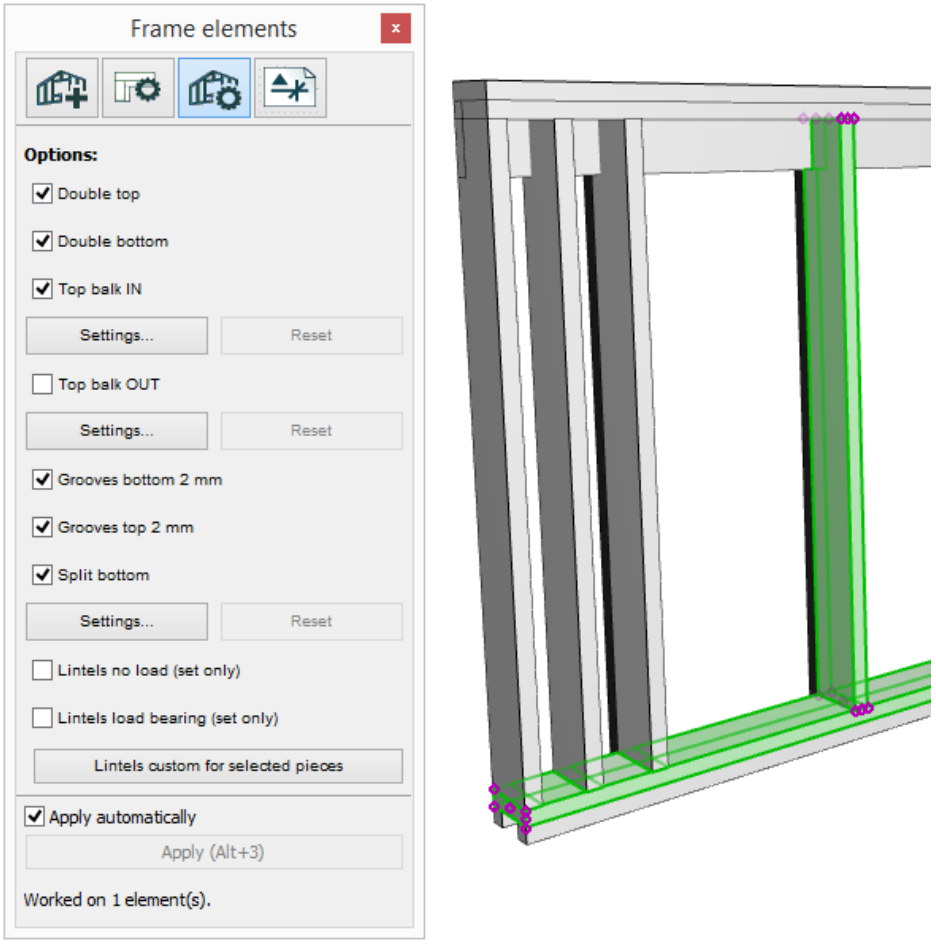
Here is a table of different corner types before and after planking. The wall element type used is called Demo Ext1 and contains (from inside out) gypsum, studding, framing, boarding, horizontal studding, vertical studding, and cladding.

Corner type	Before planking	After planking
Frame elements, Alt+Shift Corners: Left: Short	DEMO EXT1	DEMO EXT1

Corners: Left: Short 3		
Corners: Left: Long L right layered Long L right layered		
Corners: Left: Long L right layered 3 Long L right layered 3		
Corners: Left: Long L right		
Corners: Left: Long 3		
Corners: Left: Long 2		
Corners: Left: Long		



11.4 Element options tools



Sets various options to the element. The options are automatically applied when recreating the elements. Some options like lintels cannot be unchecked. The options are defined in ArchiFrameElements.xml with Lua-scripts and can be customized.

When changing many options to a big element it may be useful to uncheck *Apply automatically*, set all the options and apply those once with *Apply*-button.

Planks to opening sides and lintels custom can be limited to only selected pieces (please select the top plank for lintels). The element should be replanked after changing option *Planks to openings sides* since the option affects studs from spacing rule. Or the option should be initially on before creating the planks.



Links to videos:

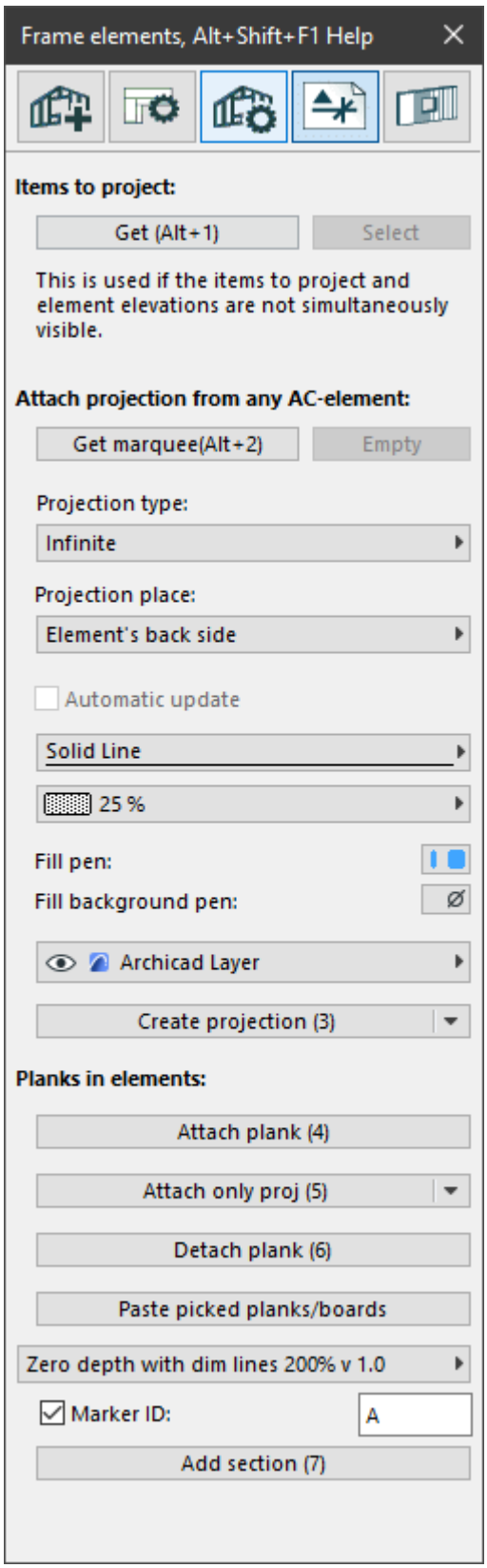
- Double top and double bottom: https://vimeo.com/173890540.

11.5 Projection tools

An example video:
https://youtu.be/GV1QUg24sZo

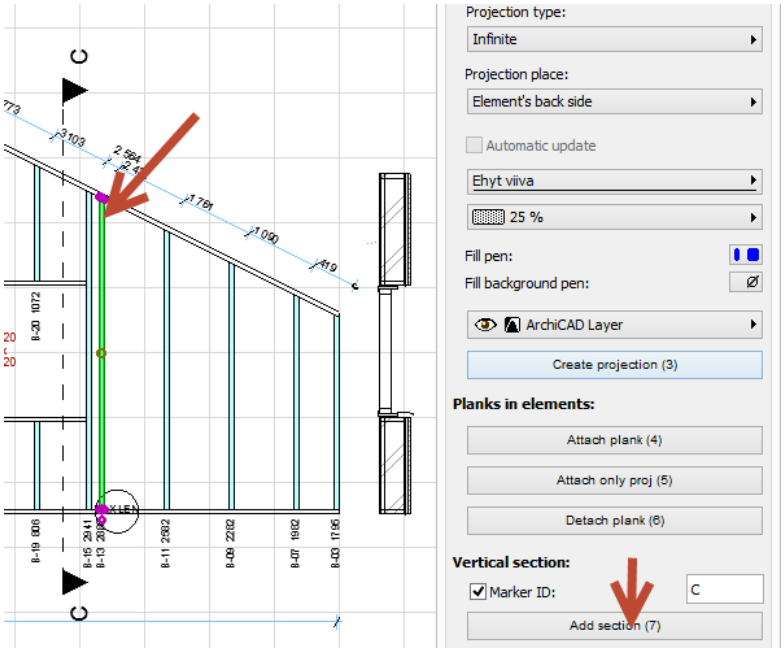
Projection tools are used to attach plank projections or projections from any Archicad element to the element elevation. *Attach projection from any AC-element* creates the projection as fill and it will not be updated if the original element is changed and vice versa. Moving the projection fill will not move the original Archicad element. *Planks in elements* tools makes fully updating projections from ArchiFrame planks only.

Projections can be added by selecting the elements or planks to project, at the same time as the destination projections where source elements should be projected. If this is not possible, for example, because the element elevations are in different storey than the elements to be projected, the *Items to project / Get* –command will pick the source elements. After this the projections are created with one of the commands *Create projection, Attach plank or Attach proj only.*



- *Get* will take source elements, for example, from another storey to be projected. All source elements to create projections or attaching to elements must first be picked with this. Can be used also to pick Archicad walls/roofs/slabs to redefine sources for ArchiFrameElement-objects.
- *Get marquee* picks current Archicad marquee to define projection clipping area. It is useful to show, for example, just a part of long beam in single element elevation.

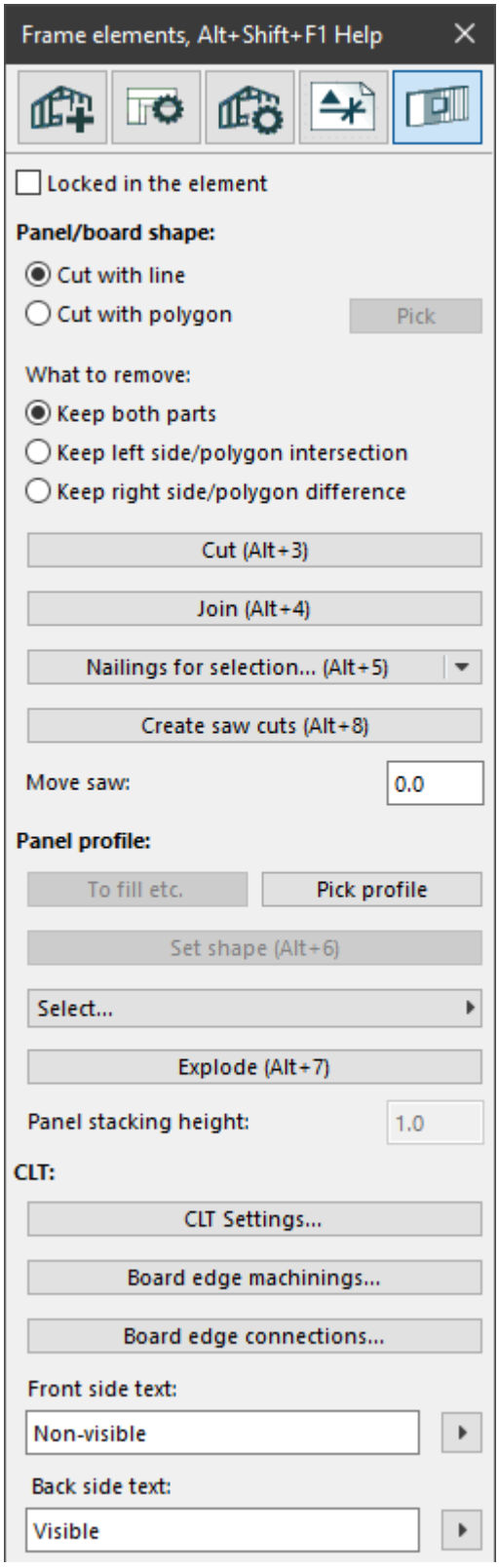
- *Projection type* can be zero depth or infinite. For example, if there is a log joint at the projection plane (the element's front or back side), the log will appear as shaped from the joint if using zero depth type.
- *Projection place* is either on element's front or back side.
- Rest settings in the group define the resulting fill. Archicad fill tool settings are used as well.
- *Create projection* creates selected or picked source elements' projections to selected element elevations as Archicad fill.
- *Attach plank* joins selected ArchiFrame plank to be part of the element. The attached plank is deleted when element is replanked. *Attach only proj* places only projection to the element elevation. The original plank is not deleted if the projection is deleted or element is replanked. This is the recommended option to add projection of a beam to the element elevation.
  If the picked sources were Archicad walls/roofs/slabs, attach will redefine sources for ArchiFrameElement-objects. This will allow ArchiFrame to for example update openings or calculate door/window weights for copied ArchiFrame-elements.
- *Detach plank* deletes selected plank projections without deleting original plank. In addition, the link from the original plank to the element is removed. This operation can be issued having only original plank(s) selected. In this case, it will remove every related projection and makes the plank independent from any element. Another option is to first use *Get* to pick planks/boards to be detached, then select ArchiFrameElement-objects or parts related to those. In this mode, planks will be detached only from related ArchiFrameElement-objects.
- *Paste picked planks/boards* will copy the picked planks to currently selected element(s) maintaining the geometry of the picked planks relative to their owning ArchiFrameElement. If picked elements and selected targets are from a single composite layer, ArchiFrame will copy the pieces not checking the layer types. Otherwise, it will match the layer types from source to target (boarding_ext2 to boarding_ext2 etc.). Please see a video here.
- *Add section* creates vertical section of the element. To use it select suitable layer combination (to see windows, the wall layer must be on), select one piece from projection, click section line, side and place of resulting section. ArchiFrame creates a temporary section and converts it into static line drawing and places the result to floor plan. The drawing is not automatically updating.



11.6 Board and panel tools

These tools are used to edit polygon shape of the boards and panels. The difference to for example planks' groove tool is that it makes a new machinings – this tool just changes the polygon. There may be a big difference in resulting cnc-.

Please note that element main tools' operations to edit the outline *To fill* and *Pick from fill* can be used to boards and panels as well as for elements.

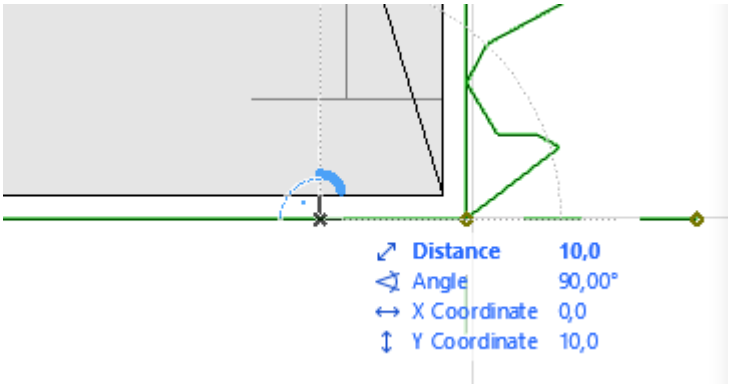


- *Locked in the element* sets or resets locking state for a board. When recreating the whole element or just boarding (having a board selected and clicking element main tools' button *Recreate boarding*) all locked boards remain at place and the rest is done according to locked ones.
- *Cut with line* allows splitting boards and panels into two or more pieces or just trimming existing ones.
- *Cut with polygon* first a working polygon is picked, then the target boards. This command is useful for example to change paneling just at part of an element (below windows for example). To use a polygon it must be first picked with *Pick*-button from existing fill or marquee.
- *Keep both parts* removes nothing. Instead with this option selected existing pieces are split into two or more pieces.

- *Keep left side/polygon intersection, Keep right side/polygon difference*, with polygon operations the target pieces must be selected. The operation results as (red one is a board/panel and blue one the picked polygon):



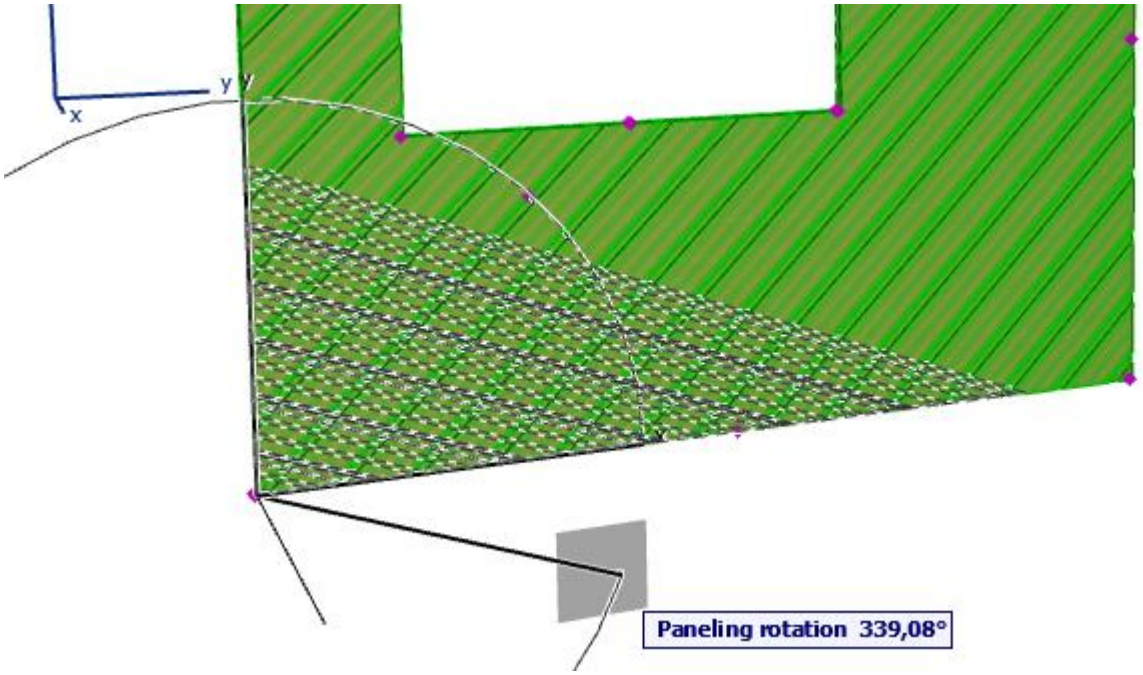
- *Cut* cuts boards.
- *Join* joins selected boards and panels together. The polygons of the boards to join must touch each other. If not, the outline needs to be edited prior to the operation.
- *Nailings for selection* places nail/screw lines to the boards. See Element nailings dialog.
- *Create saw cuts* makes board cutting objects to be used for cnc. Please see Panel cuts and Lua script for saw cuts for more details.
- *Move saw* offsets the saw from board edges, for example with value 10 mm:



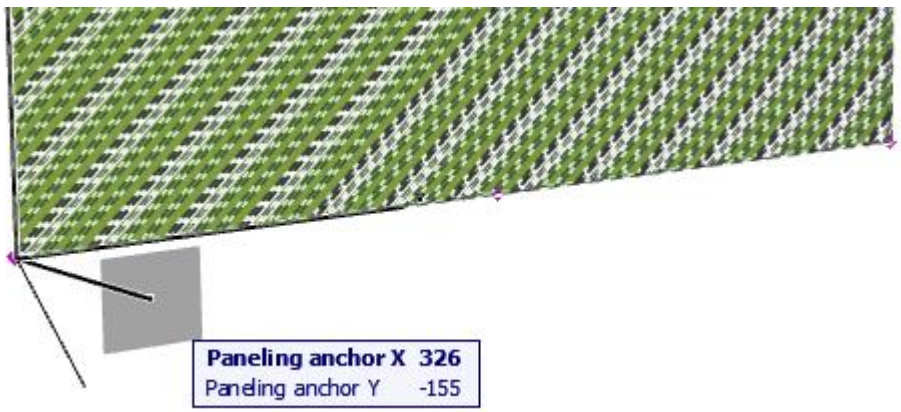
  This setting is used also when writing wup/btl cnc-files to connect saw cuts to the elements. At least as big value as used when creating saw cuts should be specified here when writing the cnc-files.
- *Pick profile* picks the profile from existing panel or specific panel profile drawing. Pressing shift when clicking the button dumps the profile as text to be copy & pasted as a preset item into ArchiFrameElements.xml. Please see section Panel profile definition.
- *To fill etc* creates Archicad fills, hotspots and lines for previously picked panel profile. These Archicad elements can be edited to edit the paneling profile.
- *Set shape* sets previously picked shape for selected boards/panels.

- *Select…* sets a predefined panel type (in ArchiFrameElements.xml) for the selected boards/panels.
- *Explode* explodes panel object into separate pieces for further editing or cnc-production. Please see Panel cuts for more details. Script based cut list requires that all ArchiFrameBoardPanel-objects are exploded if one is exploded. The non-exploded ArchiFrameBoardPanel-objects will be skipped in that case. Cladding layer may contain normal boards and they are included in the cut list.
- *Panel stacking height* sets the distance between exploded planks. This is used to match result with current cladding batch since actual height of the panel may vary.
- *Front/back side text* and related popups are used to set the texts in the CLT-boards front and back side. Please read this.

Panel profile rotation is set with a hotspot in 3D, floor plan or element elevation:

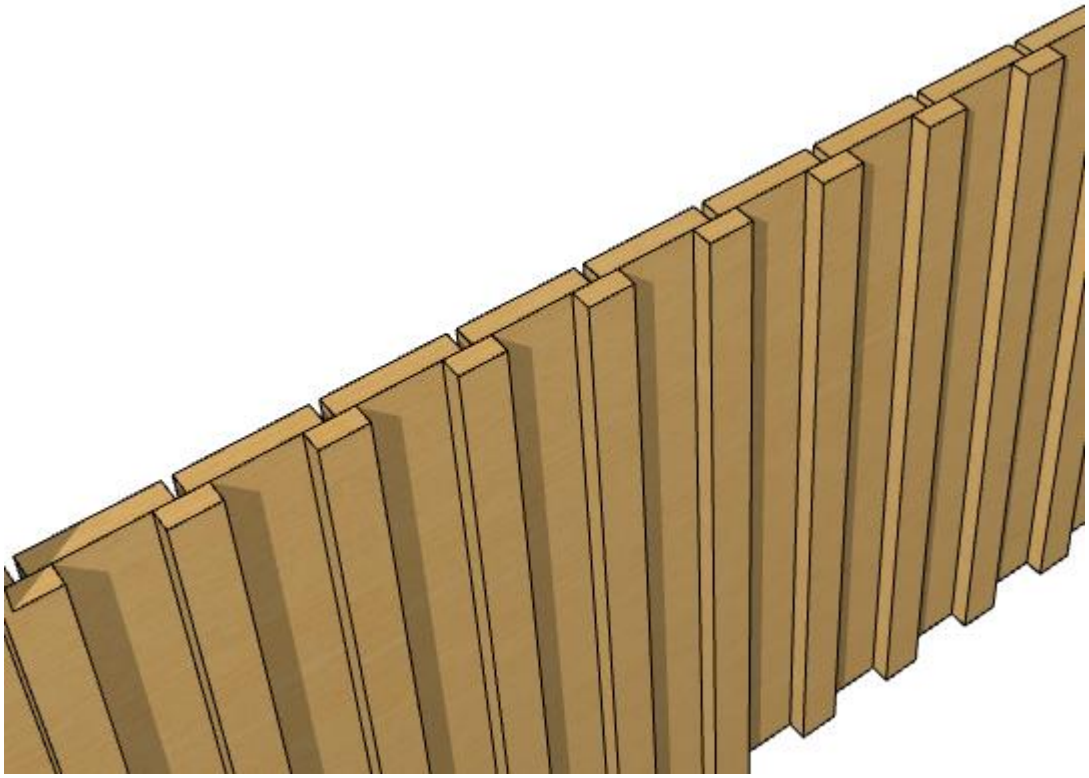


Also there is a hotspot to set the paneling achor point:



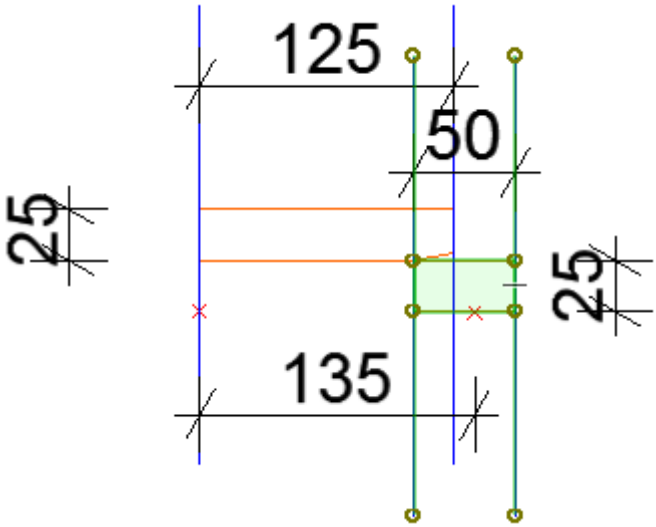
11.6.1 Panel profile definition and board/panel object's information

Let's use following profile as an example:



The paneling consist of two different pieces: 25 x 125 mm and 25 x 50 mm planks. The paneling profile definition contains definitions:

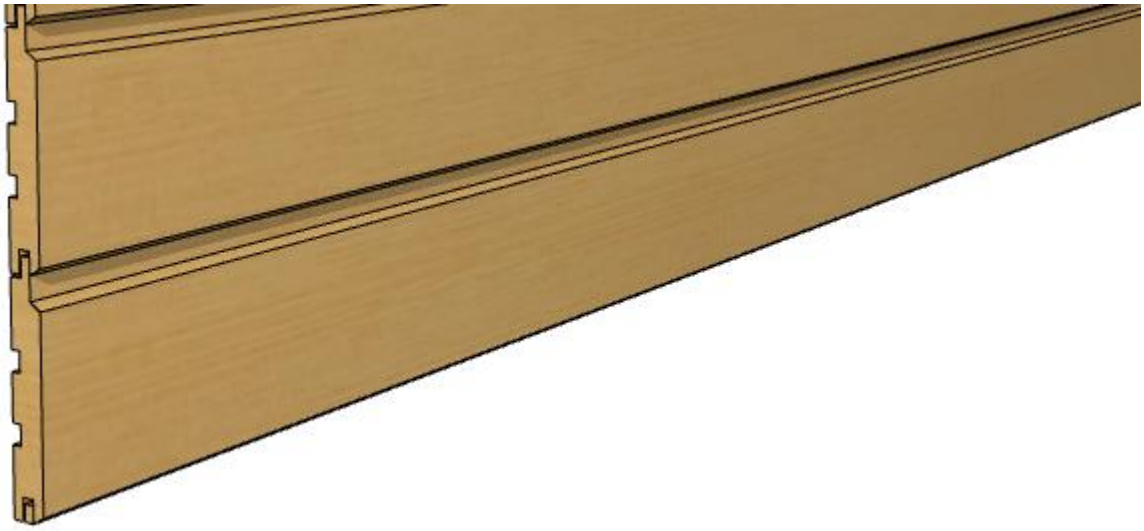
- Separate profile and placement for each plank type as Archicad fills.
- Lines to show in section (not to use real profile producing possibly much too many lines) as Archicad lines.
- Repetition anchor point and distance for the profile. In this example it is 135 mm. These are defined with Archicad hotspots. The left most anchor point works also as origin.



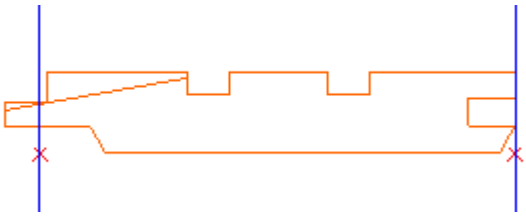
The image above has elements in two groups: First one contains the bigger one and it must contain also two hotspots defining the origin and the repetition distance (135 mm). All parts should

have positive y-coordinates. The second group is selected here and it contains the profile fill and two lines used to show the profile in sections. Please note that the profile is mirrored along if the element is looked from inside. The default *data*-folder contains similar panel definitions for both watching directions.

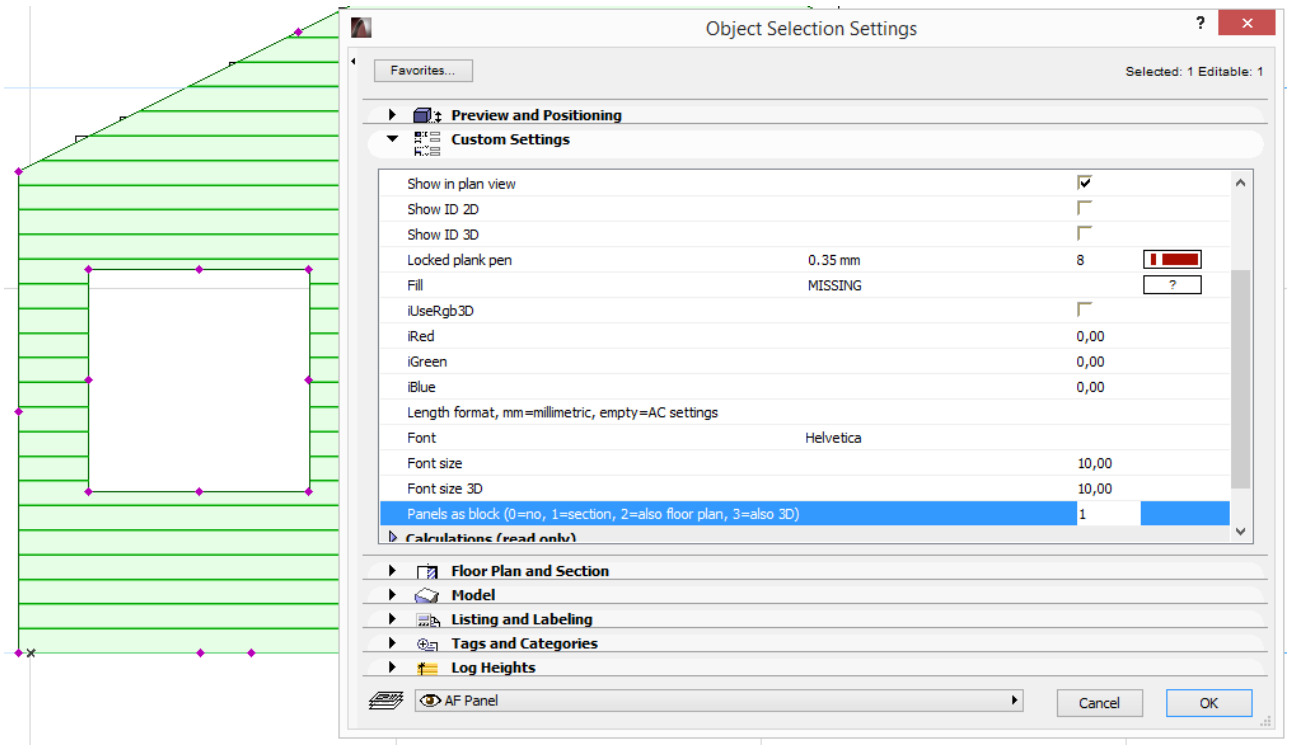
The meaning of the section lines is clearer for following paneling profile:



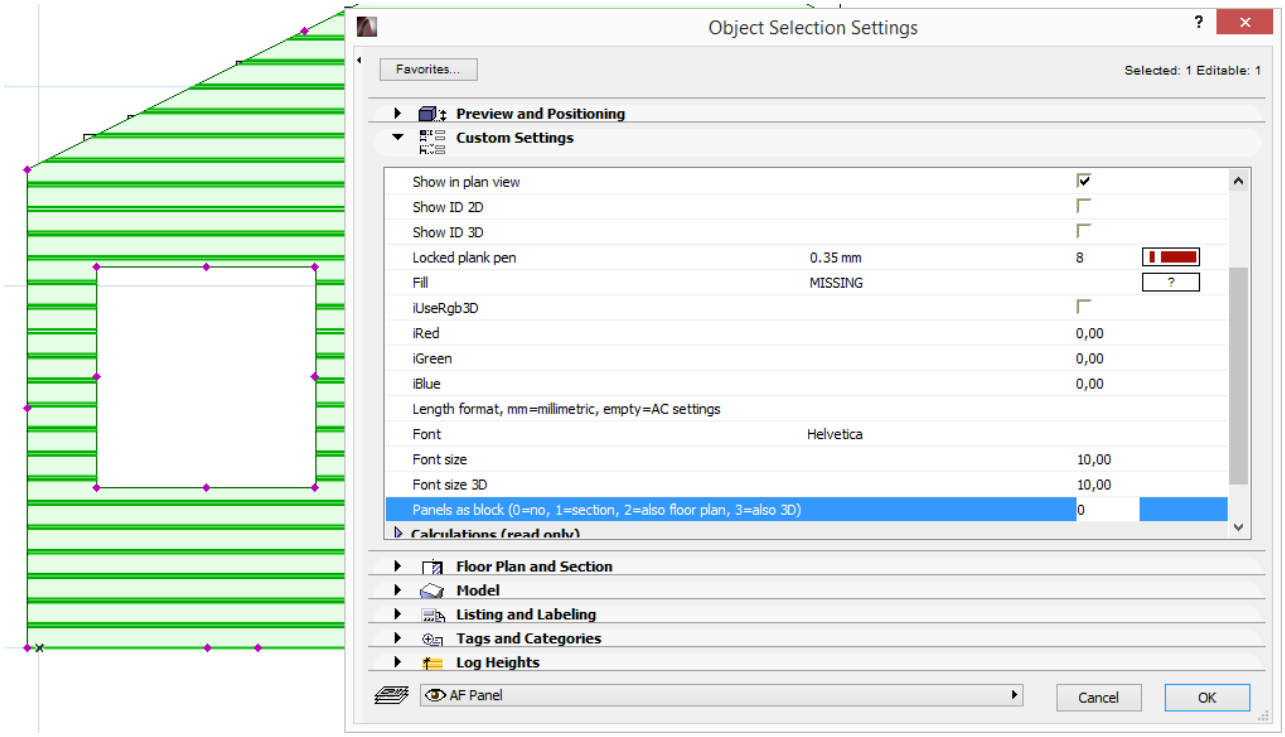
As panel definition:



With the default setting showing the lines only (Object tool/object parameters/View settings/Panels as blocks), the elevation looks like:



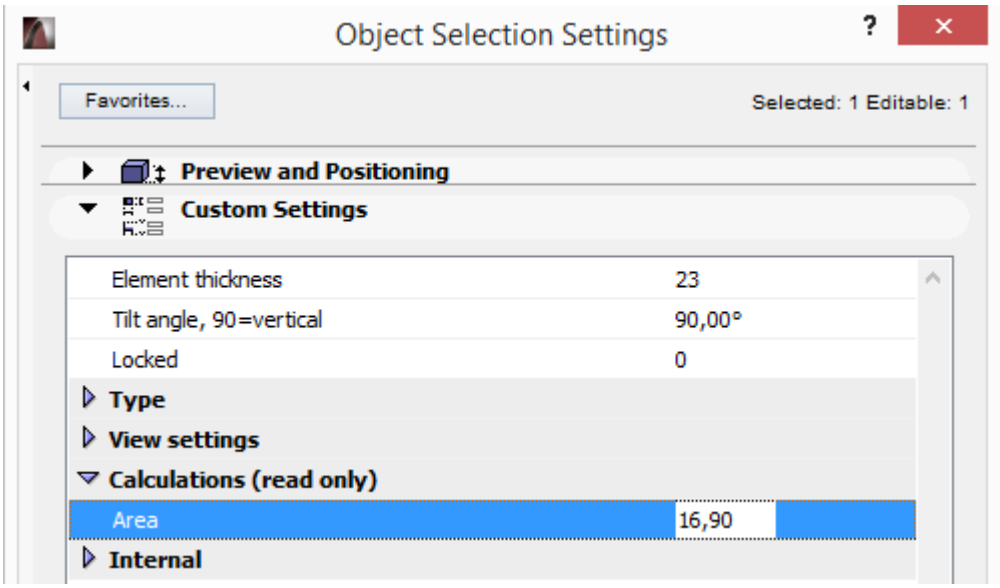
The same view showing the paneling with real profile is stuck with lines:



11.6.2 Technical information about the board/panel object

Preset profiles are saved in ArchiFrameElements.xml section <panelings>.

For calculations there is a parameter telling the net area of the board/panel (the area with holes reduced):



The panel profile is saved into ArchiFrameBoardPanel's table-parameter iPanelProf. It has triple items x, y, status. The special status codes are:

- -1, end of contour and next point is hole's first coordinate. Contour/hole starting point must be duplicated.
- -100, single panel profile ends, x,y unused.
- -101, size of current piece if modeled as a block.
- -102, Y-offsets from origin x = y1; y = y2.
- -103, X-offsets for lines in section x = x1; y = x2.

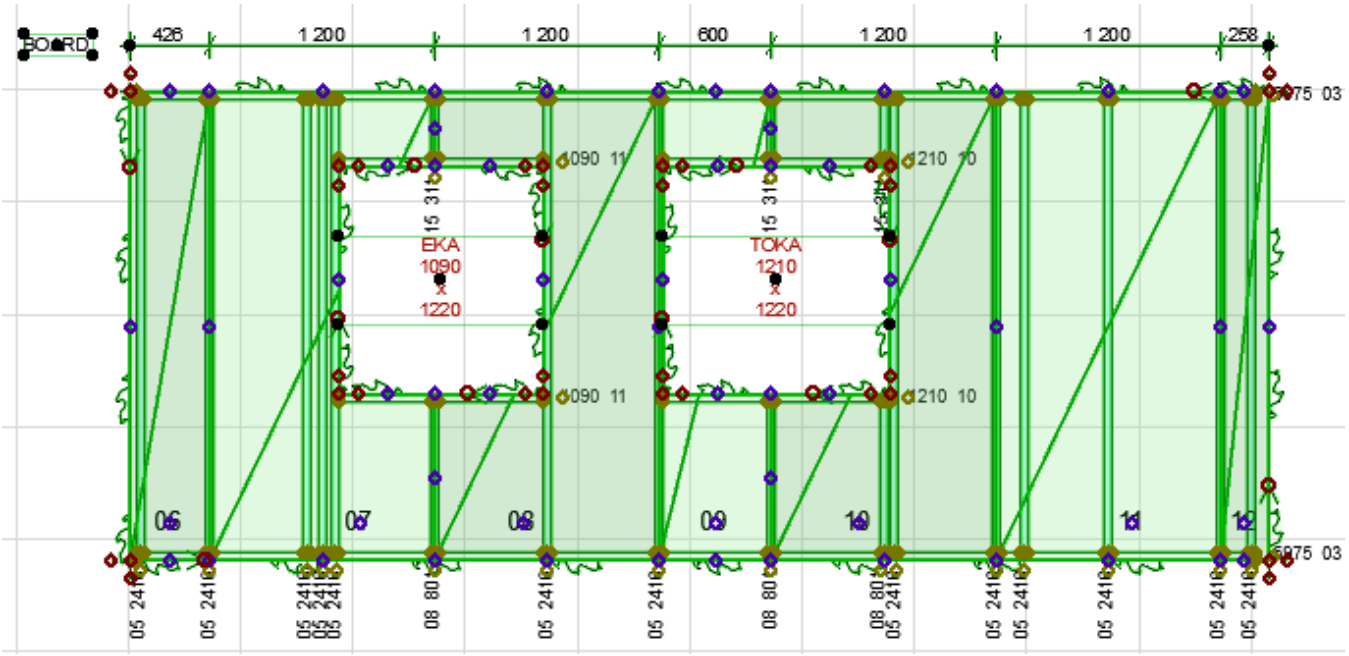
11.6.3 Panel/board cuts

There are two ways to define cuts for cnc-production: either use visible ArchiFrameSawCut-objects or rely on ArchiFrameBoardPanel edges. If there are any visible ArchiFrameSawCut-objects related to the target element layer (boarding or cladding), those will be used. ArchiFrame will warn about ArchiFrameSawCut-objects on hidden layers when creating wup cnc. If there are no ArchiFameSawCut-objects placed to the target boarding/cladding layer, the cuts are defined in the related ArchiFrameBoardPanel-objects. There should be just single boarding/cladding layer per projection with ArchiFrameSawCut-objects to give a clear one to one relation.

11.6.3.1 Panel/board cuts with ArchiFrameSawCut-objects

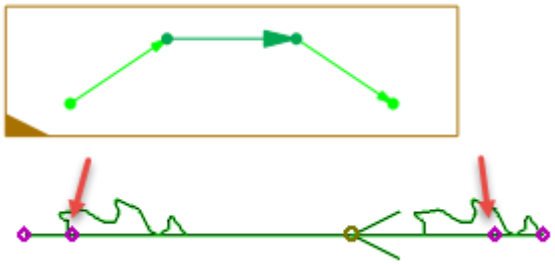
Saw cuts are placed by selecting the target layer boards and all existing saw cuts to remove those and then clicking *Create saw cuts* from the element board tool palette. Planks in selection are ignored. For example, to create saw cuts for the boards in this elevation it is ok to select everything

using selection rectangle – here existing saw cuts are removed, and new ones created:

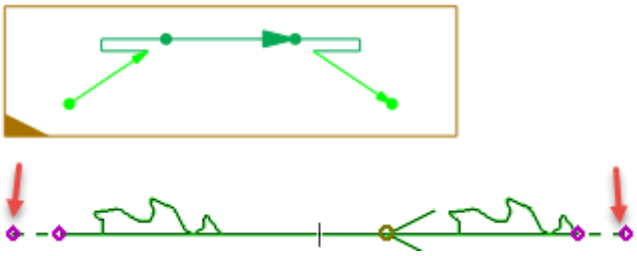


ArchiFrameSawCut-object has hotspots:

- At the ends to define whether it is an undercut:



Or overcut:

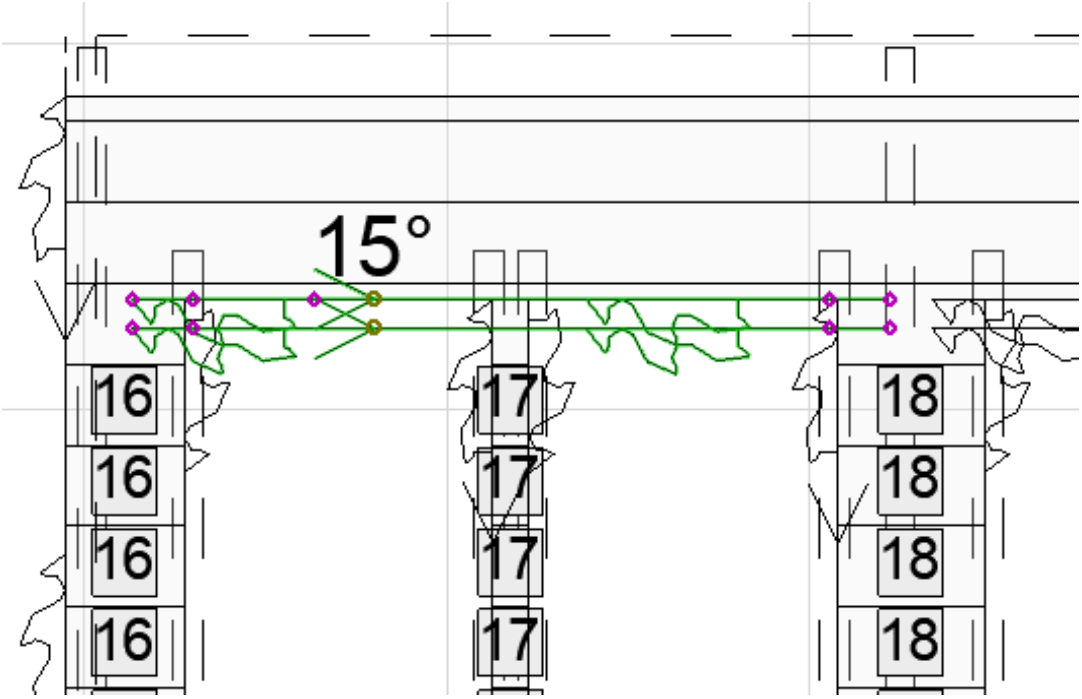


- Movable hotspots to drag the ends

To change blade side if automation fails or for special cases, the ArchiFrameSawCut-object can be mirrored. Deleting the saw cut removes the saw cut from the resulting wup-file.

ArchiFrame will automatically switch saw blade to router if the cut is undercut in both/other end and the cut is shorter than specified in saw blade default settings in ArchiFrameElements.xml <sawcut **routermaxlen="0.3"**>.

It is possible to add any custom rules in a Lua-script when creating saw cuts. Contact us to get more information. An example of adding double and extended angled cuts above openings:



11.6.3.2 Panel/board cuts with ArchiFrameBoardPanel-objects

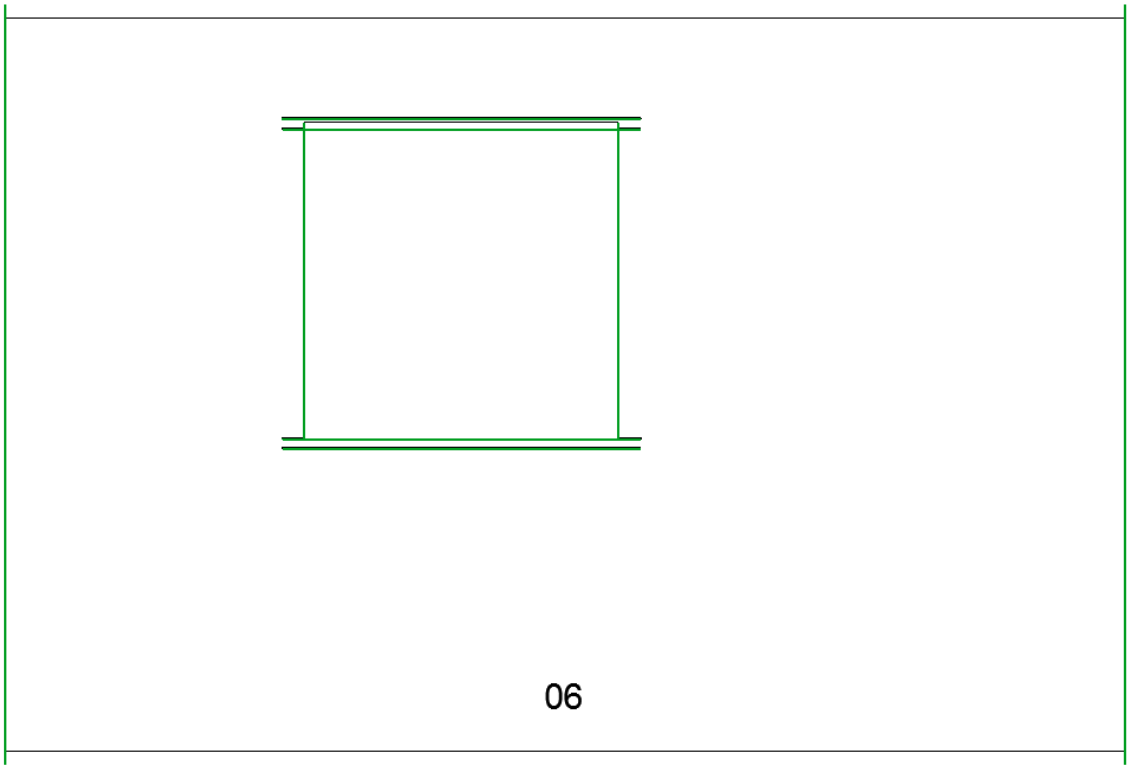
Panel cuts are used for cnc-production. When exploding paneling layer into separate planks, ArchiFrame leaves original ArchiFrameBoardPanel object in place and does following:

- Applies settings defined in ArchiFrameElements.xml path elem/settings/boardexploded. The default settings define that board object will not show the panel division any more, fill is changed to Air space and parameter iSawLines is set to value 15 that causes ArchiFrame to set cutting lines.

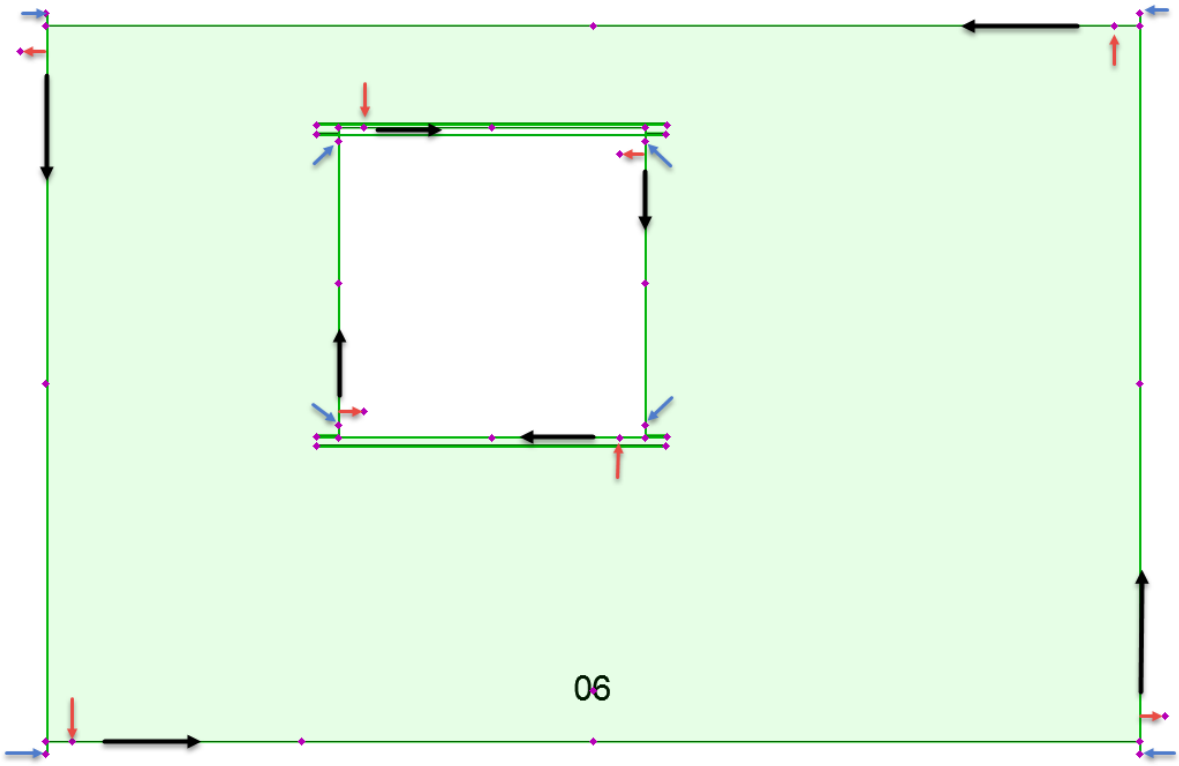
```
<boardexploded>
  <objparam name="iPanelBlock">4</objparam>
  <objparam name="iFill">Air space</objparam>
  <objparam name="iSawLines">15</objparam>
</boardexploded>
```

- There may be manufacturer specific processings in the *data*-folder that creates more cuts.
- Leaves the original board object(s) selected. The user can delete these objects by pressing Del after exploding.

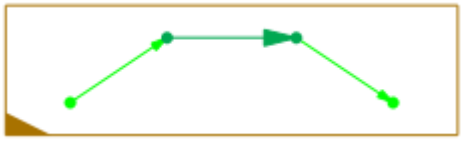
The image below shows the automatic cutting lines in green. ArchiFrame defines only the edges that are not parallel to paneling as cutting lines (in this case the paneling is horizontal). All opening edges will be cut lines even if parallel to the paneling. This example contains also manufacturer specific cutting lines for opening top and bottom:



The same with movable hotspots:



- Black arrows show the polygon edge lines direction and begin of the edge. Contour line is counter clockwise and holes are clockwise. Outside is always to right the hand side of the edge.
- Red arrows define whether the edge will be cutting edge or not: Having the hotspot on the line means no cut, having it to the right-hand side of the edge means that there is a cut. This hotspot is always 10 cm from begin of the edge.
- Blue arrows define whether the end of the cut edge should be overcut or undercut. Overcut means that the board/paneling cutting tool (saw blade or a small cutter/router) should travel over the corner to make sure that the cut is complete. For example undercut:

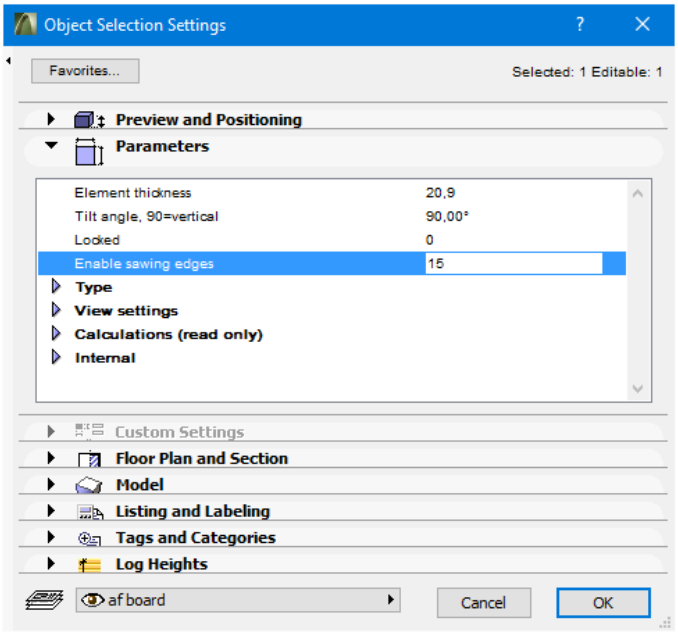
  

  And overcut:

- Other unmarked hotspots are to define the paneling anchor point and the polygon edges. The manufacturer specific special cuts also have editable hotspots.

These cuts are initially handled by ArchiFrame automation so you cannot edit those. To make the cuts editable there is parameter *Enable sawing edges* in the ArchiFrameBoardPanel-object:



It is sum of values:

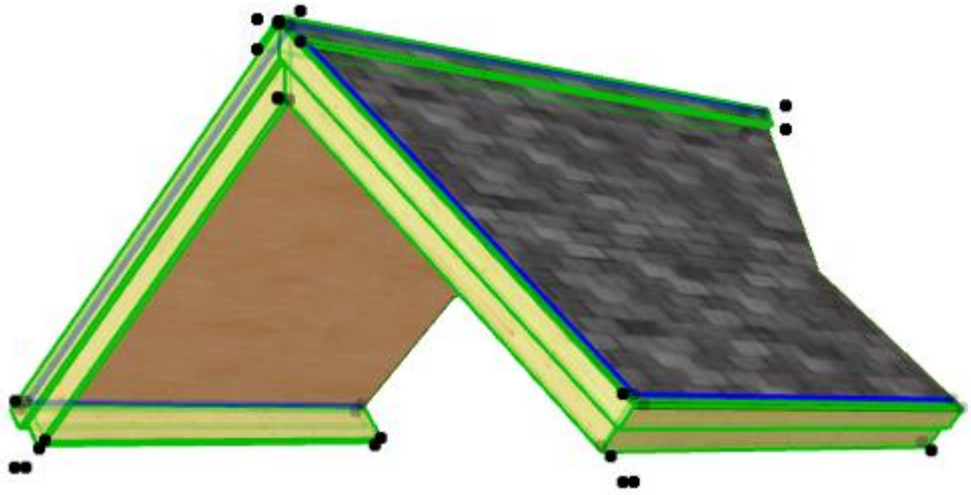
- 1, let ArchiFrame manage the polygon edge cutting definitions.
- 2, let manufacturer specific script manage the special cuts.
- 4, reserved for future.
- 8, reserved for future.
- 256, enable cuts even if no automation is wanted.

Initial value 1+2+4+8 (15) enables all automation for the cut lines. To continue from initial cuts manually the parameter *Enable sawing edges* is changed to value 256. Now the user has full control over the cuts.

12 Fascia tool

The related video here.

The fascia tool is made to help placing this kind of fascia boards semi-automatically:



The fascia boards must be modelled with Archicad profile beam tool. Also, before using ArchiFrame's fascia tool the fascia profiles must be defined already preferably in the start-up template.

The process of using the fascia tool is:

- Model the first roof manually
- Select the ready roofs and the related fascia beams
- Teach those to ArchiFrame
- Apply taught fascias to new roofs

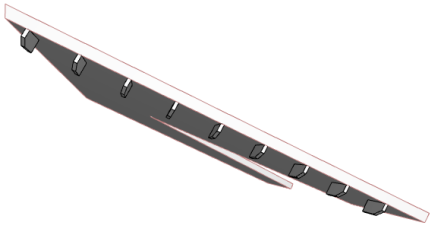
13 Modal dialogs

13.1 Multiply dialog

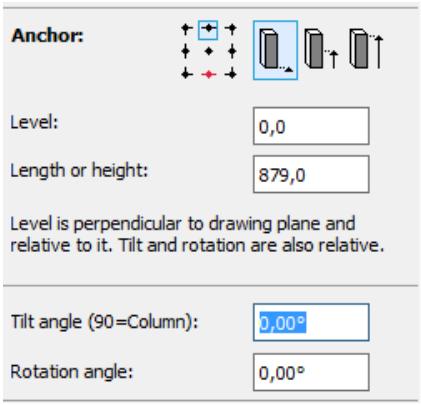
This dialog gives two extra features in addition to Archicad's standard multiply-dialog:

- It follows ArchiFrame drawing plane if it is set.
- It is possible to define maximum distance between pieces and divide those with similar spacing on pointed line.s

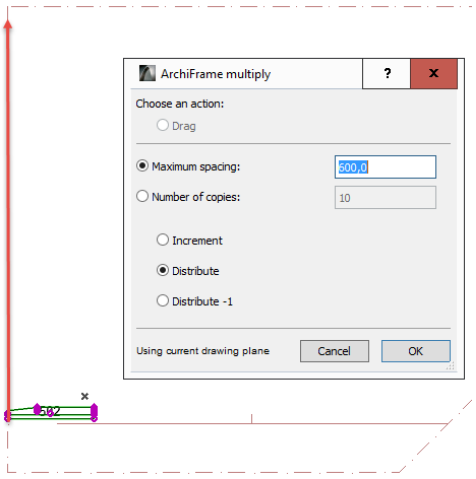
For example, to get the joists evenly spaced like this:



1. Pick drawing plane from the bottom of the roof.
2. Place the first joist setting the anchor point to middle top and set level, tilt angle and rotation angle to zero (these are relative to the drawing plane):

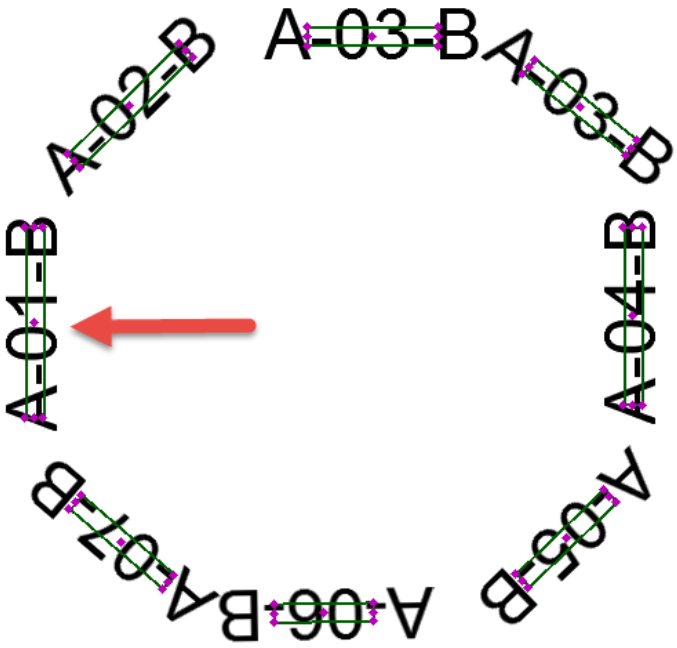


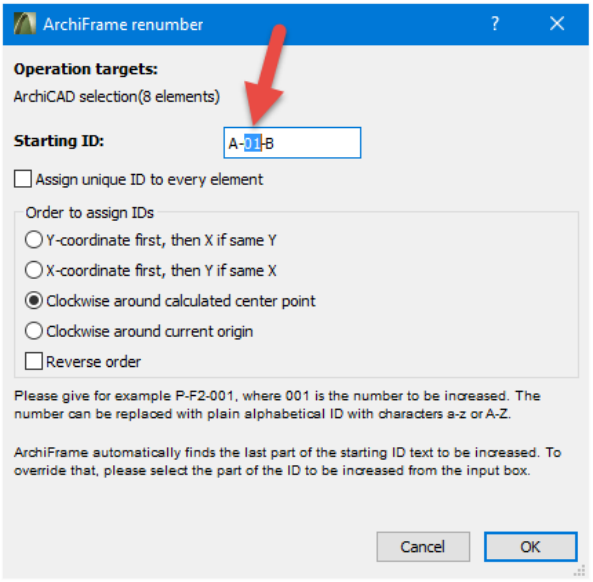
3. Select the first joist, open *Multiply* dialog and adjust settings like below. Finally show the line where the joists should be placed:



13.2 Assigining IDs

This dialog is used to assign IDs to ArchiFrame planks and element objects:

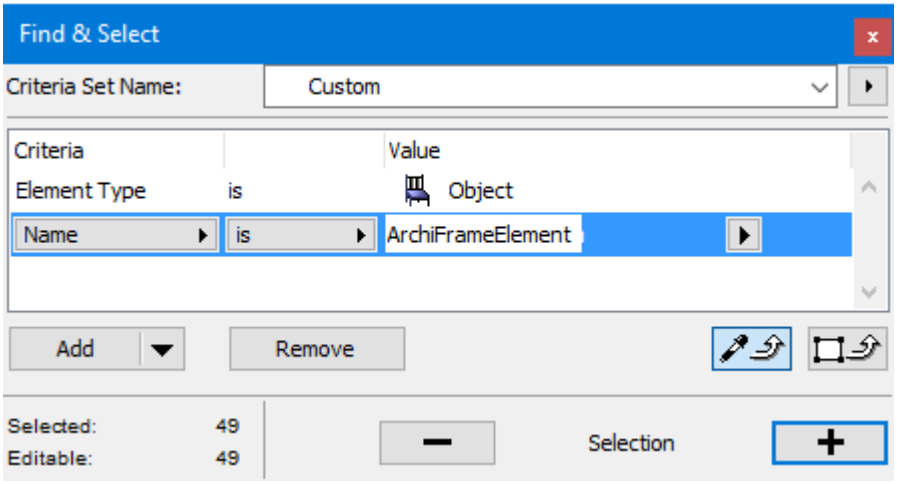




It is possible to define the part of the ID to be changed by selecting it before closing the dialog with OK. Also, any target plank's object parameter can be referred with syntax: A-01**[iElemModule]**

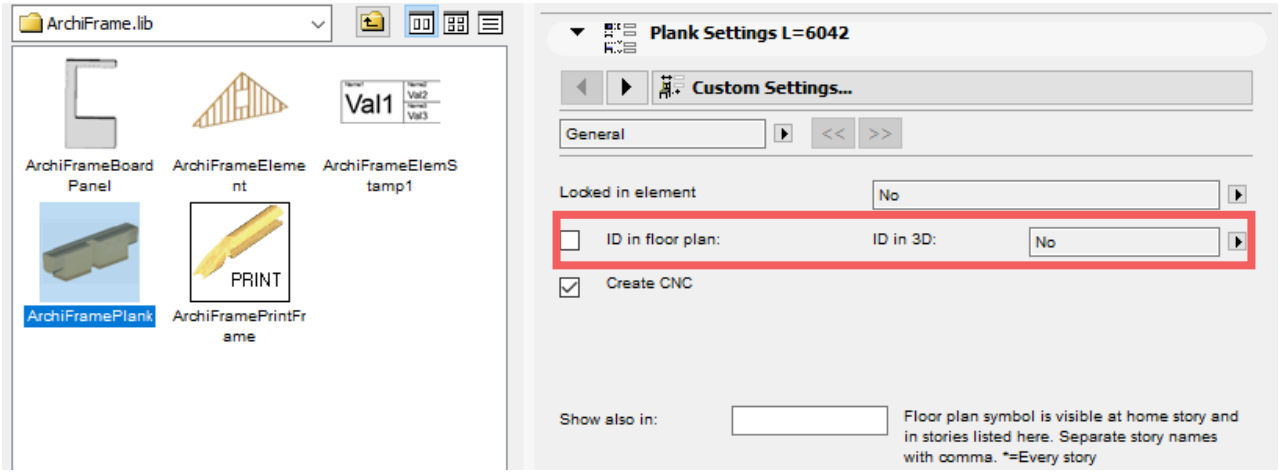
The last item selected will be the first one to get assigned ID. For element objects following procedure is helpful:

- Use Archicad Find & Select to select all element objects:

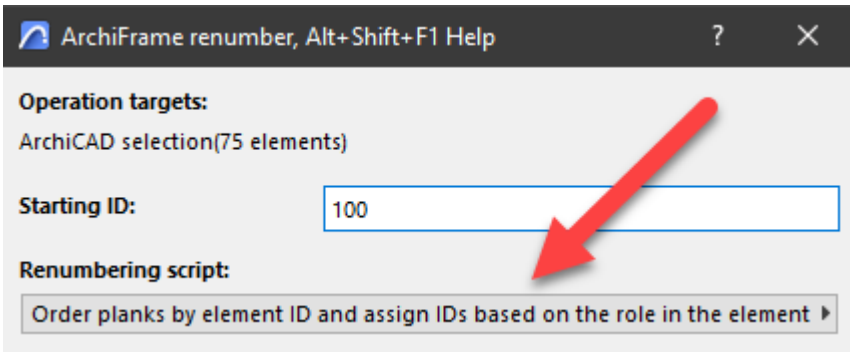


- Make sure that suspend groups is off, then de-select and select the first item again to be assigned shift-clicking it. Please note that selecting with rectangle will not work in this case.

In case the IDs don't appear in the floor plan or 3D view, go to the object's settings and make sure ID display is turned on. Object settings can be accessed by selecting the object and opening the Archicad object tool (or pressing Ctrl + T).



It is also possible to write a Lua script containing custom rules to assign IDs:



When the renumbering dialog is opened, ArchiFrame scans the base and user specific data-folders for all files matching ArchiFrameRenum*.lua. This script has two fixed name functions to integrate with ArchiFrame:

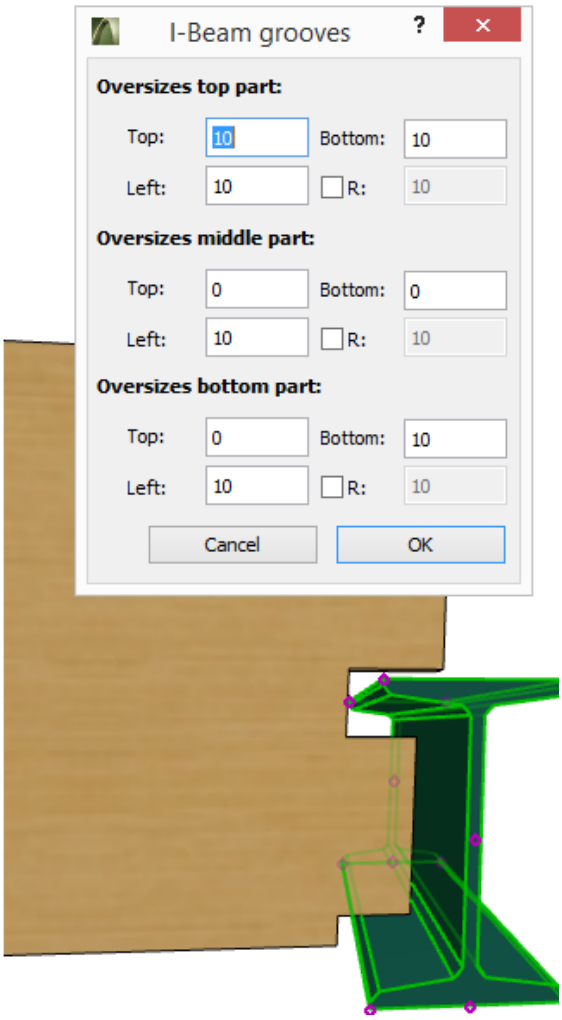
```
function RenumGetUI()
function RenumDo(tblGroupedElems, tblSettings)
```

RenumGetUI() is called to get the name of the numbering rule into the user interface and RenumDo() when user selects OK from the renumbering dialog. ArchiFrame has grouped similar elements into parameter tblGroupedElems unless setting *Assign unique ID to every* element is checked. Anyway, the script has freedom to do anything with the passed elements.

Please open an example script ArchiFrameRenumByPlankRoleSortByElemId.lua from the data-folder for more information.

13.3 I-beam grooves

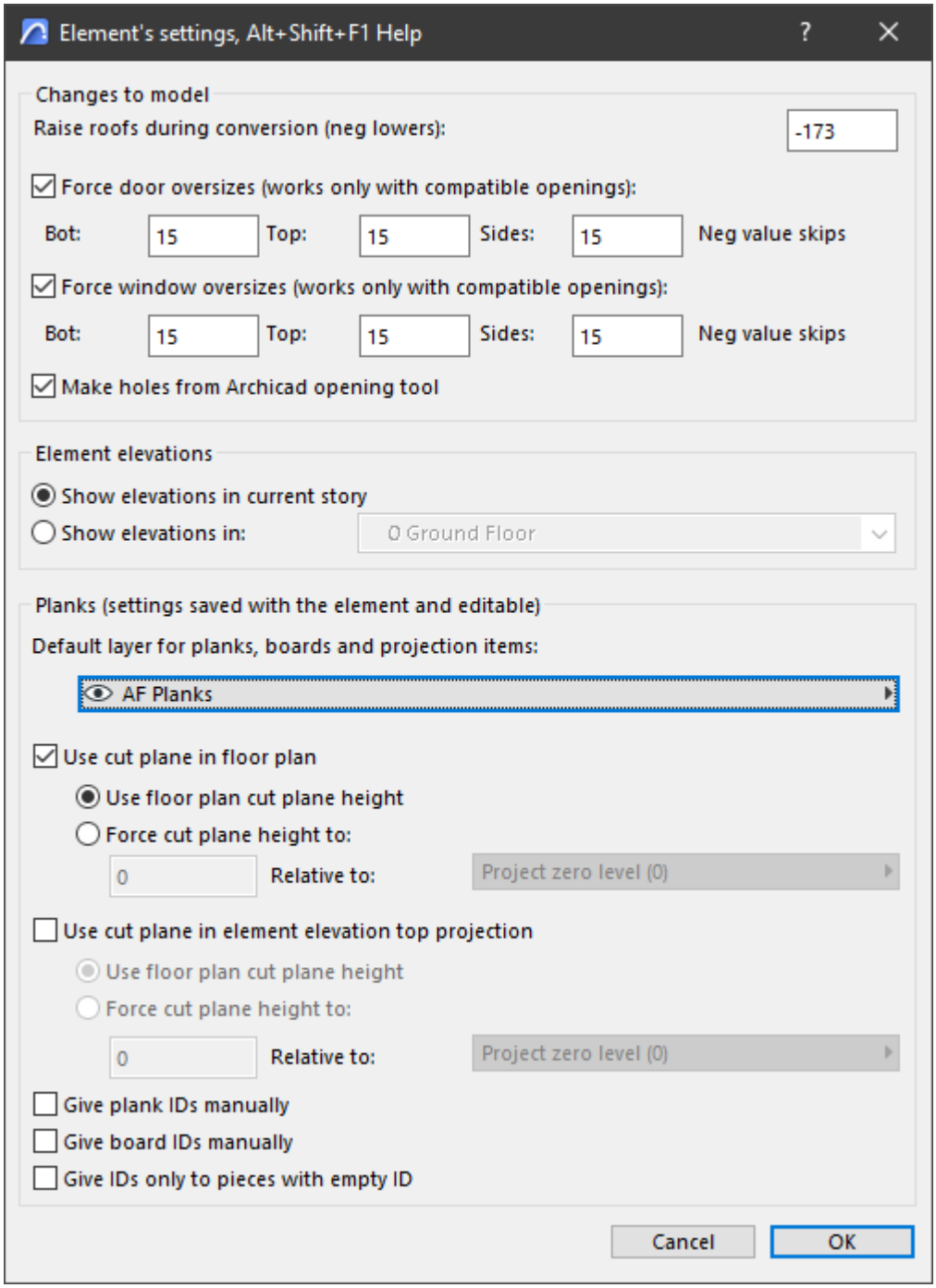
This dialog is used to make three grooves with three set of oversize-settings:



These oversize-settings are added to the main groove tool oversize-settings.

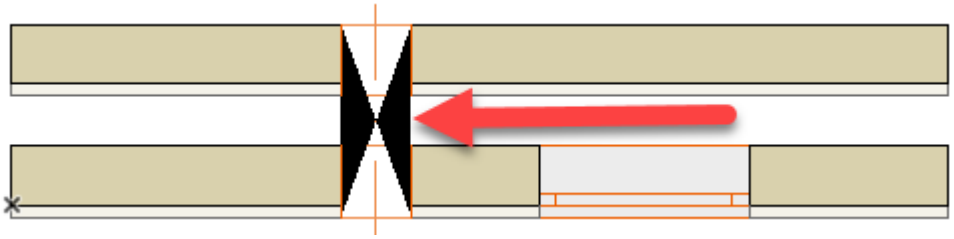
See also presets.

13.4 Element settings dialog



- *Raise roofs* lifts roofs temporarily during the element creation, perpendicular to the roof slope.
- *Force oversizes* sets oversizes to doors and windows that follow Archicad standards. Some openings may use the standard parameters in their own way – it is important to verify that forcing the oversizes actually works with used library. Forced oversizes are set to library part parameters:
    - xl, xr, yu, yl: Used in some old openings, xl = x left/left side, xr = right side, yu = upper, yl = lower.
    - gs_left_oversize, gs_right_oversize, gs_upper_oversize, gs_lower_oversize.
    - ac_left_oversize, ac_right_oversize, ac_upper_oversize, ac_lower_oversize.

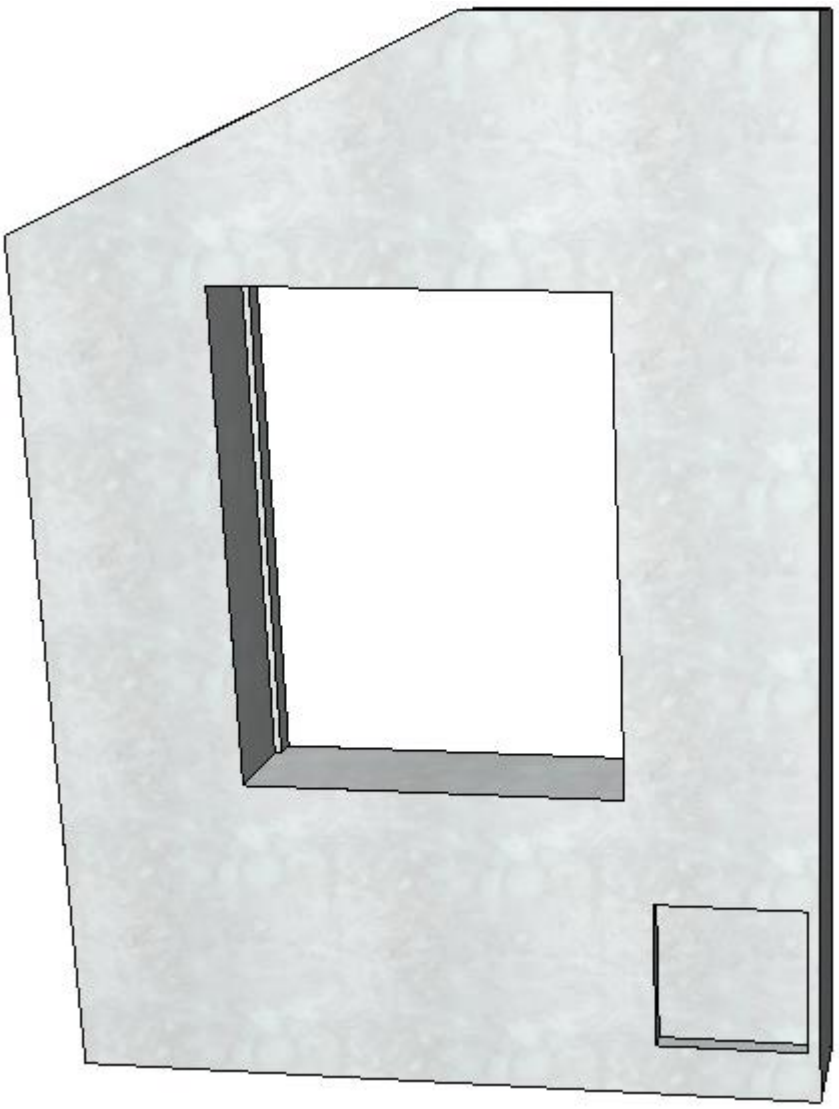
- *Make holes from Archicad opening tool* is by default checked. Unchecking it allows you to skip any hole generated by an opening element:



This is useful for example if exporting drillings and other cuts to CLT boards as IFC and the openings are used as cutting operators in the final CAD-software driving the CLT-production.

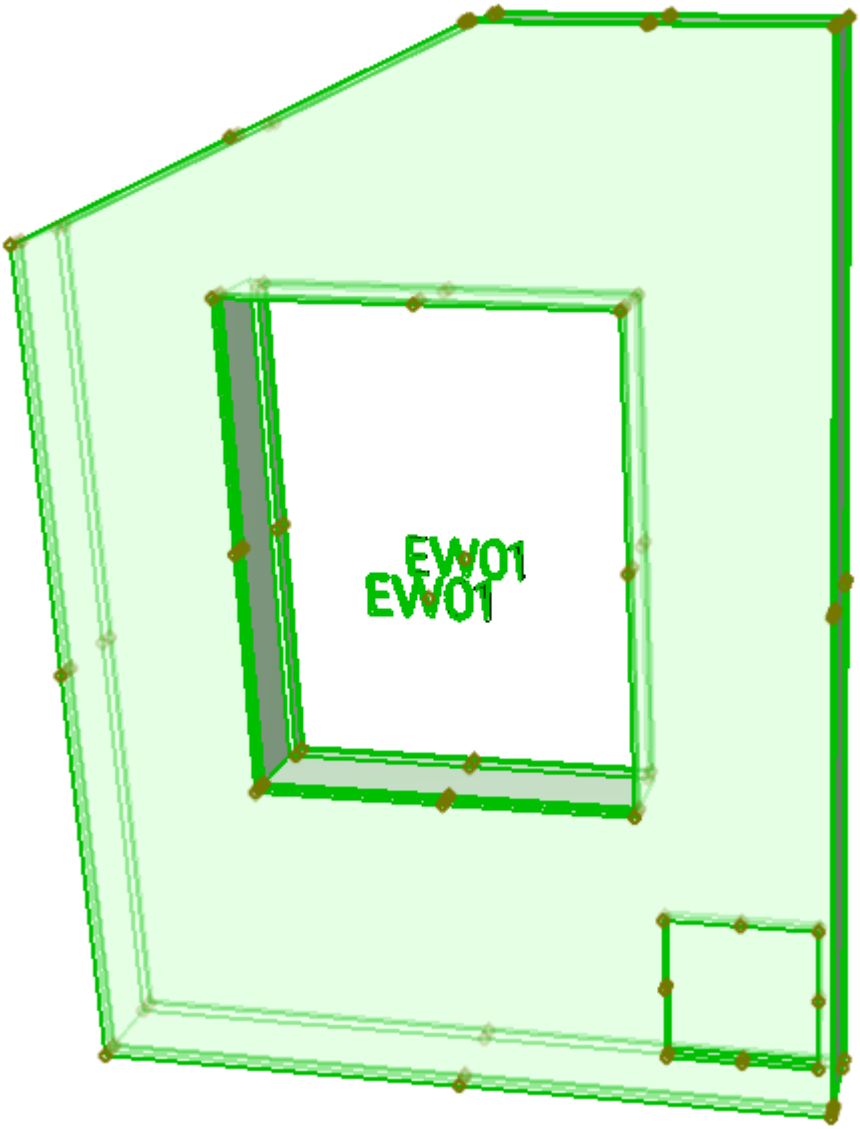
- *Take projections per element layer for newly created elements* will take max 10 mm thick section from the center of each composite layer instead of elevation kind of infinite projection of the whole source element(s). For example, having the option checked the following wall with opening at the right-hand side:





Will produce an ArchiFrameElement-composite having two first layers cut by the Archicad-

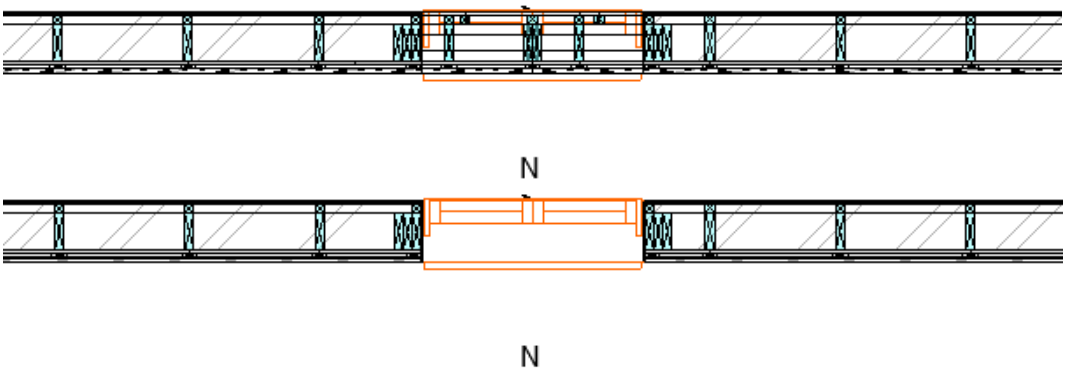
opening:



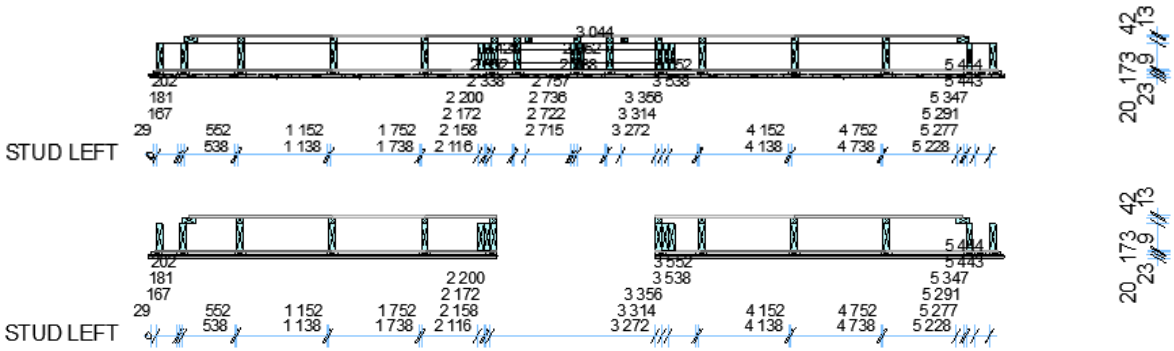
- Element elevations are perhaps best to be shown in separate storey with settings from *Element elevations*.
- Settings under planks-group can be edited for existing pieces. The changes affect all pieces belonging to selected element(s). Usually it is best to have suspend groups off and select every layer of multi-layered element and change the settings for all.
- *Default layer for planks, boards and projection items* is used when initially creating planks or regenerating projections under element tool's *Update*-button. This setting will not edit

existing pieces since it is possible to define different Archicad layers for each layer in the Custom Element Layer Settings dialog.

- Use cut plane in floor plan off and on makes this change (also the original Archicad wall is visible here):

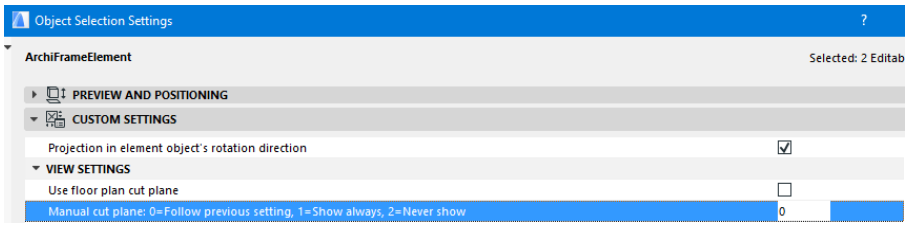


- In top projection the diffence is like below (the element must be updated with *Update*-button to get the dimension line(s) updated):

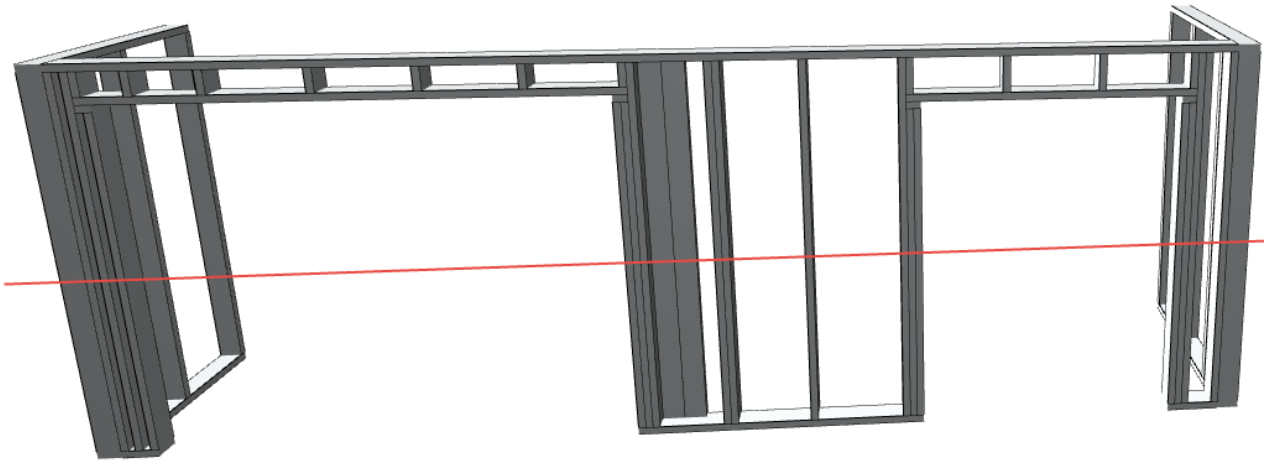


- *Give plank IDs manually* prevents ArchiFrame from assigning IDs to planks in elements when updating elements. This is useful to have same ID for similar planks in the whole building. But it requires that IDs are assigned manually just before creating end results like listings, cnc-file and layouts.
- *Give board IDs manually* does the same for boards.
- *Give IDs only to pieces with empty ID* is useful if there are changes after the prints have gone to production. The process of adding new planks using this option is:
    - Add new pieces and make sure their ID is cleared.
    - *Give IDs* and *Update* to assign IDs just to new planks and to update the cut list and dimension lines.

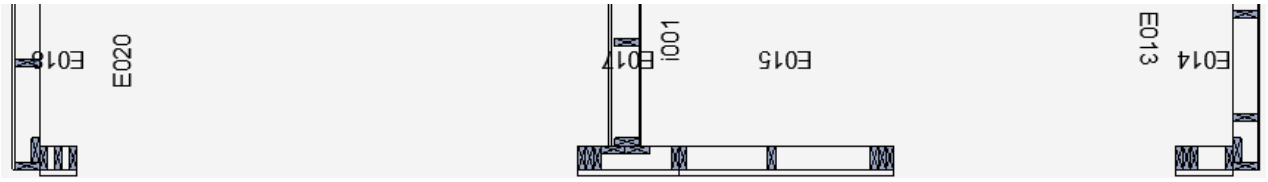
It is also possible to override the element specific cut plane setting for any element, plank or board object. They all have this setting:



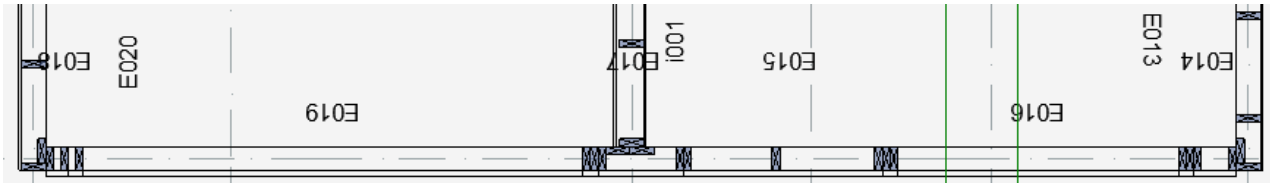
For example, in a case like this (red line shows the cut plane height):



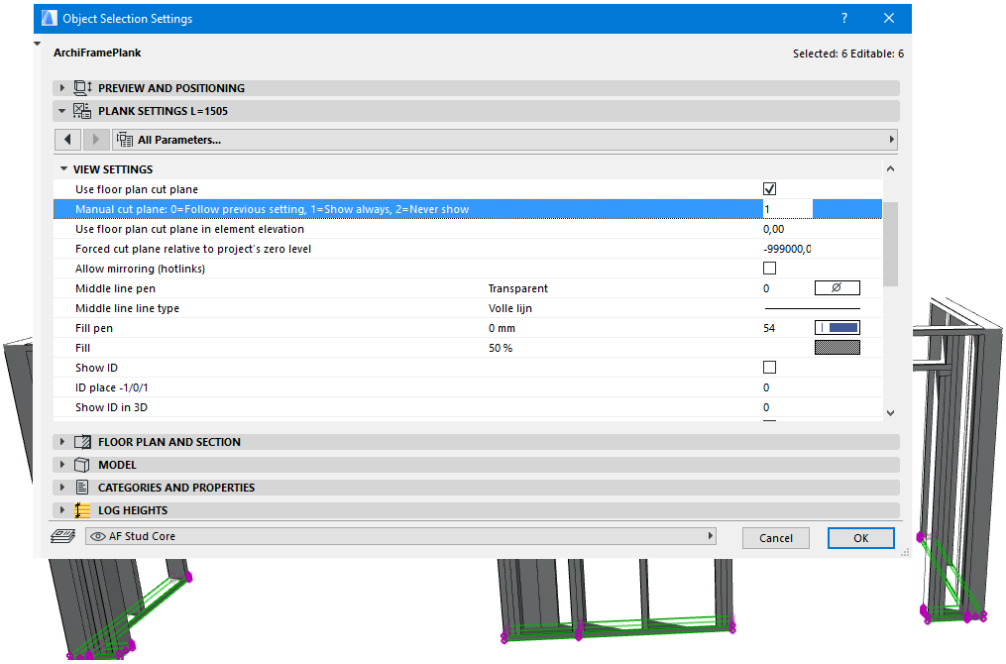
Using the cut plane, structure above the opening would remain hidden in the floor plan like this:



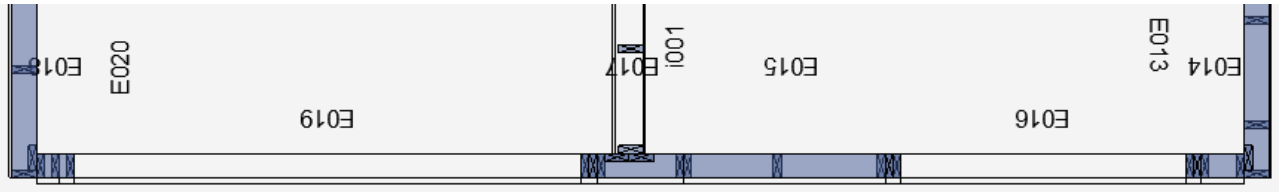
To have at least the element object visible in the floor plan *Manual cut plane* is set to value 1 for the element object and the floor plan looks like:



Another additional option is to set fill to bottom plates and force those visible also in floor plan:

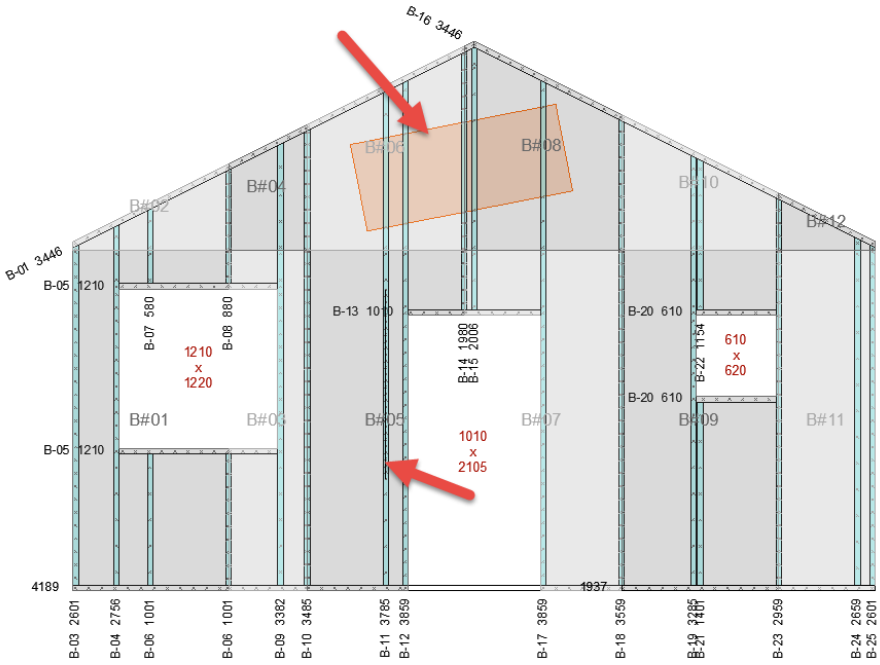


In floor plan it will look like this:



13.5 Element nailings dialog

This operation places nailing/screw lines to the selected boards based on selected framing planks. Nailing the planks is also supported.



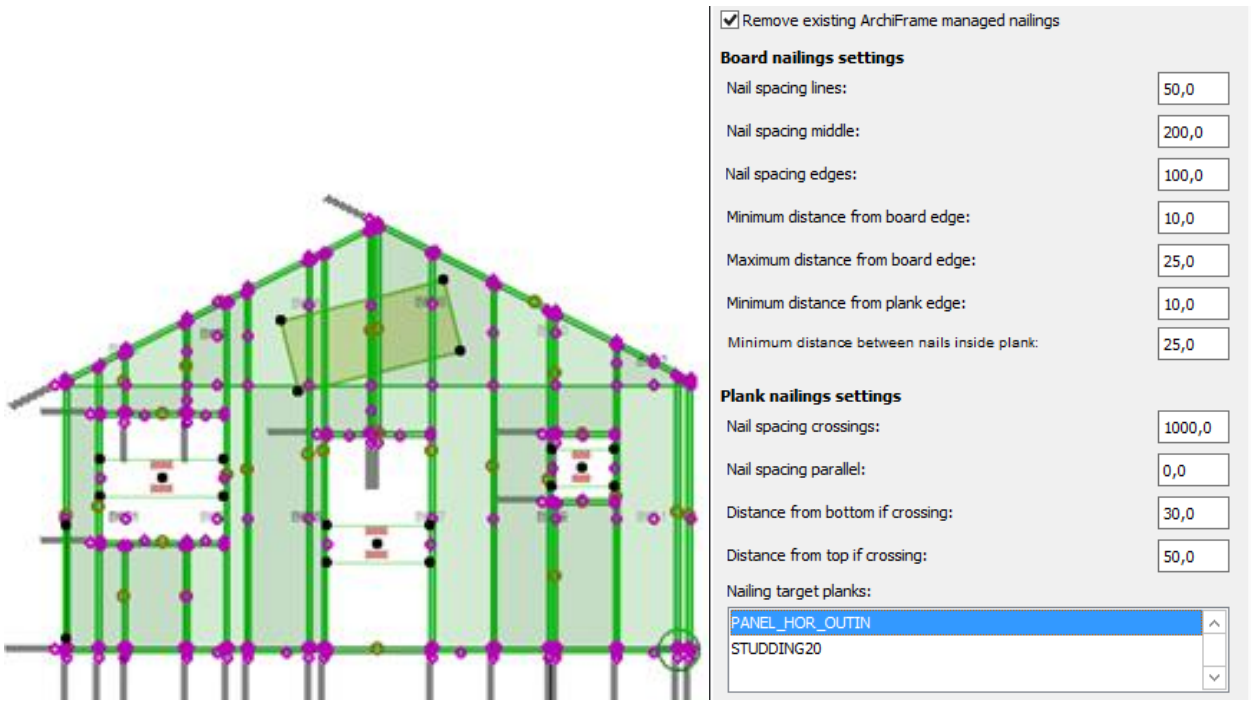
It is also possible to add ArchiFrameNailing-objects over the elevation to produce point nailints in any position. Please note that there must be a nailing target like a batten or stud behind the ArchiFrameNailing-object.

The process to add the nailings is to select all the target boards and related framing planks (usually everything in an elevation). If there are multiple studs next to each other, only the ones to have nailings to the boards should be selected. The selection may contain fills to exclude parts for nailing and lines to set specific type of nailings according to the lines. Also, selection may contain placed ArchiFrameNailing-objects to define individual point nailings. The nail gun number for nailing points can be forced in ArchiFrameNailing-object's settings. Only one layer should be done at a time since added nailings will be avoided when processing other layers if done for all layers at the same time.

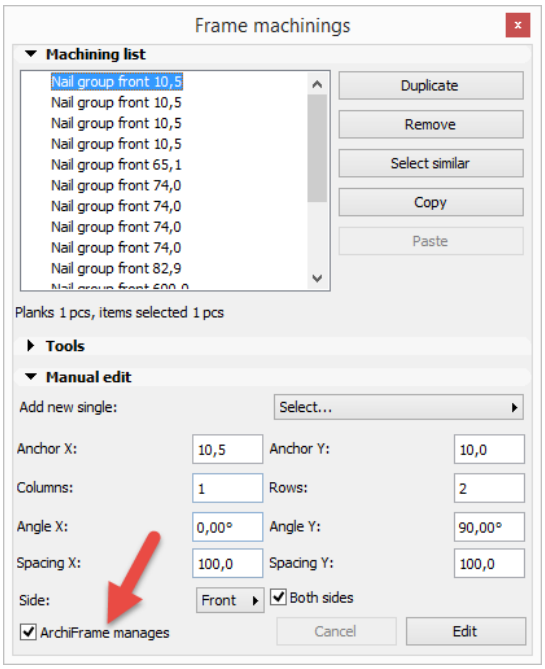
No nailings at the same place: Nailings are placed so that minimum distance with the nails in the same element group is this:



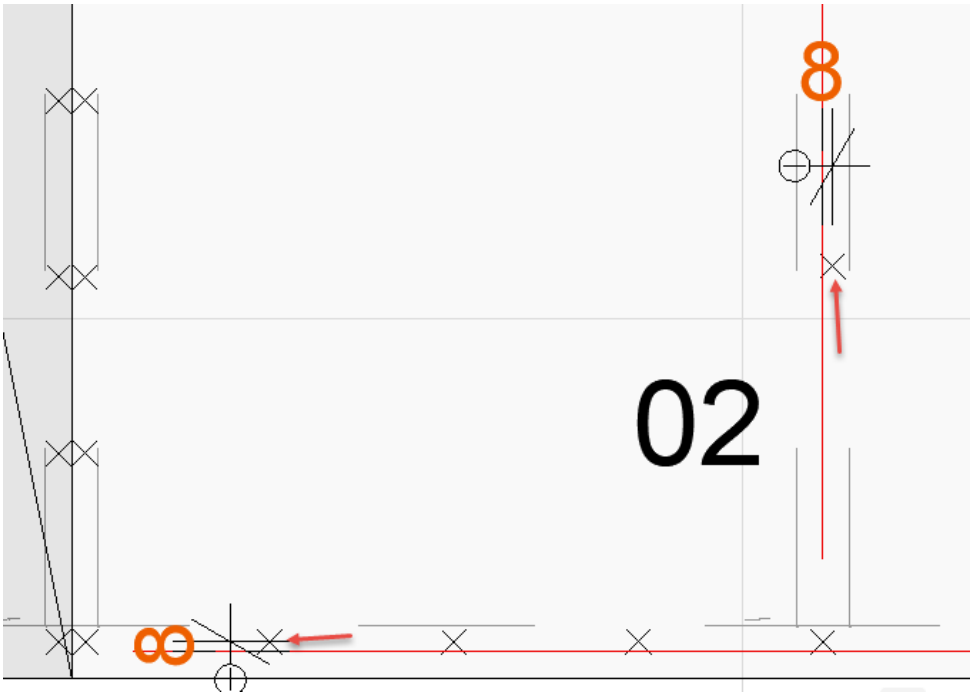
For example like this:



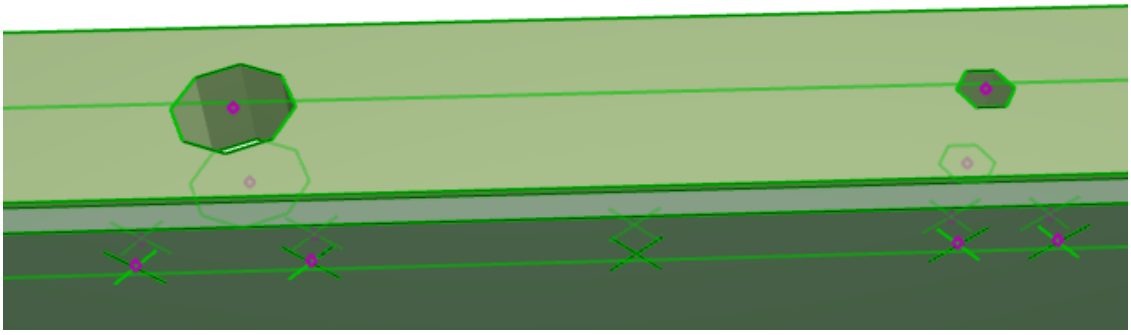
- *New* creates new preset
- *Duplicate* duplicates current settings for further editing.
- *Delete* deletes current item.
- *Load types* deletes all the presets and loads everything from an external file. This is very useful for example if there is a single person in the company taking care of the structures and shares the types to other users.
- *Add types* is as *Load types* except it does not delete current types. It merges the types from the file using any type having the same id from the given file.
- *Save as* saves all presets to an xml-file to be used for *Load types* or *Add types*. The file can be edited using text editor for example to limit types to import to another user.
- *Remove existing ArchiFrame managed nailings*, if checked all existing nailings having ArchiFrame manages on will be removed before adding new ones:



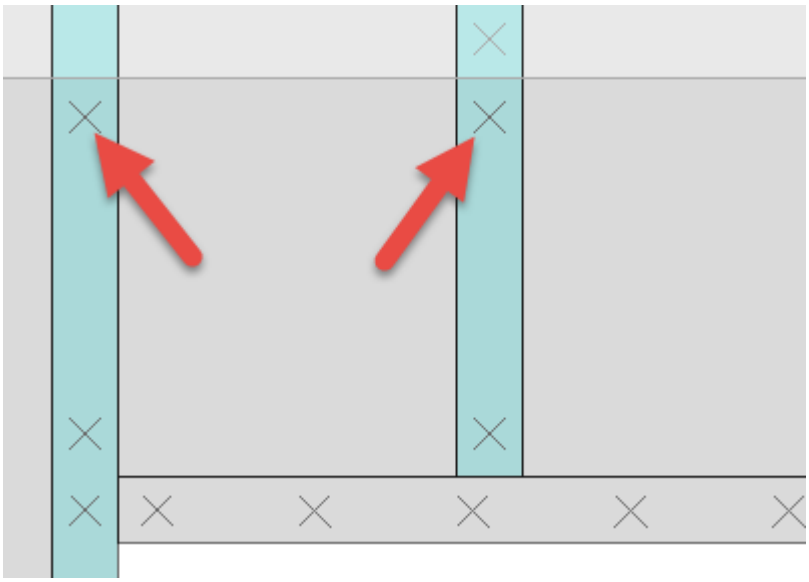
- *Minumum distance from plank edge* defines how close the nailing can be to the plank edge.
- *Minumum distance between nailings inside plank* is used to remove nailings, for example, if there are two edges at the same framing plank closer than this value. Also, this affects avoiding collision to existing nailings.
- *Maximum distance from nail target to framing* tells ArchiFrame to search for suitable framing pieces from selection no further than given value from the nail target surface.
- *Force nail gun number* sets a custom value to be used in CNC-output.
- *Max z-distance when searching existing nailing* defines the distance in nail's direction to limit searching of existing nailings. The measure is taken from nailing target surface. For example, this can be used when first the cladding nails are added to cladding with thickness 18 mm and airspace strips 32 mm (18+32=50 mm). When making nailings for the windshield boards, setting this value to 49 mm let's ArchiFrame to place board nailings ignoring cladding nailings.
- *No nails to first and last position* can be used when the airspace strips are nailed from the ends by CNC operator and the CNC-machine will shoot the remaining nails.
- *Move right if center of a stud* can be used if there are multiple boarding layers or air space strips and nailings should be on different lines. This value is used to move the nailing line given amount right from the nail target plank. For example, moving 8 mm for vertical piece:



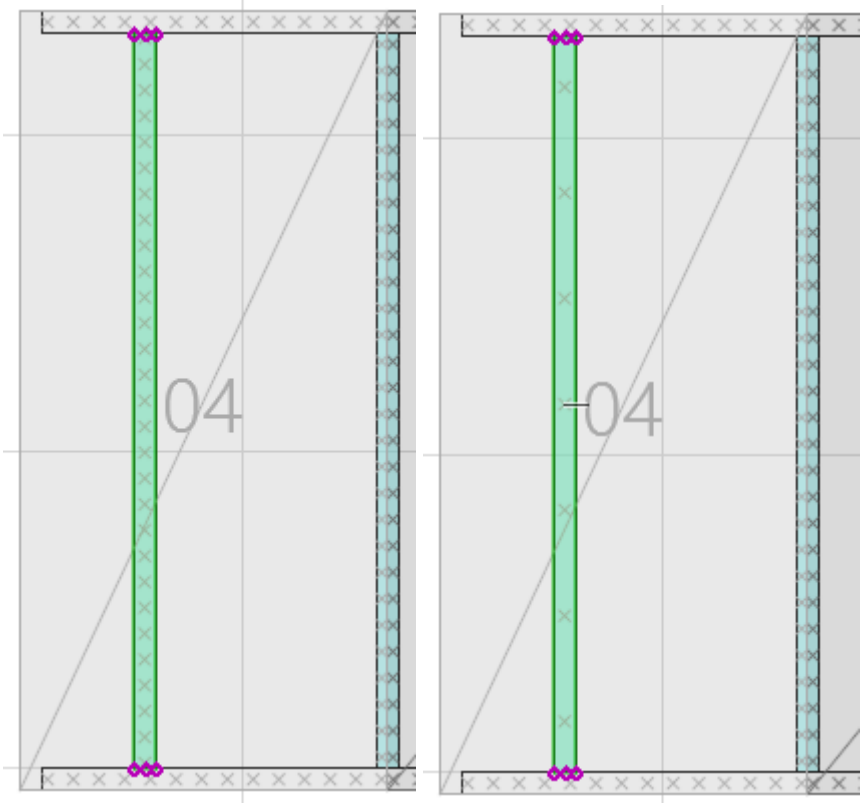
- *Min distance from drill or groove* defines the extra space left to both sides of a related machinings. Grooves are checked only from the surface towards the board and drillings for the side surfaces. An example of avoiding top plate drillings:



- *Nail spacing lines* defines the spacing used in the lines placed over the elevation. With value 0 no nalings are placed.
- *Nail spacing middle* defines the spacing used at the middle of the boards.
- *Nail spacing edges* defines the spacing used at the board edges including opening edges.
- *Minimum distance from board edge* defines no nailings are placed closer to the board edge.
- *Maximum distance from board edge* if there is a wide framing plank at the edge of the board, this value is used instead of the center line of the framing plank under the board. Also, the distance for nail from board edge if a framing plank is at middle of the board:

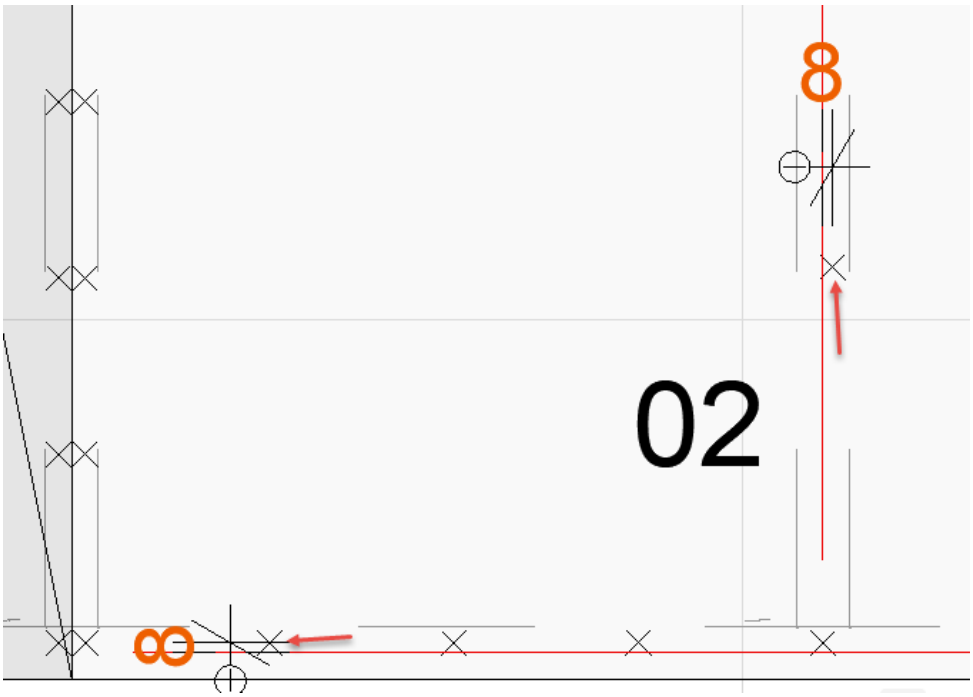


- *Extended edge spacing* will make ArchiFrame to use edge nail spacing for the furthest supporting pieces even if it is far from the board edge - checked it will produce result like this and right hand side without it:

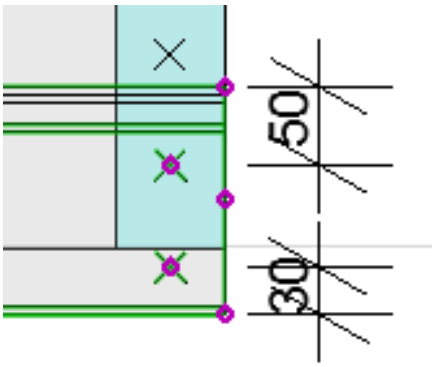


- *Move inside board if edge over 75% of target plank* can be used if there are multiple boarding layers or air space strips and nailings should be on different lines. This value is

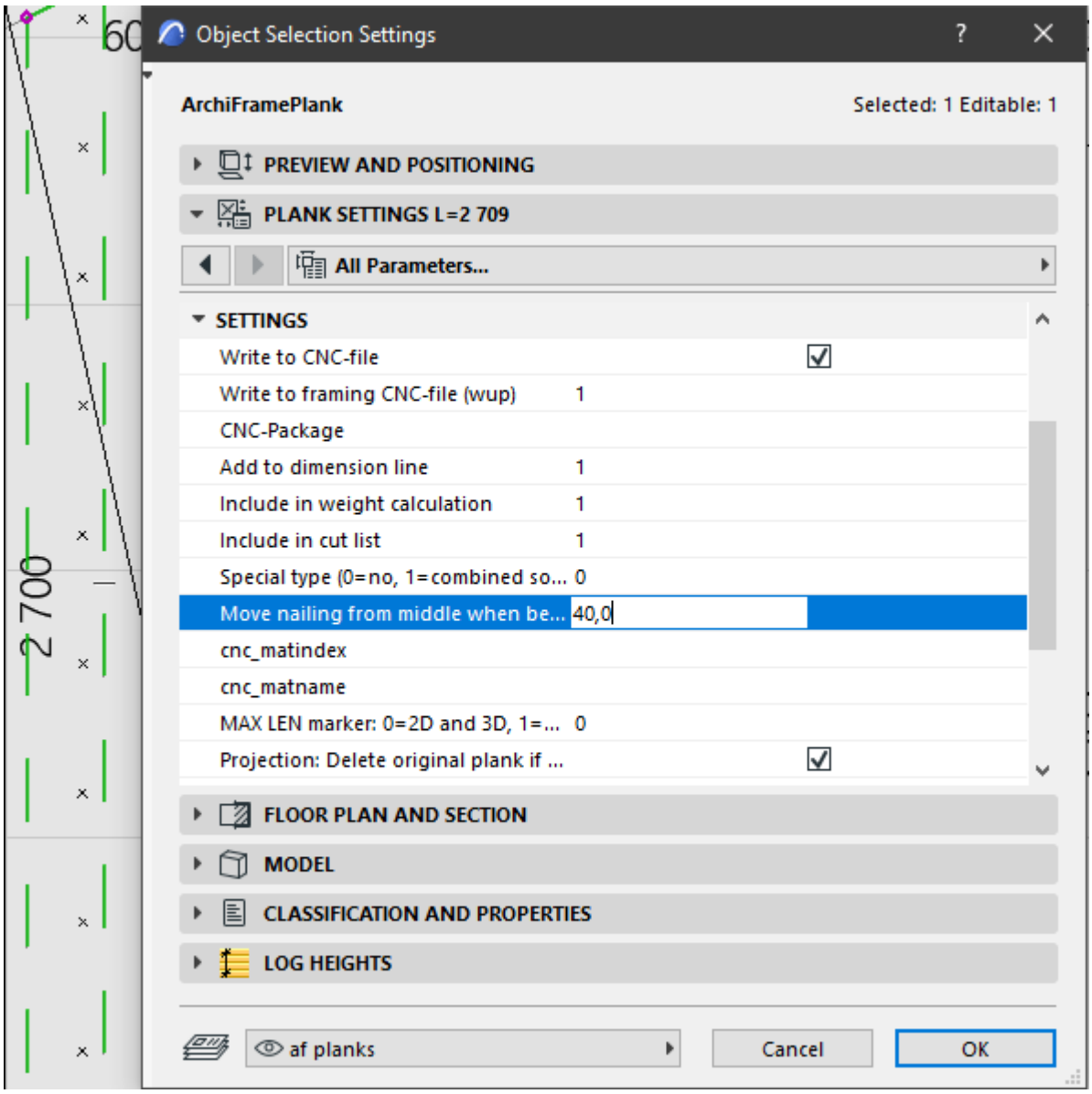
used to move the nailing line given amount right from the calculated nailing line which is in center of common area for the board and nail target plank. For example, moving 8 mm for horizontal piece, negative value will move towards board edge:



- *Nail spacing crossings* define the spacing used for crossing planks. If target plank is not wide enough to have two or more nailings, just one is placed at the plans' middle line intersection.
- *Nail spacing parallel* is the spacing used when the planks are parallel. With value 0 no nails are placed.
- *Nailing target planks* define the layer(s) to be nailing targets. It is possible to select many layers by pressing *Ctrl*-key when clicking the items.
- *Distance from bottom/top if crossing* is useful for panel/cladding nailings. Values given here will put single nailing given distance from the plank's top/bottom side. Note that single nailings are placed only if spacing crossing is bigger than the height of the piece. In the example the value is 1000 to do that. Value 0 means no special handling for the top or bottom.

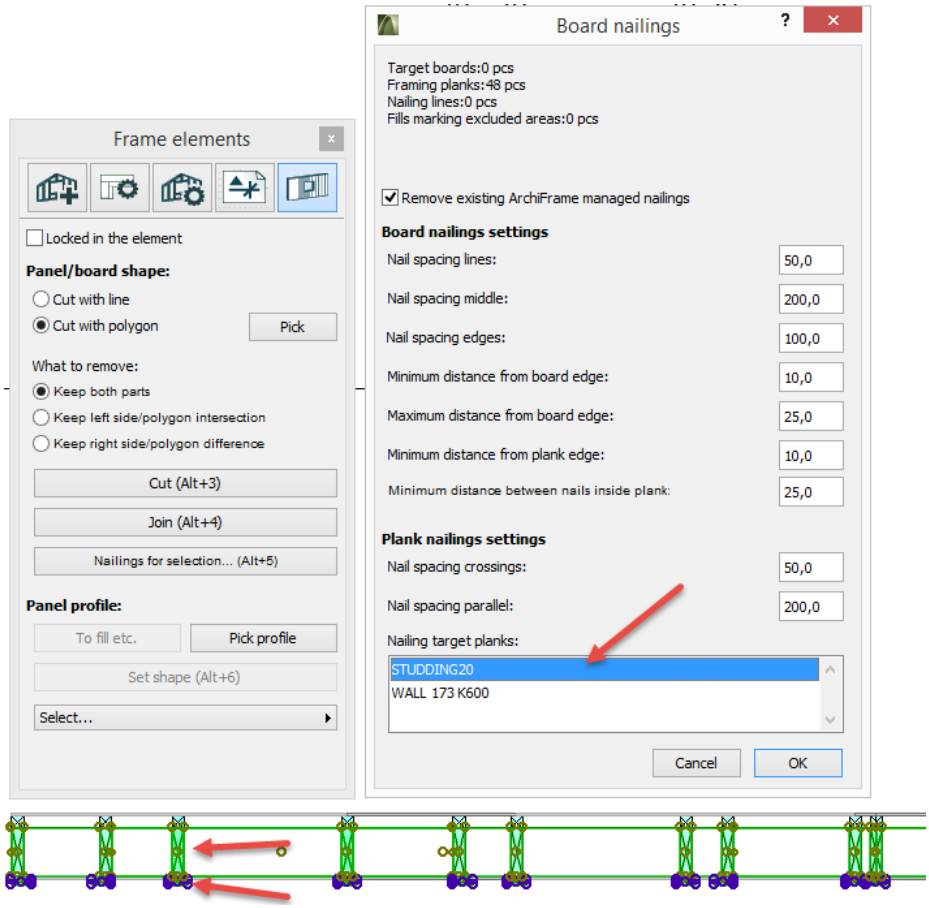


By default, the nail line comes to the middle of the plank. This behaviour can be changed from the plank that is behind the target piece. Positive moves to right and negative to left:



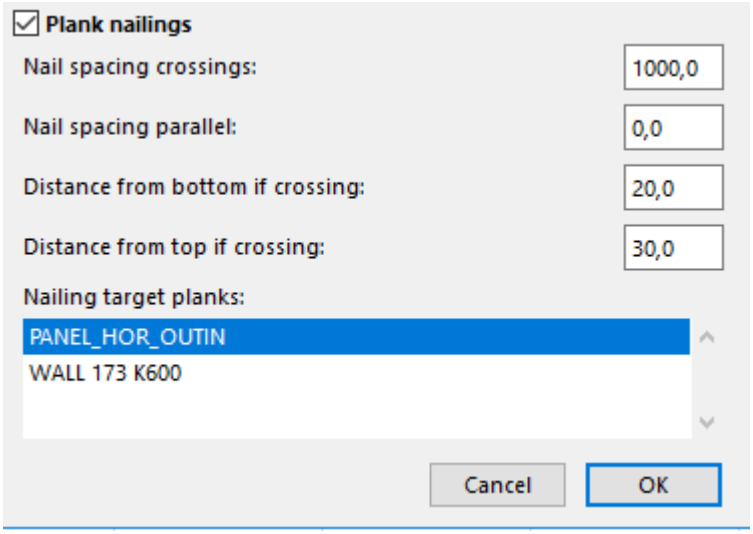
13.5.1 Example nailings for planks

Select two plank layers that should be nailed together:



Then select the nailing target layer.

For cladding it is possible to define nailing distances from cladding piece top & bottom. For example, to get nailing 20 mm from bottom and 30 mm from profile top:



Here *Nail spacing crossing* value 1000 mm indicates that we want nails to studding and cladding intersection. 1000 mm is bigger than cladding piece height, so it will not produce any nail. *Distance from bottom/top if crossing* is used instead – settings produce nailing 20 mm from bottom of the cladding and 30 mm from top of the cladding profile.

13.5.2 Nailing scripts

Nailing script can be added to add rules of what areas to exclude from the nailings. Also, additional nailing lines and points can be specified. Please see the example file ArchiFrameNailingNoBottomCladding.lua from the default data folder. All script-based definitions are relative to single plank and are in its local coordinates. The script has these global Lua-variables:

IN:
gtblSources, 1-based array of source elements each having fields:

- istarget (if flagged as target layer)
- typenum, 1=plank, 4=board
- guidmaster, guid of the 3D-model plank
- guidproj, nil=not started from projection

OUT:
gtblExclude (fills), item:

- ptr, related master plank ptr from plankinfo. Must be a nailing target.
- prio: UI/default=20
- side, 1-4 (currently not used, must be compatible with the nailing target, default=2/front)
- x1,y1,x2,y2 bounding box and currently the rectangle to exclude
- poly (reserved for future)

gtblLines, item:

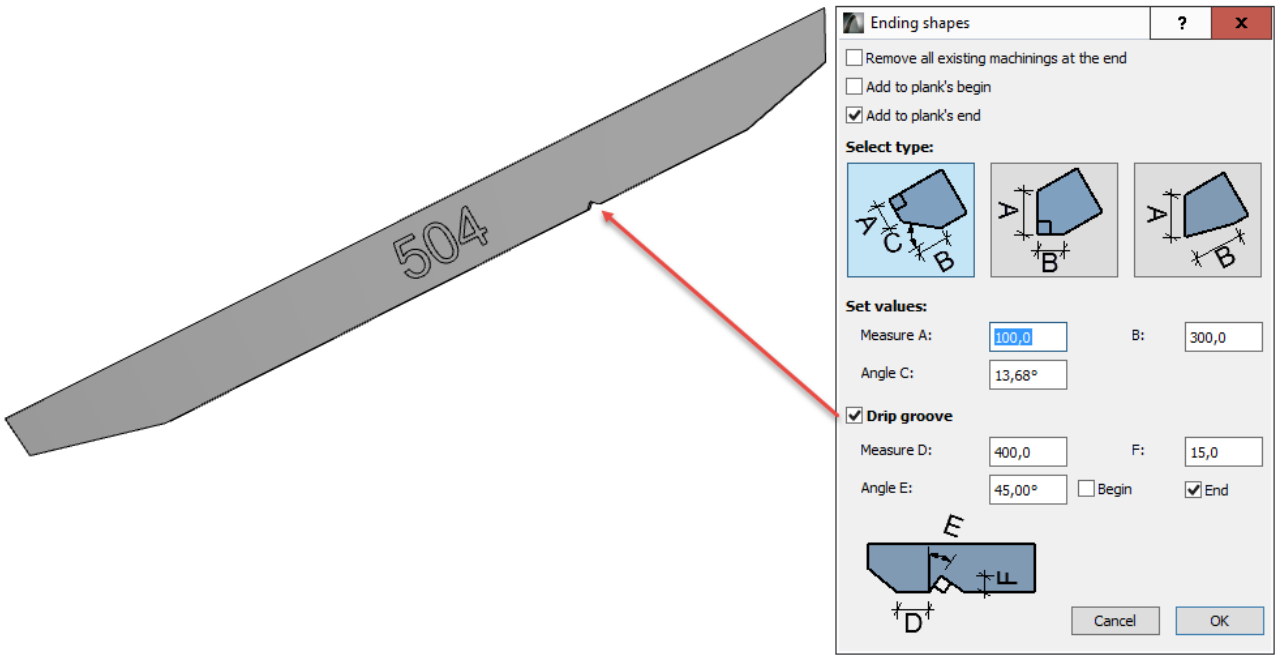
- ptr, related master plank ptr from plankinfo
- prio: UI/default=10
- side, 1-4
- spacing, nail spacing in meters that will not be exceeded
- x1,y1,x2,y2
- nailgun, missing or 0=default

gtblPoints, item:

- ptr, related master plank ptr from plankinfo
- prio: UI/default=10
- sidePlank
- x,y
- nailgun, missing or 0=default

13.6 Rafter endings

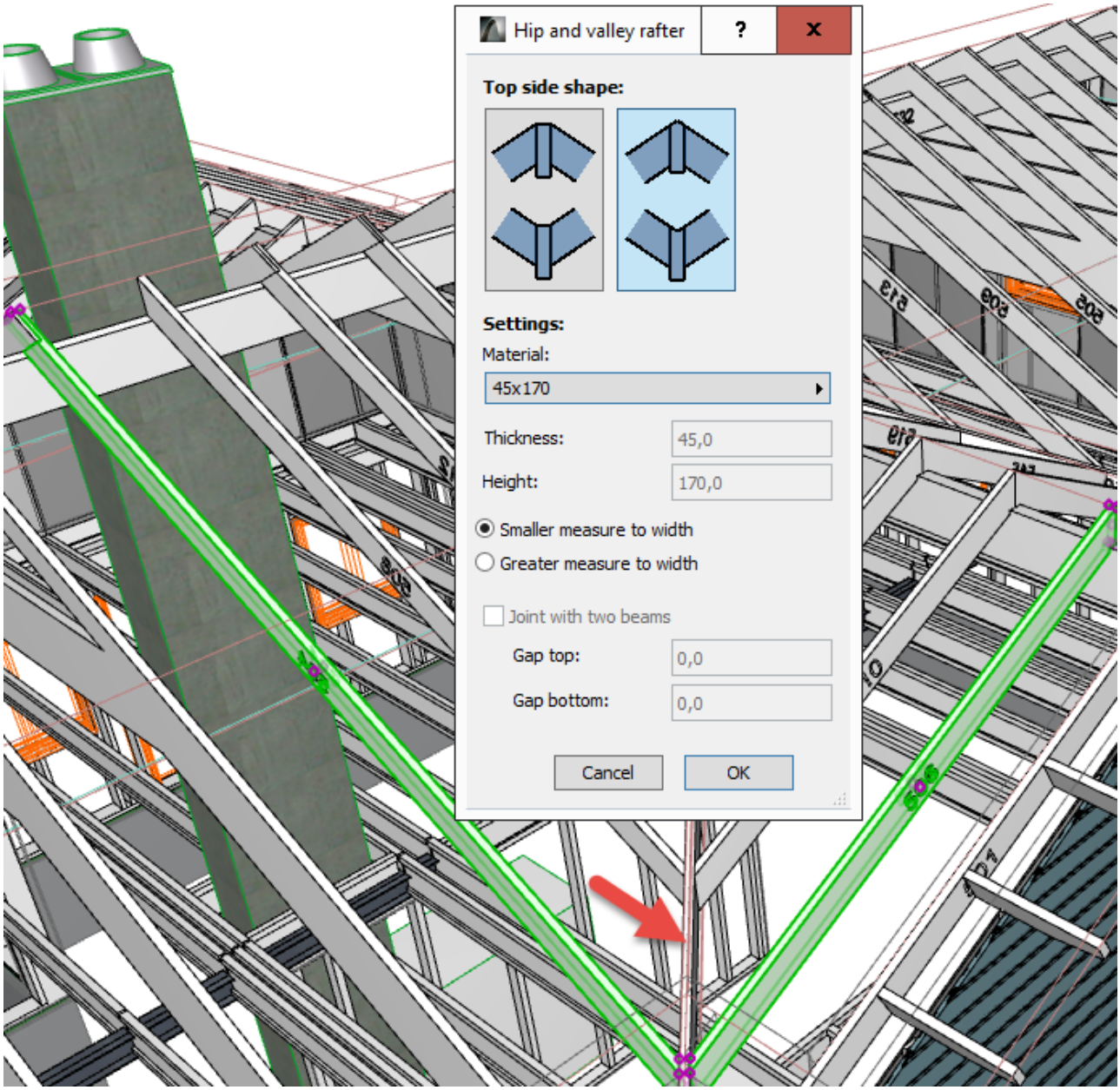
This dialog is used to to make typical endings for selected rafters.



- *Remove all existing machinings at the end* needs to be unchecked if adding different shapes to begin and end. The direction of the destination is shown by the ID text.
- *Add to plank's begin/end* define which end of the plank to set.
- *Drip groove* is handy at upper end of rafter if it is exposed to the weather and the rafter is made with cnc-.

13.7 Hip and valley rafter

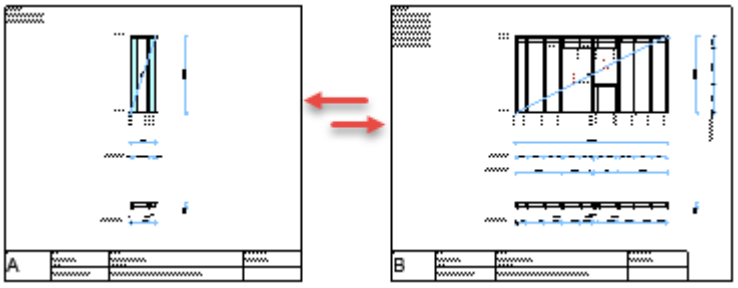
This tool helps placing these special rafters. To use it, select two existing pieces defining the two roof planes. ArchiFrame will calculate the position of the new piece.



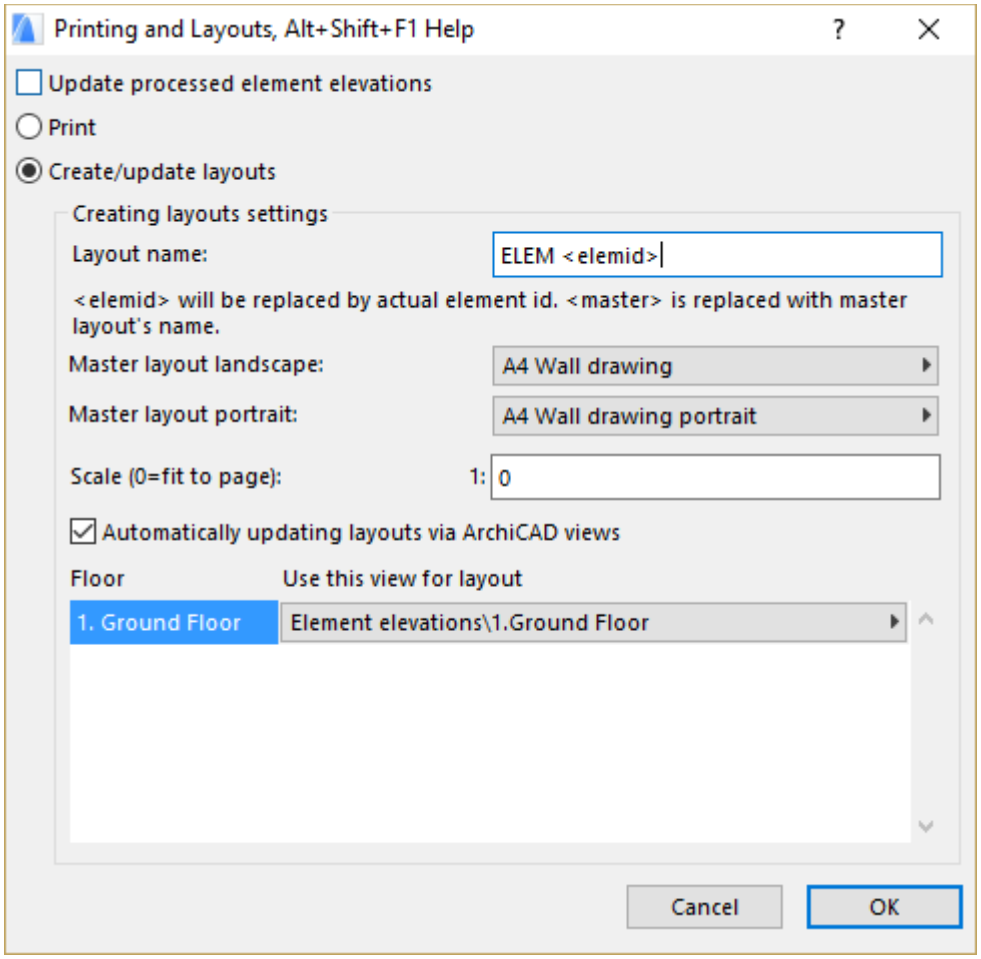
- Top side shape has two options: straight or V-cut. V-cut is useful if the planks are produced with cnc-.

13.8 Printing and layouts

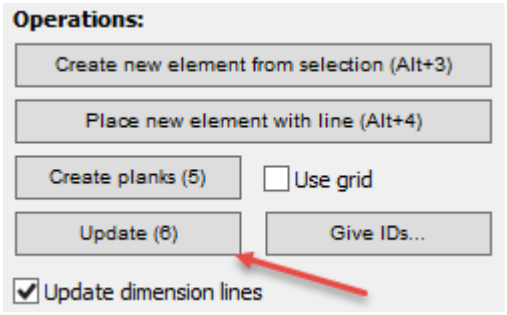
This tool handles areas in current storey marked by *ArchiFramePrintFrame*-objects:



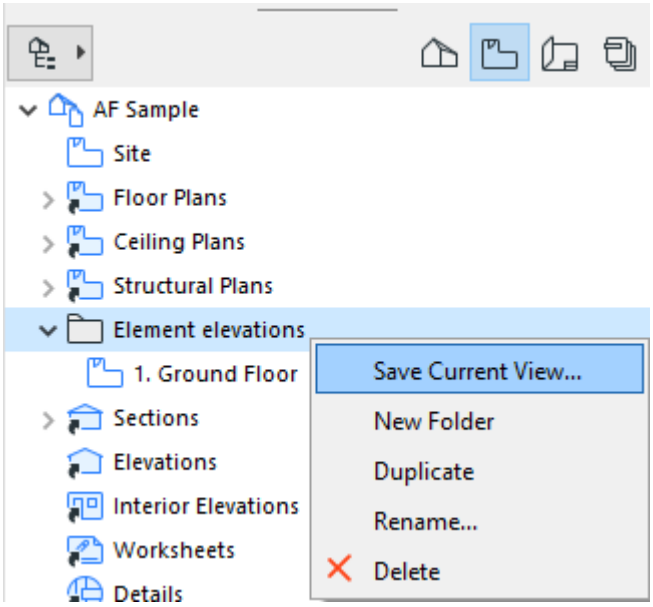
To limit the operation, select the target frame objects.



- *Update processed element elevations* does the same as Add & Edit element tool update:



- *Print* prints the target frames directly.
- *Create/update layouts* creates Archicad layouts and is the recommended way to handle printing.
- *Layout name* is the name in Archicad layout book.
- *Automatically updating layouts via Archicad views* will place a view on the layout and limit it to cover just the frame object. Due to technical limitation, there must be a view to do this. To create the views, make sure that at least the scale is correct (1:50) and needed layers visible. One way to organize the views is to create folder Element elevations (right click to the top-level item – AF sample in this example and select *New folder*). Then right click on Element elevation folder for each floor to save the views:

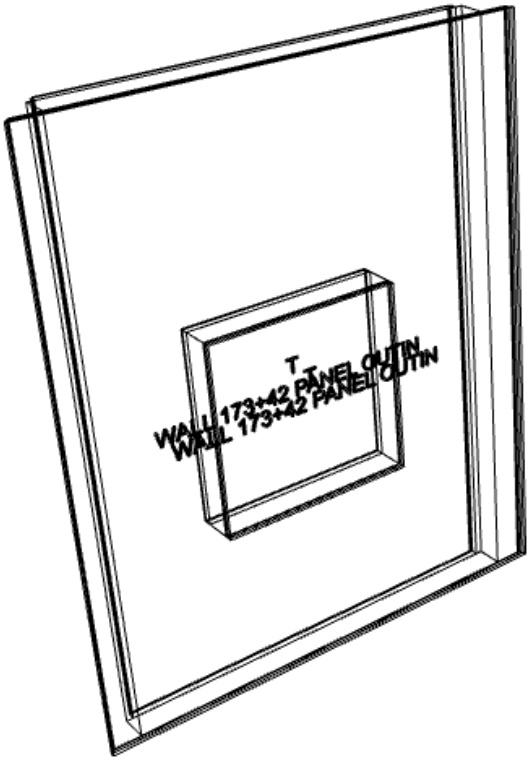


13.8.1 ArchiFrame Model View options

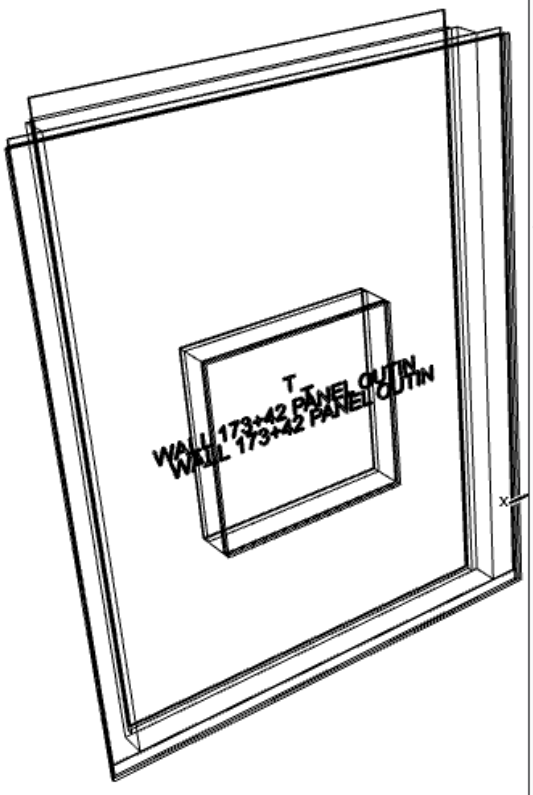
Please see here. These options are saved and applied by Archicad views.

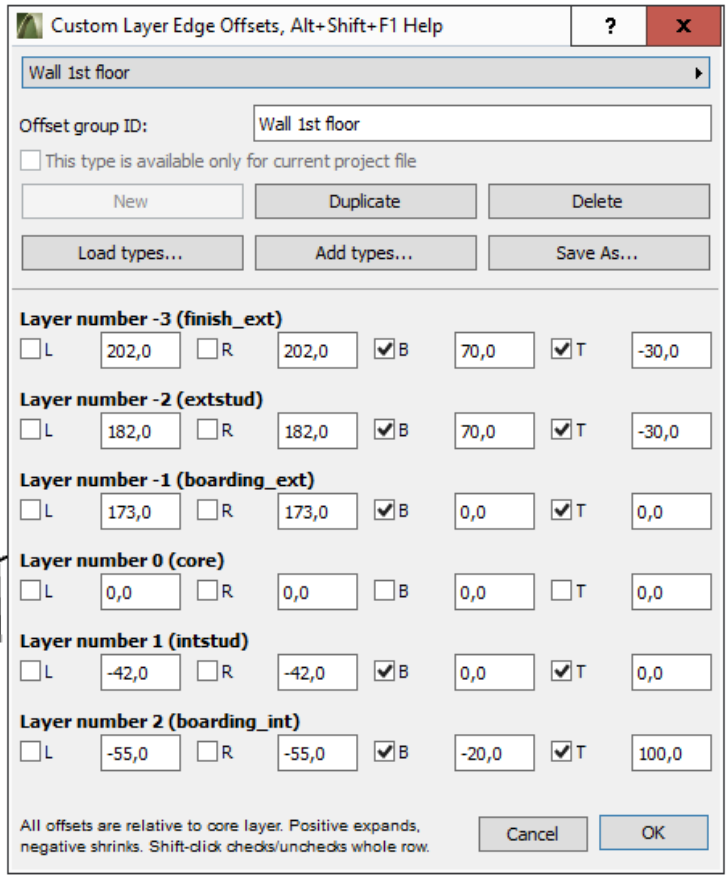
13.9 Custom layer edge/opening offsets

This tool defines distances from other layer edges to the main core layer. For example, an element having corners already set with the Element corner tools but top&bottom offsets zero:



Extend bottom 70 mm and lower top by 30 mm for exterior finishing. And lift bottom by 20 mm and top by 100 mm for the inside gypsum:

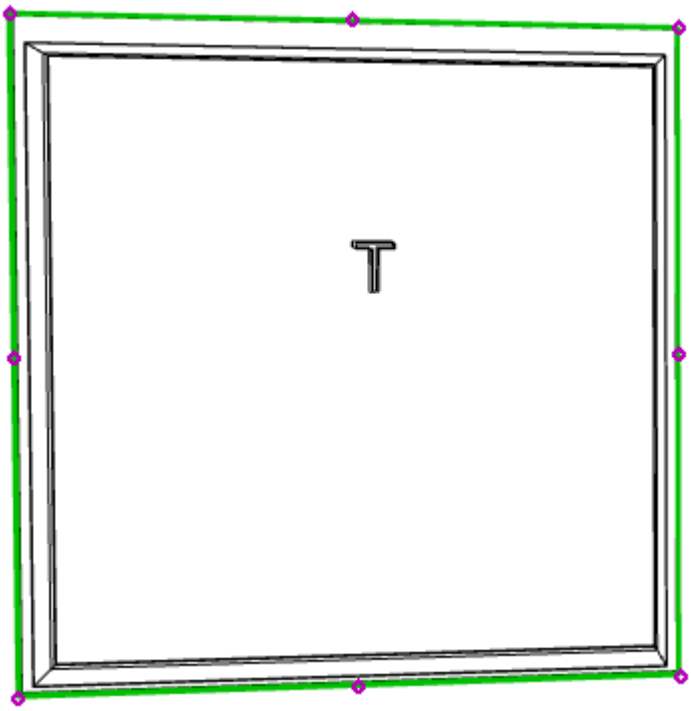




- Layer offset preset list contains all settings that have been given an ID.
- *Offset group ID* is the name for preset-list.
- Check box *This type is available only for current project file* is currently not available.

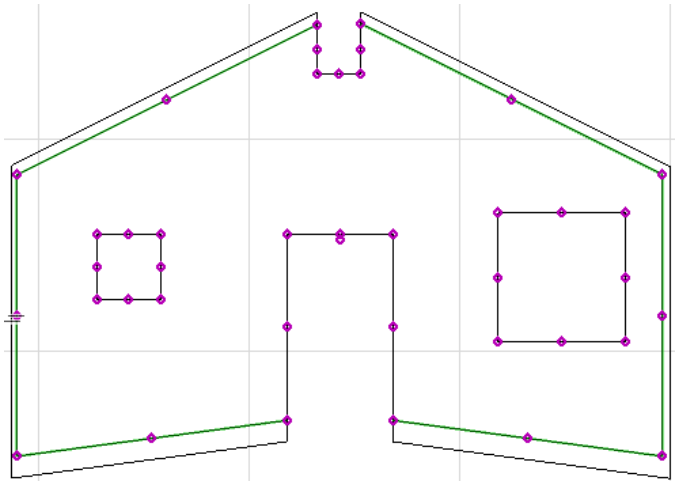
- *New* is currently not available.
- *Duplicate* duplicates current settings for further editing.
- *Delete* deletes current preset.
- *Load types* deletes all current presets and loads everything from external file. This is very useful for example if there is a single person in the company taking care of the presets and shares the types to other users.
- *Add types* is as *Load types* except it does not delete current list. It merges the types from the file using any type having the same id from the given file.
- *Save as* saves all presets to an xml-file to be used for *Load types* or *Add types*. The file can be edited using text editor for example to limit types to import to another user.
- Check boxes *L, R, B, T* and related length edit defines offsets for the left/right/bottom/top edges. Positive value extends the layer or the opening, negative makes it smaller. Only checked items will be applied to the element. For example, when defining top and bottom offsets, only T and B fields should be checked to let the corners be as they are.

Offsets for openings can be used for example to make the opening larger in the exterior finishing and air space layer:



ArchiFrame builds the layer list from the selected element or from the selected existing preset. ArchiFrame will use the definition to similar layers if applied to different element types. For example, definition for layer boarding_ext will be applied to boarding_ext1, boarding_ext2 etc if there are many boarding layers.

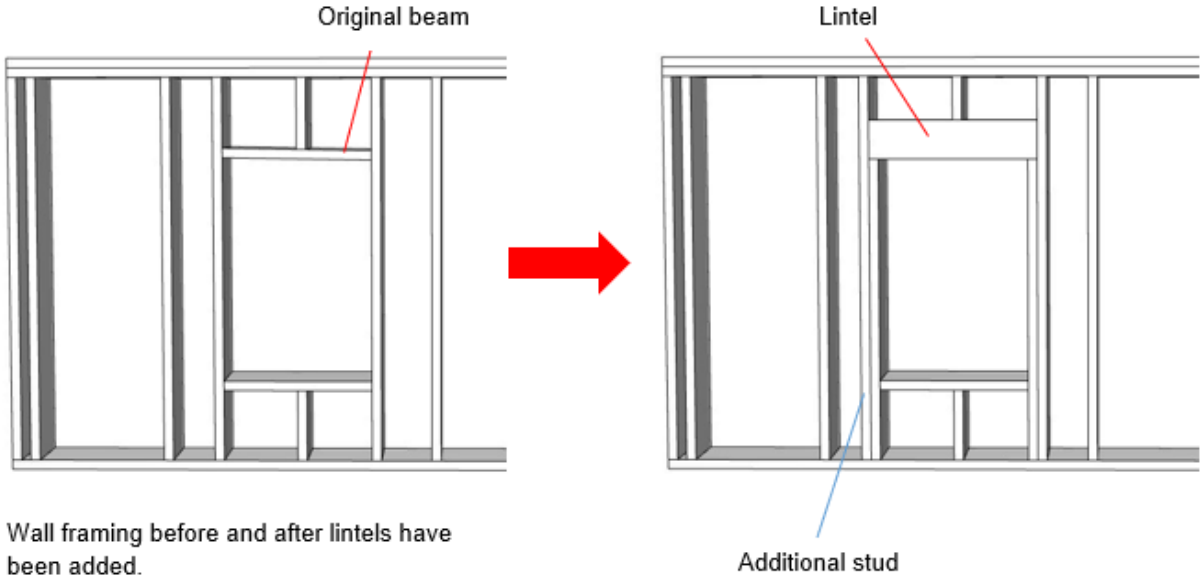
ArchiFrame will handle angled endings like below:



Here the rules are:

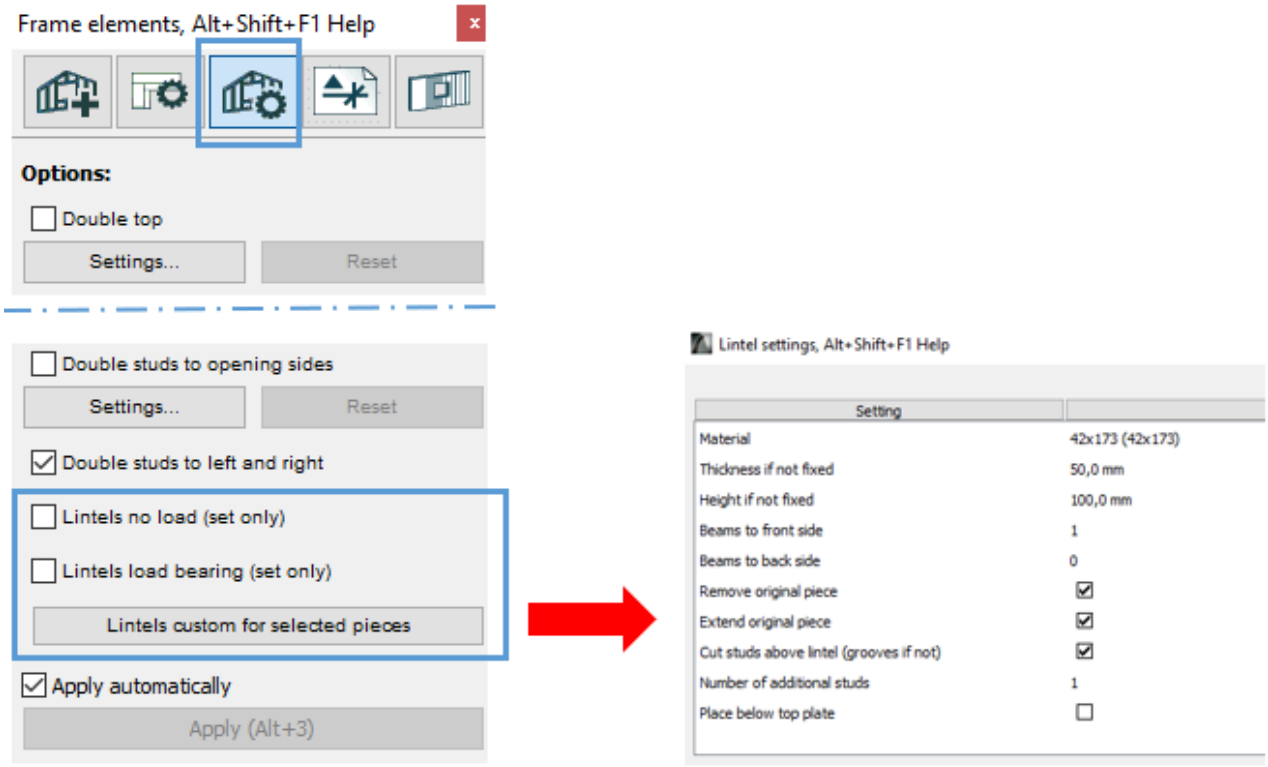
- Only top & bottom edges can have angled parts.
- Offset value is perpendicular to the line in question.
- Edge shape is taken from core layer and offset values are applied to that shape.
- ArchiFrame will find first and last part for the edge scanning for pieces that have angle -89…89 degrees to the edges normal (that is a vector pointing outside the edge).
- As a special case ArhiFrame will not apply edge offsets to three succeeding lines looking like door opening (like in the image above).
- Automation may fail in some cases. Then the element shape must be edited by hand using *To fill* and *Pick from fill* buttons or by other means.

13.10 Lintels



Wall framing before and after lintels have been added.

ArchiFrame has a special tool for adding lintels to wall openings. Lintels can be added either directly above the opening, or below the wall's top plate. Studding on either side of the opening can also be edited with this tool (Fig. 10.9.1). The tool can be found from the Set Element Options window:



For all the options, please read the [xml/Lua-script support for lintels].

13.10.1 How to create a lintel

Lintel settings can be set up for a wall element before planks are created:

- Select an element.
- Set the appropriate lintel settings.
- Create planks.

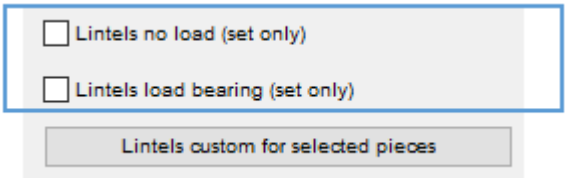
Lintels can also be added after planks have been created:

- Select the plank(s) above the opening you wish to add a lintel.
- Adjust the lintel settings.

13.10.2 Applying lintel settings from the XML file

Lintel settings are stored in the XML file in your data folder. To apply lintel settings from the file:

- Select a wall element or the piece above the opening.
- Check one of the boxes:



- Lintels will be added after planks are created.

Using settings from the XML file is useful if you are applying the same lintel settings to many different lintels, since this way the settings don't have to be set separately for each one.

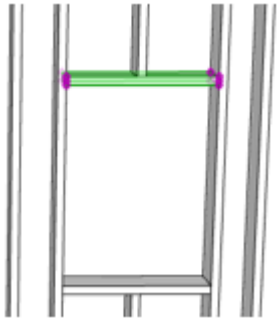
If you need to create a custom lintel type in the XML file, please contact us at ArchiFrame for help.

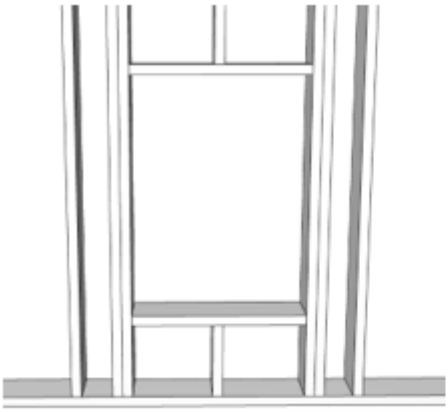
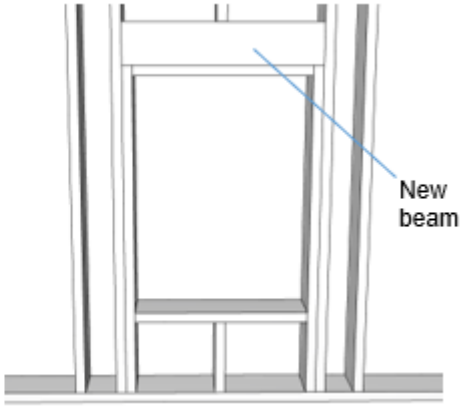
13.10.3 Custom lintel settings

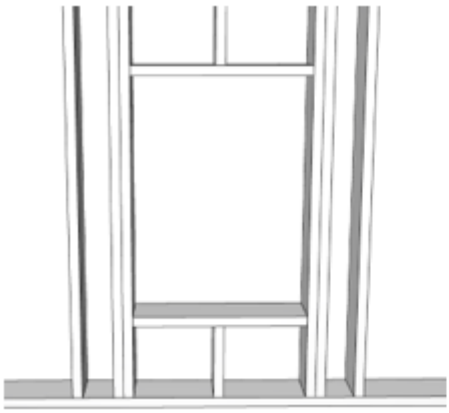
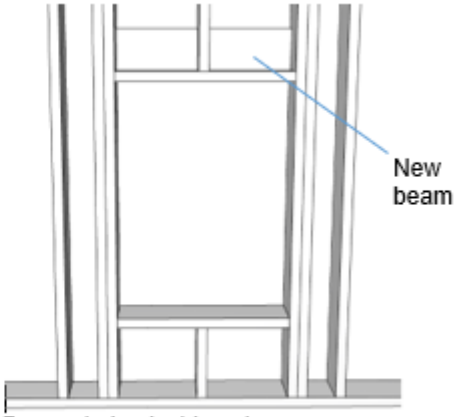
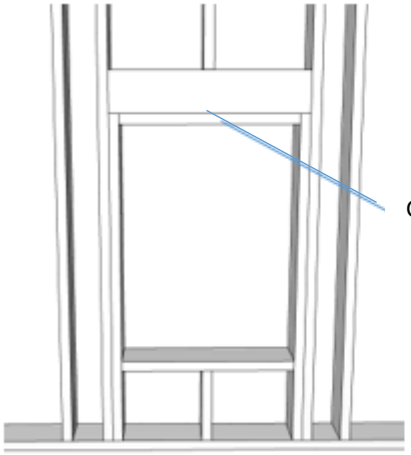
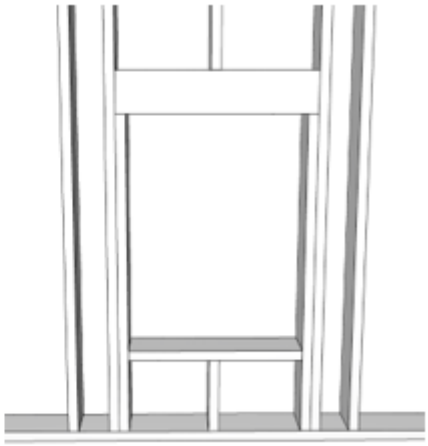
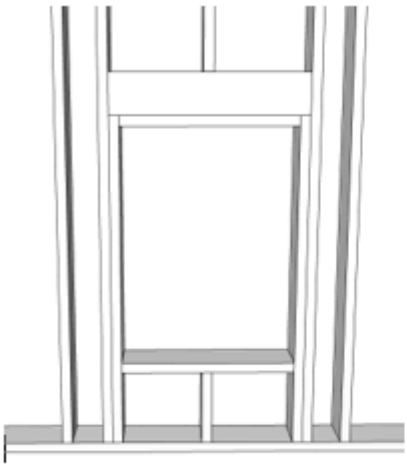
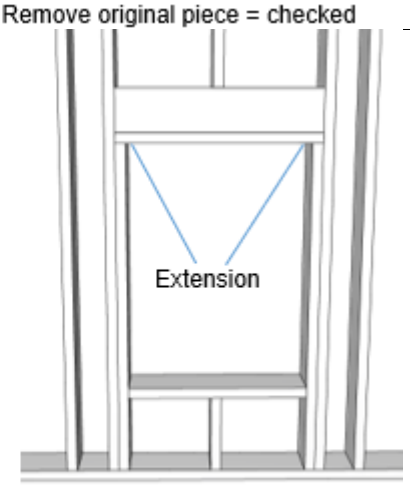
With the custom lintel settings, you can define a custom lintel type for an individual opening. To do so, first select the piece(s) above the opening (see Fig. 10.9.4), and then change the settings. Settings will be applied after you click OK.

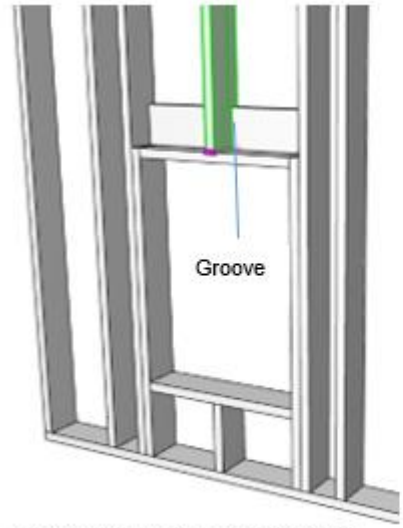
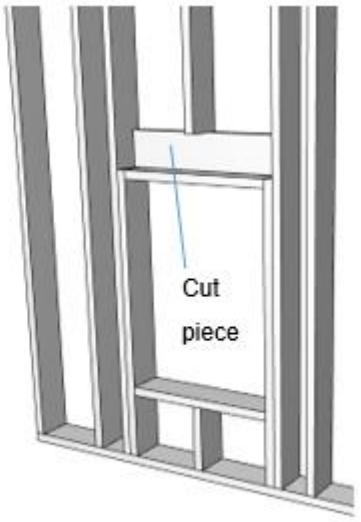
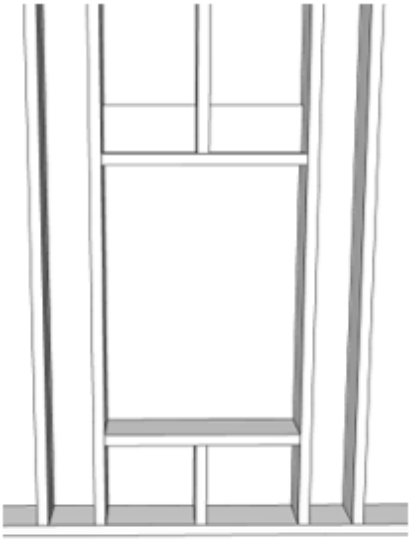
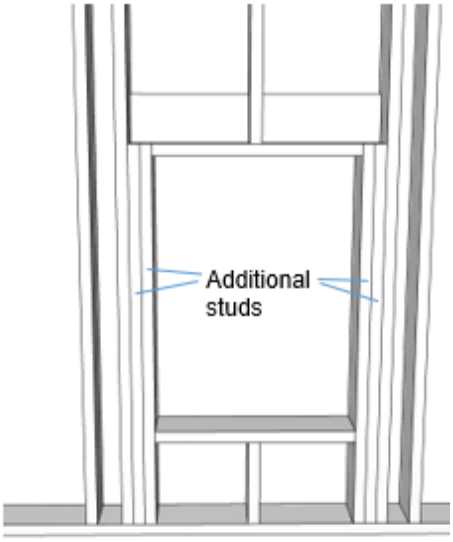
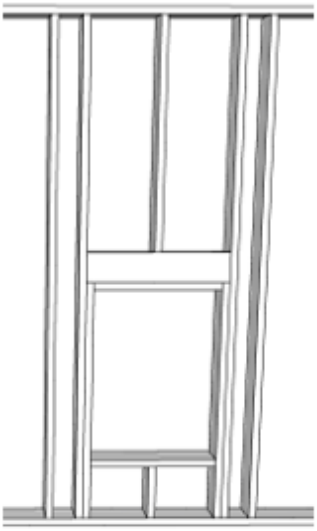
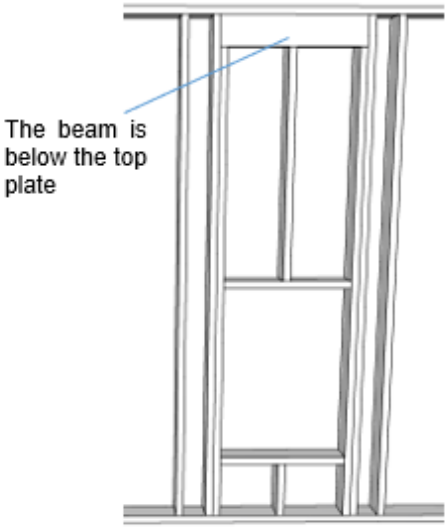
The effects of these settings are explained in the table below. All the figures in the table show the result after lintels have been created. The starting point is shown in figure 10.9.4.

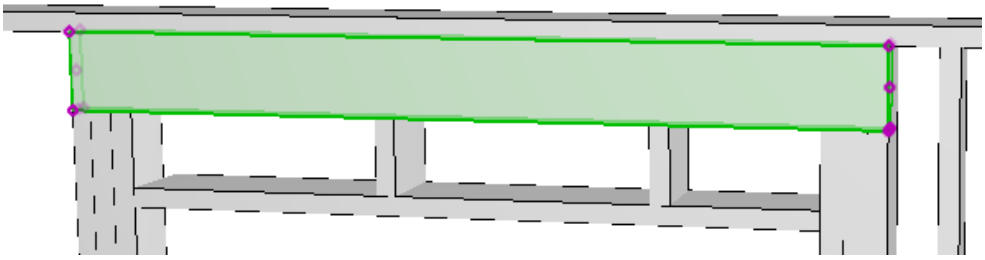
Framing before lintels have been created. The original piece above the opening is selected.



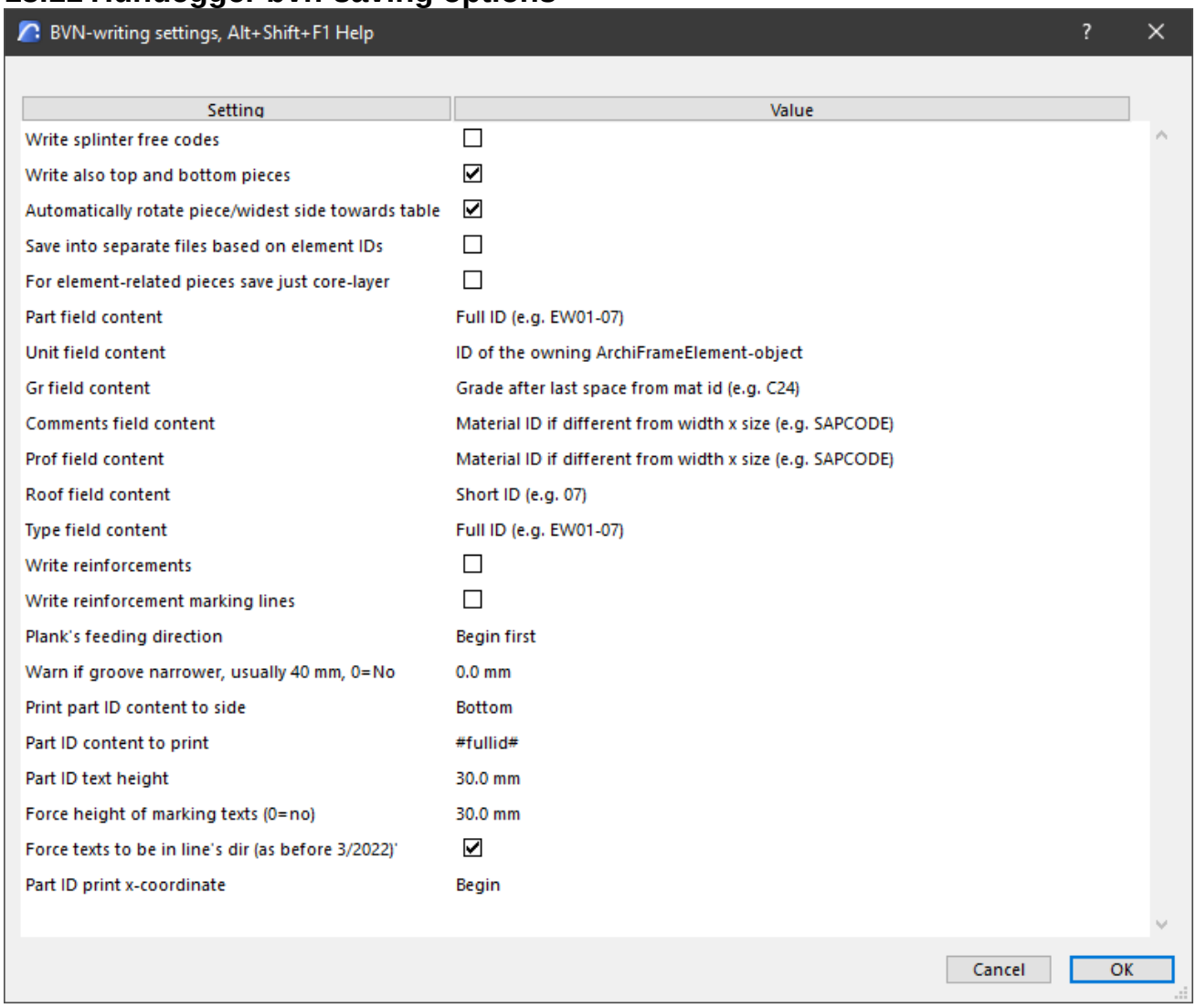
Setting	Explanation
Material	Determines the material of the new lintel piece.
Thickness	If the lintel's material was set to "block", with undefined dimensions, you can set the thickness here. • Benefit: you can create a custom lintel without defining a new material if the lintel plank is not found in the material list.
Height	See explanation for thickness above.
Beams to front side	Places the lintel on the front side of the element. In these examples, the front side faces the camera:  Beams to front side = 0  Beams to front side = 1

<table>
<tr><td>Beams to back side</td><td>Places the lintel on the back side of the element. In these examples, the front side faces the camera:

Beams to back side = 0

Beams to back side = 1</td></tr>
<tr><td>Remove original piece</td><td>
Remove original piece = not checked

Remove original piece = checked</td></tr>
<tr><td>Extend original piece</td><td>
Extend original piece = not checked

Extend original piece = checked</td></tr>
</table>

Cut studs above lintel (grooves if not)	 Cut studs to lintel = not checked  Cut studs to lintel = checked
Number of additional studs	 Number of additional studs = 0  Number of additional studs = 2
Place below top plate	 Place below top plate = not checked  Place below top plate = checked

Extend beams to furthest side of the king studs	If checked, will extend the beam to make a groove also to the king studs: 

13.11 Hundegger bvn-saving options



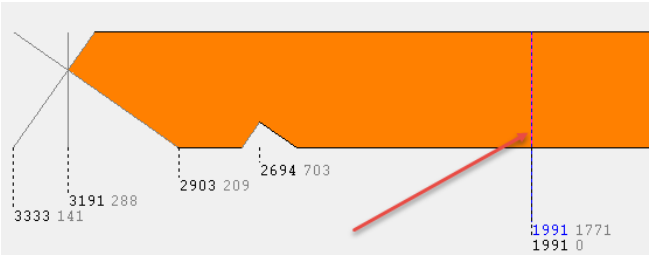
- *Write spliter free codes* defines if for example for a slot saw cuts sides of the slots before cutter to avoid splintering. Hundegger defines internally how the machining is done.
- *Write also top and bottom pieces*, if unchecked top&bottom plates are not included in the cnc-file.
- *Automatically rotate piece/widest side towards table* means that for example part 45x180 will be placed on the table the 180 mm wide surface down.
- *Save into separate files based on element IDs* will cause ArchiFrame to add element ID to given save as file name and put only pieces from single element to single file.
- *For element related pieces save just core layer* is used to skip for example air space strips from the resulting cnc-file.
- Rest of the settings define content to put into part related information fields:

Nb	Part	Req.	Cut	Width	Height	Length	Unit.	Gr	Comments	Prof.	Roof.	Type
1	Stud	1	0	36	198	2440	YV6c	C24	36x198 C24		5	YV6c-05 3

- *Write reinforcements* will write the pieces below into separate file with postfix _reinforce and it will add REINFORCED to comment field in bvn if the piece has reinforcements.



- *Write reinforcement marking lines* marks ends of reinforcements into the bvn-file:

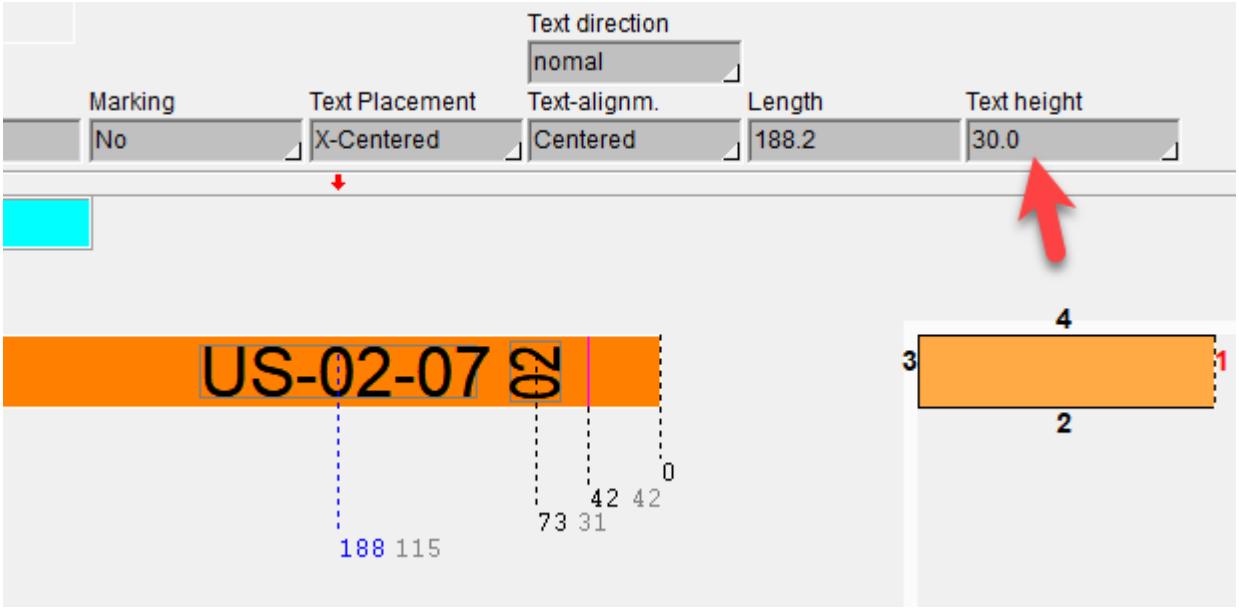


- *Plank's feeding direction* defines which end of the plank is going to the machine first. For example, it may be good to change it to End first to do this kind of studs so that the groove comes first:



- *Print part ID content to side* defines the ArchiFrame plank side to be used to print information into plank with Hundegger's inkjet printer or *Nothing* means no printing.
- *Part ID content to print* defines what to print. It may have any text and following placeholders, for example: (#plankroleshortnor# #fullid#)
    - #fullid# Full plank ID: EW01-07
    - #projid# Project ID auto text
    - #projnum# Project number auto text
    - #usage# Please see the image below
    - #shortid#
    - #plankrole#
    - #plankrolefin#
    - #plankrolenor#
    - #plankroleshort#
    - #plankroleshortfin#
    - #plankroleshortnor#
    - #ownerelemid#
    - #floornum#

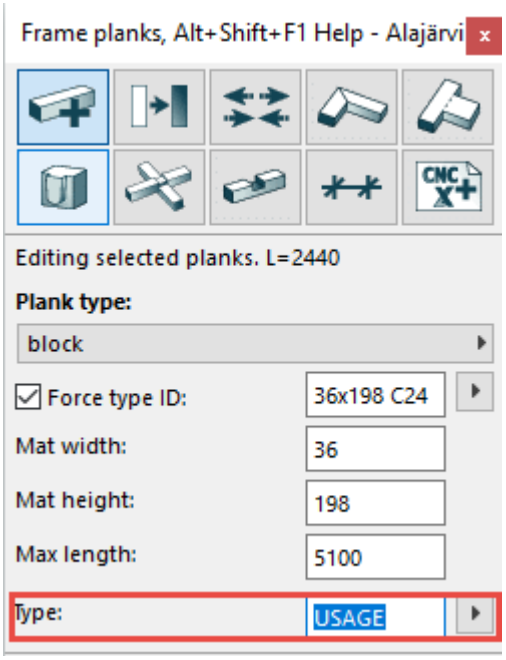
- *Part ID text height* is the height of printed part ID:



- *Force height of marking texts* can be used to set the height of the end result or more commonly: the machine prints in its own pre defined size and this is used to let ArchiFrame calculate correctly possibly overlapping texts. If the ID in above image overlaps with stud marking texts, please increase this font size and set the same value to previous setting.
- *Force texts to be in line's dir (as before 3/2022)* checked stud ID 02 as in above image, unchecked in plank's direction.
- *Part ID print x-coordinate* defines where to print the part ID. It is automatically moved not to overlap with existing markings. Please see also *Force height of marking texts* above.

The possible piece of information to select for each field are:

- The example piece is part of element EW01, its full ID is EW01-07, its type is 48x198 C24 and it is a stud. Usage is defined here:



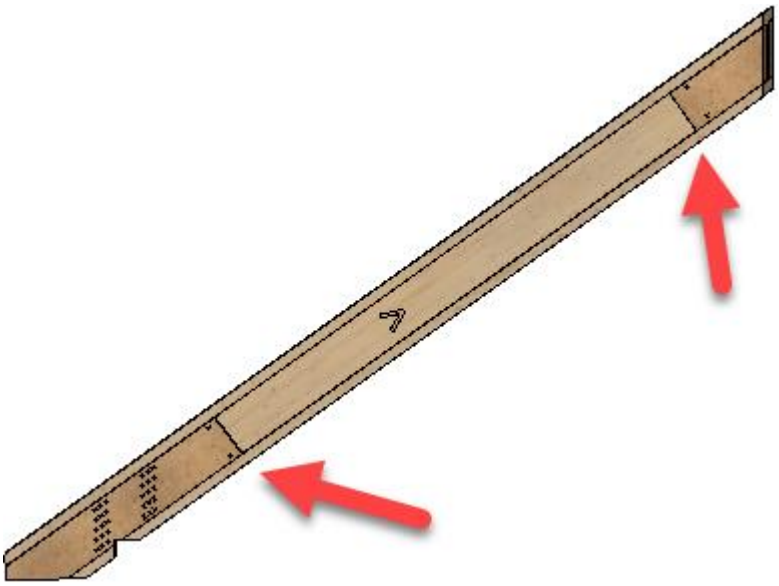
- Full ID plus usage: EW01-07 (USAGE).

- Short ID: 07.
- Full ID: EW01-07.
- Element ID: EW01.
- Usage: USAGE.
- Full mat id: 48x198 C24.
- Short mat id cut to last space: 48x198.
- Grade after last space from mat id: C24.
- Material ID if different from width x size: 48x198 C24.
- Plank role in element: Stud.
- Plank role in element Finnish: Tolppa.
- Plank role in element Norwegian: Stender.
- Plank role short in element: ST
- Plank role short in element Finnish: TO
- Plank role short in element Norwegian: ST

13.12 BTL-saving options

In both Btl and WeinmannBtl:

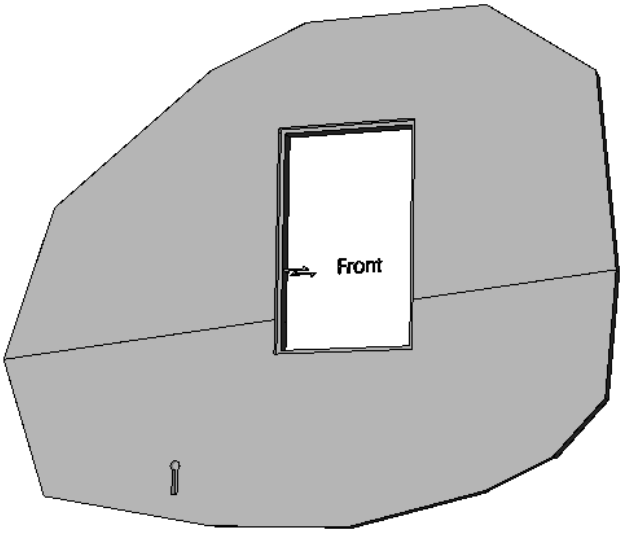
- *Mark reinforcements* will produce marking lines for the reinforcements typically used in I-shaped beams:



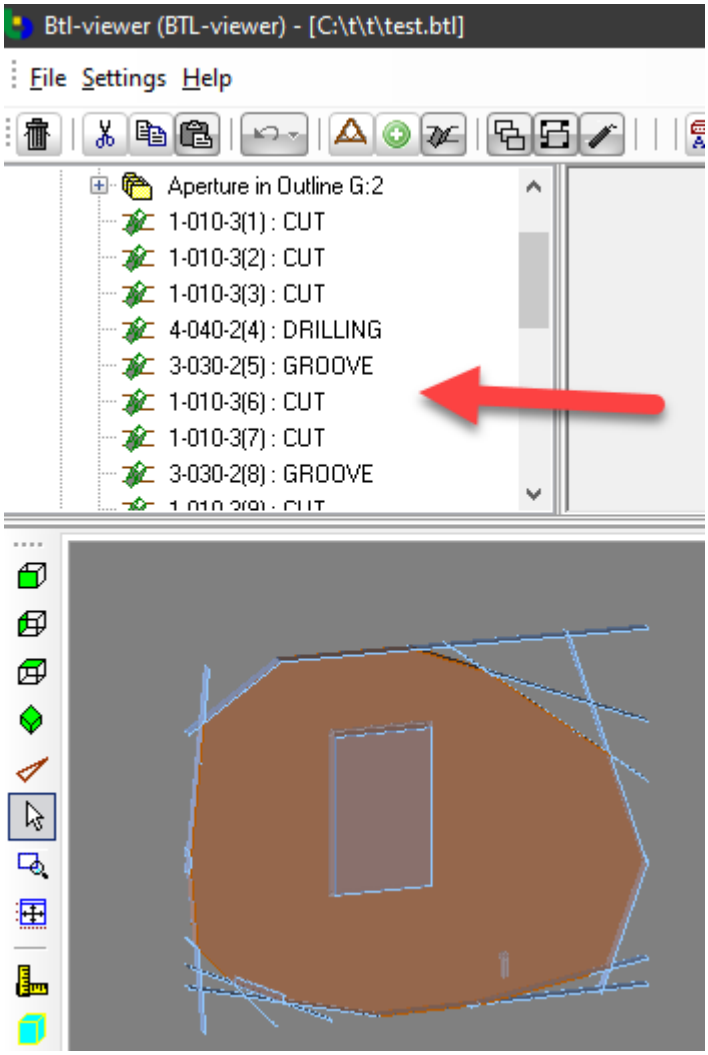
Only in the Btl-writer:

- *Element built from inside to out (layer numbers)* defines the layer numbering in btl: If checked, the layer numbers before core layer will be positive and after core negative.
- *Start and end offset* are used to tell the saw how much to cut away from each plank to make sure its ending is straight.
- *Extend selection* is useful for testing to write just a single piece into btl-file. Typically all pieces related to an element should be written.
- *DESIGNATION, MATERIAL and TIMBERGRADE* settings define what information to write from a plank to these standard Btl-fields.

- *Default saw cuts if no ArchiFrameSawCut-objects* will add sawing lines based on boarding/cladding layers' joined polygons if there are no ArchiFrameSawCut-objects placed over the elevations (single elevation per layer is required for ArchiFrameSawCuts)
- *ArchiFrameSawCut generated codes* tell what codes to write from the saw cuts: The machine will use either saw blade or a milling tool.
- *Board polygon processing behaviour* is typically needed for CLT-boards and the different values are:
    - *PROCESS: NO (default)* will write the board shape into btl-file but the cnc-machine will not make the cuts.
    - *PROCESS: YES* as previous but the cnc-machine should cut all polygon lines.
    - *PROCESS: NO + Generate machinings* will mark the contour lines as no and will add btl saw cuts and grooves to match the board shape. This is required by DDX Software Solutions software. For example, board like this:



In BtlViewer it will produce many cuts and grooves:



- *Scaleunit/Precision* will define the precision used in the resulting btl-file.
- *Destination saw type* defines the target saw type. Use other than default only if your machine is the listed one. Stromab has these changes:
    - File version is written 10.4 instead of 10.6
    - Groove X-030-X parameter P09 is written as 0 deg instead of 90 deg.

Only in the WeinmannBtl-writer:

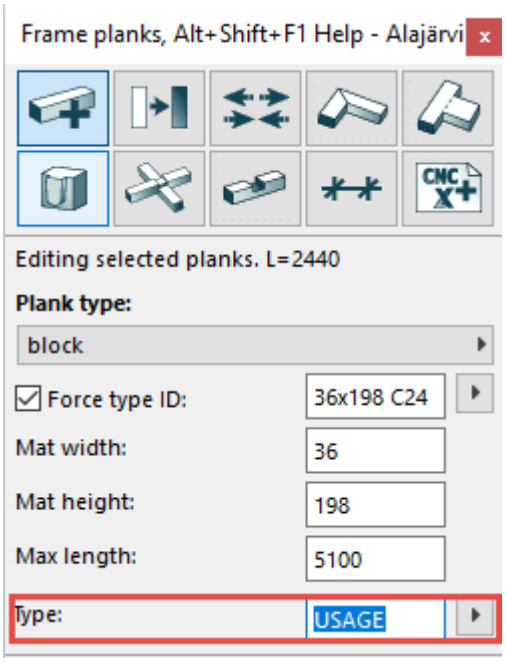
- *Groove minimum width* defines the tool's diameter and ArchiFrame will issue warnings if there are smaller grooves.
- *Write top & bottom plates to separate files* will write top and bottom plates to separate file(s) adding postfix to the base file name.
- *Move grooves on side 1 mm from end (only top&bot pcs)* will cause ArchiFrame to move grooves 1 mm inside the plank to make Weinmann saw to make the grooves the same way as it does for the grooves in the middle of the plank. Unchecked Weinmann saw may start making the grooves from the end of the piece and it may end up bit different than the grooves in the middle of the plank.
- *Part ID content printed on wood* defines the text that identifies each part and is printed to the part. Option may contain any text and these placeholders that are replaced:

#fullid#	Full plank ID: EW01-07
#projid#	Project ID auto text
#projnum#	Project number auto text

- *Part ID mid from end (pos=from end, neg=from beg)* defines the distance of middle of the printed ID from plank's end with positive value. Negative value anchors the part ID to the begin of the plank.
- *Part ID top from surface bottom* defines the part ID text top distance from the surface bottom. Please note that BtlViewer does not show this as Weinmann control software.
- *Save into separate files based on element IDs* will cause ArchiFrame to add element ID to given save as file name and put only pieces from single element to single file.
- *For element related pieces save just core layer* is used to skip for example air space strips from the resulting cnc-file.

The possible piece of information to select for each field are:

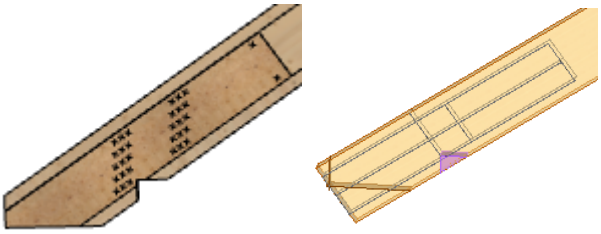
- The example piece is part of element EW01, its full ID is EW01-07, its type is 48x198 C24 and it is a stud. Usage is defined here:



- Full ID plus usage: EW01-07 (USAGE).
- Short ID: 07.
- Full ID: EW01-07.
- Element ID: EW01.
- Usage: USAGE.
- Full mat id: 48x198 C24.
- Short mat id cut to last space: 48x198.
- Grade after last space from mat id: C24.
- Material ID if different from width x size: 48x198 C24.
- Plank role in element: Stud.
- Plank role in element Finnish: Tolppa.
- Plank role in element Norwegian: Stender.

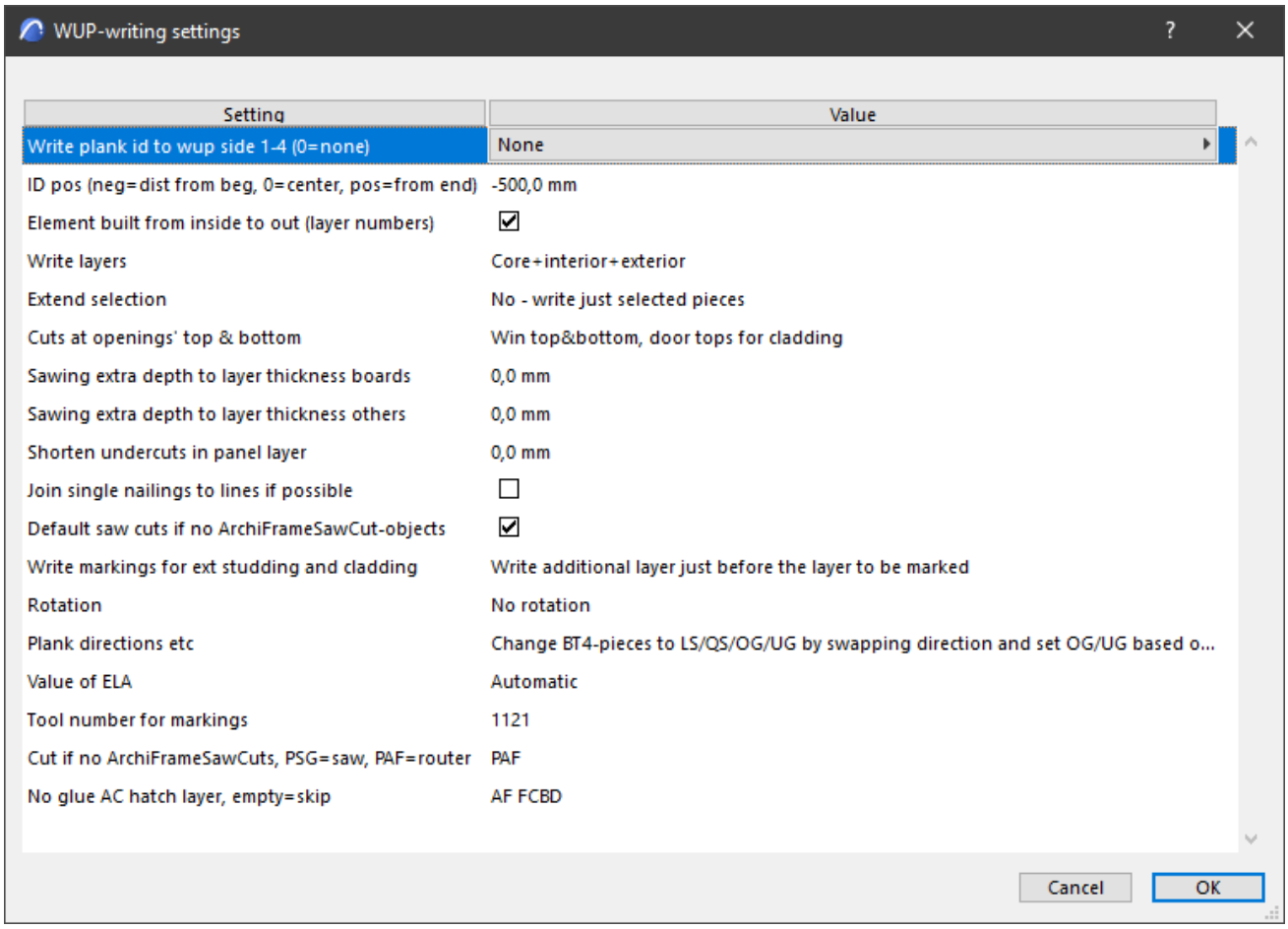
Options continue:

- *Mark reinforcement* defines whether to mark positions of reinforcements with lines:



- Boards' polygons PROCESS-value will set the btl-code's for boards' polygons values to either:
    - PROCESS: NO
    - PROCESS: YES
      Value YES allows some post processors board nesting tool to work.

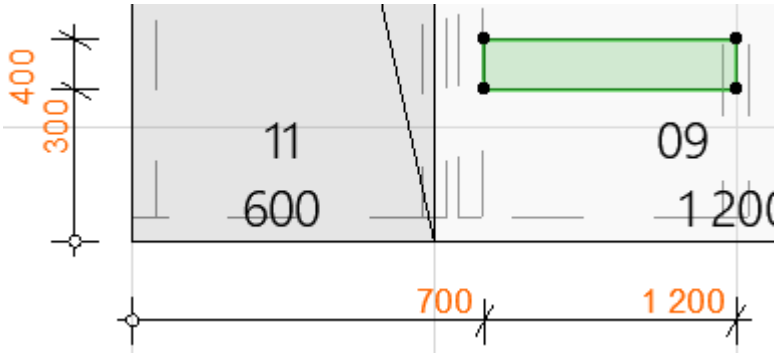
13.13 Wup-saving options



Defaults can be set per element type in composite type's cnc_wupsettings string. The settings are:

- *Element built from inside to out* defines if layers before core are PLIx (interior) or PLAx (exterior). If checked, the layers before core will be PLIx and layers after will be PLAx. Unchecked the other way round. ArchiFrame will give warnings if composite layer types have different classification (for example layer type before core is boarding interior and this option is unchecked).
- *Cuts at opening's top & bottom,* to add special saw cuts for fascia boads.
- *Sawing extra depths* will adjust sawing depth for example to windshield boards to make sure that the board is fully cut.

- *Shorten undercuts in panel layer* will shorten interior corner cuts in cladding layer with given value to make sure the corner comes out uncut.
- *Join single nailings to lines if possible* makes ArchiFrame to convert point nailings to lines with spacing if there are many adjacent point nailings on the same line with same spacing.
- *Nailings ordering* will set the way ArchiFrame writes the nailings. The machine will anyway does its own optimization so it may be best to set this to *In plank's order* to let the operator easily remove not wanted nailings for a single cladding piece for example.
- *Default saw cuts if no ArchiFrameSawCut-objects* will add sawing lines based on boarding/cladding layers' joined polygons if there are no ArchiFrameSawCut-objects placed over the elevations (single elevation per layer is required for ArchiFrameSawCuts)
- *Write PLA2 markings for ext studding and cladding* will add PLA2-layer after windshield board marking the airspace strips and every fourth cladding piece.
- *Rotation* rotates the structure by given value
- *Plank directions etc* swaps plank directions for wup-file so that framing codes can be used instead of BT4-codes. Useful at least when using rotation.
- *Value of ELA* sets the value of wup-file's header line ELA INTERIOR; or ELA EXTERIOR; Default setting is automatic. May be needed if rotation swaps the watching direction.
- *No glue AC hatch layer* defines the Archicad layer where user can place hatches to define such areas. The hatches must be drawn over projections – not into the floor plan. The resulting wup-codes are (tool number one defines no glue area in wup):

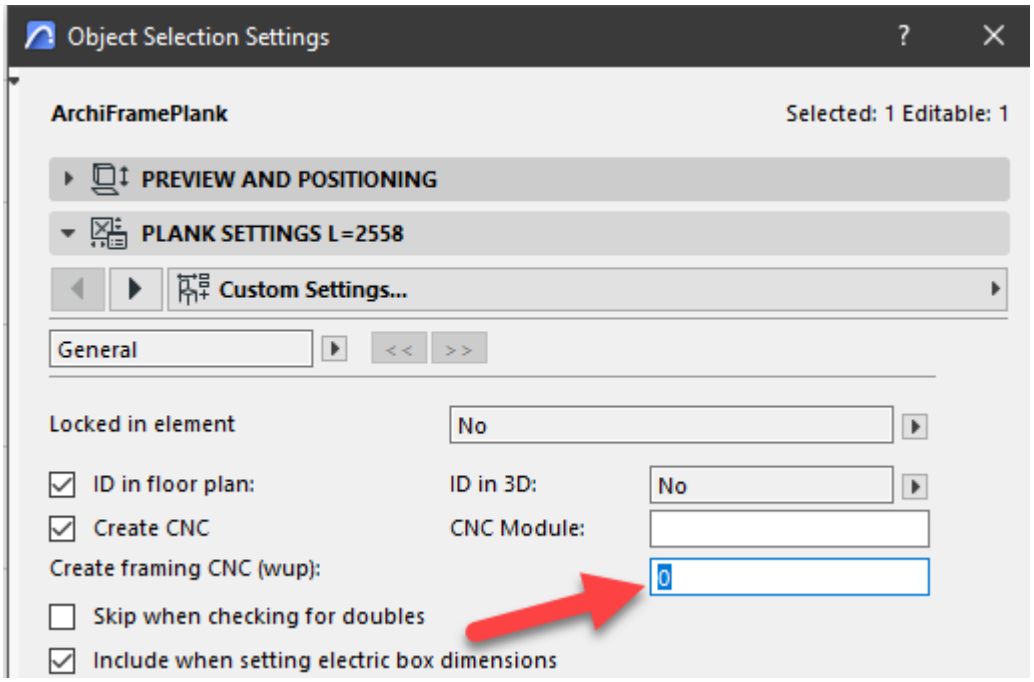


PSF;
PP 1200,400,9,1,0,-9;
PP 1200,300,9,1,0,-9;
PP 700,300,9,1,0,-9;
PP 700,400,9,1,0,-9;
PP 1200,400,9,1,0,-9;

- *Combine marking lines max dist,* if value is bigger than -1 meter, the wup-writer will combine parallel marking lines on same line. Negative distance means that the lines must overlap given distance, positive means there may be max given distance between the marking lines' endings. Value -1 or smaller negative value disables combining marking lines.
- *Warn about skipped machinings* is meant to be used if wup-file is used for beam processing. When selected, there will be warning of any normal framing cut (grooves, saw cuts, drillings, lengthwise saw cut) that was not written into the resulting wup file. If the beams are processed with another cnc-file like bvn/bvx/btl, this can be unchecked since it is used just for wup visualization.

13.13.1EBT parts

EBT parts are core layer parts that are there but should not be handled by the framing station. To produce EBT-parts, this plank setting is used:



Values:

- 1 is normal part
- 0 is written as EBT
- -1 part is not written at all to wup

13.13.2More information about the nailings

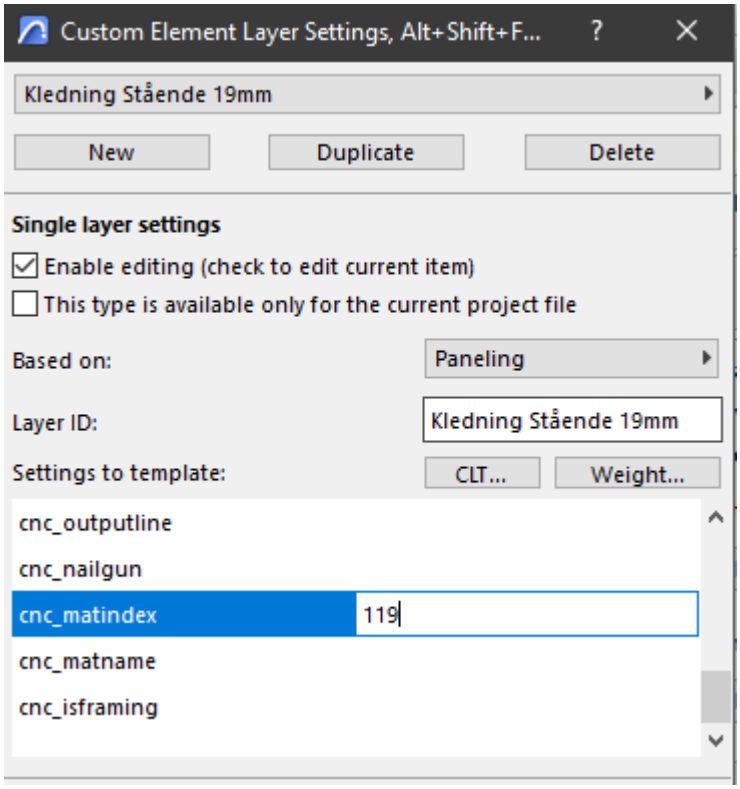
Weinmann will readjust cladding nailings and speed up the nailing operation if you set the cnc_materialindex for the layer correct. For cladding, the value 119 will enable Weinmann's

optimizations (a screen shot from wup specification):

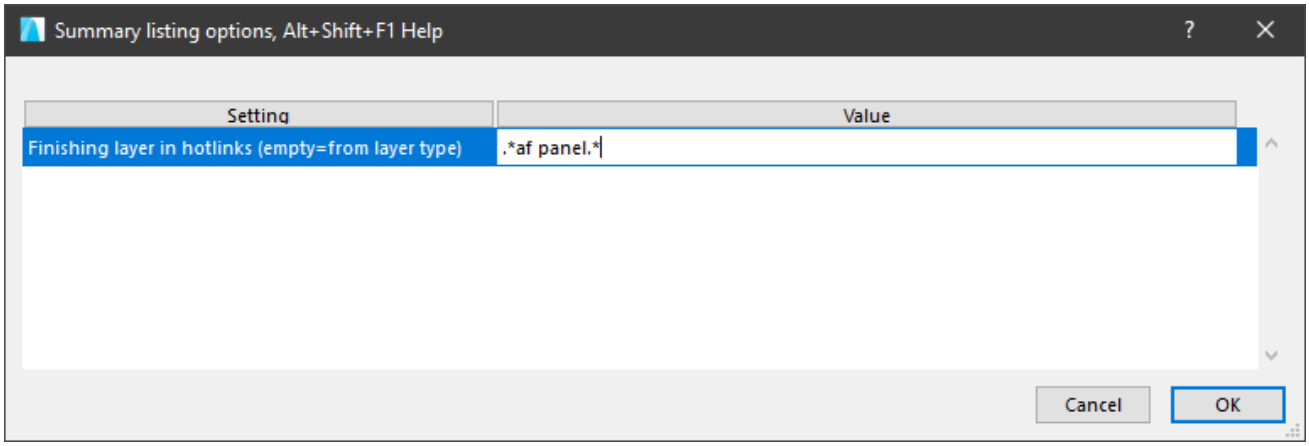
The material index identifies the type of panel.

Material	Index
Wood component	01-09
Fermacell	10-19
Soft fiber panel (Gutex,...)	20-29
OSB	30-39
chip board	40-49
Plaster-base sheeting	50-59
Plaster	60-69
Gypsum plasterboard	70-79
Plastic panel	80-89
Plywood panel	90-99
Plaster	100-109
Shuttering	110-119
Three-layer panel	120-129
Glue	130-139
Insulating plate (Diffutherm)	140-149
Insulating plate (Heraklith)	150-159
Floorboards	160-169
Adhesive tape	170-179
Film/vapor block	180-189*
Plywood panel	190-199
Hardboard	200-209
Profiled panel [1)]	210-219
Porous concrete	220-229
Cavity insulation: cellulose	230-239
Cavity insulation: soft wood fiber	240-249
Cavity insulation: mineral wool	250-259

It is the own element settings where to adjust the cnc_materialindex:



13.14 Summary listing options



In hotlinked master file ArchiFrame has no access to saved element data and cannot detect the cladding layer from the own element type's layer definition. To get the 3D-materials in hotlink master file, it is possible to enter layer name identifying finishing parts. Layer name is case insensitive limited to standard characters A-Z, special characters like ÅÄÖ are case sensitive. Layer name may contain wild parts defined with two characters: .*

For example .*af panel.* will pick all layers having text af panel anywhere in the layer name. For full description of patterns, please read this article.

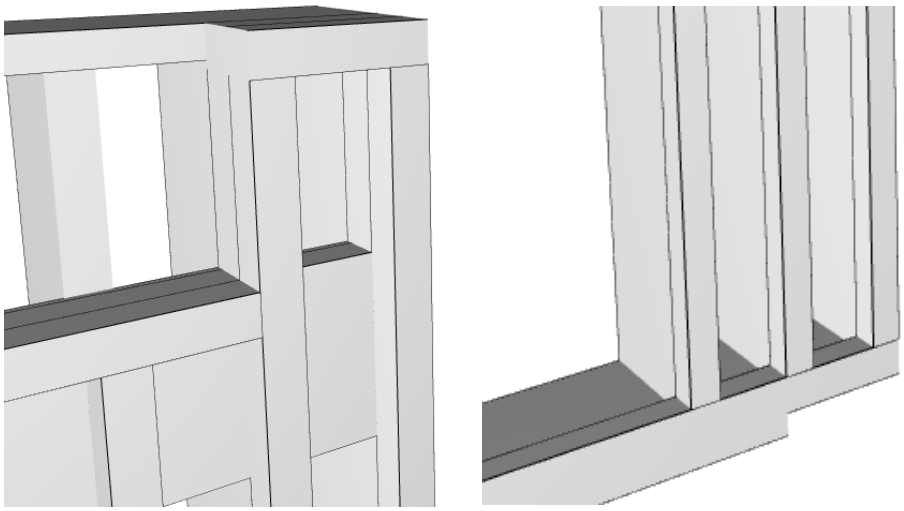
The quality and type fields are defined in the material list ArchiFrameBlocks.xml or ArchiFrameBlocksChanges.xml in tags excel_quality and excel_type:

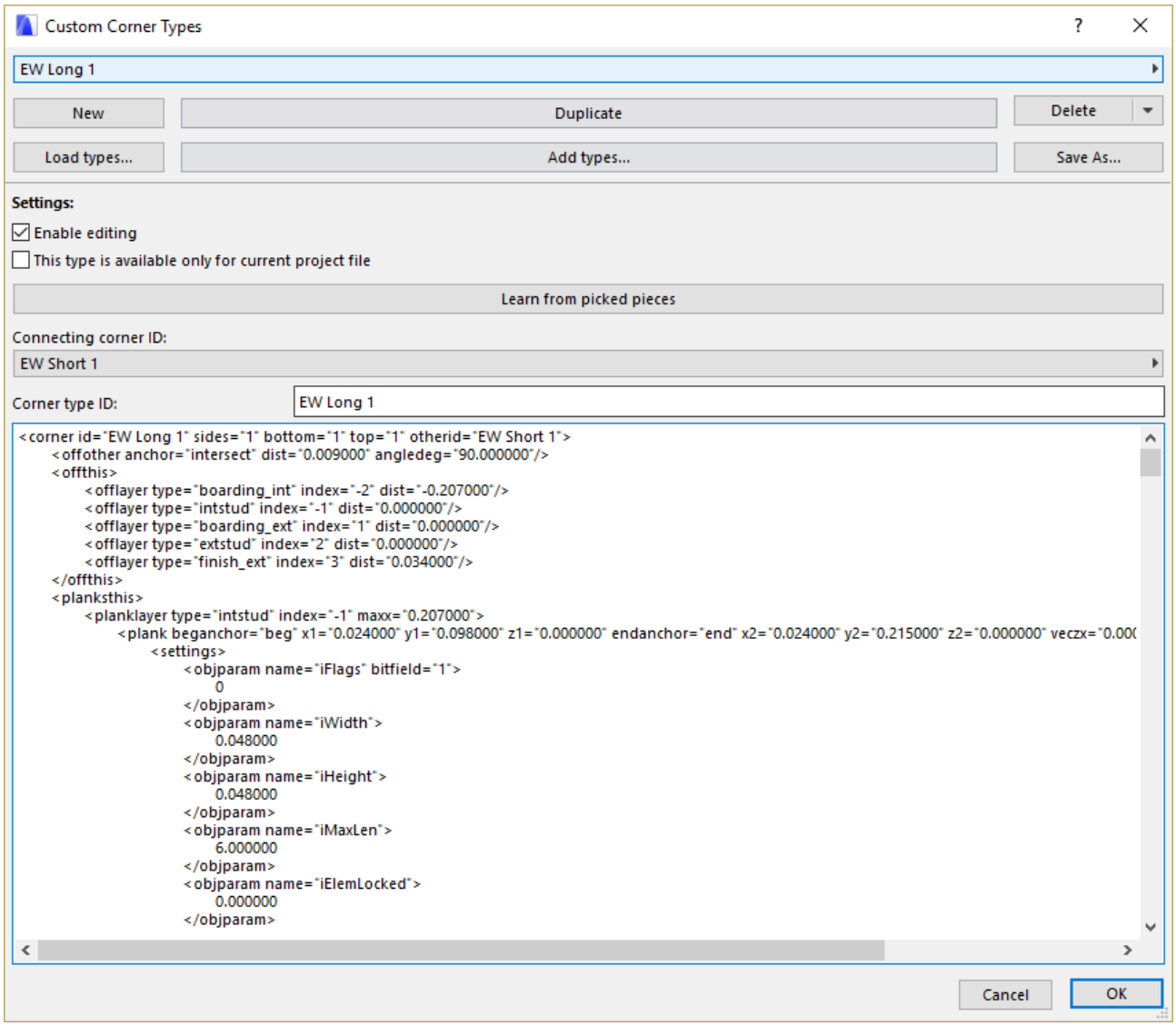
```
<material id="LP 90x270" name="LP 90x270" thickness="0.090" height="0.270" shape="block"
maxlen="99" m3factor="0.000" excel_quality="C22" excel_type="GL">
</material>
```

13.15 Custom corner types

Video: https://vimeo.com/277423174
https://player.vimeo.com/video/277423174

Ability to teach corner types allows very complicated connections. For example, like below:





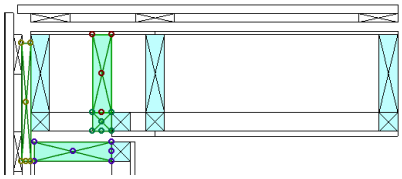
Dialog parts are:

- Corner type dropdown list, this list contains all defined types.
- *New* creates new empty corner type definition.

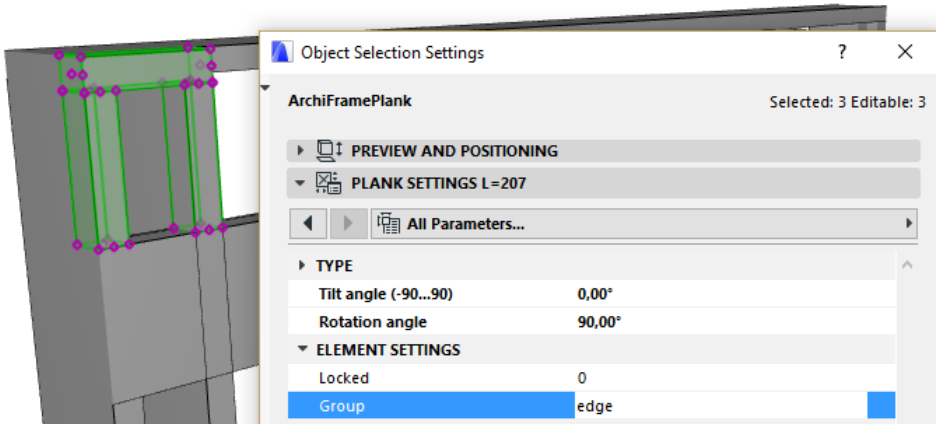
- *Duplicate* duplicates current type for further editing.
- *Delete* deletes current type. Be very careful not to delete any structure that is in use. The custom corner types are shared among all projects as default. So, delete a type only if you are absolutely sure that it is not needed any more.
- *Load types* deletes all custom corner types and loads everything from external file. This is very useful for example if there is a single person in the company taking care of the structures and shares the types to other users.
- *Add types* is as *Load types* except it does not delete current types. It merges the types from the file using any type having the same id from the given file.
- *Save as* saves all custom types to an xml-file to be used for *Load types* or *Add types*. The file can be edited using text editor for example to limit types to import to another user.
- *Enable editing* must be checked on before making any changes. Its purpose is only to avoid editing existing types accidentaly.
- Check box *This type is available only for current project file* defines that the type is available only in current Archicad project file (pln-file).
- *Corner type ID* is the unique ID for the custom corner type. The ID should never be changed if the corner type is in use. It is advisable to design the ID types carefully before actually adding elements.

Steps to teach a corner:

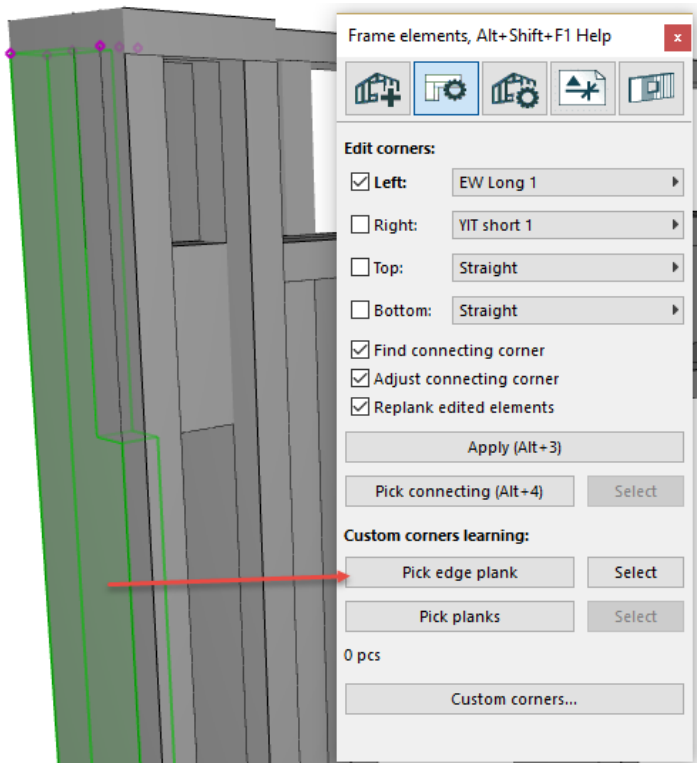
1. Model the corner as it should be (we will first teach the long/horizontal wall). *Important! ArchiFrameElement-objects must be first adjusted to have correct layer offsets – moving just the planks will not work.*



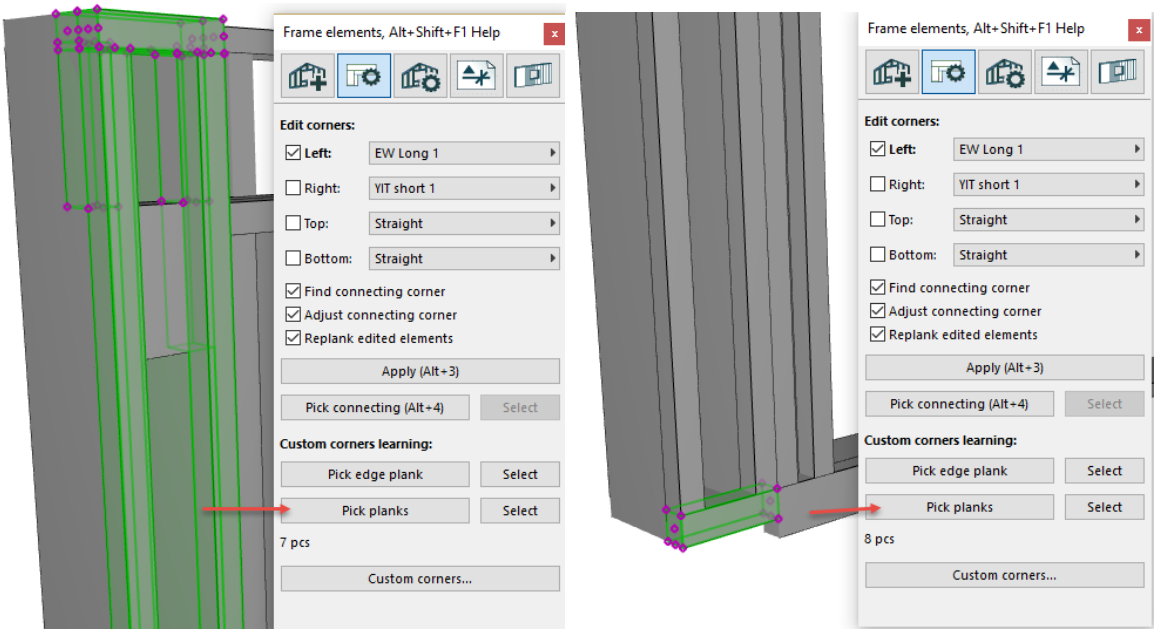
2. Change the element group of special pieces to value edge. If these were original (top_force, vertical_force and vertical_y) ArchiFrame would trim the pieces away when applying the framing rules. Normal studs should not be changed to allow for example *Marking grooves* and *Double top* options to work:



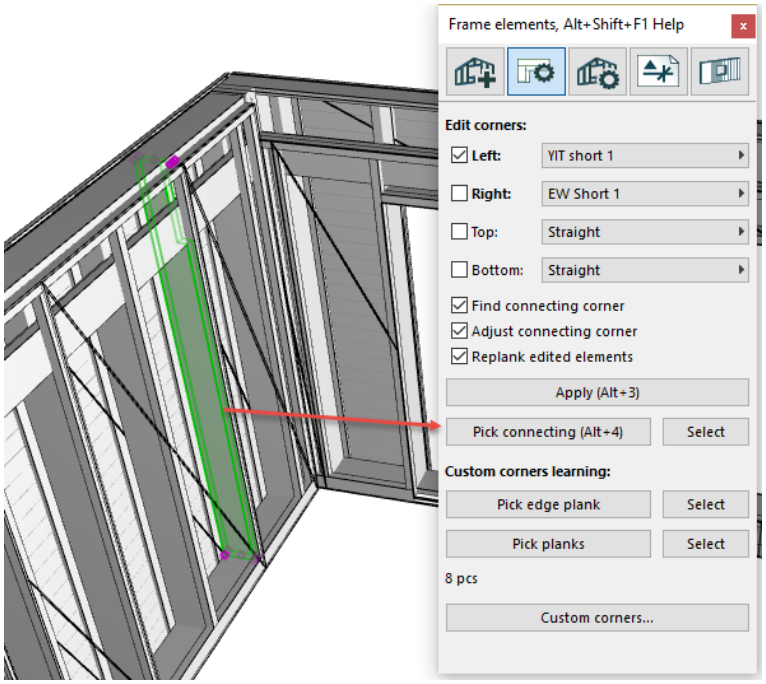
3. Pick the edge plank that tells ArchiFrame which edge to learn from:



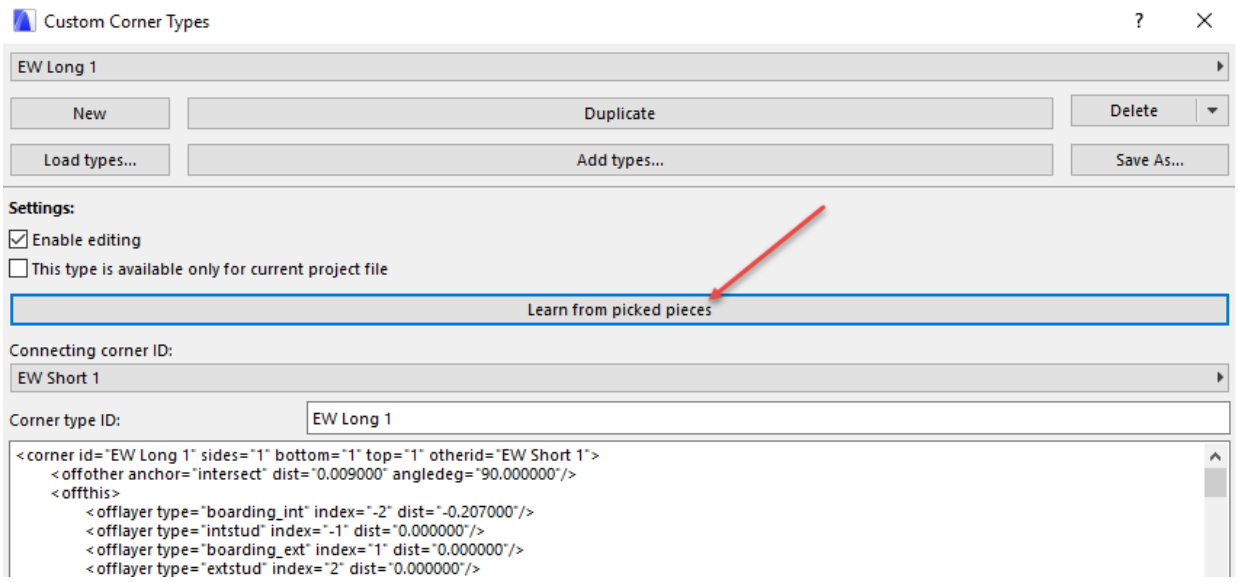
4. Select planks to learn, *Pick planks* is used twice – first at top and then at bottom of the wall:



5. Pick the connecting element (having any plank/board relateted to the connecting element is ok):



6. Open *Custom corners* dialog and click New to make new corner type (or select old one to edit its definitions), then click *Learn from picked pieces*:



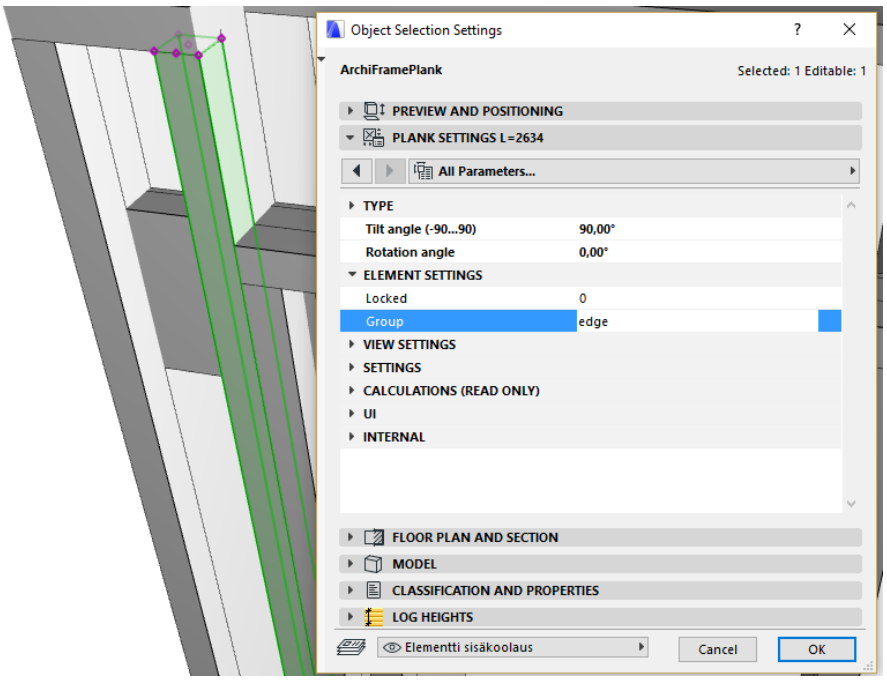
In special cases it is possible to edit the resulting xml-definition. ArchiFrame will save this information about the corner:

- Distance to connecting element's core-layer and angle (0…90 deg) between edge normal vector and connecting element's X-axis. This angle is used when finding related *ArchiFrameElement*-objects – only parts with matching angle are picked.
- Current element's layer offsets (for example, how much far the cladding extends from core layer).
- For each layer the handled area where other framing rules may not add planks. Attribute maxx="value" defines the distance from the edge that marks no framing area. If there was a plank to be created that intersects that area, the plank will not be created. For example, the definition below sets the area to be 485 mm from the edge of the element layer:

<planklayer type="core" index="0" **maxx="0.485000"**>
If the edge rule is applied to top/bottom of the element, this rule will prevent creating spacing-rule studs at all so you will need to edit the value to:
<planklayer type="core" index="0" **maxx="0"**>.

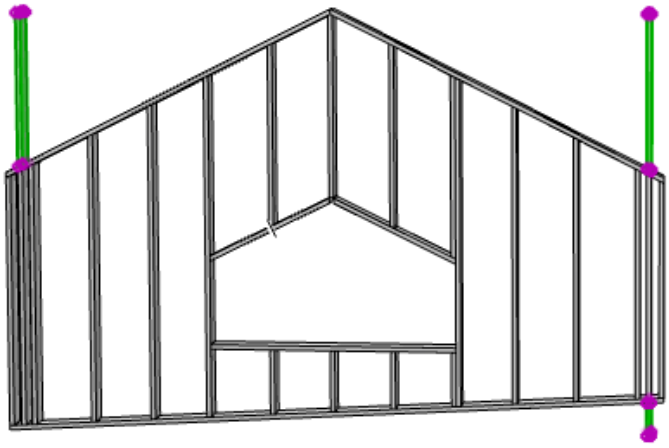
- For planks following information is saved:

    - Begin and end position relative to source edge's top and bottom coordinates.
    - Placement relative to source edge's geometry.
    - Numerous parameters.
    - The element role is set to edge unless it contained force originally:



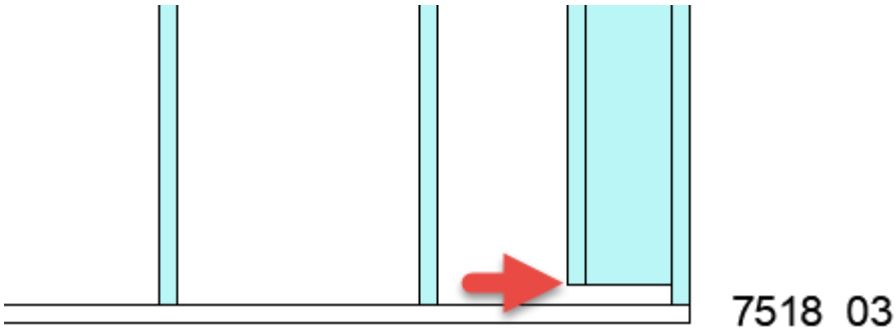
This group setting affects how planks are handled in element options tool.

- Does the learnt piece intersect with its owning parent element? This information is used to automatically clean up pieces outside the element in cases like a gable wall.



This is defined by attribute xelem = "0" (part does not intersect the element – do not cleanup) or xelem = "1" (part intersects with element, delete all parts in target that are outside the element).

By default, saved edge pieces have precedence over other parts. In other words, they cut other parts. To chage that the other way round for example to teach this kind of situation where you can have single or double bottom plates:



The steps are to teach this so that the vertical parts reach the lowest bottom plate. Then, following line is needed to xml-definition after each <plank xxx>-line to allow ArchiFrame to adjust the related pieces to current top & bottom plates:

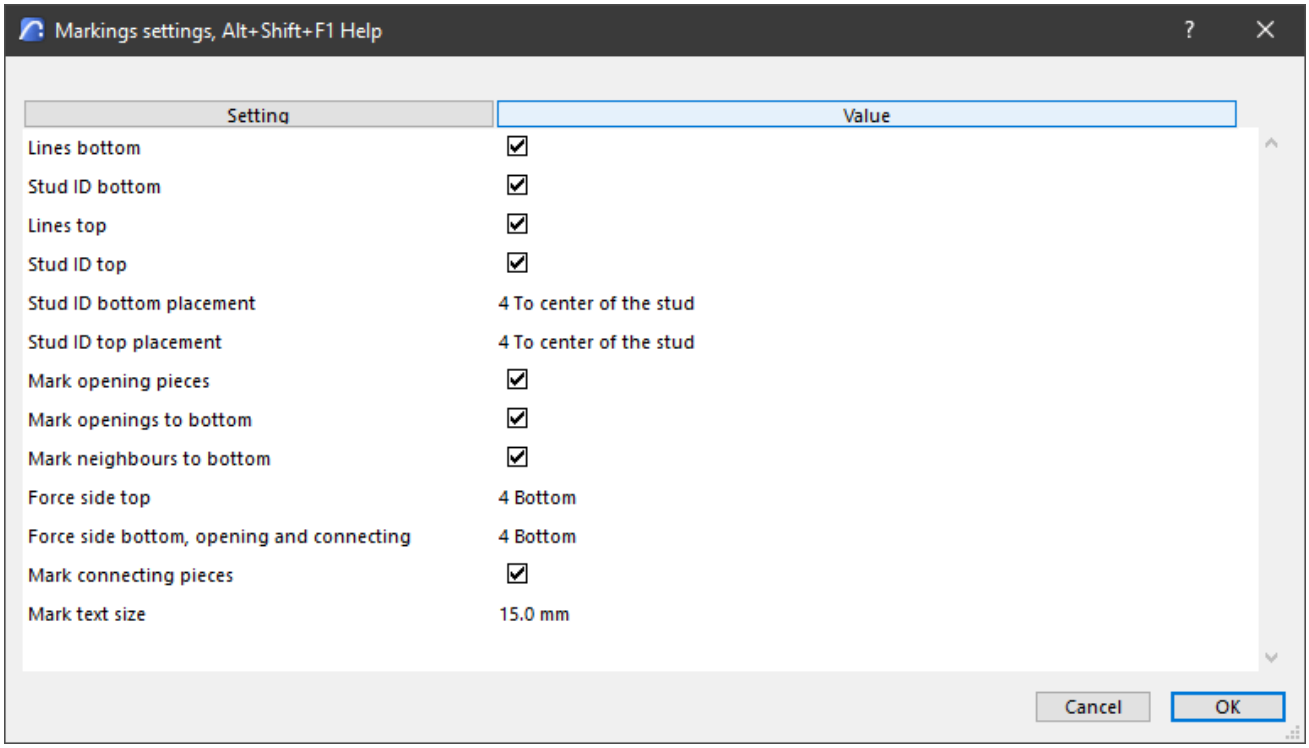
<collisions allowall="1"/>

```
<corner id="NL1_pitkä_Sievitalo" sides="1" bottom="1" top="1" otherid="NL1_I
    <offother anchor="infurther" dist="0.000000" angledeg="90.000000"/>
    <offthis>
        <offlayer type="finish_ext" index="-3" dist="0.054000"/>
        <offlayer type="extstud" index="-2" dist="0.000000"/>
        <offlayer type="boarding_ext" index="-1" dist="0.000000"/>
        <offlayer type="intstud" index="1" dist="-0.250000"/>
    </offthis>
    <planksthis>
        <planklayer type="core" index="0" maxx="0.282000">
            <plank xelem="1" beganchor="beg" x1="0.261000" y1="0.000000
                <collisions allowall="1"/>
                <settings>
                    <objparam name="iFlags" bitfield="1">
                        0
                    </objparam>
```

If the line is already there, ArchiFrame has saved information about allowed collisions and it is not wise to edit the definitions.

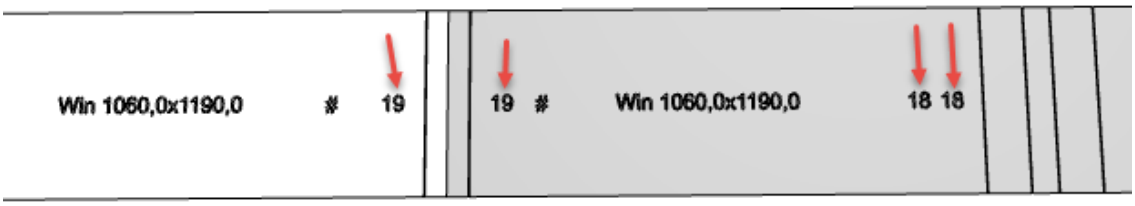
13.16 Element markings

Please see a related video here.

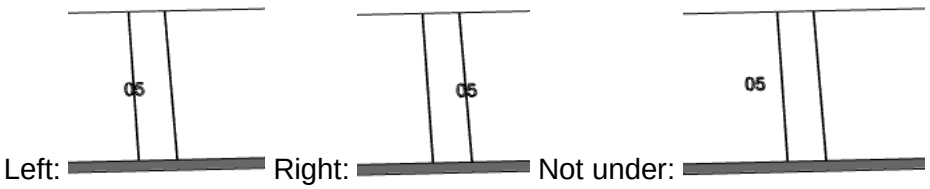


Dialog parts are (example has two separate pieces):

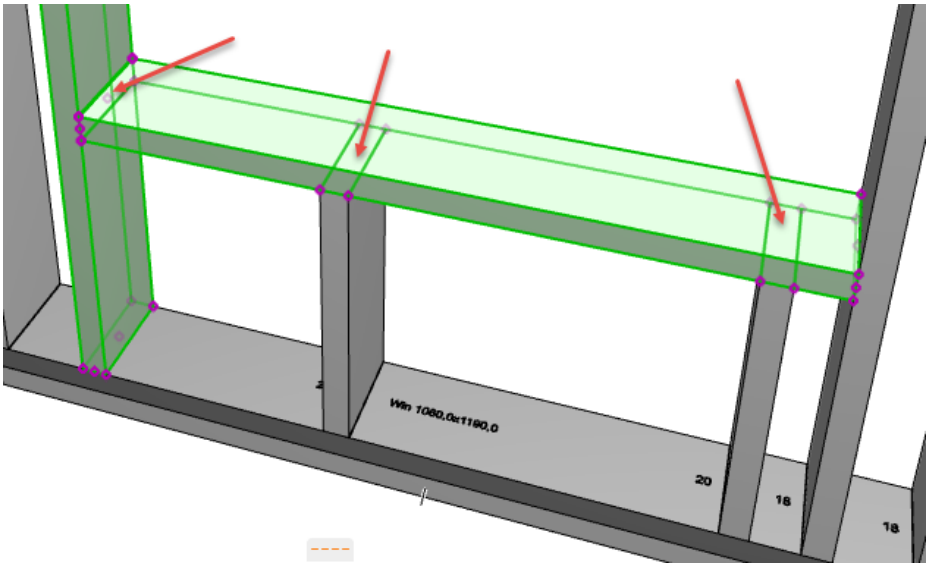
- *Lines bottom/top* creates marking lines for studs at bottom/top plate
- *Stud ID bottom/top* adds the related stud's ID to the marking lines (18 and 19 are stud IDs)



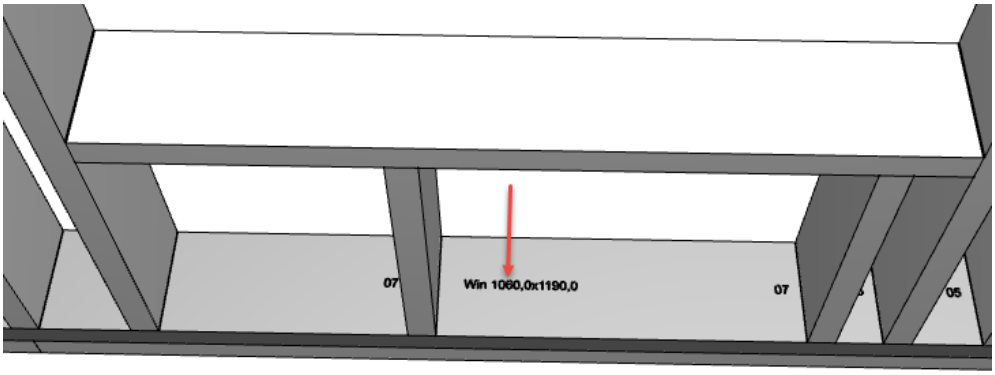
- *Stud ID xxx placement* works like this:



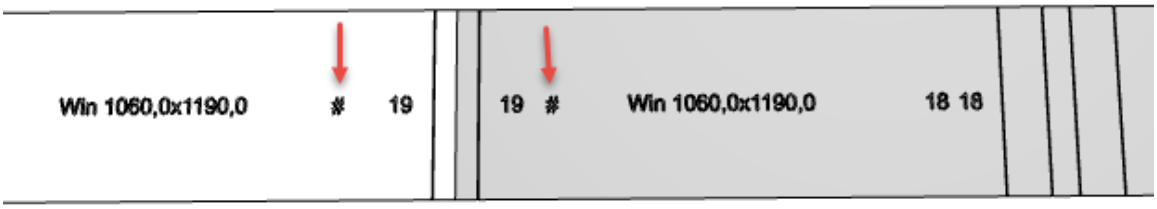
- *Mark opening pieces* adds marking lines also to opening-related pieces:



- *Mark openings to bottom/top* adds opening information:



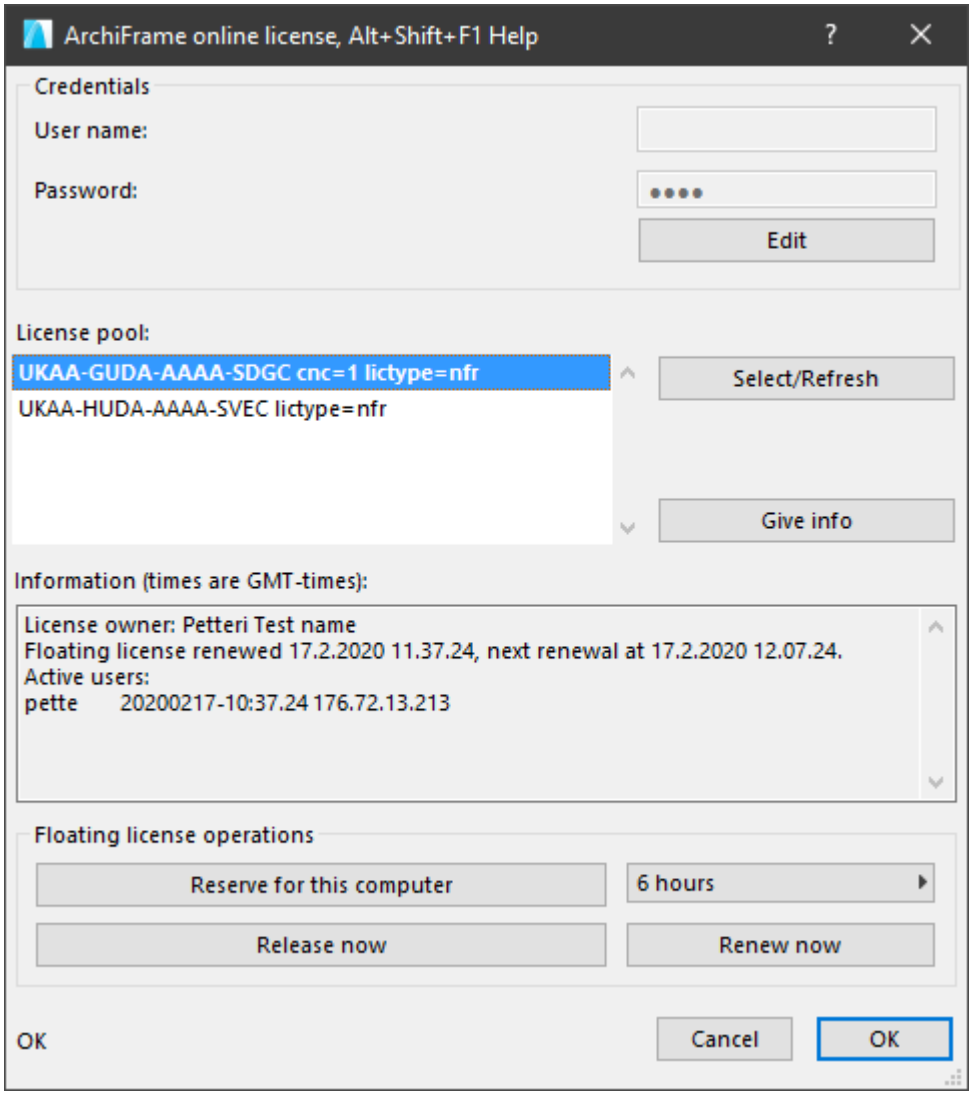
- *Mark neighbours to bottom* adds IDs of neighbour elements to the ends of the bottom plate.
- *Force side* allows user to define the plank surface where the markings are done. Typically the narrow part in front elevation is bottom side:
- *Mark connecting pieces* puts markers to cut planks on the same line. First connection has single marker #, second two ## and so on.



ArchiFrame uses followings characters to distinguish different plank types:

  - = bottom plate
  - # secondary bottom plate
  - + top plate
  - $ secondary top plate
  - * everything else
- *Mark text* size sets the font size used. To avoid overlapping texts this should be close to the size the CNC-output is using.

13.17 Online license



The structure of licensing is that each user has a login and under that free number of different licenses. Each license is floating allowing limited number of concurrent users. In default mode licensing needs Internet connection. However, the license can be reserved for single computer for maximum one month. This is useful if the user does not have Internet connection available for certain time.

ArchiFrame automatically reserves a license when there are any ArchiFrame tool palettes visible and releases when all tool palettes are closed. That way it is possible to use just Archicad without reserving an ArchiFrame license.

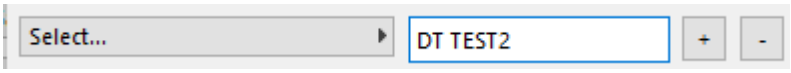
The parts of the dialog are:

- *Username and Password*, the credentials acquired from the reseller.
- *Edit* allows the credentials to be changed. After changes clicking *Login* will login to the license server and list available licenses for new credentials.
- *License pool* lists available licenses showing basic license information. Bolded item is the currently used license.
- *Select/Refresh* will select current license and renew its reservation updating the information below.
- *Give info* justs gives information of currently selected item in the license pool.

- *Information* tells the status of current license.
- *Reserve for this computer* reserves current license for specified period for current computer.
- *Release now* releases the license to the license pool. ArchiFrame tools that require a license are not available in current workstation.
- *Renew now* will switch to normal floating license renewal cycle: The license will be reserved every 30 minutes.

13.18 Preset settings

ArchiFrame gives user possibility to save settings as presets in various places. The preset controls are:



The dropdown list contains all presets and at the bottom three additional operations:

- *Load replacing all presets* will clear current list and add items from given file.
- *Add presets from file* keeps all current items and adds new ones from file replacing similar named existing presets from the file.
- *Save as* saves all presets to a file.

The presets are saved under user specific settings folder with file name depending on where the presets are used.

13.19 Combine elements

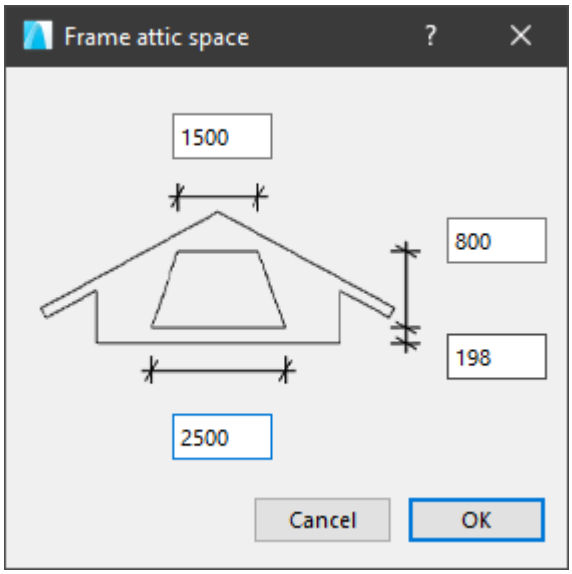
Video showing the steps here:
https://youtu.be/TfuuZhaJztI

Combine elements feature works the same way with walls, floors, and roofs. Combined elements must have the same height in element's y-direction which is up for walls.

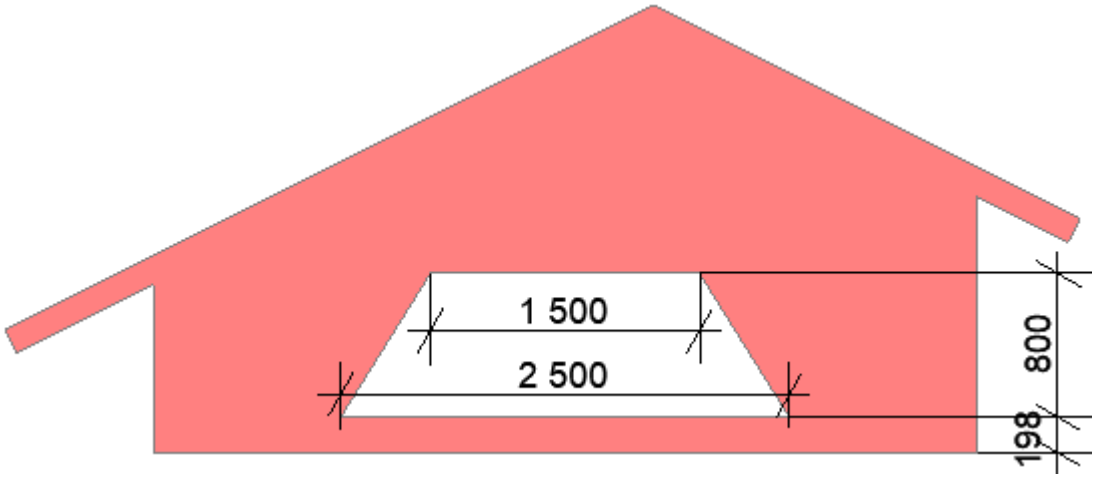
Currently combined element is totally independent – it has no links to the source elements.

13.20 Truss attic space

Defines the space reserved for the attic. For example, an attic like this:



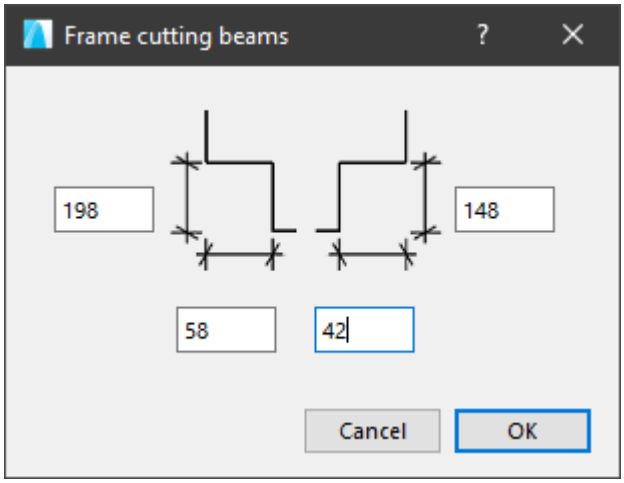
Produces this kind of hole to the truss polygon:



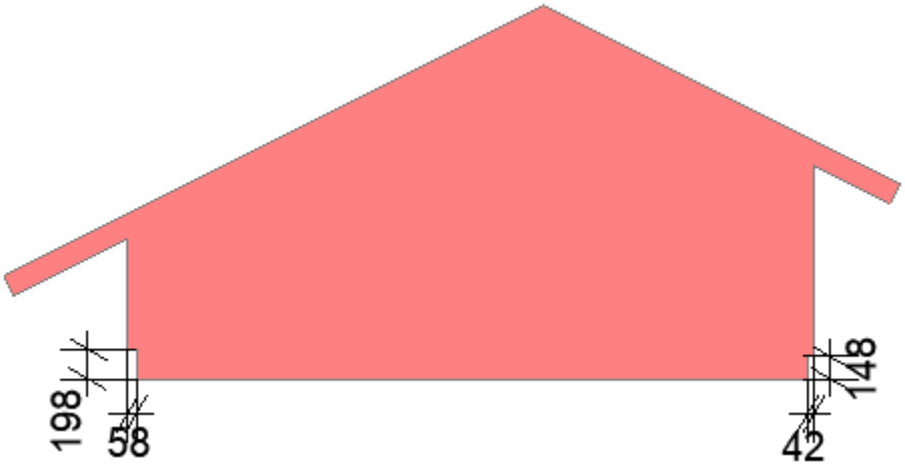
The attic is always horizontal and bottom dimension is to the lowest part of the truss.

13.21 Truss cutting beams

Defines size of the beams to the sides of the truss. For example, these values:



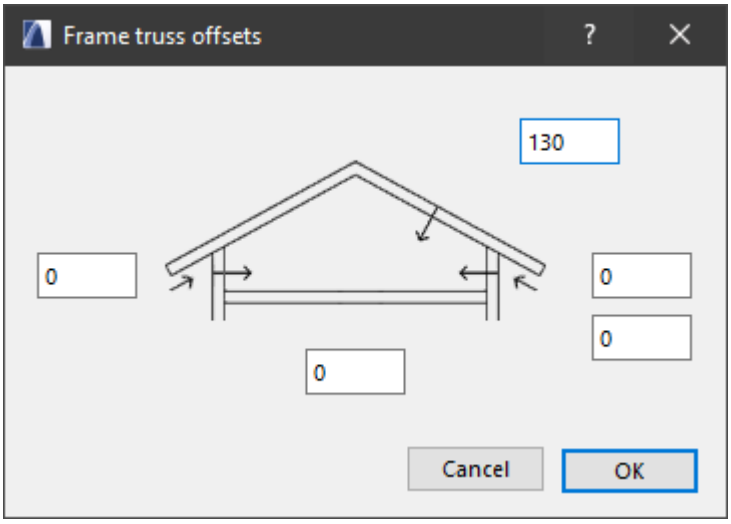
Produce these cuts:

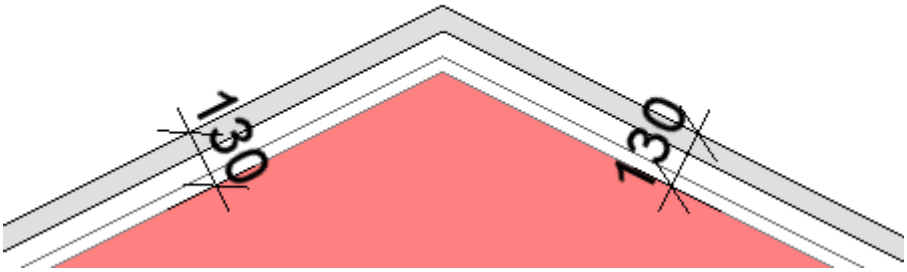


13.22 Truss offset dialog

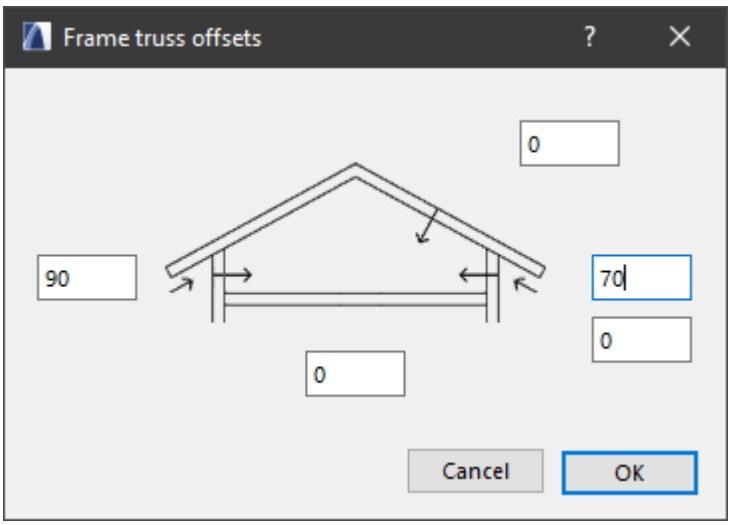
Defines changes from Archicad model or to be more precise from the support heights to the resulting truss. Different fields are like below.

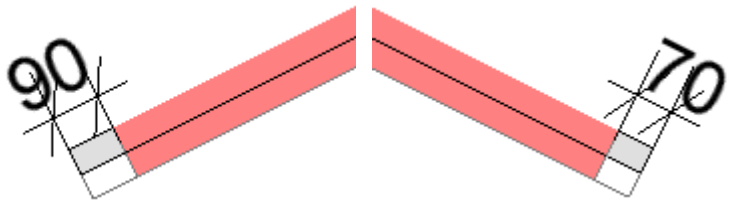
- Top offset:



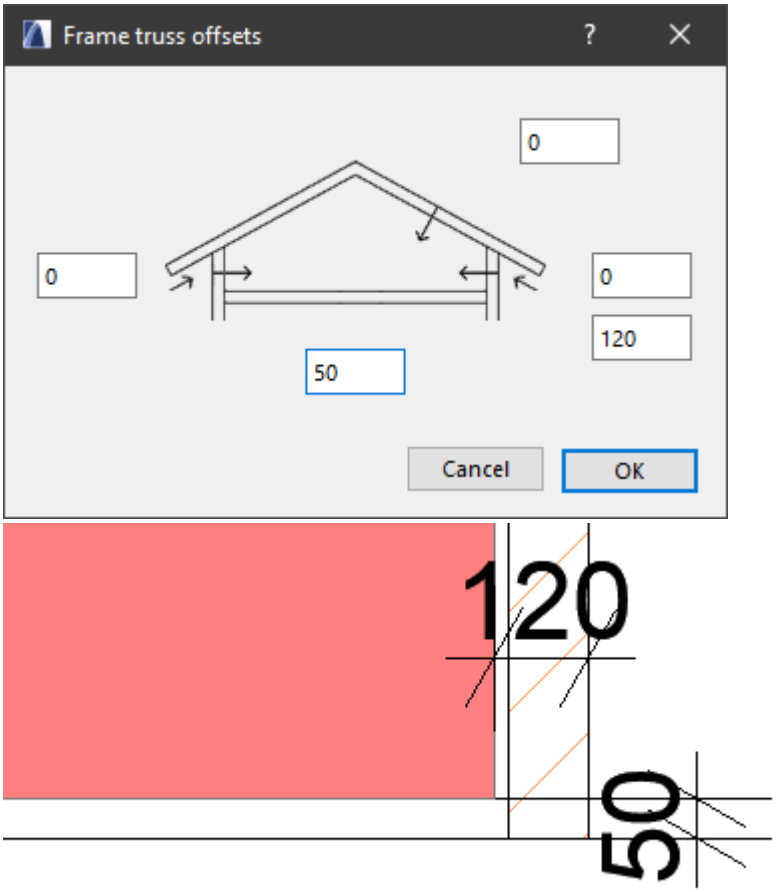


- Eaves offset:

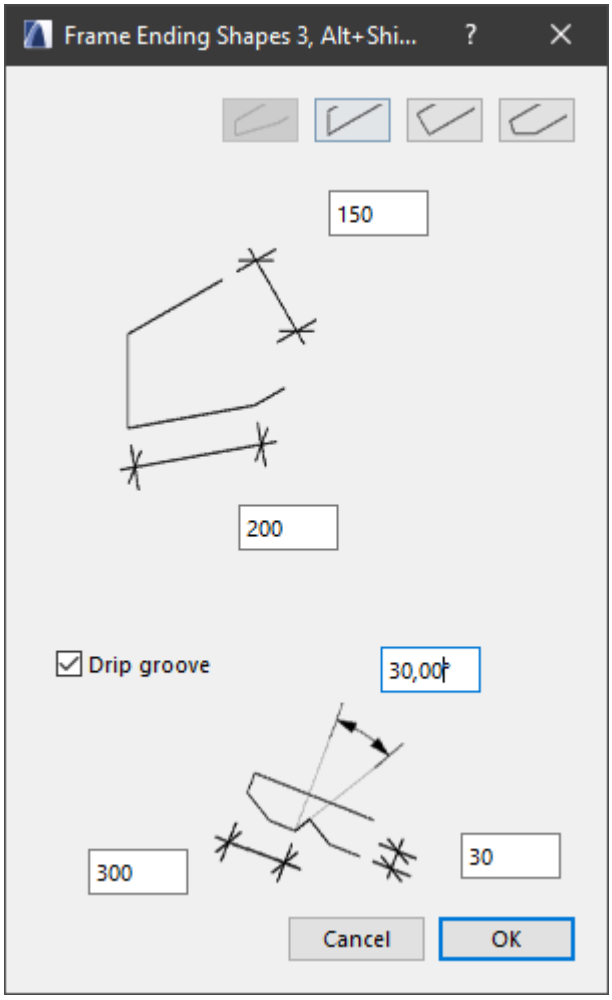




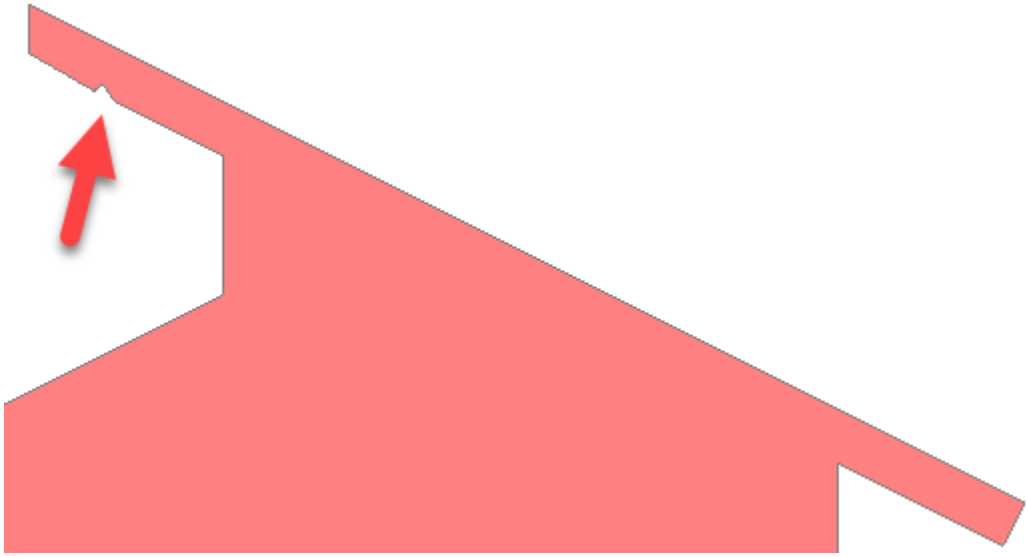
- From bottom and sides of the placement line:



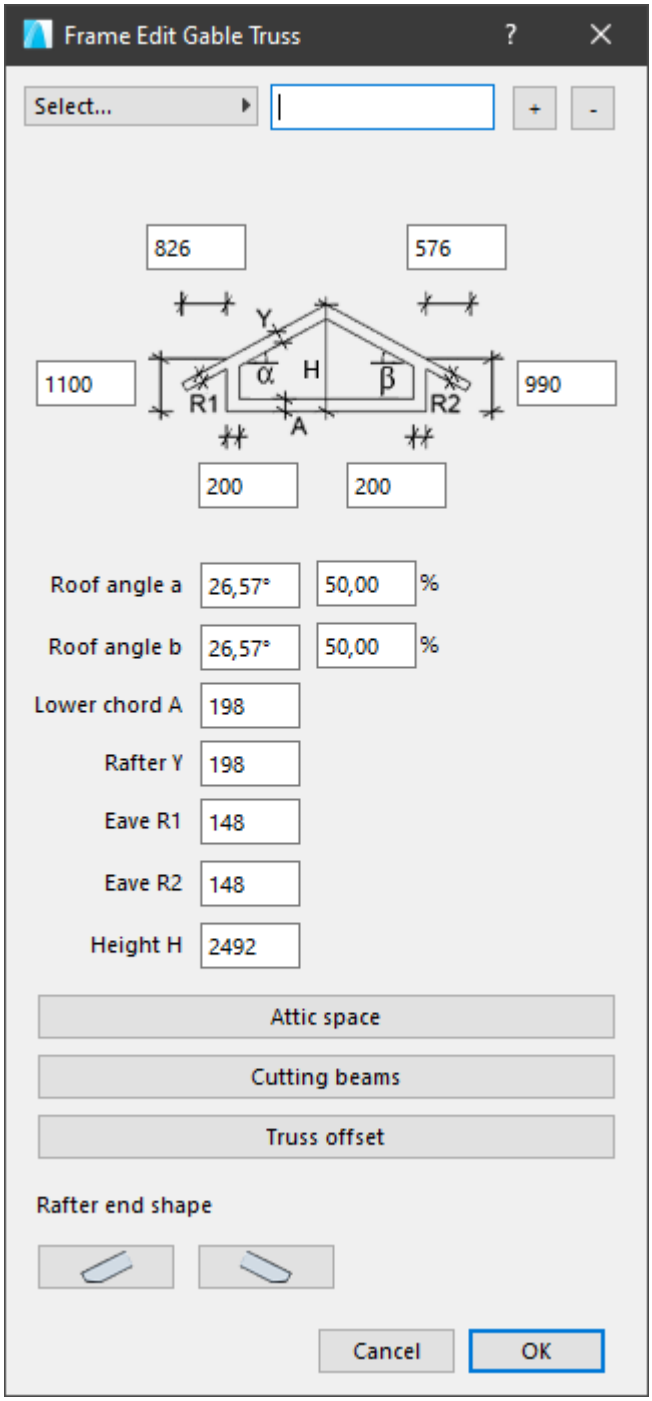
13.23 Truss rafter ending dialog



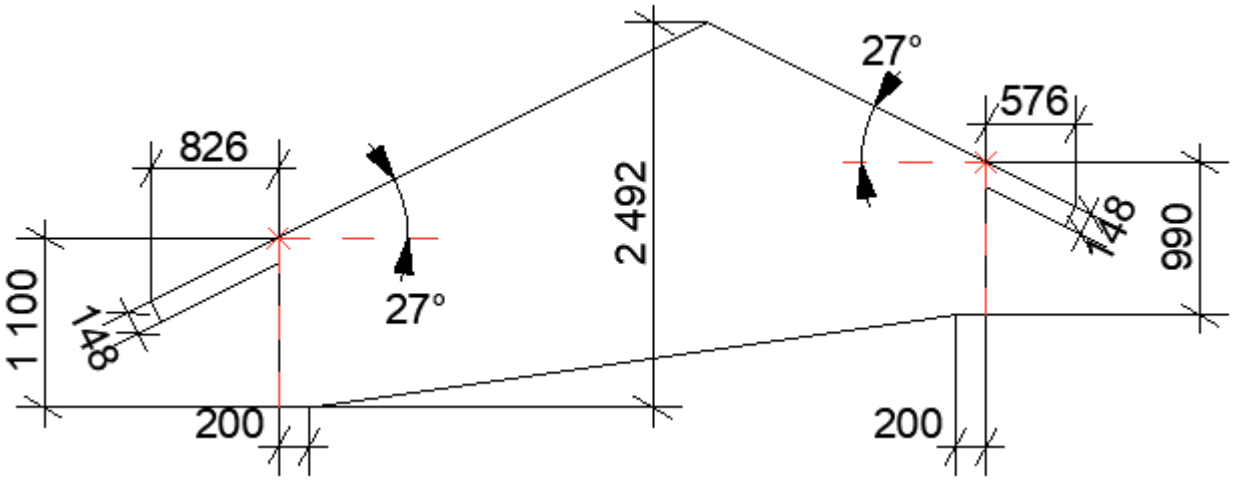
Defines shapes for selected end of the truss. Currently selected shape shows as grayed. Drip groove is meant for mono or clerestory truss to avoid water coming inside. The option is available only for upper ends:



13.24 Gable truss dialog



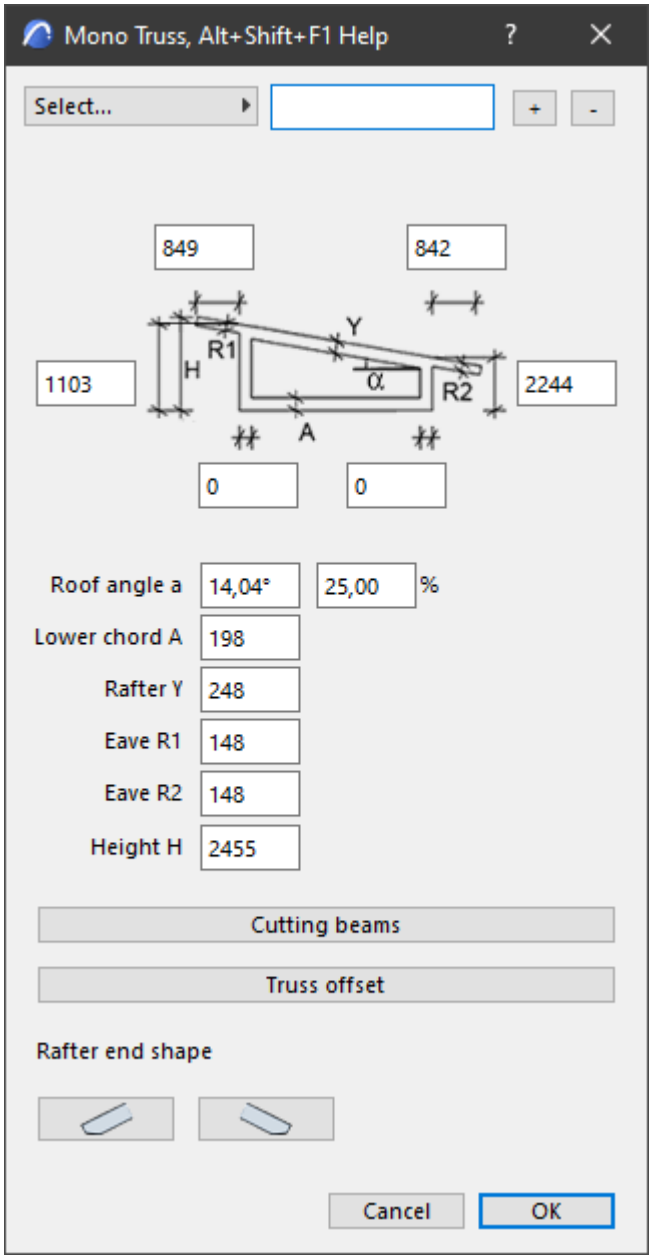
The values in the truss:



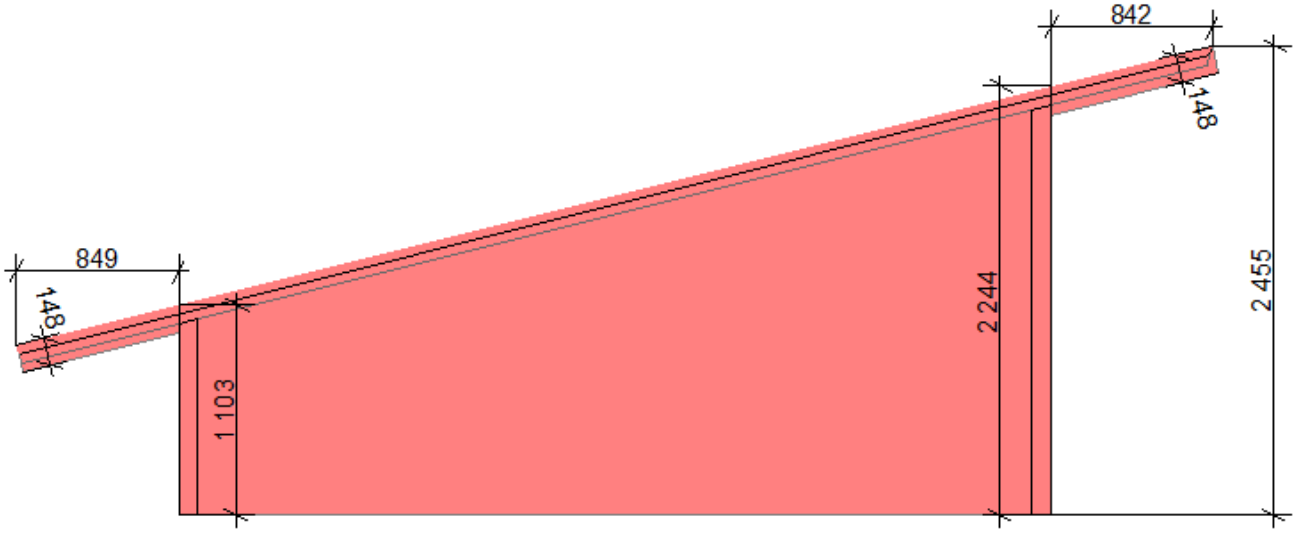
See also:

- Trusses
- Presets
- Attic space
- Cutting beams
- Truss offset
- Truss ending shape

13.25 Mono truss dialog



The values in the truss (Archicad walls and roofs as lines behind):

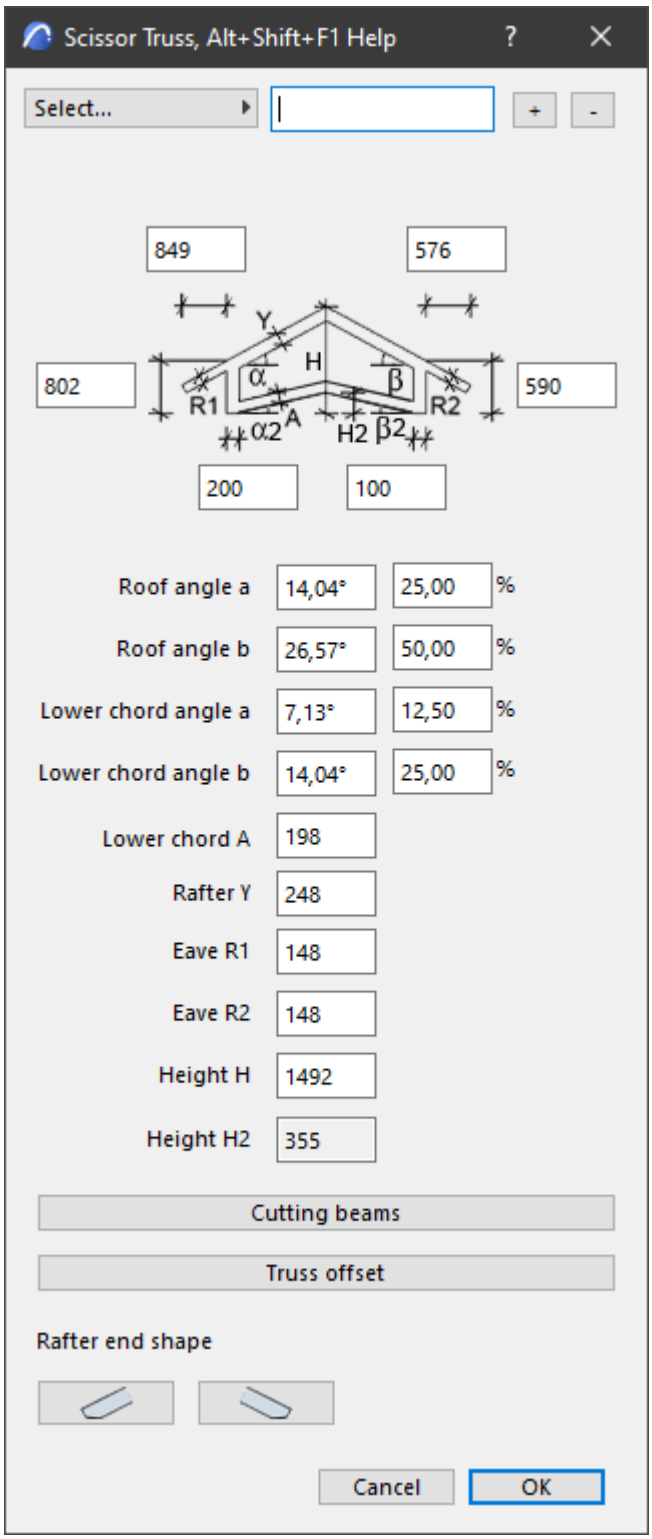


See also:

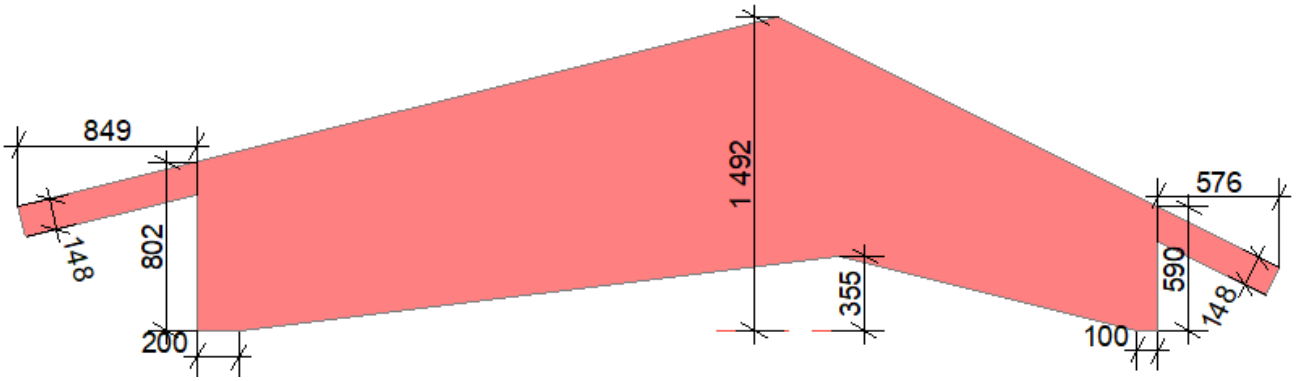
- Trusses
- Presets
- Attic space
- Cutting beams
- Truss offset
- Truss ending shape

13.26 Scissor truss dialog

Please note that in truss dialog the lower angled part cannot be read from the model. Instead the values are entered in the truss editing dialog.


Scissor Truss, Alt+Shift+F1 Help
Select...
+
-
849
576
802
590
200
100
Y
H
α
β
R1
R2
A
α2
H2
β2
Roof angle a
14,04°
25,00
%
Roof angle b
26,57°
50,00
%
Lower chord angle a
7,13°
12,50
%
Lower chord angle b
14,04°
25,00
%
Lower chord A
198
Rafter Y
248
Eave R1
148
Eave R2
148
Height H
1492
Height H2
355
Cutting beams
Truss offset
Rafter end shape
Cancel
OK

The values in the truss (Archicad walls and roofs as lines behind):

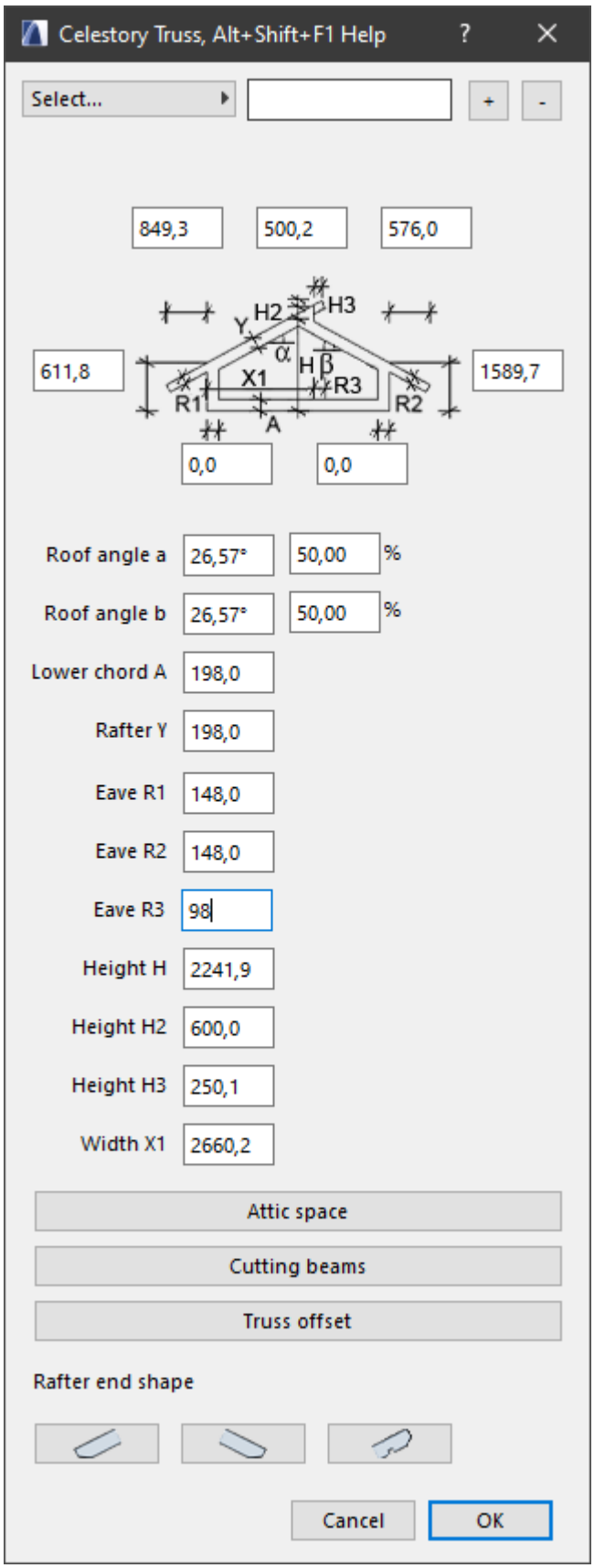


See also:

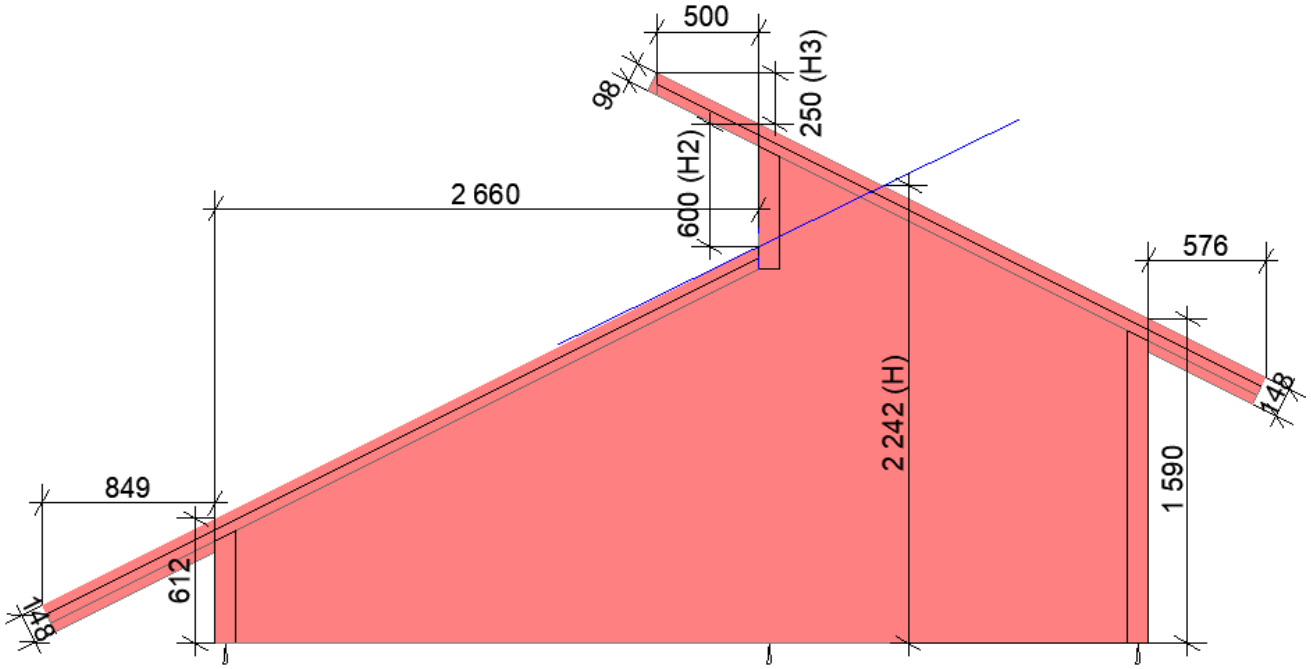
- Trusses
- Presets
- Attic space
- Cutting beams
- Truss offset
- Truss ending shape

13.27 Clerestory truss dialog

Please note that the X1-value must be given manually – ArchiFrame is unable to detect it from the model.


Celestory Truss, Alt+Shift+F1 Help
Select...
+
-
849,3
500,2
576,0
611,8
1589,7
H2
H3
Y
α
H
β
X1
R3
R1
R2
A
0,0
0,0
Roof angle a
26,57°
50,00
%
Roof angle b
26,57°
50,00
%
Lower chord A
198,0
Rafter Y
198,0
Eave R1
148,0
Eave R2
148,0
Eave R3
98
Height H
2241,9
Height H2
600,0
Height H3
250,1
Width X1
2660,2
Attic space
Cutting beams
Truss offset
Rafter end shape
Cancel
OK

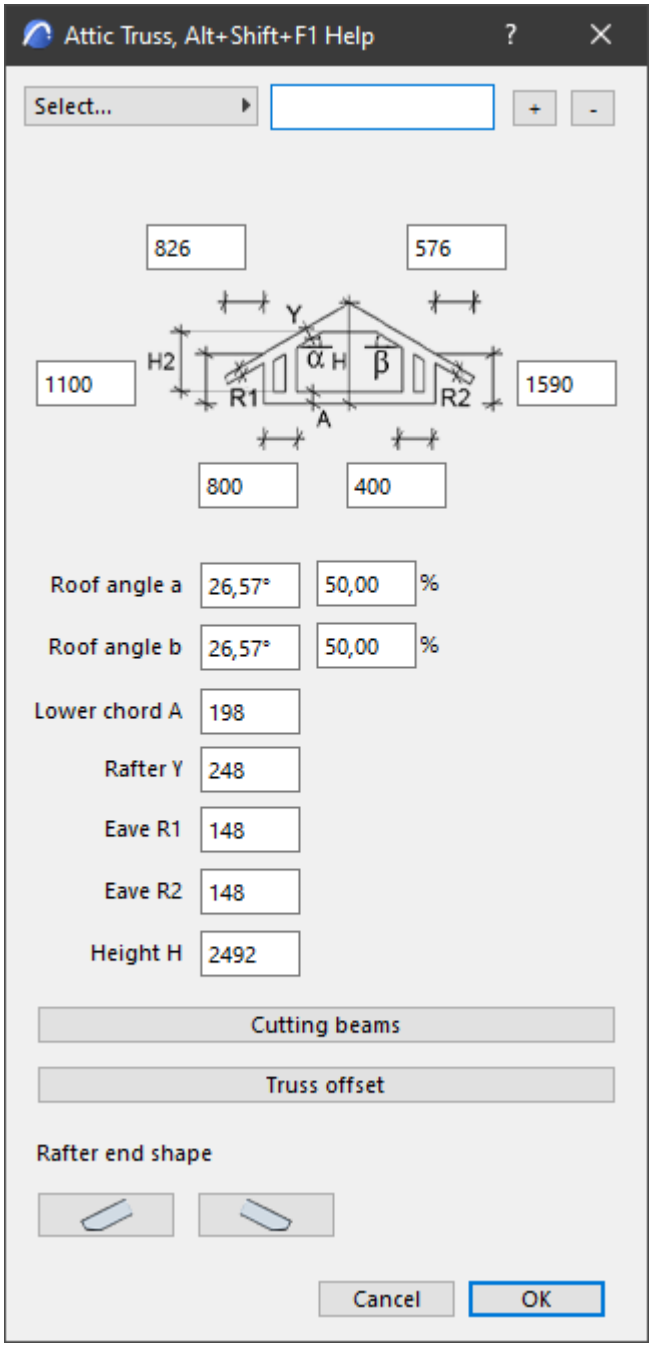
The values in the truss (Archicad walls and roofs as lines behind):



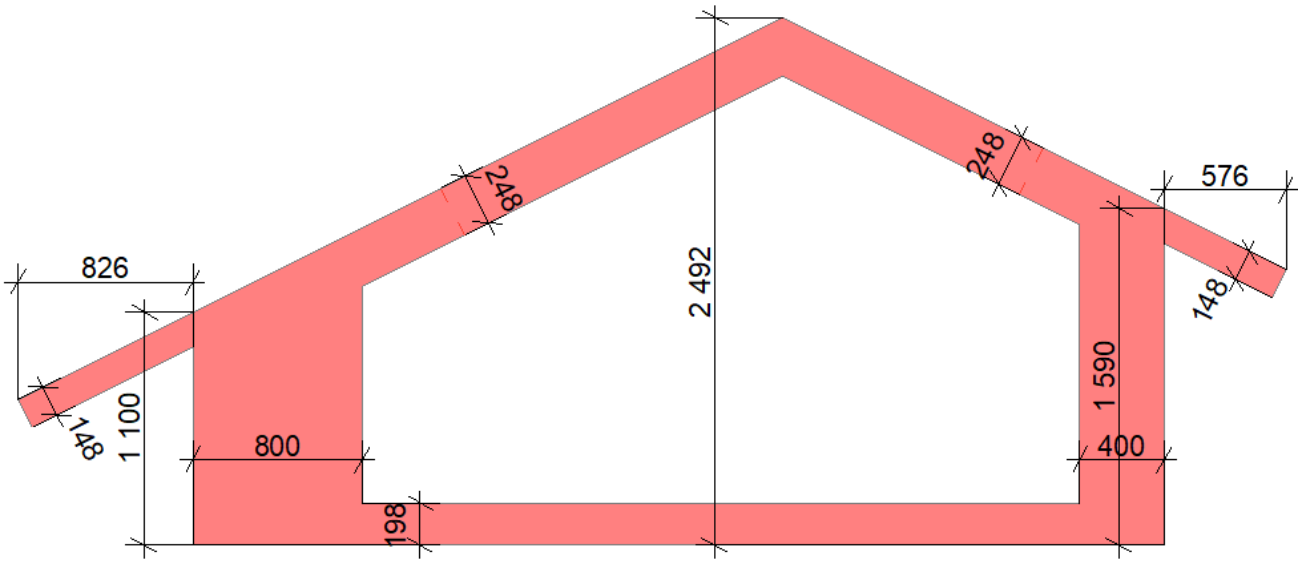
See also:

- Trusses
- Presets
- Attic space
- Cutting beams
- Truss offset
- Truss ending shape

13.28 Attic truss dialog



The values in the truss (Archicad walls and roofs as lines behind):



See also:

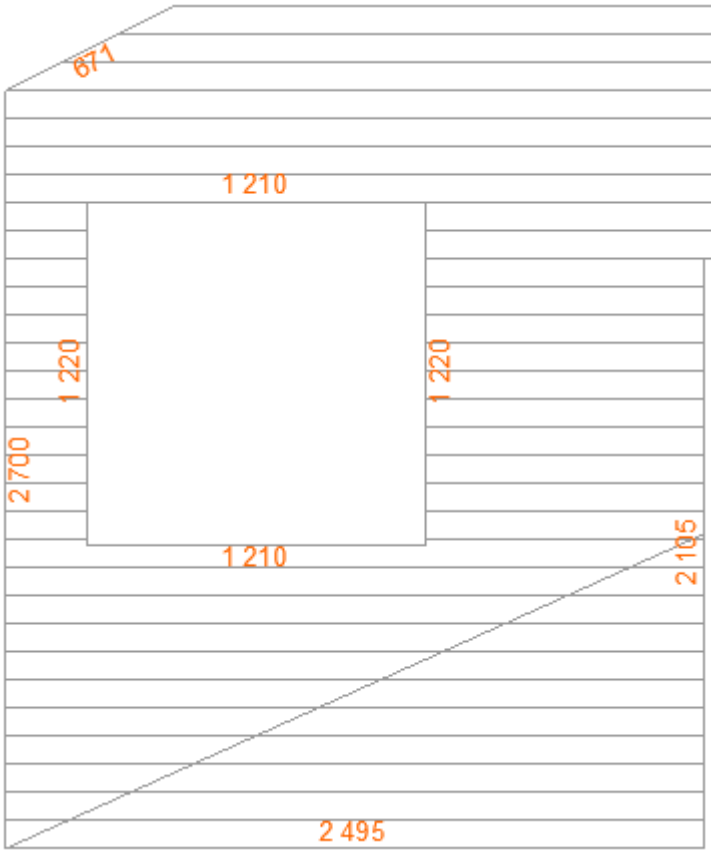
- Trusses
- Presets
- Attic space
- Cutting beams
- Truss offset
- Truss ending shape

13.29 Model View options

These options are saved with Archicad view. The options are:

- *Hide IDs* will hide plank & board IDs in 2D and 3D. One usage case is collision checking – having IDs visible while checking may give false collisions.
- *Hide lengths* will hide lengths.
- *Hide contour lines of ArchiFrameElements, show just ID and type* is handy to show just element IDs with the real framing.
- *Hide nailings* will hide the nailing X-marks in 2D and 3D.
- *Special view* contains view types useful for laser projector and for 3D documents of timber frame.
- *Fast 2D projections without any cuts/machinings, boards without fill* may be used when editing the element elevations to speed up operations.
- *CLT settings* define how to show CLT (cross laminated timber) boards in 2D, 3D, element top projection and element front/back projection. Value <=0 means do not show that part, nonzero will show it (bigger numbers than 1 are reserved for future options). It is possible to override the visibility of all parts in every separate board. In that case negative value should be used here to tell the style but not to produce the part as default:
    - *Layers* defines if the CLT board layers are shown (nonzero) or board is drawn as single block (0).
    - *Direction arrow* defines whether the front & back grain directions are shown.
    - *Marking* defines if the marking text from the object is shown.

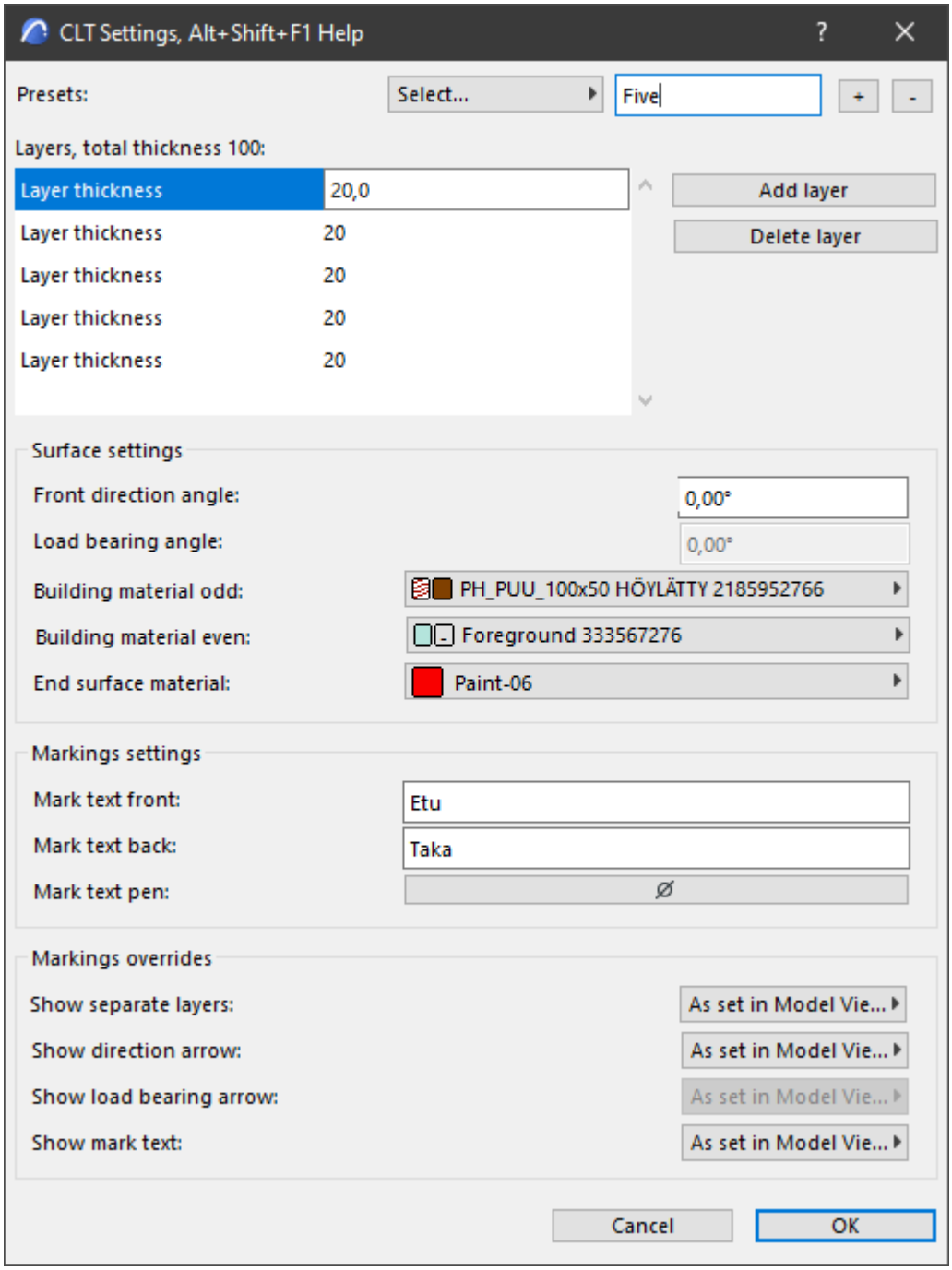
- *Show surface direction with lines in 2D* selected will show the grain direction in floor plan with lined hatch:



- *Ifc-export* will set these:
  - Resolution for circles (drillings) to be at least 36 segments per 360 degrees
  - Disable ArchiFramePlank to add a plane to each side surface which makes Archicad to add lines between two touching object with the same surface material.

See also CLT Settings dialog.

13.30 CLT Settings dialog



Tutorial video here.

Dialog parts are:

- Presets
- *Layer list* shows all layers and their thicknesses.
- *Add layer* adds new layer before current one.
- *Delete layer* deletes current layer.
- *Surface settings* define the front layer grain direction. Zero is vertical and 90 degrees is horizontal. Please note that the texture image should be vertical to correctly visalize ArchiFrame CLT-board.
- *Load bearing angle* is currently unused
- *Material odd* is a building material that is used in the front layer and then in every other layer.

- *Material even* is a building material that is used in the second layer and then in every other layer.
- *End surface material* is shown at the end of the layer and should look something like this:



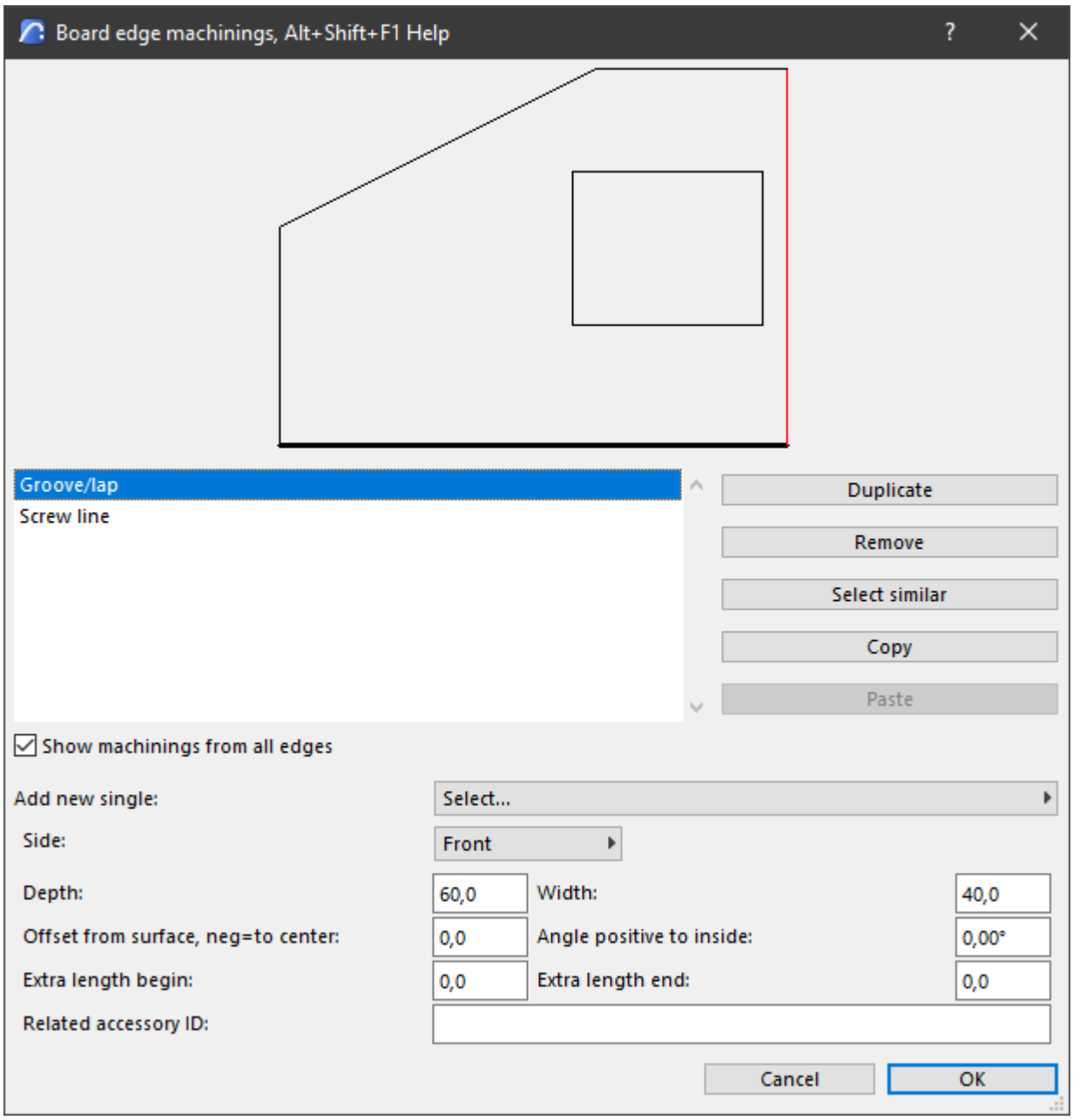
- *Marking front and back* are the texts that are shown in the corresponding sides of the CLT-board.
- *Mark text pen* is the pen used to render the text
- *Marking overrides* can be given to override current CLT-related model view options for a single board.

13.31 Board edge machinings dialog

Tutorial video here.

Board edge machinings are connected to board edges. For example, this edge has a groove and a screw line:





The board polygon's contour line is always counterclockwise and the holes are clockwise. That way the inside of the polygon is always on the left-hand side of the edge.

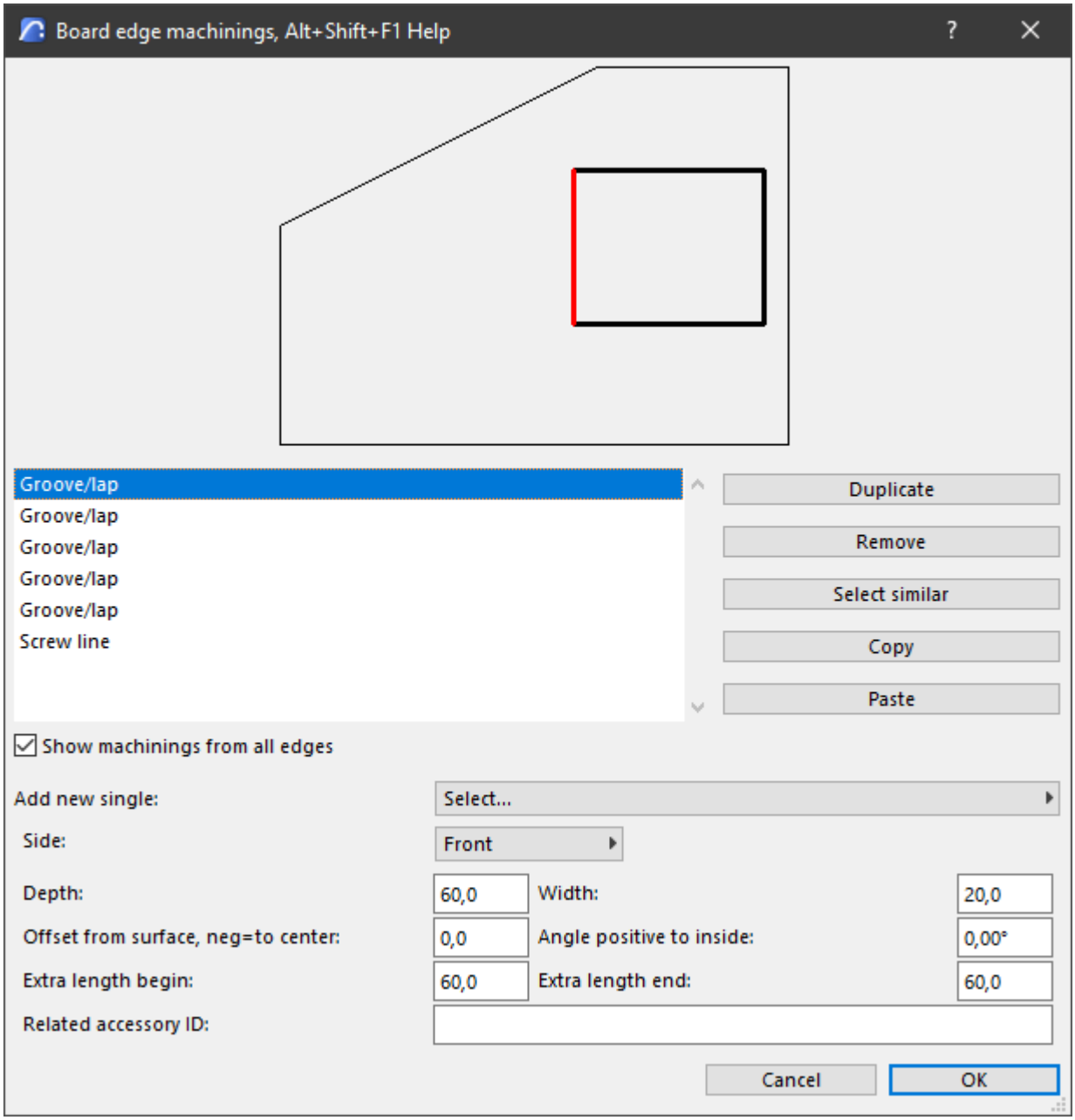
Parts of the dialog are:

- Top shows the shape of the board. The edge for machining selected from the list (Groove/lap) is shown with red colour. Heavy black or red colour shows the active edge where new machining will be added. It is possible to select multiple edges by pressing shift while clicking edges.
- The machining list shows all machinings from selected edge or all.
- *Show machinings from all edges* will list all machinings of the board if checked, and if unchecked, just machinings from selected edge are listed.
- *Duplicate* will duplicate current item to be further edited and finally added.
- *Remove* removes current item.
- *Select similar* selects all machinings that are like the selected one.

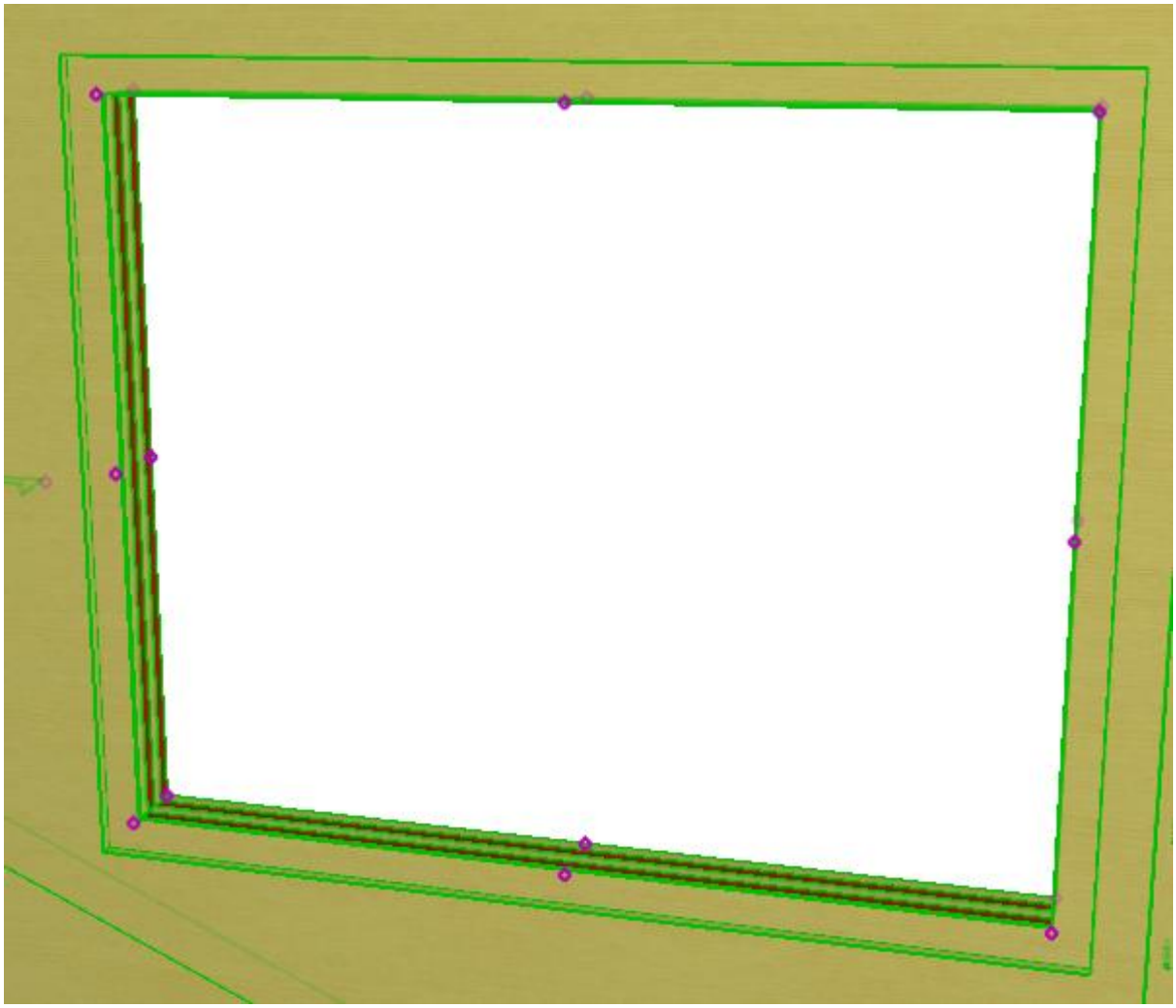
- *Copy and Paste* are used to copy machinings from another board or edge to new boards or edges.

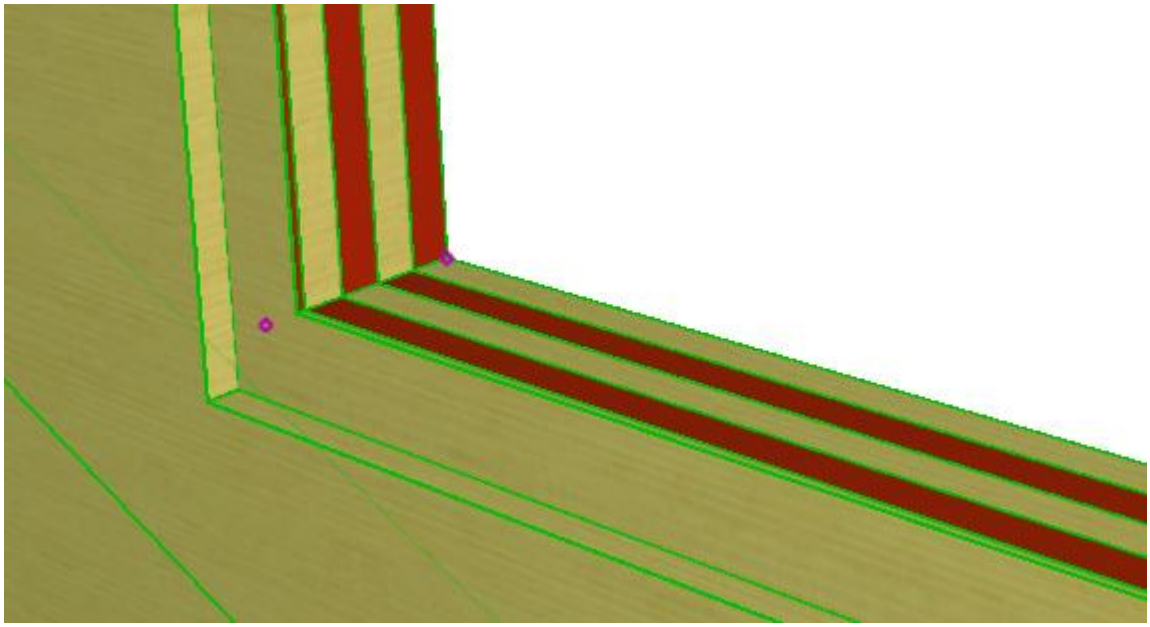
13.31.1Groove/lap settings

Settings like below:



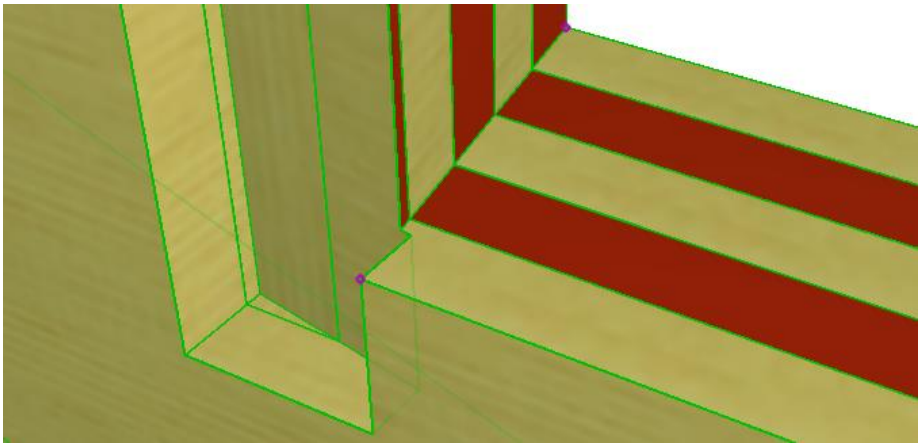
Produce this kind of cuts around the window:





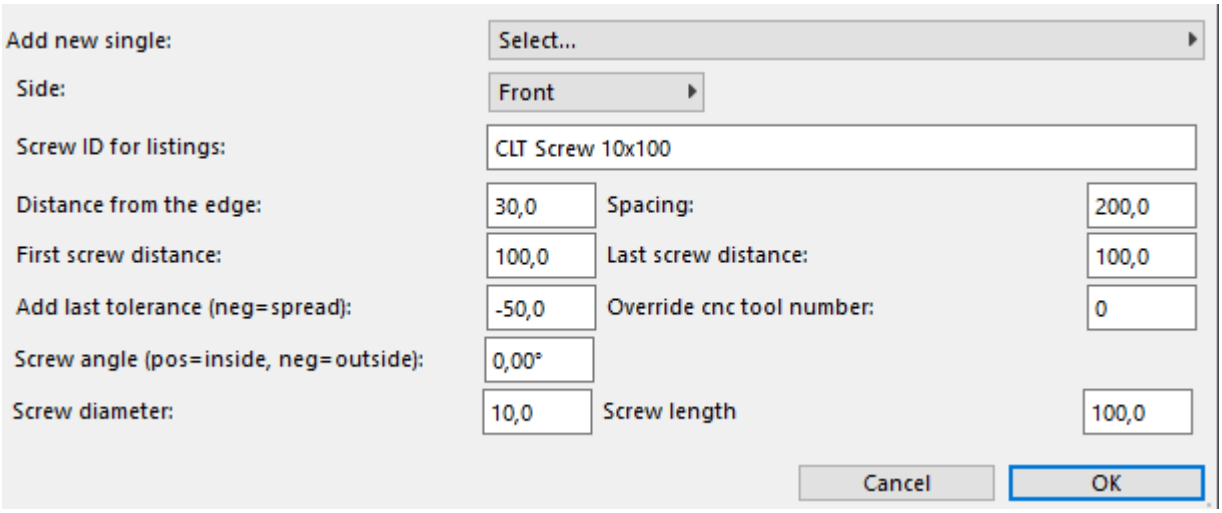
- *Side* is the anchor side of the groove.
- *Depth* is the depth form the board edge inside the board. In the above image depth is 60 mm and width 20 mm.
- *Width* is the width of the groove looking to the board edge

- *Offset from surface* moves the groove into inside the board. Any negative value will put the groove to the middle of the board.
- *Angle positive to inside* tilts the groove, here used value 10 deg:



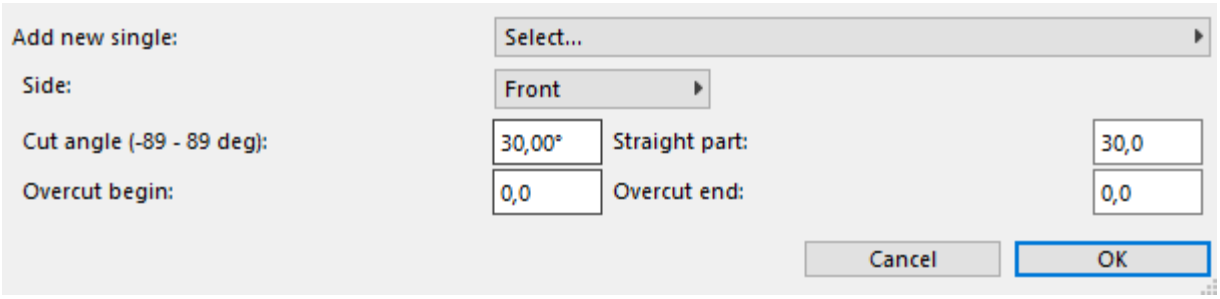
- *Extra length begin/end* extends groove by given value from the edges. For the window the value of connecting groove's depth is used.
- *Related accessory ID* is used for example to calculate connection pieces between the boards. In ArchiFrame's summary listing single groove is calculated as 50% of its length and 0.5 pieces since there is a pair groove somewhere.

13.31.2Screw line

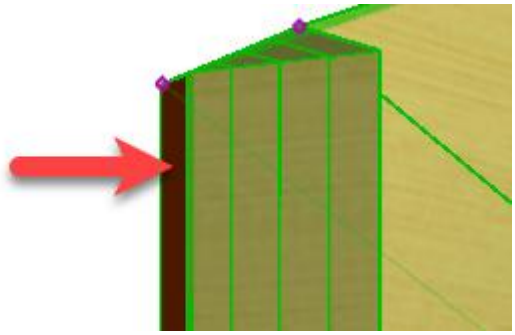


- *Screw ID for listings* is used for quantity takeoffs.
- *Spacing* defines spacing of the screws or if spreading with even spacing, the maximum spacing that can be used.
- *First/last screw distance* define the distance of these screws from the edge's begin and end.
- *Add last tolerance* defines the maximum distance from the last screw to the end of edge (taking last distance into account). If the distance is more, there will be extra screw. Negative value instructs to use even spacing not exceeding given spacing value.
- *Override cnc tool number* is used when producing cnc from the board.
- *Screw angle* defines angle, zero is straight through.
- *Screw diameter and length* are for future. ArchiFrame may visualize the screws later.

13.31.3Angled cut



- *Side* defines which side the cut is anchored.
- *Cut angle* is the angle, positive cuts the anchor side more and negative the opposite.
- *Straight part* is the uncut part of the edge:

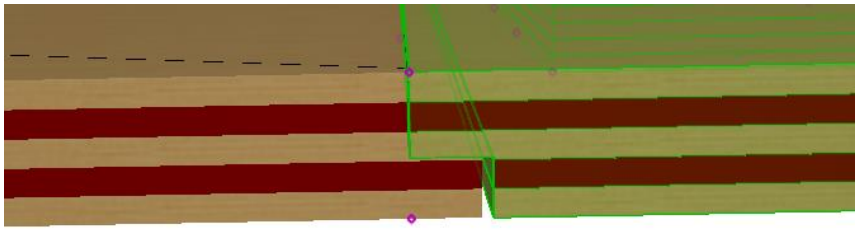


- *Overcut begin/end* defines how much the cur should be extended to cut away everything needed. In above example some overcut is needed.

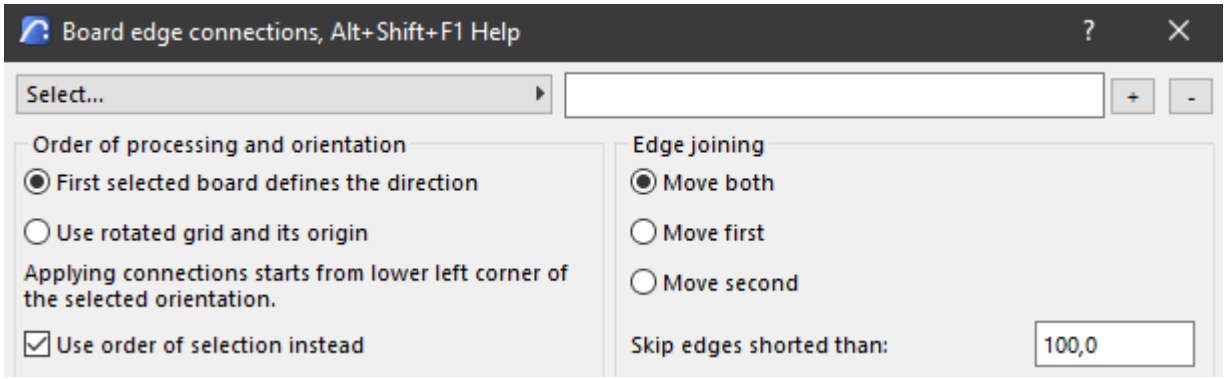
13.32 Board edge connections dialog

Tutorial video here.

Board edge connections dialog is used to connect boards together for example like this:

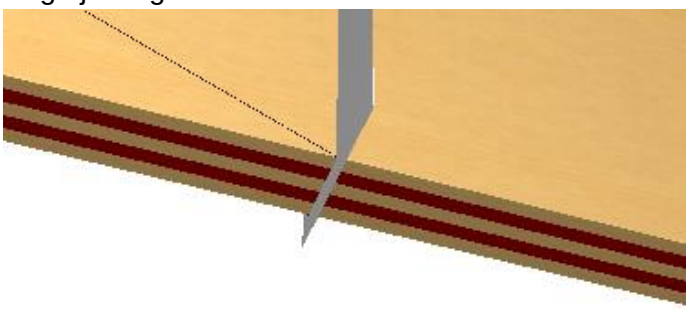


Currently the operation is limited to parallel boards only. Corner and wall & floor -connections must be made using the plank tools like earlier.

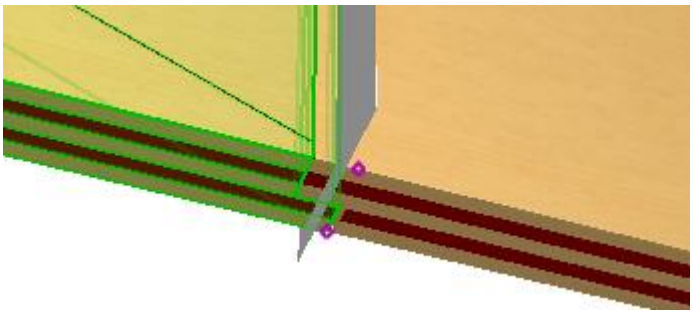


- Preset settings, see help here.

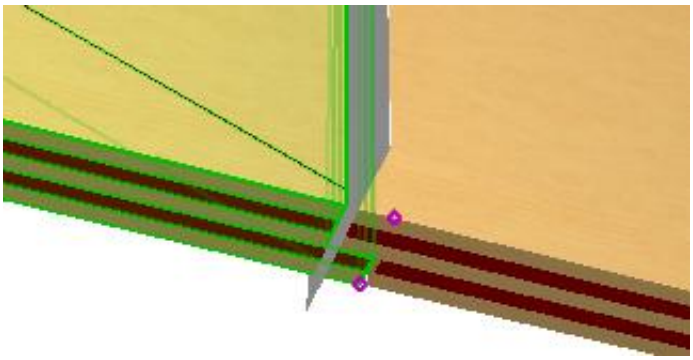
- *First selected board defines the direction* tells that processing geometry world is defined by selection.
- *Use rotated grid and its orientation* is useful when working with floor boards.
- *Use order of selection instead* makes ArchiFrame to make connections in order of selection. This is typically the most logical way to work.
- Edge joining is like this state before and left-hand board is the first to be processed:



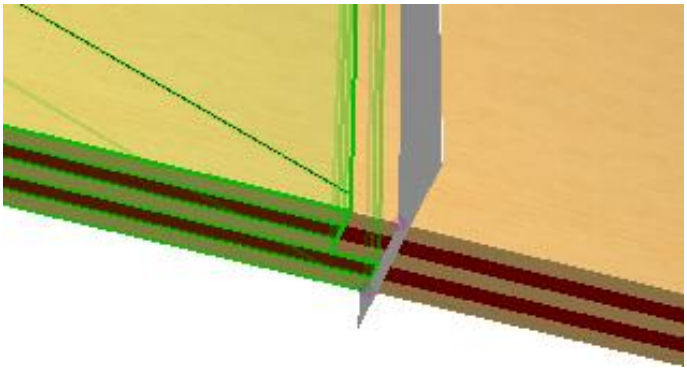
- *Move both:*



- *Move first:*



- *Move second:*

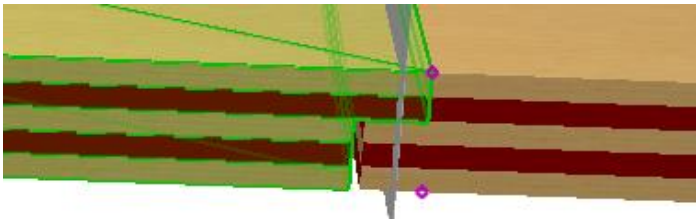


- *Skip edges shorter than* causes ArchiFrame not to process edges shorter than given value.

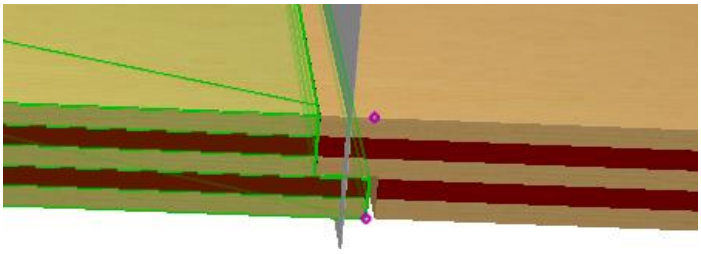
13.32.1Lap joint settings

Lap joint types for the target piece (first to process) looking at the front of the board

- *Lap joint front:*



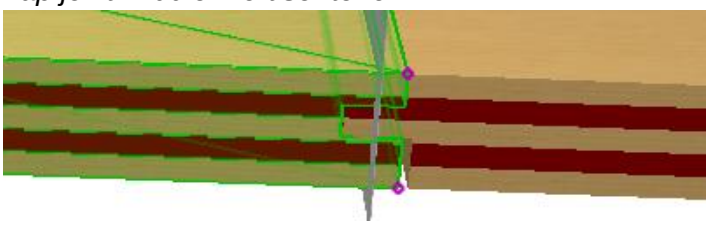
- *Lap joint back:*



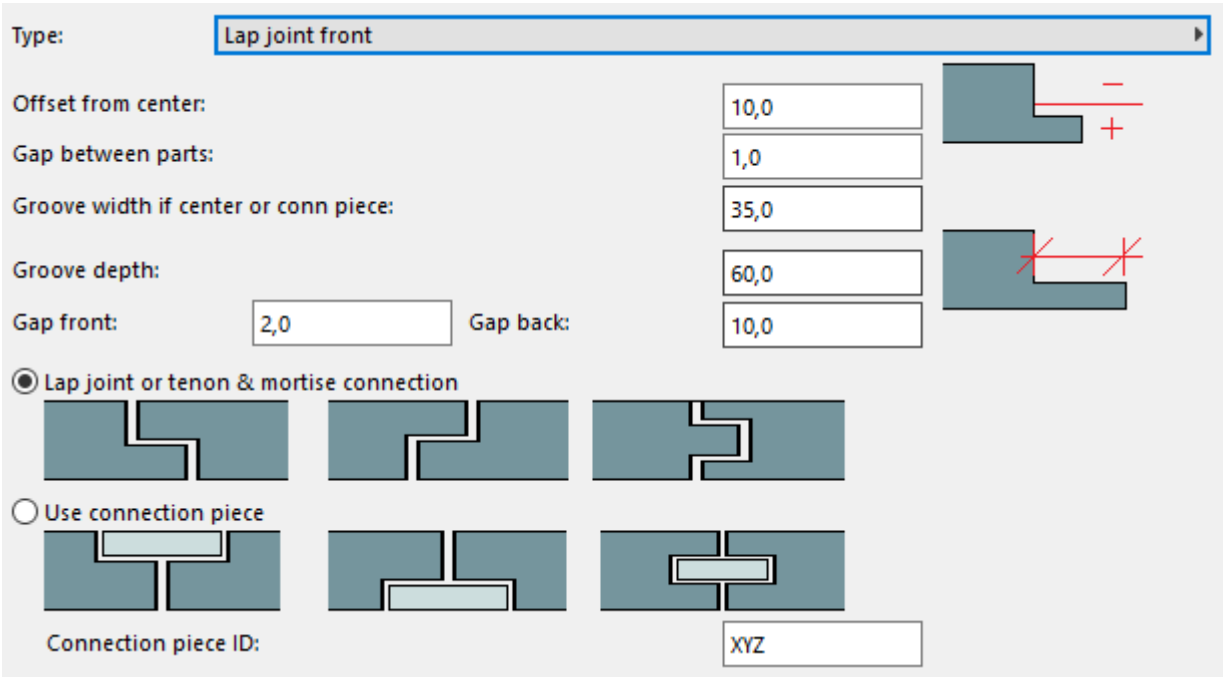
- *Lap joint middle tenon+mortise:*



- *Lap joint middle mortise+tenon:*

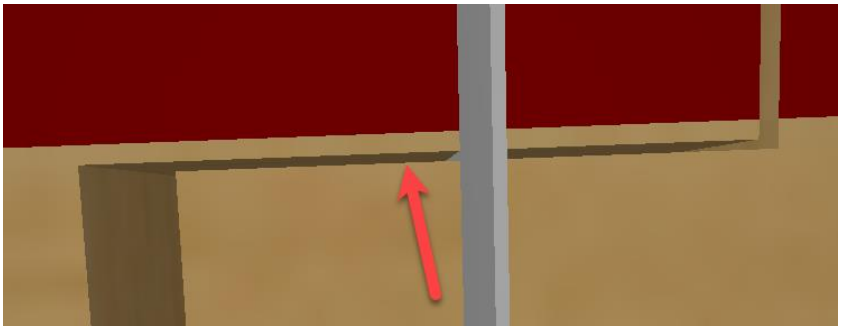


Above example connections are done with these settings:

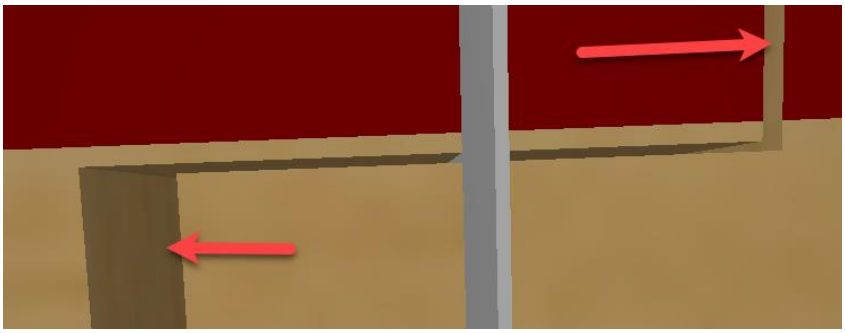


- *Offset from center* makes the tenon part smaller if at front/back. With tenon & mortise it moves tenon towards the front surface.

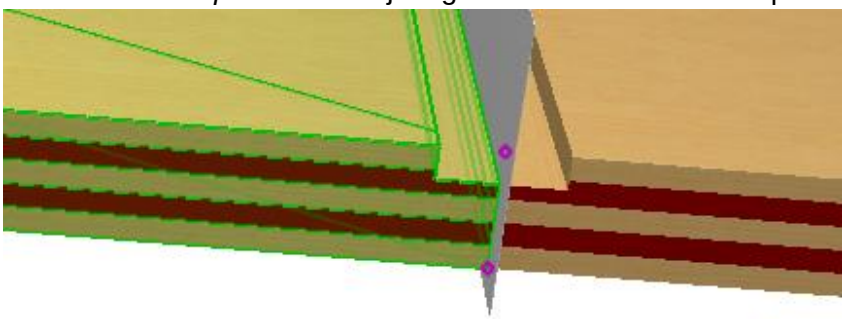
- *Gap between parts* is this dimension:



- *Groove depth* as in the dialog's image
- *Gap front and back* are these. The working geometry defines which is front and which is back.



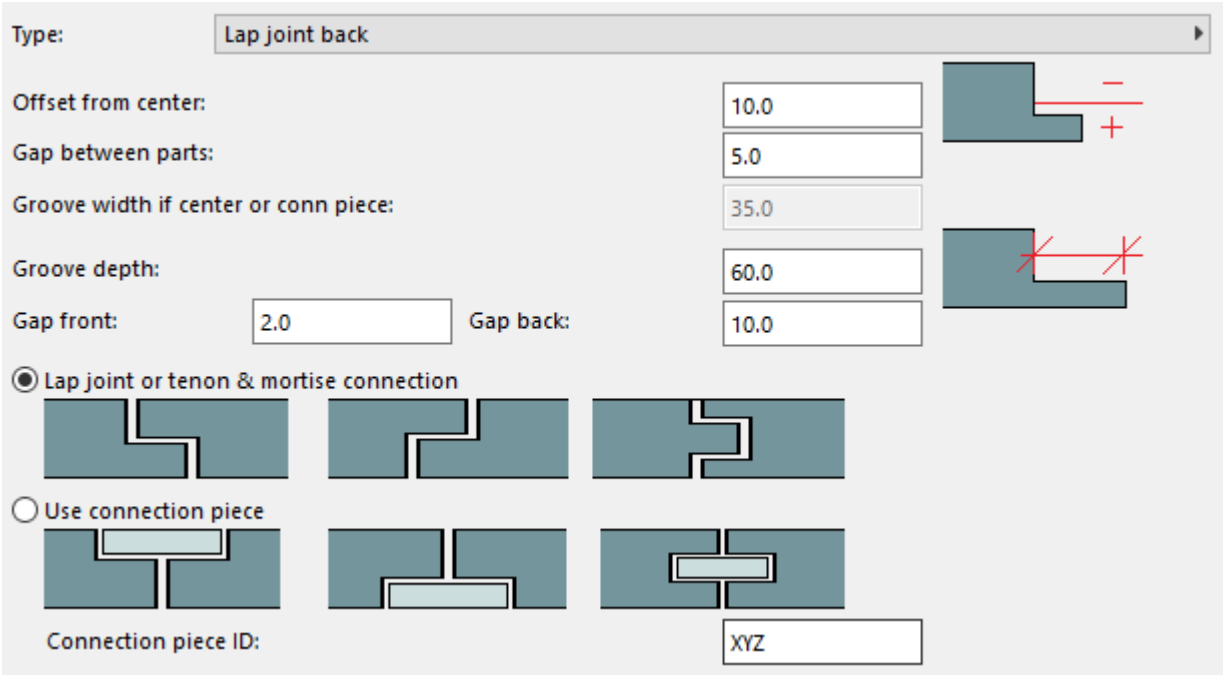
- *Use connection piece* makes just grooves for a connection piece:



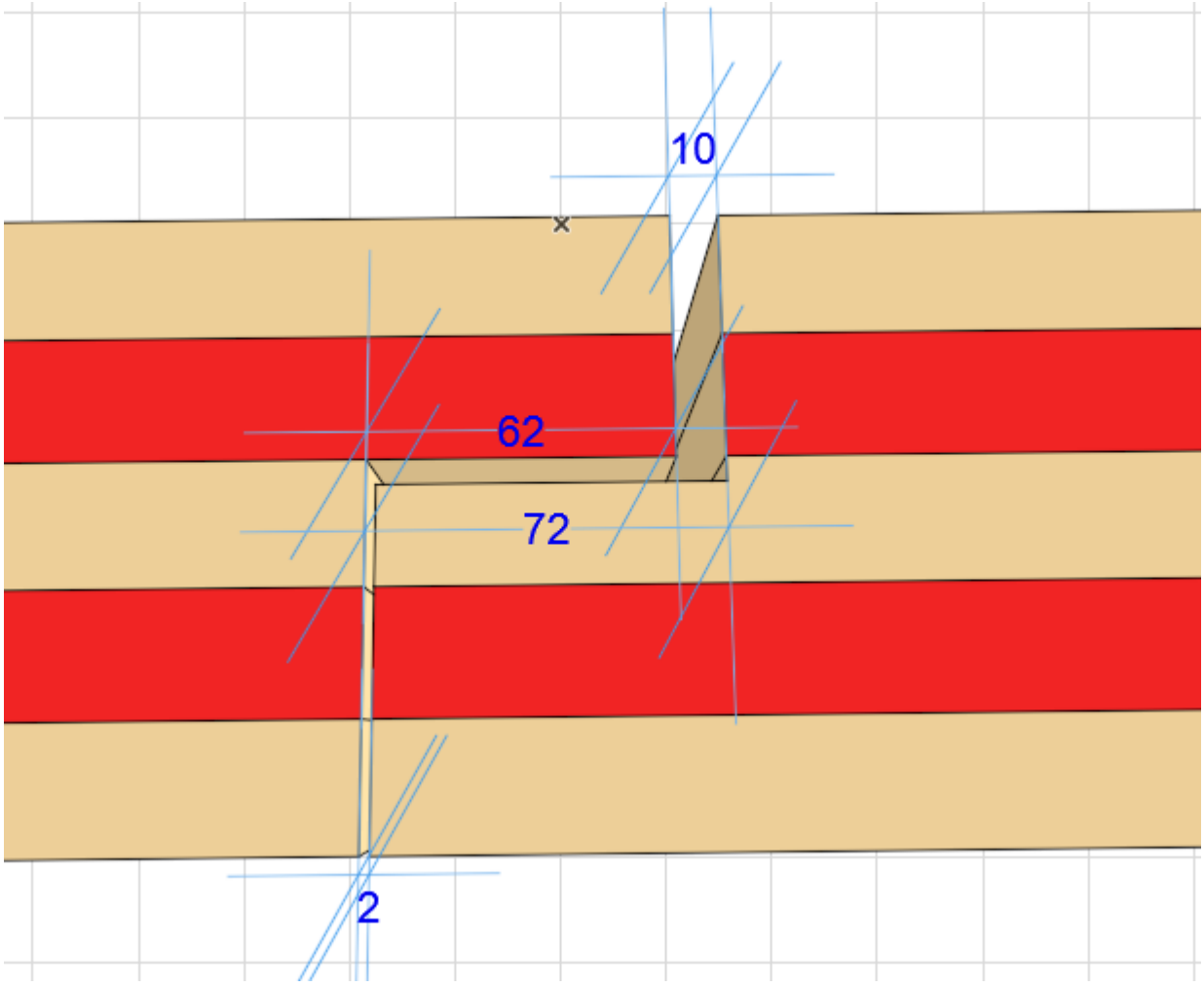
- *Connection piece ID* is used in ArchiFrame's summary Excel-listing. Each groove produces half of its length and 0,5 pcs of the piece to the listing. That is because the connection piece extends over two boards.

13.32.2Lap joint without connection piece settings

These settings:



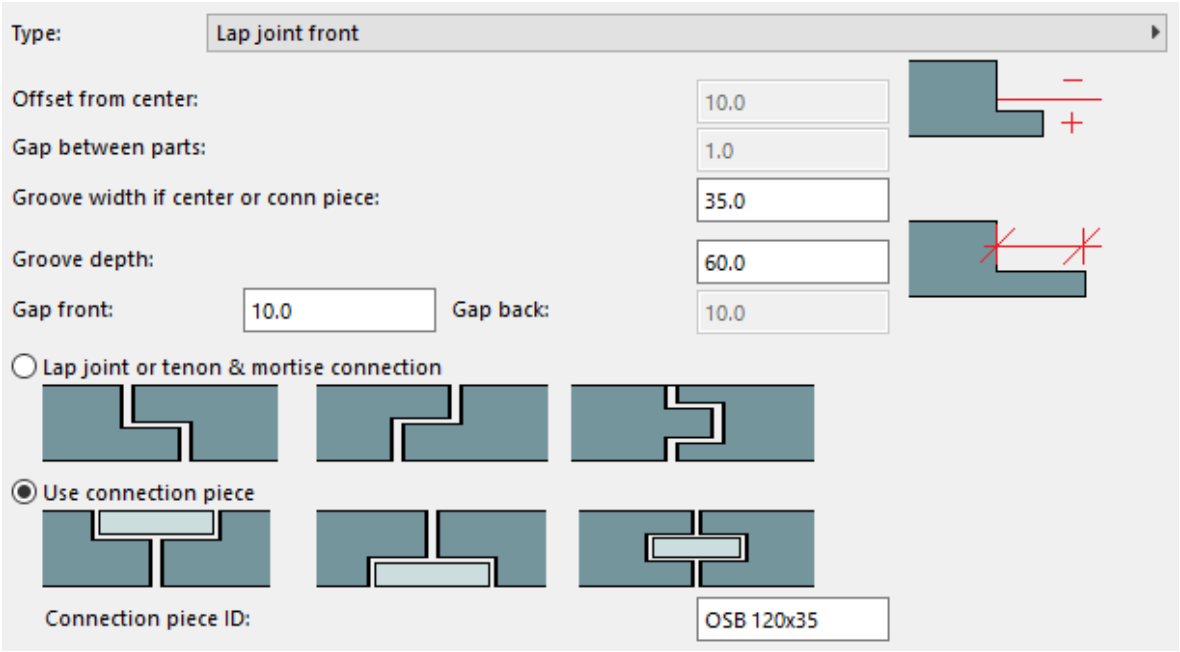
Produce this kind of result (in the image below the front side is at the bottom):



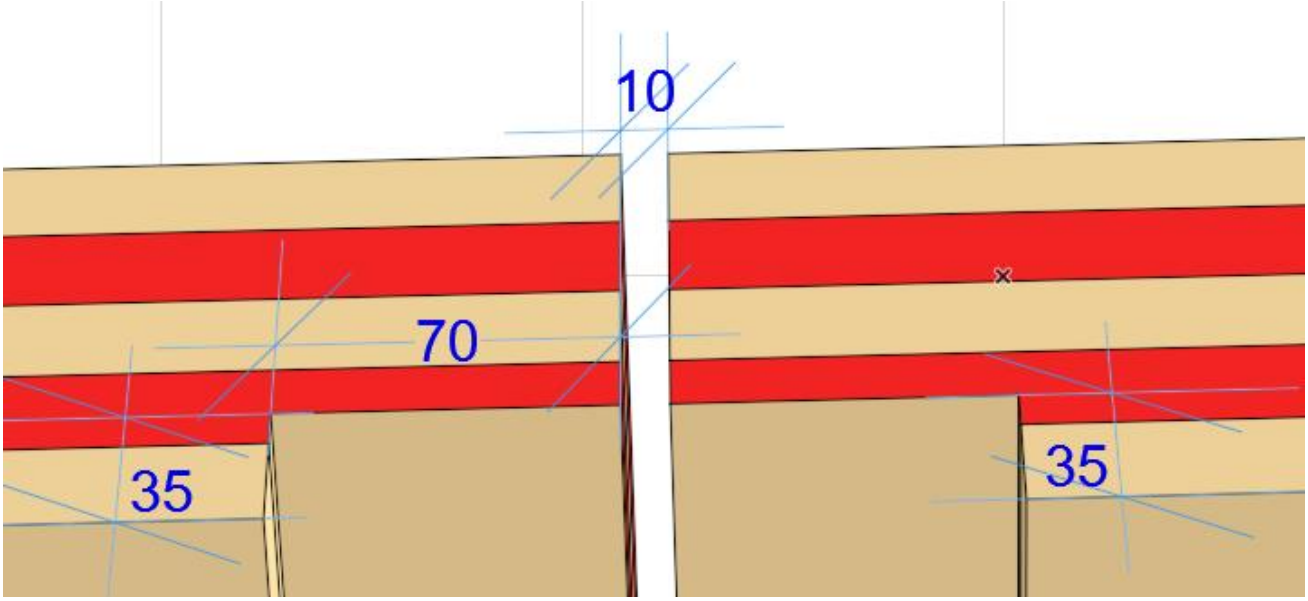
5 mm *Gap between parts* is used at the middle and *Offset from center* has moved the joint 10 mm back from the center of the board.

13.32.3Lap joint with connection piece settings

The meaning of lap joint front/back has been changed to define onto which side the connection piece will be placed. Front means that it comes to the front side of the target board. The settings when using connection piece are:

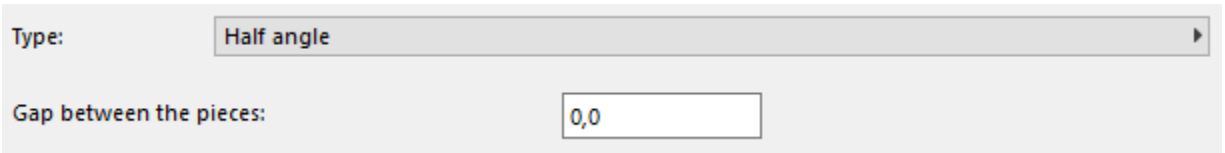


And these settings produce this kind of result:



The setting Gap front is applied between the connected boards and to the groove depth (60+10=70).

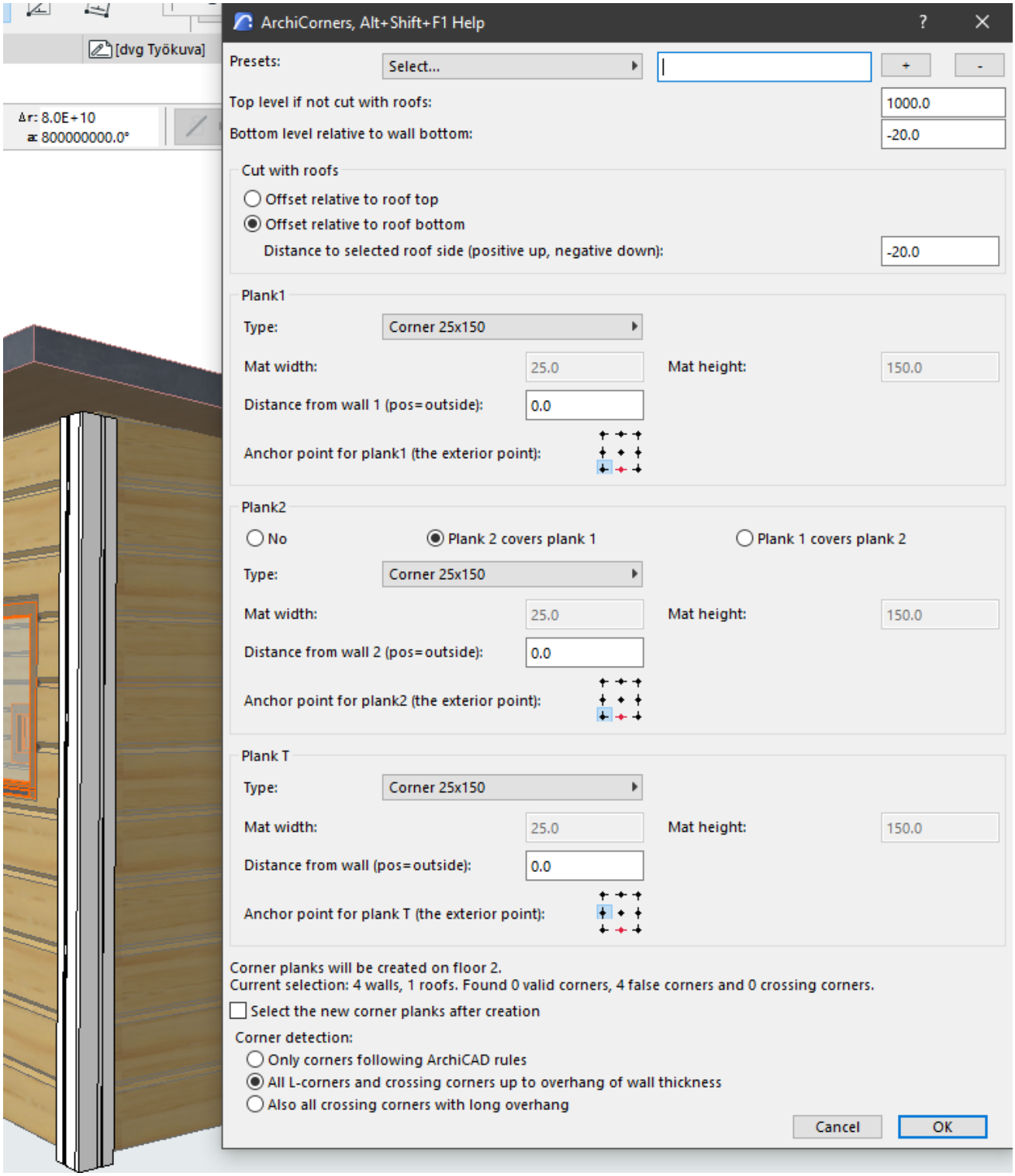
13.32.4Half angle



If the boards are parallel, this type is used to remove any old connections. Currently only parallel boards are supported.

13.33 ArchiCorners / Corner covering planks

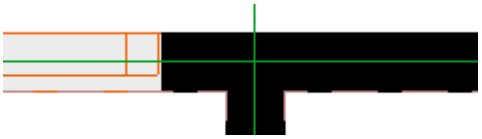
This tool helps adding the white pieces in the image below:



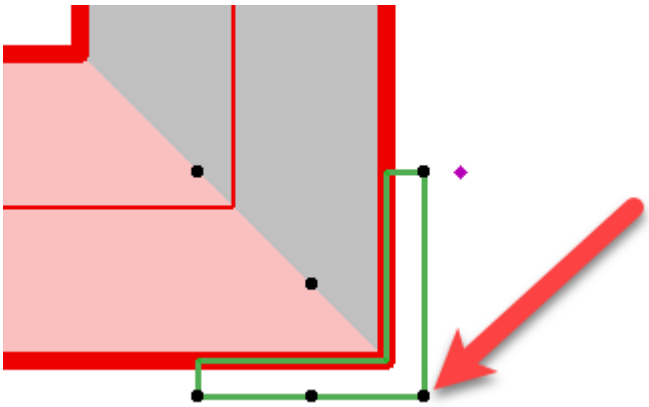
To start the tool, there must be at least two walls in selection. If selection also contains roofs, those are used to trim the corner plank tops. The dialog parts are:

- Preset settings, see help here.
- *Top level if not cut with roofs* tells offset from top of the wall to the top of covering planks. Positive value moves up and negative down. If there are roofs in selection to trim the corner planks, this value should be big enough for the corner planks to intersect with the roof and get trimmed by the roof.

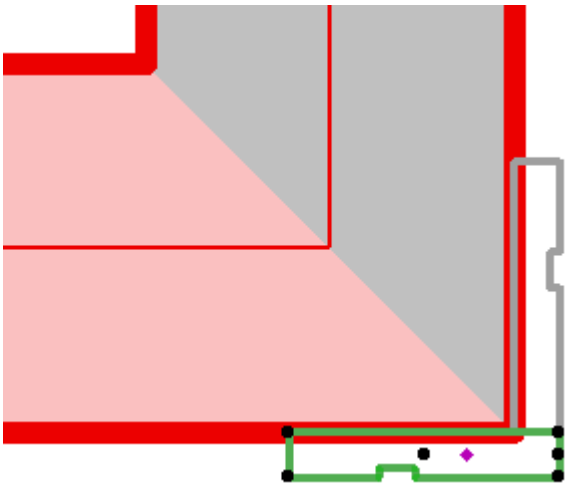
- *Bottom level relative to the wall bottom* offsets bottom of the covering planks. In the image above the corner planks are extend 20 mm below the wall bottom.
- *Cut with roofs* controls how to adjust to the roofs. The distance is perpendicular to the roof.
- *Plank1/2* type of the plank to add.
- *Plank T* settings are used for T-connections:



- *Type* is the material type from ArchiFrameBlocks*.xml to add.
- *Mat width/height* are available only for the block material type.
- *Distance from wall* defines the distance from wall exterior side to the anchor point of added cover plank.
- *Anchor point* is the plank anchor point to place to the offseted corner exterior point. Please note that when looked from above, the plank points are mirrored on vertical axis. For example, the resulting cover plank from above settings has the anchor point on the right hand side looked from above:

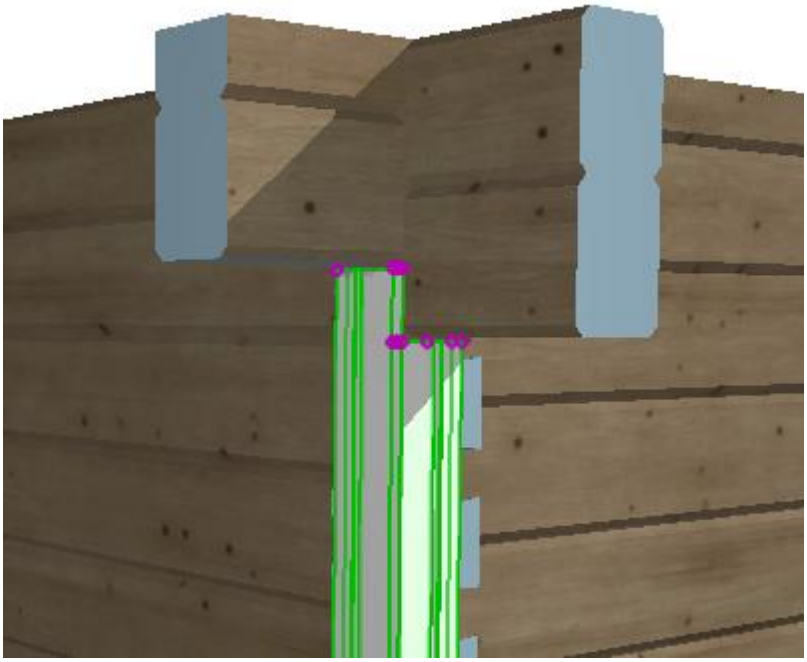


- *Plank1 covers Plank2* and the other way round defines how to place two plank corner, here the selected plank covers the other one (is long one in the corner):



- Current selection: describes the found wall connections. *Valid corners* means the connecting walls have proper outside corner, while *False corners* means the facing of connecting walls are wrong or the resulting corner is not convex. *Crossing corners* shows the number of X-type connection of selected walls (these can be still good, for example if they have overhang).
- *Select new corner planks after creation* is used to select the newly created pieces when dialog is closed. This will ease further editing.

- *Only corners following ArchiCAD rules* will detect the exterior side from the wall outside surfaces. Concave (inner) corners will not be handled.
- *All L-corners and crossing corners up to overhang of wall thickness* places corner planks to all corners to the side having bigger angle. It may cause extra pieces to inner corners that must be removed manually.
- *Also all crossing corners with long overhang* is used in cases like this:

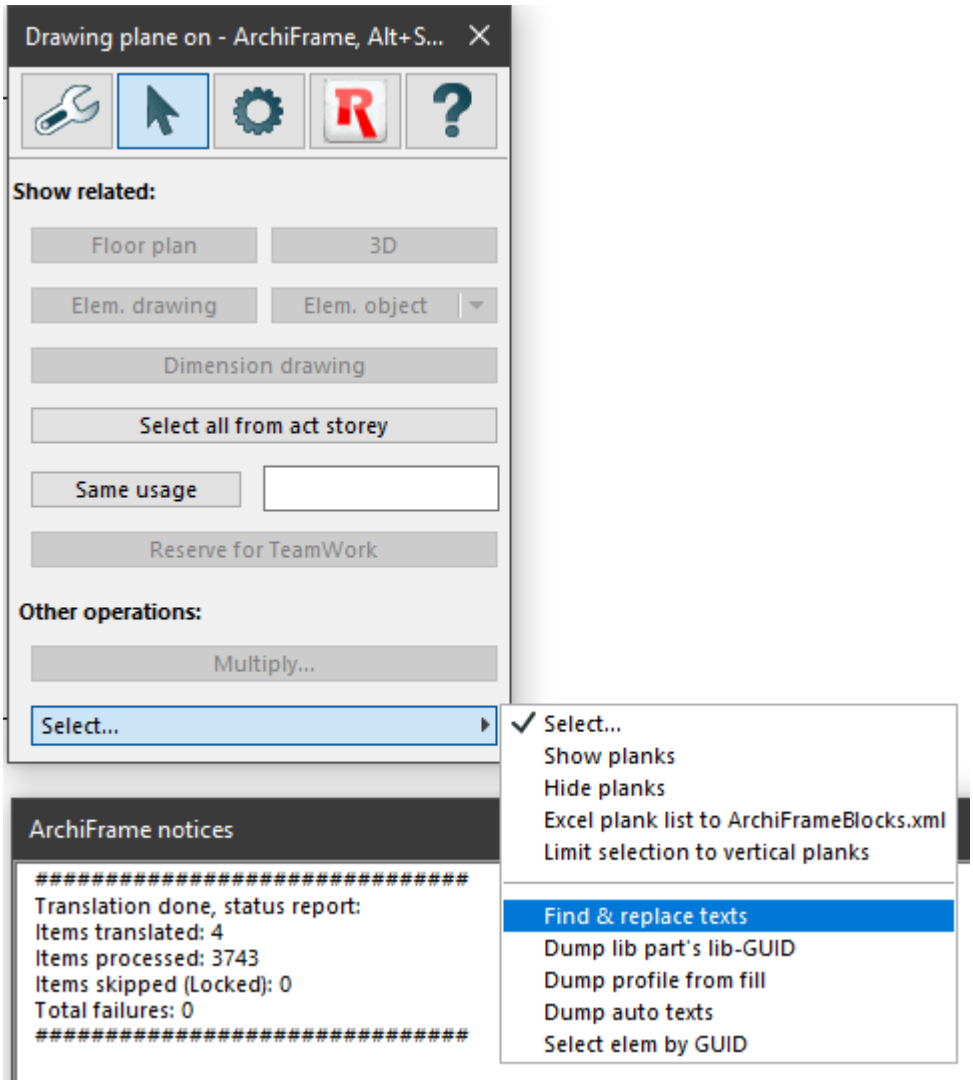


Please note that in this mode the inner (concave) corners will get planks, remove them manually if they are not needed.

13.34 ArchiTranslate / Global Find & Replace texts

The related video is here.

The tool is found here:



It is used to find & replace texts in the whole model. The translations and settings are defined in an Excel file. Please open ArchiTranslate.xlsx from ArchiFrame's Samples-folder to see example definitions.

Excel-file column definitions are:

Column A, targets
Targets defines which parts of the Archicad-file to process, possible values are:

- floor, floor plan items
- layout, all layouts including master layouts
- detail, detail windows
- elevation
- section
- worksheet

Multiple values are given comma separated, for example:

floor,layout

Column B, settings
Settings, comma-separated search options:

- scope=full/word/part
  - full means that the whole source text must match the search text.
    An example text
  - word means that source text must be a word.
    An **example** text
  - part will match just any part of the text
    An ex**ample** text
- casesensitive=yes/no/keepfirst
  - yes requires source text case to match the search text
  - no makes case insensitive search
  - keepfirst makes case insensitive search and keeps the case of the first letter in the text, for example with search & replace definition dove to fish:
    Dovetail is used *to* **F**ishtail is used
    Connected with a dovetail *to* Connected with a **f**ishtail

For example, the default which is used if this column is empty:
scope=word, casesensitive=keepfirst

Column C, element types
Element type filter, comma separated list of elements for translation. Possible values are:

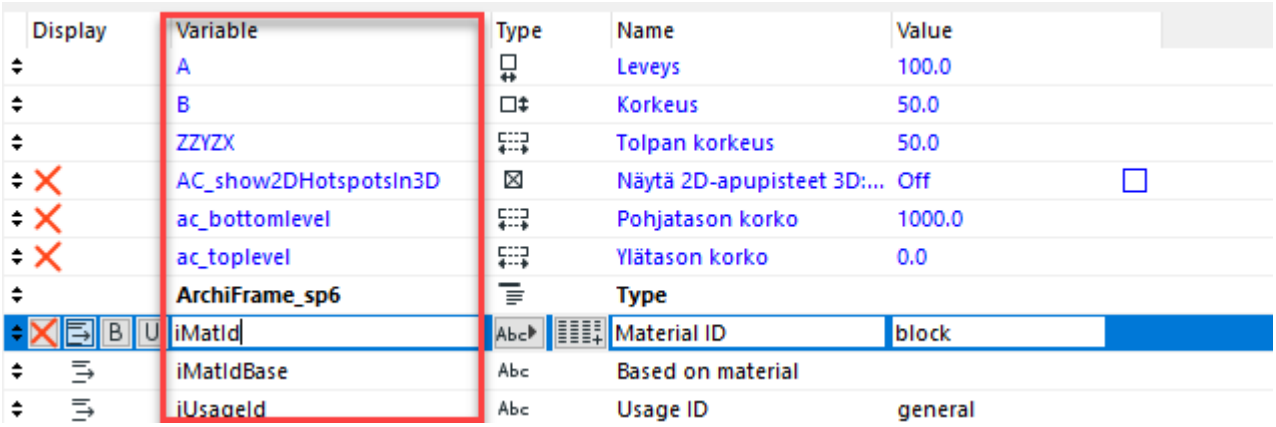
- projinfo, process texts under Archicad menu-item File/Info/Project Info
- text, process text elements
- label, process label elements
- window
- door
- zone
- object
- sectelem, process section/elevation markers

Starting the column with character # switches the definition from *include listed* to *exclude listed*. For example, to process all but project information:

#projinfo

Column D, object parameter names
Object parameter name filter, contains list of object internal parameter names to handle during the process. Please note that the internal parameter name is this one not directly visible to the user but

will remain unchanged in different object language versions:



Starting the column with character # switches the definition from *include listed* to *exclude listed*. Wild card * can be used in the parameter names. For example, process all but parameters ending with ID:

#*ID

Column E, target library part names
Lists included target library part names or GUIDs. Names can include wild cards *. For example:

MyObjectRevision*,Baar*stool

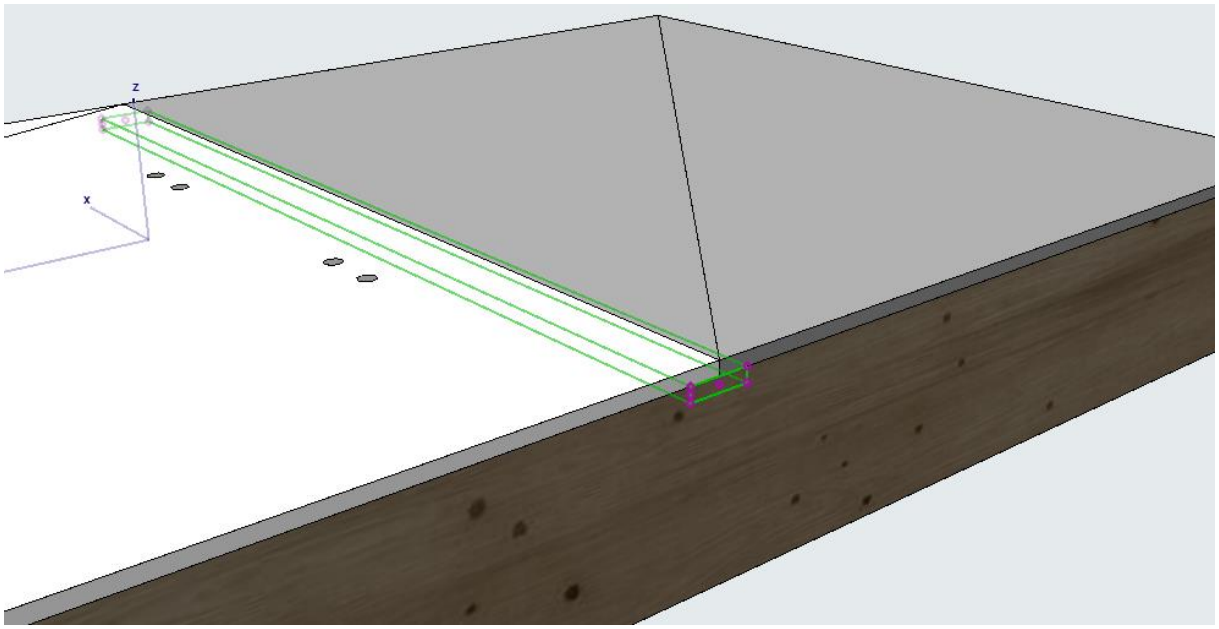
Column D, Lua-script for library parts
This column is used to run a script that takes care of the translation. Column has two comma-separated value: the script file name and the function name to be called to do the translation. If the scipt file name does not contain path (recommended way), the scipt is first loaded from the user specific settings folder and if it is not found there, from the base data folder. The called function has this form:

function DoTranslate(strSourceLang, strDestLang, guidObject)

Please see the default data folder for an example translation Lua-script ArchiFrameTranslateLua.lua. Lua script itself must be utf8-encoded.

13.35 Board edge support pieces

This option is be used to place supporting pieces to the board edges:

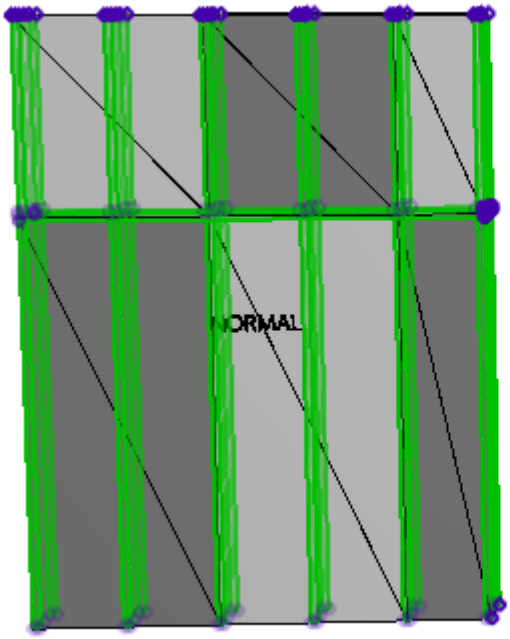


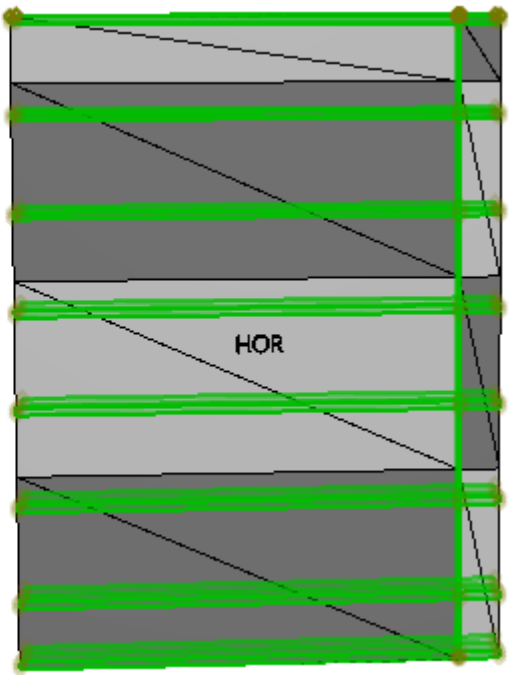
The support pieces can be set three ways from *Spacing rule*-setting:

- Fixed spacing, the starting offset and constant spacing value is given.
- Override, all positions are given in maximum 10 spacing override input fields
- Automatic, ArchiFrame searches the board edges from connected boarding layer. If the boarding changes and the support pieces need to be recreated, switch the option off and on again.

Parts of the setting dialog are:

- Material, the material used for the support pieces.
- Material rotation, typically for narrow pieces 90 degrees is used.
- Framing rotation should be set to 90 if the framing is in 90 degrees angle to the element's direction, for example the right hand side below is horizontal:



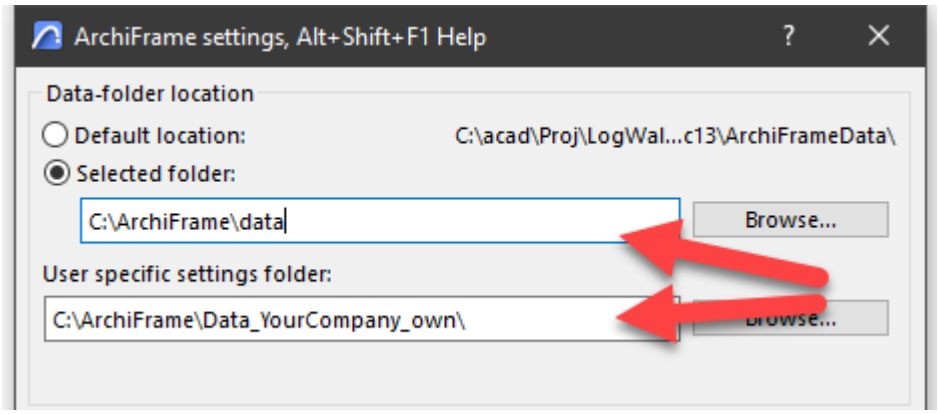


Please note that if framing layer is rotated using rotframing-setting, this setting should remain as zero.

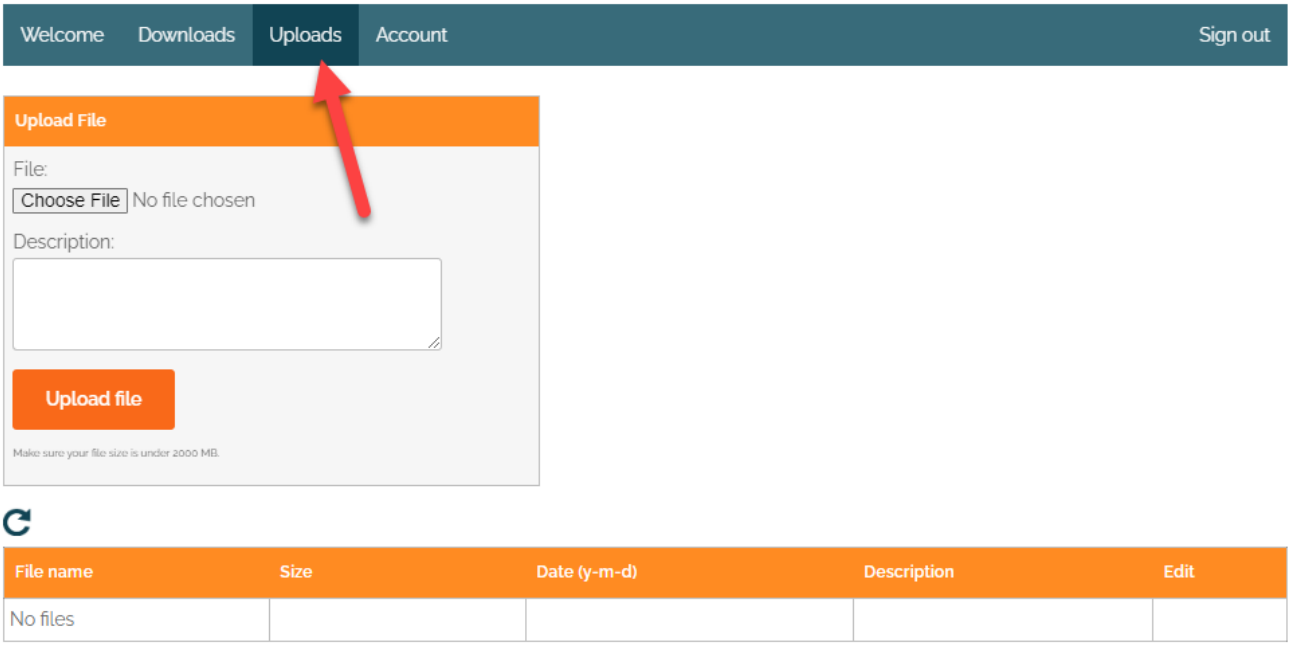
- Spacing rule start, start spacing either from element bottom or top
- Spacing start offset, move spacing rule begin by given value. Negative outside the element and positive towards the spacing direction.
- Fixed spacing value, used if *Spacing* rule is set to *Fixed spacing*.
- Oversize top/bottom/behind, how much bigger the slot is than the used plank.
- Spacing override 1-10, used if *Spacing* rule is set to *Override*. Value >= 100 m ends the list.

13.36 Support package zip file

Support package/Create will save current AC-file as Archicad archive having file extension .pla, add it to a zip file and add content of the current data and user specific settings folder into it:



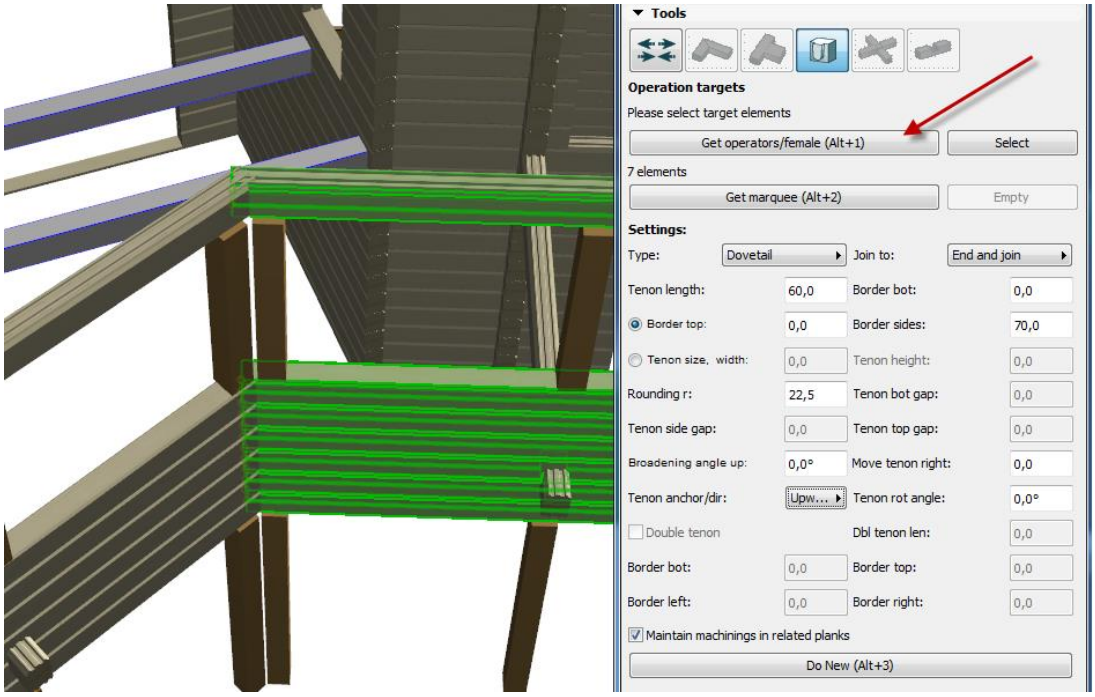
This feature can also be used to archive the current model with current ArchiFrame-settings. However, it was made to ease sending information for ArchiFrame support. You can use your own favourite file sharing service to send it to the support@archiframe.fi. Another option is to use your ArchiFrame login and use our upload service from here:



14 Examples of using log structures

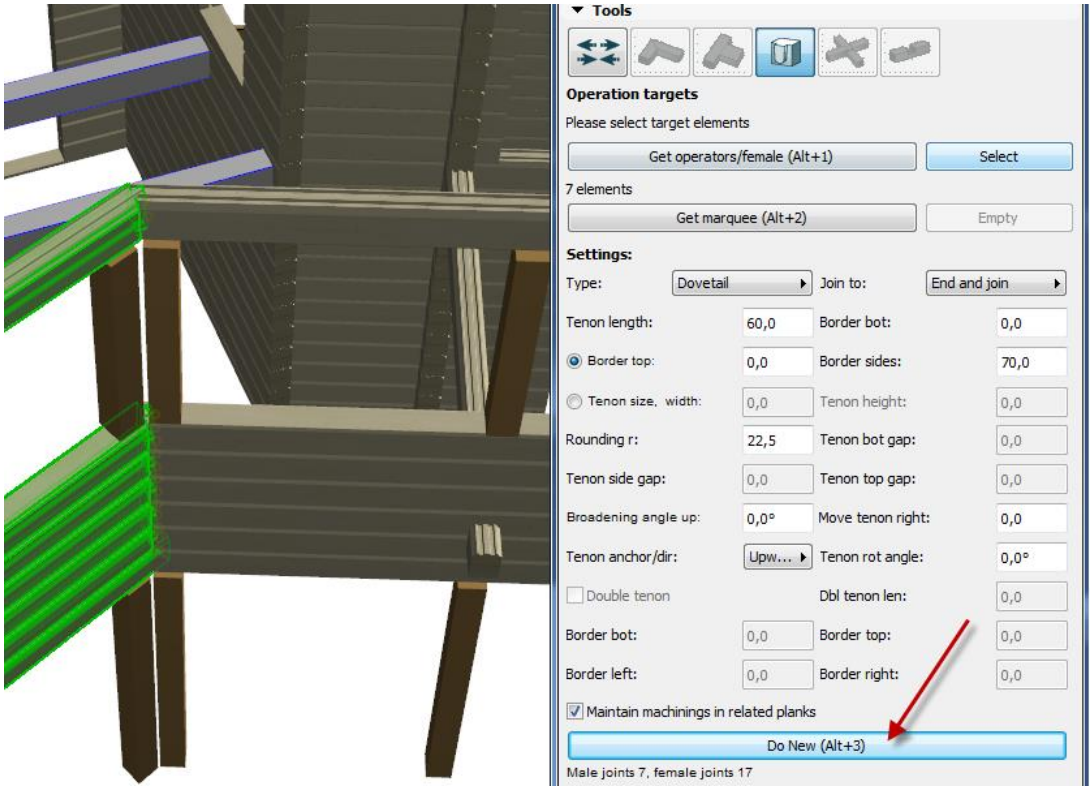
14.1 Joining beams or logs from ends with continuous dovetail

Select pieces to have DT-mortises:



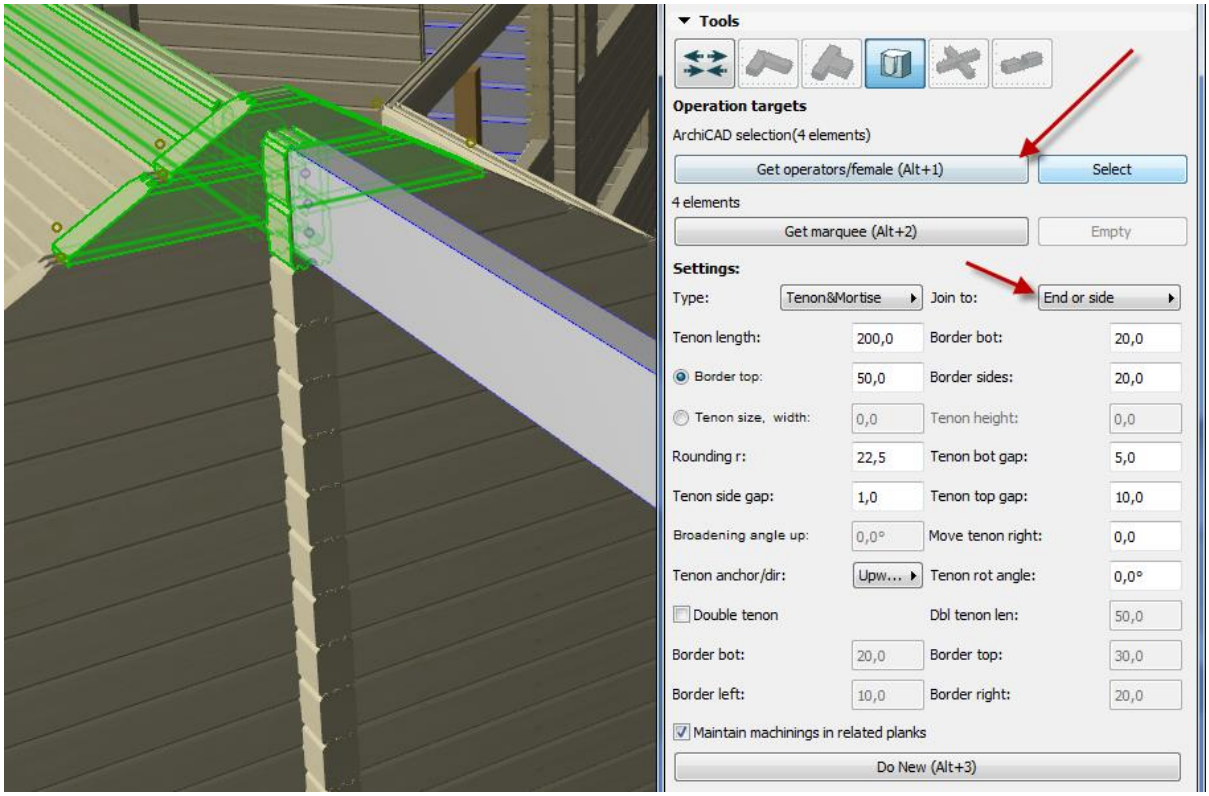
Adjust settings like above (zero at border bottom and top and at broadening angle up will produce wanted continuous dt-joint).

Then select male parts and create the joint (please notice that *Join to* be set to *End and join* – it will halve the angle between planks automatically):

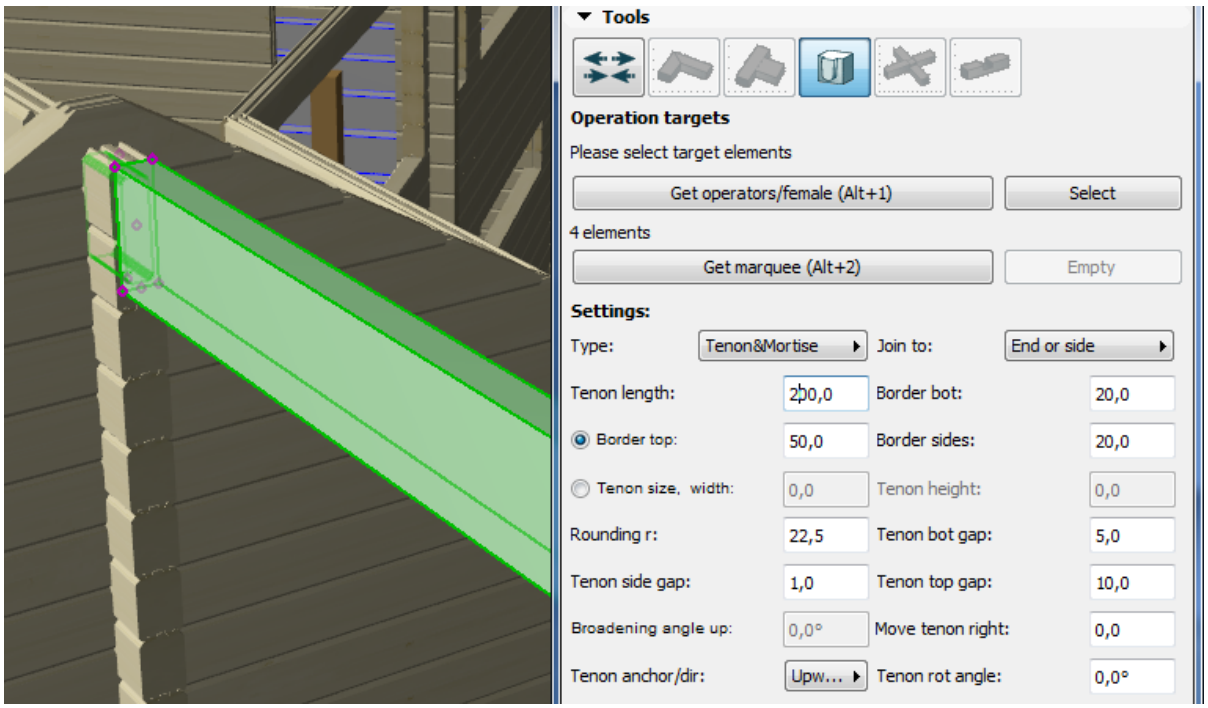


14.2 Join balk to the end of logs with tenon & mortise

Select the mortise/pocket parts and adjust settings (please note Join to-setting is *End or side*):

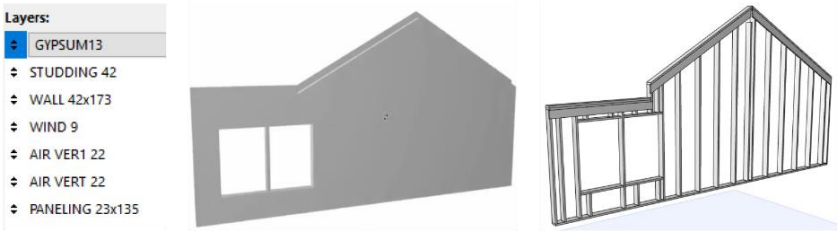


Select the male part and create the joint:

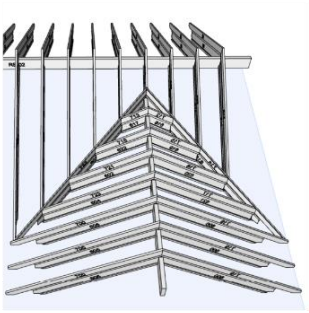


15 Examples of using frame structures

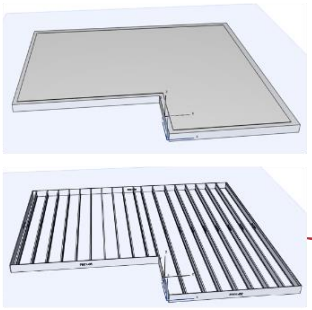
For an example of modelling wooden structures with ArchiFrame, please see our online videos. These videos show how the ArchiFrame demo house was created. The demo house file can be found in C:\ArchiFrame\Samples\ArchiFrameDemo2016. Here we provide an overview of the different parts of the demo house, as well as the contents of the videos.



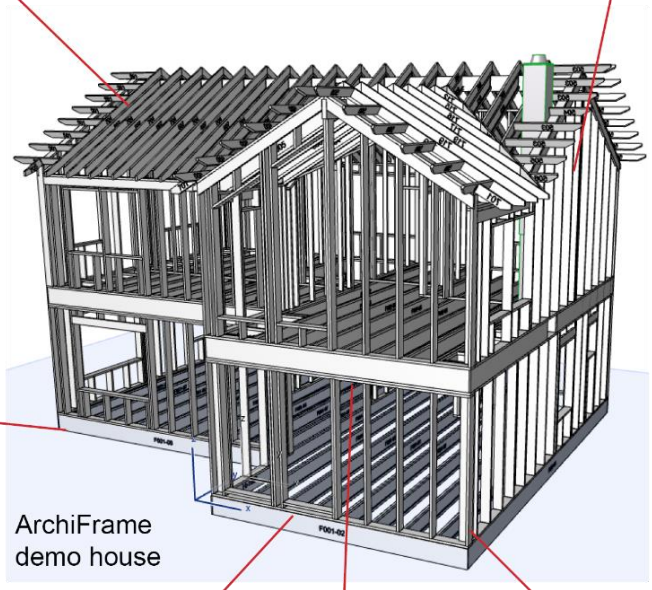
1. Walls are created by defining the element type (what layers the wall consists of), creating an element, and finally creating the planks. Besides rectangular walls, the videos show how to create walls with a complex shape.



6. The roof contains two ridge beams, two valley rafters, main rafters and smaller rafters that support the top layer of the roof.



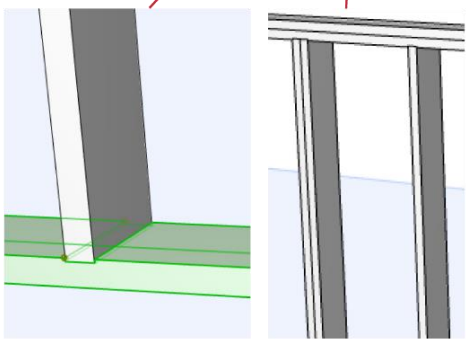
5. Floors are modelled as elements, in a very similar way as walls



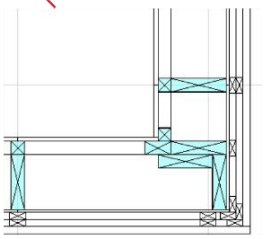
ArchiFrame demo house



4. ArchiFrame automatically creates elevation drawings and shoplists of all the elements in your model.



3. Examples of wall settings that can be edited. Left: marking grooves on the bottom plate. Right: double top plate.



2. ArchiFrame allows you to create custom corner definitions.

Creating wall types and elements

- Workflow: using ArchiFrame elements (video)
- Defining a custom wall type (video)
- Creating wall elements for
  - rectangular walls (video)
  - custom-shaped walls (e.g. with triangular tops) (video)
- Creating planks (video)

Custom corners between two walls (video)

Changing wall settings (e.g. adding double top plates or marking grooves) (video)

Creating elevations, drawings and shop lists (video)

Floor (video)

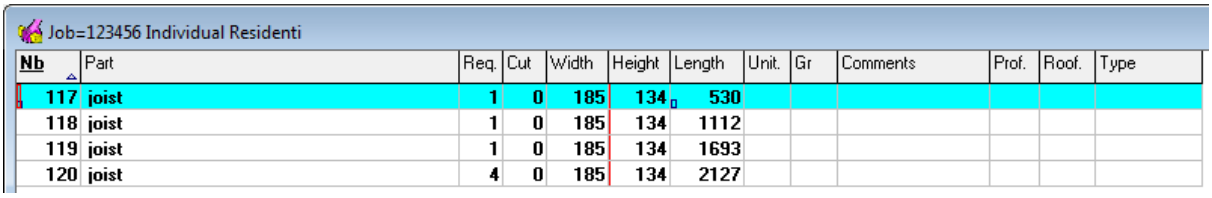
Roof (video)

16Listings

16.1 Hundegger cnc-file

File extension is bvn. The planks will get the ID like below:

- In case of fully numeric ID, it will be copied to Hundegger's number column and Part column will contain the set usage type:



Nb	Part	Req.	Cut	Width	Height	Length	Unit.	Gr	Comments	Prof.	Roof.	Type
117	joist	1	0	185	134	530						
118	joist	1	0	185	134	1112						
119	joist	1	0	185	134	1693						
120	joist	4	0	185	134	2127						

- If the ID contains non-numeric digits, ArchiFrame will assign consecutive numbers to the number columnd starting from one. Part column will contain ID (usage type):

Nb	Part	Req.	Cut	Width	Height	Length	Unit.	Gr	Comments	Prof.	Roof.	Type
1	C-001 (ELEM)	2	0	195	41	127						
2	C-002 (ELEM)	1	0	195	41	6000						
3	C-003 (ELEM)	2	0	195	41	2886						
4	C-004 (ELEM)	2	0	195	41	2918						

16.2 Editing listings

Note! Always edit a copy of the *data*-folder, for example C:\ArchiFrame\Data_own. This way the changes will remain during installation of updates.

The file to be edited is ArchiFrameListing.lua inside the data folder (see Lua scripts). In addition to the default listing there are special listings defined in global variable gtblListings. One item in the table contains fields:

Field	Description
strName	Name of the listing.

strUIClass	Special listings can be invoked from another place in the UI other than the default listing tool: • Cover, listing for weatherboards.
strPreCollectFunc	Called before planks/boards are collected. Function may show settings dialog and affects how ArchiFrame collects planks & boards. Returns nil=cancel the process or a table having fields: • collectspecial, bit mask of what special planks to collects. Elements having parameter iSpecialType zero are always collected. Default=2 (include just combined ones)
strOnInitFunc	Function that is called at the start of listing.
strOnSaveListFunc	Function that creates the actual listing.

strFilterFunc	Function that filters planks to collect, "" = no filter. For example:
	`function FilterPlank(sGuid)`
	`  local sUsage`
	`  ac_objectopen(sGuid)`
	`  sUsage=ac_objectget("iUsageId")`
	`  ac_objectclose()`
	`  if sUsage==nil then`
	`  return false`
	`  end`
	
	`  return string.upper(sUsage)=="BUCK"`
	`end`

nSorting	Sorting criteria: • 0, by plank ID. • 1, material id, height, thickness, ID, length. • 2, material id, height, thickness, length, ID.
nAllowSameId	Many different planks have the same ID: • 0, no (default). • 1, may have.
nCollectElem	Collect all planks related to the elements (use possible selection to collect every plank from elements related to the selection): • 0 or missing, no (default). • 1, collect all.
nCollectType	ArchiFrame collects planks only to table gtblPlanks. This field is used to change that. • Missing, default and collect planks only. • 0, don't pre-collect anything. • 1, collect planks. • 4, collect boards. • 1+4, collect planks+boards.

ArchiFrame also sets Lua-global gbPlanksSel to tell if source for the operation was element selection (true) or not selection (false). If listing script sets Lua-global gnStatusCount, it will be shown in tool palette's status text after the operation. Script may also set global gbCnc-IncludeNoCnc- to process also planks having parameter icnc- set to zero (usually skipped).

17 Material list ArchiFrameBlocks.xml

This file contains all pre-set material types and their settings. Editing the file is best to start from Materials <materials>/<material>.

To edit raw plank materia list, please check this video.

17.1 Settings <settings>

```
<archiframe>
  <settings>
    <misc>
      <manufacturer name="xxx"></manufacturer>
```

Manufacturer name affects possible customisations programmed into ArchiFrame.

17.1.1 Printing settings

```
<print layoutname="Elem drawing &lt;elemid&gt;" frame="0"
       masterland="A4 Wall drawing" masterlandxsize="297" masterlandysize="210"
       masterport="A4 Wall drawing portrait" masterportxsize="210" masterportysize="297">
  <empty_custom name="MyMaster" left="0" top="0" right="100" bottom="0"></empty_custom>
</print>

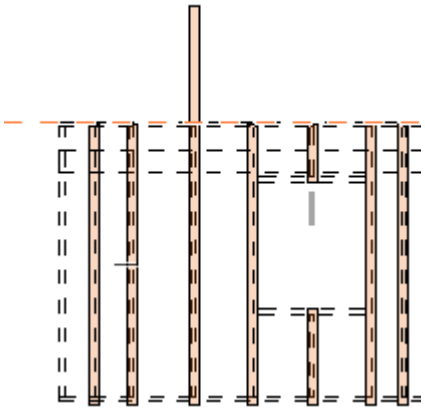
<printframe dimtitle="Frame ">
  …
</printframe>
```

Tag <print> defines master layout names and dimensions for landscape and portrait types. Both can be defined with same dimensions to have always the same orientation.

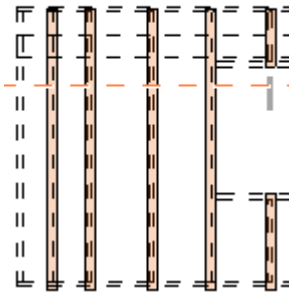
<empty_custom> defines how much space should be left at each side of the print area. This is useful if there is a stamp in master layout that should not be tampered. The example leaves 100 mm of the right side of the layout untouched. This setting is used if master layout name matches with name-attribute. Name may contain wildcards (name = "*") will match everything. It may also contain attributes:

Xlm-attribute	Description
valign	Align vertically in the master layout: • -1, top. • 0, center. • 1, bottom (default).
halign	Align horizontally in the master layout: • -1, left (default). • 0, center. • 1, right.

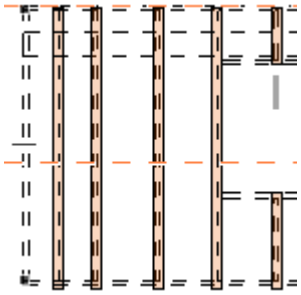
An example of different vailgn values for a projection with layoutminheight="5",
To bottom, valign="1":



Center, valign="0"



Top. valign="-1":



<*printframe*>-tag contains XML-settings for an Archicad element.

Xlm-attribute	Description
layoutname	Default text for print dialog.
frame	Values: • 1, include the frame (default). • 0, no frame.
skiplibpart	If given the listed library parts are skipped when creating the layouts, for example, to skip standard element stamp, specify skiplibpart="9FD03D30-6A12-4B0F-AB3A-01410A529E38". Add multiple values using comma as a separator.
masterland	Name for landscape drawing (width>height).
...xsize, ...ysize	Paper size in millimetres.

masterport	Name for portrait drawing (height>width). Both landscape and portrait definitions must exist even if not used.
dimtitle	Title for single plank dimension drawings. List of plank Ids will be added after the title.

17.1.2 Plank dimension drawings <projections>

```
<projections layer="xxx" maxwidth="80.0" margboxes="3.0" margleft="1.0" margtop="1.0"
margright="1.0" margbot="1.0" margproj="0.7">
```

Xlm-attribute	Description
layer	Layer to place the dimension drawings. If the layer does not exist, it is created. Empty or missing layer name places dimension drawings to the same layer as the original plank.
maxwidth	Dimension drawing line maximum length in meters. If drawing does not fit to current line, it is placed to the next line above.
margboxes	Marginal between the dimension drawing frames.
margxxx	Marginal between the frame and markings inside it.
margproj	Vertical marginal between dimension drawings from different plank sides.

Global settings for any projection element in dimension drawings and element projections:

```
<projbase>
  <!--layer>Name, auto created</layer-->
  <!--elemparam name="pen">1</elemparam-->
</projbase>
```

It is possible to list any XML-settings for an Archicad element. In this place as exception unknown lines do not cause any message. That allows setting for example object parameters even for dimension lines.

Tag projplank contains settings for the projection. It may be useful to set the line type to solid instead dash & dot etc:

```
<projplank>
  <elemparam name="linetype">1</elemparam>
  <objparam name="#useobjlinetype">0</objparam>
</projplank>
```

Dimension settings:

```
<dimsettings>
  <drill midout="0"></drill>
</dimsettings>
```

Xml-tag	Description
drill	Settings for drill dimensions, attributes: • Midout, default is 1. With value 0, drillings outside the plank surface will not have dim line for Y-coordinate.

17.1.2.1 Dimension drawings titles <projections>/<text_title>

This section defines the titles and text settings for dimension drawings from different plank sides.

```
<text_title yoff="0.10">
  <settings>
    <elemparam name="fontname">Arial</elemparam>
    <elemparam name="fontsize">1.2</elemparam>
```

```
    <elemparam name="fontstyle"> </elemparam>
    <elemparam name="pen">11</elemparam>
  </settings>
  <side1>
    <text>TOP</text>
  </side1>
  <side2>
    <text>FRONT</text>
  </side2>
  <side3>
    <text>BOTTOM</text>
  </side3>
  <side4>
    <text>BACK</text>
  </side4>
</text_title>
```

Xlm-attribute or tag	Description
yoff	Vertical marginal between the plank projection and the text.
<settings>	Settings for the text element, see XML-settings for an Archicad element.
<sideX>	Text for different surfaces.
lang	To be used with *<text>*-tag. If current ArchiFrame language matches with the setting, that tag will be used. Otherwise first *<text>*-tag is used.

17.1.2.2 Dimension line settings <projections>/<dim_xxx>

Contains settings for different kinds of dimension lines, see XML-settings for an Archicad element.

Dimension line types are:

Xlm-attribute or tag	Description
<dim_main>	Main dimension line.
<dim_mcmain>	Machinings dimension line.
<dim_angled>	Angled machining.
<dim_mcdet>	Additional machining dimensions.
<dim_mcdetangle>	Angled machinings additional dimensions.
<text_mcdet>	Settings for text details. Additional attributes for this tag: • Diapre = "Ø ", text to show before actual drill diameter.
yoff	Dimension line vertical distance to the plank.

17.1.2.3 Dimension drawings summary text <projections>/<text_summary>

Summary text is placed below the dimension drawings. See XML-settings for an Archicad element.

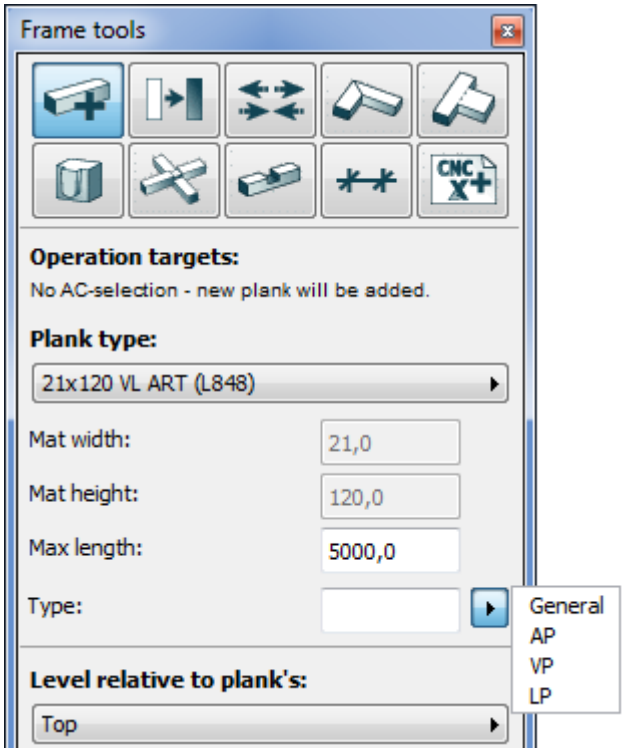
```
<text_summary height="1.4" minwidth="3.0">
  <settings>
    <elemparam name="fontname">Arial</elemparam>
    <elemparam name="fontsize">1.5</elemparam>
    <elemparam name="fontstyle"></elemparam>
    <elemparam name="pen">11</elemparam>
  </settings>
  <text lang="fin">Mat: [mat]\nLaatuluokka/Quality: [quality]\nNimi/Name:
[usage]\nSamanlaisia/Similar: [count]\nTunnukset/IDs: [idlist]</text>
</text_summary>
```

Text may contain parts to be replaced:

Part	Description
[mat]	Material name and ID, for example 41x195 (XYZ 123).
[quality]	Quality class from the plank object.
[usage]	Usage text from the plank object.
[count]	Number of similar planks.
[idlist]	IDs of similar planks.
\n	Line feed.

17.1.3 Plank usage pre-set list <usagetypes>

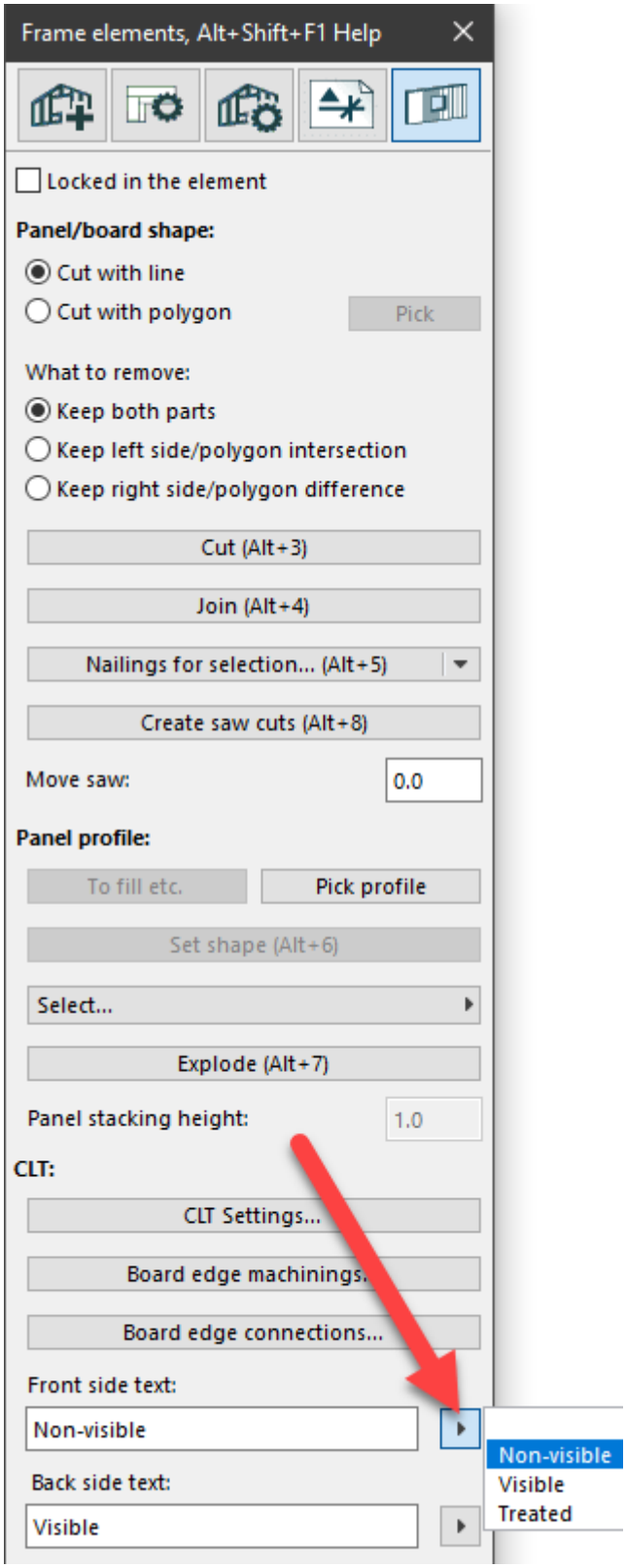
This list is put here:



Attribute name is the name for the list and id is the actual value to save. Name and id should be the same unless there is a need to use very short abbreviations.

17.1.4 CLT-boards front and back text presets <clt_frontback>

This list is put here:



Own definitions should be put to user specific settings/ArchiFrameBlocksChanges.xml, please see <clt_frontback> from the ArchiFrame Samples/Data_Yours_BASE_XXX_own (default place in Windows C:\ArchiFrame\Samples\Data_Yours_BASE_XXX_own).

17.1.5 Hidden balk joint list <balkhiddenjoints>

This section defines machinings for different hidden balk joints and accessory codes to be used in custom listings.

```
<balkhiddenjoints>
  <balkhiddenjoint name="Hidden balk joint 1" id="HIDDENBALKJOINT1" cutwidth="0.050"
cutheight="0.150" cutdepth="0.005" grodepth="0.100" growidth="0.008" drilly1="0.050"
drilldist="0.080" drillsize="0.010" drillspace="0.040" drillcount="3">
  </balkhiddenjoint>
</balkhiddenjoints>
```

Xlm-attribute	Description
name	Joint name.
id	Beam shoe id for custom listings.
cutwidth	Shoe back plate width.
cutheight	Shoe back plate height, shoe is machined into the plank from top surface downwards.
cutdepth	Show back plate thickness.
grodepth	Length of the shoe part that actually holds the beam (goes into the beam).
growidth	Thickness of above.
drilly1	First drilling's distance from top surface.
drilldist	Drilling distance from plank's end.
drilldist2	The same if there is another row of drillings.
drillsize	Drilling diameter.
drillspace	Distance between the drillings.
drillcount	Number of drillings.

17.1.6 Mortise and tenon joint default dimensions <mortisetenon>

```
<mortisetenon len="0.056" lenrelative="0.5" bordertop="0.02" borderbot="0.005"
borderside="0.01" roundingr="0.0225" endgap="0.001" sidegap="0.001" topgap="0.010">
</mortisetenon>
```

Xlm-attribute	Description
len	Tenon length.
lenrelative	Overrides len, use tenon length relative to plank width.
bordertop	Border top.
borderside	Border sides.
roundingr	Rounding radius for corners. Affects directly 3D and Hundegger cnc- so that zero means rectangular and any other value will create rounded tenon (rounding radius is defined by the machine).
endgap	Mortise bottom gap.
sidegap	Mortise side gap.
topgap	Mortise top gap.

17.1.7 Dovetail joint default dimensions <dovetail>

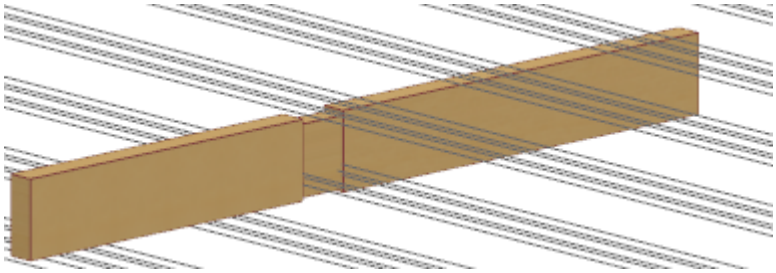
```
<dovetail len="0.038" bordertop="0.00" borderbot="0.050" bottomwidth="0.06"
roundingr="0.0225" endgap="0.000" sidegap="0.000" angle="6" >
</dovetail>
```

Xlm-attribute	Description
len	Tenon length.
bordertop	Border top.

borderbot	Border bottom.
bottomwidth	Dovetail tenon width at bottom (in pocket the width at plank surface).
roundingr	Rounding radius, dovetail is always rounded – this affects only the 3D.
endgap	Pocket bottom gap.
sidegap	Pocket side gap.
angle	Opening angle.

17.1.8 Narrowed balk default dimensions <balkborder>

Narrowed balk is used for example when beam goes through a log wall:



```
<balkborder bordertop="0.02" borderbot="0.005" borderside="0.01" overtop="0.001"
overbot="0.000" overside="0.001" overlen="0.000" >
</balkborder>
```

Xlm-attribute	Description
bordertop	Top side narrowing.
borderbot	Bottom side narrowing.
borderside	Side narrowing.
overtop	Oversize top.
overbot	Oversize bottom.
overside	Oversize sides (to be used to both left and right).
overleft	If different on the sides.
overrigh	If different on the sides.
overlen	Lengthwise oversize. For example if the wall is 200 mm thick and this value is 1 mm, the narrowing will be 201 mm long.

17.1.9 Intact balk default dimensions <balkcut>

As narrowed balk but no machinings in the balk.

```
<balkcut overtop="0.01" overbot="0.000" overside="0.001" >
</balkcut>
```

17.1.10 Default settings <plankdefaults>

Plank default settings. When adding new planks these settings are set to the plank. See XML-settings for an Archicad element.

```
<plankdefaults defaultlayer="Wall drawings.2d" plankminlen="0.01">
  <elemparam name="pen">11</elemparam>
  <objparam name="iLengthFormat">mm</objparam>
  <objparam name="iFontSizeBase">10</objparam>
</plankdefaults>
```

Xlm-attribute	Description

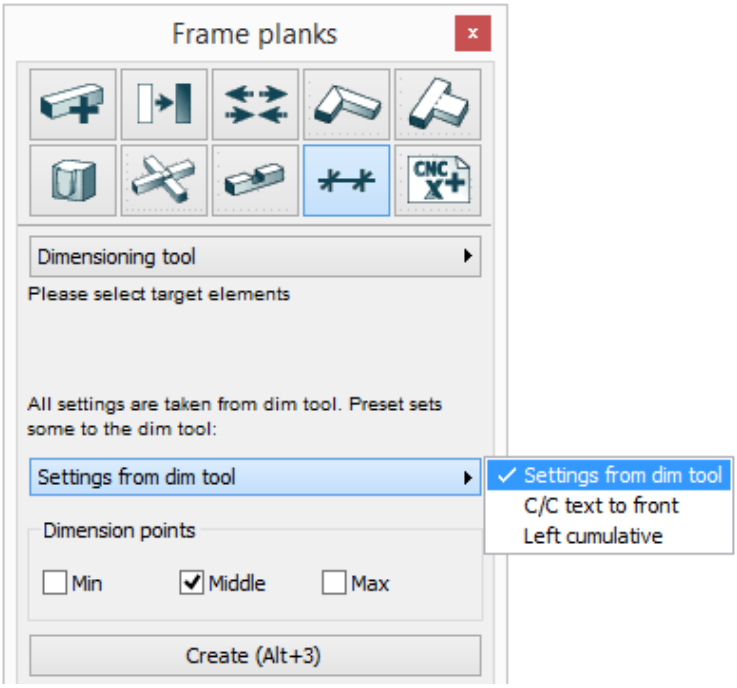
defaultlayer	Default layer for add & edit tool palette. If the layer is forced, it is given the normal setting.
plankminlen	Shorter planks are not created and are removed automatically.

17.1.10 CNC-limitations <cnc-limits>

Xlm-attribute	Description
gromaxdepth	Groove maximum depth. Groove tool palette uses this value to select groove target side in the plank. The side that does not exceed the maximum is better than the one that exceeds the maximum value.
nailmindistplank	Nail minimum distance from plank edge.
nailmindistboard	Nail minimum distance from board edge.
elemgeo	Value: • 0, no cnc- for element geometry, give same ID if planks are similar mirrored or direction swapped. Using this value still assigns different ID to the planks in case of certain machinings like angled ending having angle <> 90 degrees. Give value 2 if it is OK to combine these also. • 1, element geometry cnc- needed – give similar IDs only if similar unmirrored and not swapped (default). • 2, see value 0.

17.1.11 Dimension tools presets

These definitions affect here:




```
<dimpresets>
  <dimpreset name="Settings from dim tool" name_fin="Asetukset mittatyökalusta">
  </dimpreset>
```

```
  <dimpreset name="C/C text to front" name_fin="C/C teksti alkuun" setmin="0" setmid="1"
setmax="0">
    <dimlinesettings>
      <elemparam name="dimensiontype" >linear</elemparam>
      <elemparam name="fontname">Arial</elemparam>
      <elemparam name="fontsize">1.5</elemparam>
      <elemparam name="fontstyle"></elemparam>
    </dimlinesettings>
    <text content="C/C" textid="0" anchor="6" marg="0.2">
      <elemparam name="pen">11</elemparam>
      <elemparam name="fontname">arial</elemparam>
      <elemparam name="fontsize">1.5</elemparam>
      <elemparam name="fontstyle"></elemparam>
    </text>
  </dimpreset>

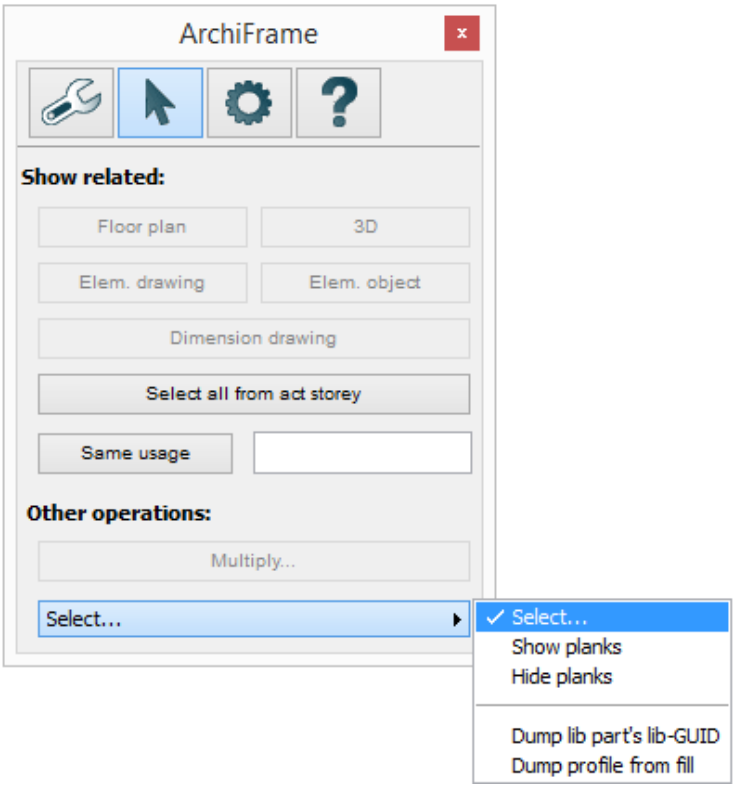
  <dimpreset name="Left cumulative" name_fin="Vasen juokseva" setmin="1" setmid="0"
setmax="0">
    <dimlinesettings>
      ...
    </dimlinesettings>
    <text content="LEFT" textid="0" anchor="6" marg="0.2">
      ...
    </text>
  </dimpreset>
</dimpresets>
```

Xlm-attribute	Description
name	Preset name in the tool.
setmin	Default value for check box Min.
setmid	Default value for check box Middle.
setmax	Default value for check box Max.

It is possible to attach text element to the dimension line using <*text*>-tag, for details please see Single dimension line settings <dimline>.

17.1.12 Selection tool automations

These definitions affect here:




```
<selautomations>
  <script name="Show planks" name_fin="Näytä kapulat" openundo="0">
    <![CDATA[
ac_environment ("layer", "af *", 1)
ac_environment("rebuild", 1)
]]>
  </script>
  <script name="Hide planks" name_fin="Piilota kapulat" openundo="0">
    <![CDATA[
ac_environment("layer", "af *", 0)
ac_environment("rebuild", 1)
]]>
  </script>
</selautomations>
```

The items may contain any kind of Lua scripts. ArchiFrame provides interface to Archicad. Please see Lua scripts.

17.2 Materials <materials>/<material>

Please see the related video here and list of all xml-settings here.

Material list contains pre-set materials. In addition to these the default material list contains freely adjustable block, planed and round materials. Single material is defined with *<material>*-tag and inside it is possible to give Settings <settings> except <projections>-part. In the example the pen for the material is set to value 22.

```
<material id="XYZ200" name="Weather A" thickness="0.025" height="0.120" shape="block"
maxlen="5.0" m3factor="0" fixlen="0.210" tolist="0">
  <settings>
    <plankdefaults defaultlayer="Wall drawings.2d" plankminlen="0.01">
      <elemparam name="pen">22</elemparam>
    </plankdefaults>
  </settings>
</material>
```

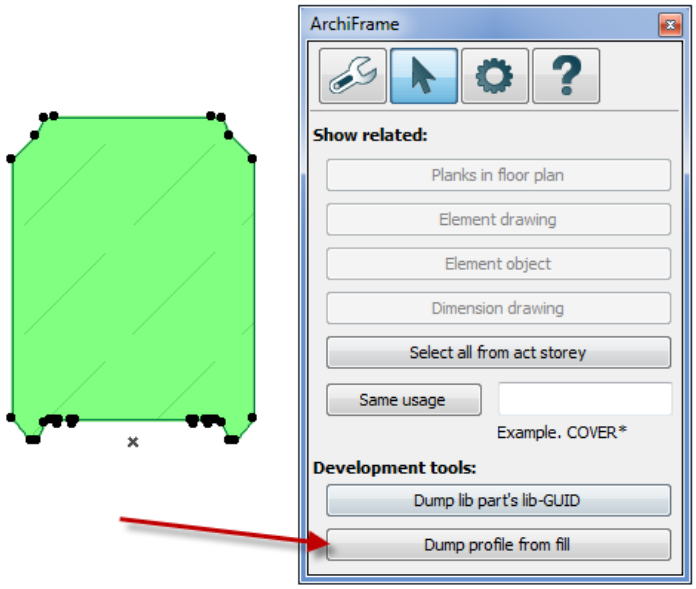
Xlm-attribute	Description
id	Material internal ID. Name can change in different languages but the material can still be matched by the same ID.
name	Material name to be listed for the user.
thickness	Material thickness, usually the smaller extent is given here.
height	Material height.
shape	Material basic shape: • Block, rectangular. • Plane, planed from corners. • Round, circular shape. Basic shape must be given even if material profile is provided.
maxlen	Maximum length. If exceeded the pank displays MAX LEN-remark.
m3factor	For listings to define volume: material volume = meters*m3factor.
fixlen	If given, the material length is always fixed.
tolist	Value 0 removes the material from Add & Edit tool palette. Material can still be used for example for weatherboards.
density	Material density (kg per dm3), default value is 0.460 which can be used for dry pine.
kgpermeter	Another option to give weight by kg per meter.
lbperfoot	Another option to give weight by lb per foot.
noswap	Do not allow ArchiFrame to make the plank go from left to right or bottom to top. Useful for example for weatherboard pieces having profile.
excel_quality	Used by summary list listing to produce quality-column. For example: C22, GL20
excel_type	Used by summary list listing to produce type-column. For example: Treated, Glue laminated, Massive

17.2.1 Material profile <material>/<profile>

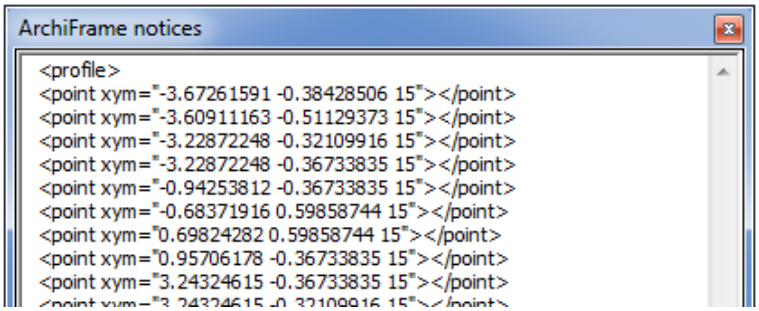
The default material list contains material XX 204x275 which contains profile.

```
<material id="XX 204x275" name="XX 204x275" thickness="0.204" height="0.275" shape="block"
maxlen="99" m3factor="0.000">
  <profile>
    <point xym="-0.10200000 0.23900000 15"></point>
    <point xym="-0.10200000 0.02000000 15"></point>
…
    <point xym="-0.10200000 0.23900000 -1"></point>
  </profile>
</material>
```

Profile is picked from fill placed to the origin. Origin defines the position of plank's reference line (bottom middle):



Profile is copied from ArchiFrame notices window and pasted to material's <*profile*>-tag.



Avoid using arcs that exist in the example as they make the 3D window slower. Arcs can be modelled with lines by changing the status code from 15 to 79. This will produce a 3D profile without unwanted lines.

17.2.2 Material lengthwise profile <material>/<profile_xz>

Lengthwise profile makes it possible to model decorative boards. However, there won't be cnc- for any profile definition. The example below contains fills for plank beginning, multiplied middle and end parts. Please note the point selected for picking is at the begin part's lower left corner:

Lengthwise profile is picked as normal profile and is placed under following tags:

```
<profile_xz>
  <beg>
    <point xym="0.00000000 0.27500000 79"></point>
    …
    <point xym="0.00000000 0.27500000 -1"></point>
  </beg>
  <mid>
    <point xym="2.00000000 0.27500000 15"></point>
    …
    <point xym="2.00000000 0.27500000 -1"></point>
  </mid>
  <end>
    <point xym="0.00000000 0.27500000 79"></point>
    …
    <point xym="0.00000000 0.27500000 -1"></point>
  </end>
</profile_xz>
```

18 Element definition file ArchiFrameElements.xml

This file is used to define all element types, element elevation layout and other settings related to placing the planks to the element. All dimensions are given in meters and a decimal separator is shown as a full stop, for example 1.234.

18.1 Settings <setting>

Contains general element related default settings.

18.1.1 Copy&paste between Archicads <trackelemsettings>

If some custom element stamps/cut list/weight marker objects are used instead of the built-in ones, those should be added here:

```
<!-- These libparts will be tracked for changes in addition to AF internal ones. Add any
libpart type here used as custom element stamp, cut list or simila. Data is used for handling
copy/paste between ArchiCADs -->
<trackelemsettings>
  <libpart guid="FD6C137D-C647-42FD-A904-181281909443"></libpart>
  <libpart guid="9056D768-6A80-423F-B548-30D0A09D857F"></libpart>
  <libpart guid="9FD03D30-6A12-4B0F-AB3A-01410A529E38"></libpart>
</trackelemsettings>
```

It will allow ArchiFrame to add needed information to get copy&paste work well.

18.1.2 Plank related settings <plank>

```
<misc minlen="0.050" idstr="%s-01" sort="xy" topbotunid="1"></misc>
```

- minlen, shorter planks are not created.
- idstr, ID string form, %s is replaced with element ID. Default is %s-001.It is possible to refer to plank's parameters like this: %s-01[iElemModule]
- boardidstr, ID string for boards. Format is like idstr.
- sort, sorting when giving IDs, possible values.
    - yx, assing from bottom to top then from left to right (default).
    - xy, assign from left to right then from bottom to top.
    - globxyz, compare global bound box, first x, then y and finally z.
- topbotunid, if set to 1 top & bottom including double parts will not get the same ID as any other piece even if those are similar (will always have unique ID). Default is 0.

18.1.3 Default settings for element drawings

```
<projections maxwidth="80">
```

- Maxwidth defines when ArchiFrame places next element drawing/element elevation to new row above existing ones. Default value is 80 meters.

18.1.4 Default settings for new element <newelem>

See section XML-settings for an Archicad element.

18.1.5 New plank's default settings <newelemplank>

Defines settings for original plank. Original plank is the one that is visible in 3D. See section XML-settings for an Archicad element.

18.1.6 Element elevation from front and back side defaults <newelemprojside>

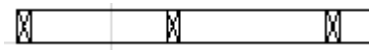
Defines settings for side projection planks. For example, ID is good to be shown here but not at the top projections. See section XML-settings for an Archicad element.

18.1.7 Element elevation from top defaults <newelemprojtop>

Defines settings for top projection planks. See section XML-settings for an Archicad element.

To get the column cross lines also to top projection, add following setting:

```
<objparam name="iColLines">3</objparam>
```



18.1.8 Dimension lines settings <dimlinesettings>

Base settings for every dimension line is read here first. See section XML-settings for an Archicad element.

18.1.9 Pre-settings for dimension lines <dimsettings> ja <dimlines id="id">

This section contains a dimension line definitions used in many element types. The section contains <dimlines id="id"> tags. Xml-attribute id is the id used to refer to the settings, for example, front side projection uses id "wall_elevation". Definition contains one or more single dimension line (<dimline>).

To use Lua-script for finding pieces to add to the dimension line attribute sript=”xxx” is added:

```
<dimlines merge="addpre" id="maindims" script="yit_projplanks">
```

This is a reference to <script id="yit_projplanks"> in xml-path archiframe/elem/settings/scripts.

18.1.10 Lua script to find relevant pieces to add to the dimension line

Used for maximum flexibility to control the content of the dimension line. An example at xml-path archiframe/elem/settings/scripts:

```
        <script id="yit_projplanks" merge="addpre">
          <![CDATA[
-- Sets ID to middle for horizontal planks
function DoProjSettings()
  local i, z

  for i=1,gnPlanks do
    ac_objectopen(string.format("#%d",i))
    -- Vertical (studs) with different fill
    z=ac_objectget("iEndZ")
    if z and math.abs(z)<0.001 then   -- nil if board
      -- Horizontal piece, ID to middle
      ac_objectset("iIDPlace", 0)
      ac_objectset("iXoffID", 0)
    end
    ac_objectclose()
  end
end

-- Dim line creator uses to find top and bottom plates only
function FindTopBottomPlanks()
  local i, s, n
  local tblRes, tbl

  tblRes={}
  n=0
```

```
  for i=1,gnPlanks do
    ac_objectopen(string.format("#%d",i))
    z=ac_objectget("iEndZ")
    if z and math.abs(z)<0.001 then   -- nil if board
      s=ac_objectget("iElemGroup")
      if s and (string.match(s, "^bottom_force%a*") or string.match(s, "^top_force%a*"))
then
        tbl={}
        tbl.index=i
        tbl.anchorfirst=1
        tbl.anchormid=0
        tbl.anchorlast=1
        n=n+1
        tblRes[n]=tbl
      end
    end
    ac_objectclose()
  end
  return tblRes
end
]]>
        </script>
```

Find function returns 1-based table where each item has fields:

- index, the 1-based (projection) plank index.
- anchorfirst/mid/last, overrides to any settings in the xml-attributes.
- anchorfirstskipnext, anchormidskipnext, anchorlastskipnext: Value 1 will not add a dimension to next point from current point.

18.1.11 Lua script to build all dimension lines

Allows script to use sophisticated logic to create only needed dimension lines and to add points using more logic than just for example adding all studs to the dimension line.

```
<dimlines id="dim_wallsect_nowin_horint_10" script="dim_wallsect"
scriptcreate="SectDimLines">
  <!-- Base dim lines could be added before script based -->
</dimlines>
```

As previous but will define everything related to the dimension lines:

- The xml to create dimension line optionally with text
- Dimension points to add

The dimension line creation script returns 1-based table each cell containing fields:

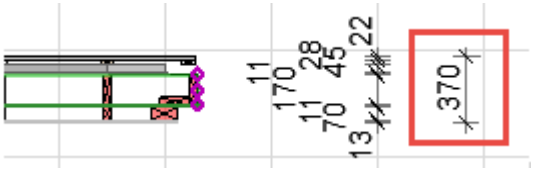
- xmldim, the xml defining the dimension line
- dimpoints, 1-based table of dimension points each containing one of the two options:
  - index, anchorfirst, anchormid, anchorlast as in previous section
  - x, y The static point to add to the dimension line

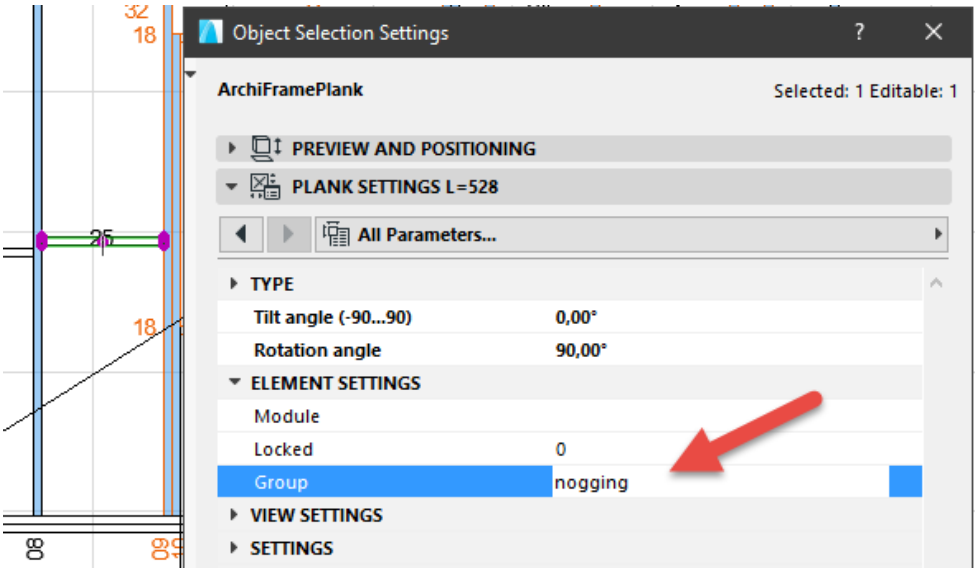
Please search for text SectDimLines in default data folder's ArchiFrameElements.xml for details (C:\ArchiFrame\data\ArchiFrameElements.xm).

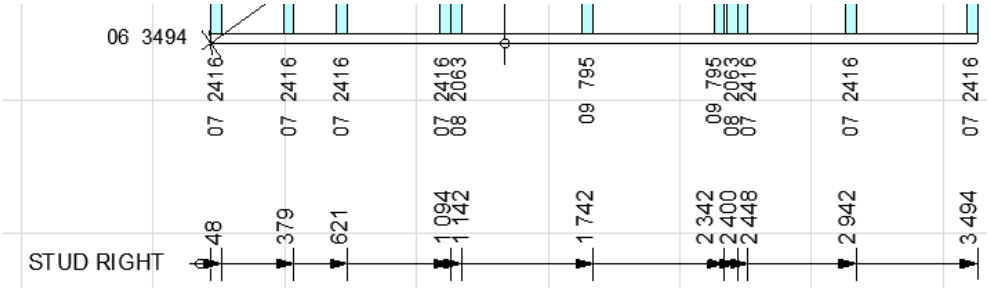
18.1.12 Single dimension line settings <dimline>

```
<dimline side="bottom" margin="1" elemvecx="1" elemvecy="0" elemvecz="0" addminmax="1"
addx="0" addy="0" addz="0" anchorfirst="0" anchormid="0" anchorlast="0" exclude="extstud">
  <dimlinesettings>
    <elemparam name="fontsize">2</elemparam>
    <text content="STUD MID" textid="10" align="6" marg="0.2">
```

```
      <elemparam name="pen">11</elemparam>
      <elemparam name="fontname">arial</elemparam>
      <elemparam name="fontsize">2</elemparam>
      <elemparam name="fontstyle"></elemparam>
    </text>
  </dimlinesettings>
</dimline>
```

XIm-attribute	Description
side	Where to place the dimension line from the projection/elevation. Value is one of "left","right","bottom","top".
margin	Marginal between projection and dimension line in meters.
marginproj	An alternative to margin=xxx, will place the dimension line given distance from the projection. ArchiFrame expands total bound box with the resulting dimension line.
elemvecx,y,z	Dimension line direction in the element coordinates. Note! In wall element Y-axis goes up, Z-coordinate is along the watching direction. Length of the direction vector must be 1.
allowanydir	If set to 1 it is possible to define right to left and top to bottom dimension lines. Otherwise ArchiFrame swaps the dimension line direction to be from left to right or bottom to top.
scriptfind	The function name in the referenced Lua-script used to find planks relevant for this dimension line. For example: scriptfind = "FindTopBottomPlanks".
addminmax	The element's contour line minimum and maximum position are added to the dim line. For horizontal the left and right side are added and for vertical the top and bottom sides. Special values: • 2, add just min. • 3, add just max. To get total thickness like below, following values are used:  • 11, add total composite min&max. • 12, add just min. • 13, add just max. Related attributes: • addminplankmindist = "0.050" adds dimension point only if the closest plank dimension point is further that 50 mm. For min end. • addmaxplankmindist = "0.050" for max end. Other way round (do not add element edge dim if plank is closer): • addminplankmindistfrom = "0.050" adds dimension point for plank only if closest plank dimension point is further that 50 mm from end. For min end. • addmaxplankmindistfrom = "0.050" for max end.
addopeningmin	For horizontal dim line: Add opening's left side to the dim line. For vertical dim line: Add opening's bottom side to the dim line.

addopeningmax	For horizontal dim line: Add opening's right side to the dim line. For vertical dim line: Add opening's top side to the dim line.
addsidesminmax	Add first and last point from the element sides. For vertical dim line both left- and right-hand side corner points are added to the dim line.
addx,y,z	Direction of the planks to be added to the dim line. Direction is defined in element's local coordinates. For example, in a wall element the vertical studs have direction addx = 0 addy = 1 addz = 0 and horizontal planks have direction addx = 1 addy = 0 addz = 0. Z-direction is probably never used in wall elements.
anchorfirst	Is the plank's first side added to the dim line. Useful for example if dimensioning from element's left side to plank's left side.
anchormid	Is the plank's middle point added.
anchorlast	Is the plank's last side added.
limitdist	To have different dim lines for studs at bottom and top. The maximum distance from plank's end to element polygon contour line.
minpoints	If defined, there must be at least given number of points for at dimension line to be created.
exclude	Exclude planks and element layers on given type = “xxx” layer. For example exclude = “extstud”. `<layer ref="WALL 19" anchorname1="Ext studs ext" type="extstud" studs="core">`
boardpanels	To be used with exclude = “*”, include layer types to handle here. Also, must be given to include boards: include = “*boarding*”.
objtype	Attribute objtype can be added to include only planks (objtype="1"), boards (objtype="4") or both (objtype="5"). Default is -1 which includes everything.
excludegroup, includegroup	Used to exclude/include planks based on their role in the element. The role is defined in ArchiFramePlank’s parameter:  For example, to collect only noggins: excludegroup="*" includegroup="nogging*" To give many names, please use comma (,) as the separator.
includeelem	To include only element object, if given, overrides include when processing element objects. To include layers not visible in the projection, the layer

	name is prefixed with exclamation mark (!). For example: includeelem="*boarding*,**!**core"
onlyelem	If given onlyelem = "1", no planks/boards are included in the dimension line – just the *ArchiFrameElement*-object making up the layer. Useful to create layer offset dimension lines.
textrot	Setting textrot = "90" will produce following result (angle is global angle):  Please note that this setting causes the text to be manually placed and Archicad/ArchiFrame will not clean up overlapping texts – that will remain as manual work. Also notice that dimension text layouts are scale specific and the dimensions are rotated only in the scale the dimension line was originally created. To force recreating the dimension line it must be deleted and recreated by element tools/*Update*-command.
textrotoffx textrotoffy	May be used with textrot-attribute. Defines the distance to move the text to left from default position in x- and y-directions. If not specified, the default value of 75 mm is used.
textrotanchor	If given, the dimension text anchor point is: APILbl_MiddleAnchor= 0 APILbl_TopAnchor= 1, APILbl_BottomAnchor= 2, APILbl_Underlined= 3 (default)

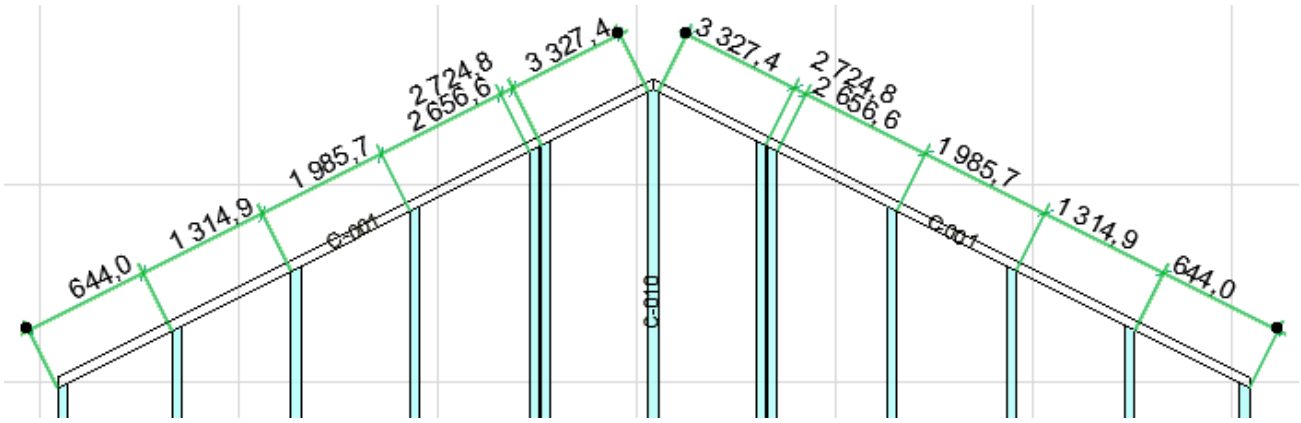
Definition may contain Settings <setting> inside <*dimlinesettings*>-tag that affect just this single dimension line. In addition it may contain text related to the dimension line. Text-element may contain attributes:

Xlm-attribute	Description
content	Text.
textid	Text ID that must be unique inside the whole element drawing. The ID is used to detect unchanged dimension lines during update. Value is from range 1-254. Value can be given as an expression that uses variable for projection number. For example, `textid="projid*20+4".`
anchor	Text anchor point and placement relative to the dimension line. For example value 6 places text to the left side of the dimension line and the text extends to left from the anchor point. Anchor points are: 123 456 789
marg	Distance to the dimension line in meters.

18.1.13 Dimension line for gable wall top planks

```
<!-- Dimension line for gable wall top tilted planks -->
<dimspec type="toptilted" margin="0.3" addminmax="2" anchorfirst="1" anchormid="0"
anchorlast="0" ltor="0">
  <dimlinesettings>
    <elemparam name="dimensiontype">cumulative</elemparam>
  </dimlinesettings>
</dimspec>
```

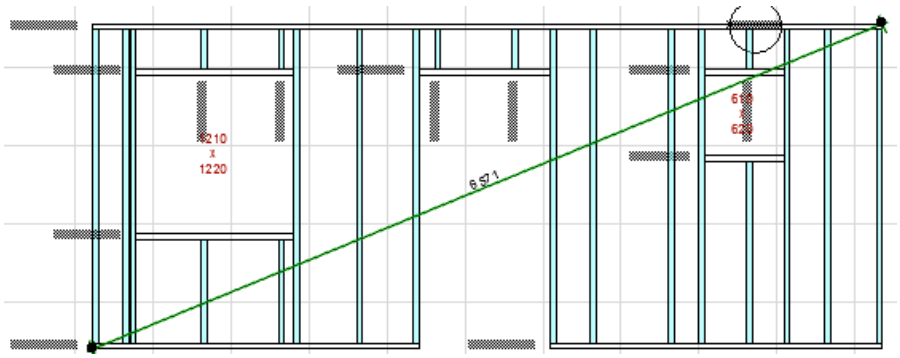
With *dimspec*-tag and attribute type="toptilted" it is possible to get the dimension lines like below. With attribute ltor="1" the dimension line is always from left to right. Otherwise it is from bottom to top. Setting ltor="2" will place dim point to the left side of the stud on the right-hand side also.



18.1.14 Cross dimension line

```
<!-- Cross dimension for the element -->
<dimspec type="cross" anchor="lb">
  <dimlinesettings>
  </dimlinesettings>
</dimspec>
```

With *dimspec*-tag and attribute type="cross" it is possible to get cross dimension line for the rectangular shapes – other shapes must be dimensioned manually. Attribute anchor can have values "lb" (left bottom) and "rb" (right bottom).

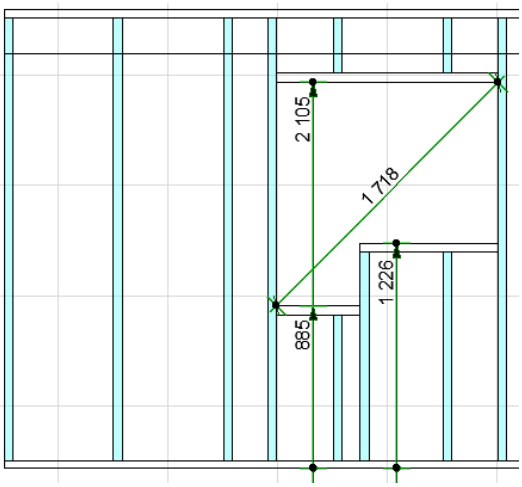


18.1.15 Openings dimension lines

```
<dimspec type="opening" include="core" vertdim="1" crossdim="1" vertanchorbot="top">
  <dimlinesettings_vert>
    <elemparam name="dimensiontype">cumulative</elemparam>
    <elemparam name="markertype">5</elemparam>
  </dimlinesettings_vert>
  <dimlinesettings_cross>
  </dimlinesettings_cross>
```

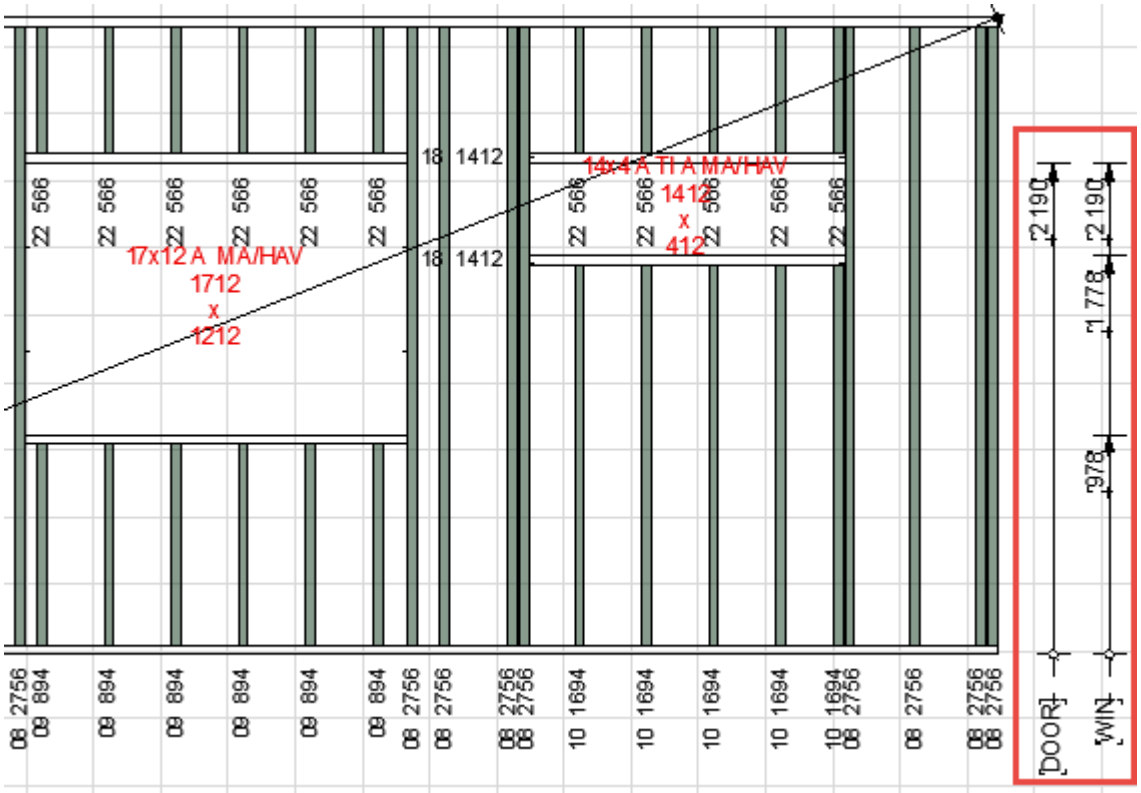
```
</dimspec>
```

With *dimspec*-tag and attribute type = “opening" it is possible to add individual dimension lines for each opening. Attributes *vertdim* and *crossdim* are used to enable/disable corresponding dimension lines.



Xlm-attribute	Description
include	Defines which class (es) of the planks visible in the projection are used to define the openings.
vertdim	Create vertical dimension 0/1.
crossdim	Create cross dimension 0/1.
vertanchorbot	Where to anchor the bottom of vertdim: default is bottom of bottom plate, value top anchors to top of the bottom plate.
vertanchortop	If missing (default), there is no dim line to the element’s top plate. If given, these values can be used: • "top", add dim point to topmost top plate point • "bottom", add dim point to lowest top plate point
vertopeningbot	-1=anchor to lower horizontal piece’s bottom side 1= anchor to lower horizontal piece’s top side (default)
vertopeningtop	-1=anchor to upper horizontal piece’s bottom side (default) 1= anchor to lower horizontal piece’s top side
grouping	Specified to group all window and/or door dimensions to single dimension line which is placed outside the projection. Possible values are: • 0, no grouping (default) • 1, group windows • 2, group doors • 3, group both to separate dimension lines

It is possible also to combine door & window dimension lines like below with grouping-attribute:




```
<dimspec type="opening" include="core" vertdim="1" crossdim="0" vertanchorbot="bottom"
vertopeningbot="1" vertopeningtop="-1" grouping="3">
  <dimline_door side="right" margin="0.25">
    <dimlinesettings>
      <elemparam name="dimensiontype">cumulative</elemparam>
      <elemparam name="markertype">5</elemparam>
    </dimlinesettings>
    <text content="DOOR" textid="projid*20+19" anchor="6" marg="0.2">
      <elemparam name="pen">1</elemparam>
      <elemparam name="fontname">arial</elemparam>
      <elemparam name="fontsize">1.5</elemparam>
      <elemparam name="fontstyle"></elemparam>
    </text>
  </dimline_door>

  <dimline_win side="right" margin="0.5">
    <dimlinesettings>
      <elemparam name="dimensiontype">cumulative</elemparam>
      <elemparam name="markertype">5</elemparam>
    </dimlinesettings>
    <text content="WIN" textid="projid*20+18" anchor="6" marg="0.2">
      <elemparam name="pen">1</elemparam>
      <elemparam name="fontname">arial</elemparam>
      <elemparam name="fontsize">1.5</elemparam>
      <elemparam name="fontstyle"></elemparam>
    </text>
  </dimline_win>
</dimspec>
```

Here tags <dimline_door> and <dimline_win> define settings for related dimension lines.

18.1.16 Common marking settings to be used in multiple places <elemmarkings>

These settings are referred from element projections and settings are given here to avoid copy & pasting the same settings many times. An example used under <elemtemplate>

```
<elemmarkings>
  <opening text="#id#\n#width#\nx\n#height#">
    <script ref="OmataloAukkoID"></script>
    <elemparam name="pen">1</elemparam>
    <elemparam name="fontname">Arial</elemparam>
    <elemparam name="fontsize">1.5</elemparam>
    <elemparam name="fontstyle"></elemparam>
    <elemparam name="just">center</elemparam>
    <framelines create="[create_openig_frame_lines;default=0]">
      <elemparam name="pen">1</elemparam>
      <elemparam name="linetype">1</elemparam>
    </framelines>
  </opening>
  <weight libname="{FD6C137D-C647-42FD-A904-181281909443}-{9060078A-0F41-48C8-8400-
8E414A838053}" create="[show_weight_front;default=3]" text="[weight_text;default=(weight)
KG]" units="[weight_unit;default=kg]">
    <settings>
      <layer>[layer_weight;type=layer]</layer>
      <objparam name="iFill">*25*</objparam>
      <objparam name="A">0.250</objparam>
      <objparam name="B">0.325</objparam>
    </settings>
  </weight>
</elemmarkings>
```

For openings minheight can be given to specify minimum height of an opening to create the marking text. Following texts are replaced for openings:

- #id#, all doors and windows located in the element hole are collected here. There may be, for example, many small windows next to each other. If there is a script defined, [id] refers to script defined text.
- #width#, #height# size of the hole in the element in setting's defined units.

For openings it is possible to use a script to define the opening ID as is used in the example above. An example script to pick Archicad window object's parameter under xml-path archiframe/elem/settings/scripts:

```
        <script id="OmataloAukkoID" merge="addpre">
          <![CDATA[
-- sOpeningGuid GUID of the related door or window
-- nFlipLR 1 if the ArchiFrameElement and original Archicad wall directions are opposite
function GetText(sOpeningGuid, nFlipLR)
  local res

  ac_objectopen(sOpeningGuid)
  res=ac_objectget("fdkarmi")
  if res==nil then
    -- Parameter missing, revert to opening's ID
    res=ac_objectget("#id")
  else
    -- Parameter found, that will be used instead of the ID
  end
  ac_objectclose()

  return res
end
]]>
        </script>
```

For weight marker the attributes are:

Xlm-attribute	Description
libname	The library part object to show the weight
create	Can be used for example in element templates. If missing or value is 1, the weight marker will be created. Possible values are: • 0, do not calculate. • 1, show the real position of the center of gravity. • 2, show above the elevation. • 3, show above the elevation and show separate value calculated. Without doors&windows if different.
text	The text to show, (weight) will be replaced by the actual height
units	kg or lb
movey	Offset for marker's y-coordinate

18.1.17 Common default settings to be used in multiple places <pre-settings>

These settings are referred from element projections and settings are given here to avoid copy & pasting the same settings many times. Settings are referred with tag name, in this case with xml-attribute settingsref = "cutlist":

```
<presettings>
  <!-- Used for cutlists -->
    <cutlist tab1="0.5" tab2="0.8" tab3="1.2" tab4="1.5" tab5="1.8" boards="1" sortby="len"
plankid="-" boardid="#">
    <elemparam name="pen">11</elemparam>
    <elemparam name="fontname">Arial</elemparam>
    <elemparam name="fontsize">4</elemparam>
    <elemparam name="fontstyle"></elemparam>
</cutlist>

  <!-- Bottom wood markings -->
  <markings id="mark_default">
    <bottomwood>
      <stud line="3" id="2" idline="1" markxpos="2" markypos="1" marksize="0"
marktextdir="1"></stud>
      <opening line="1" windowtext="Window [width]x[height]" doortext="Door
[width]x[height]"></opening>
      <neighbour begtext="&lt;[id]" endtext="[id]&gt;" marksize="0"
marktextdir="1"></neighbour>
    </bottomwood>
  </markings>
</presettings>
```

<*cutlist*>-tag may have following xml-attributes (<cutlist tab1 = "0.6" tab2 = "0.9" boards = "0" sortby = "id">):

Xlm-attribute	Description
create	If set to zero, the cut list is not created (useful for element templates)
tab1	Tab stop for number of similar pieces in meters.
tab2	Tab stop for start of IDs, also tab3, tab4 …tab10 can be given.
plankid	Separator character for plank ids: • Missing or empty, use full plank ID (default). • Separator character -, include just the number part. • "No", no plank IDs.

boards	Values: • “0”, boards are not included into cut list. • “1”, boards first then planks (default). • “2”, planks first then boards.
boardid	Include board ids to the cut list, possible values: • Missing or “no”, do not include (default). • Separator character #, include just the number part. • Empty, full ID.
sortby	As default the planks are sorted by length, sortyby=”id” changes this to ID.
exclude	Exclude planks on given type = “xxx” layer. For example, exclude = “extstud”. <layer ref=”WALL 19” anchorname1=”Ext studs ext” type=”extstud” studs=”core”>
matname	How to show material id and name in the cut list, default shows just the material id. String contains parts [name] and [id] which are replaced with the current values for the material. For example, matname = “[name] ([id])”. Used only if material id and name are different.
settingsref	Use settings from given tag.
elemid	Include element id to begin of the cut list text. Default value "1", "0" = do not include element id in the cut list. It may also be free text with tags to replace. For example, elemid = "#id# - #floorname#". See the tags here.
script	Construct whole content after possible elemid with given script, see Lua-script for cutlist.
libpart	Use library part for the cut list. For example to refer ArchiFrameTable, use libpart="{9056D768-6A80-423F-B548-30D0A09D857F}".
libscript	Must be used in conjunction with attribute libpart. Builds up the cut list.
libscriptid	Since script is loaded only once per handing of all element’s projections. Must be specified to use different script(s) to buid the cut list.

18.1.17.1 Custom door and window weight calculation <doorwinweight>

If given, under this tag it is possible to define a Lua script that takes opening GUID and returns its weight in kilograms. For example, the default placeholder:

```
        <doorwinweight>
          <![CDATA[

-- Script loaded once and called for each opening
-- Return 0.0 to have default weight calculation, otherwise returned value will be used
function OnCalcDoorWinWeight(guid)
  local res

  res=ac_elemget(guid, "props=1,ifc=1")
  return 0.0
end
]]>
        </doorwinweight>
```

18.1.18 Lua-script for cutlist

There are two options: to build the textual cut list or to build an object containing all needed information. To build the content of the cut list ArchiFrame calls Lua-function DoCutList (nSeqNum). Parameter nSeqNum (0…N-1) tells the sequence number of the cut list for current composite element. For multiple cut lists for an element, ArchiFrame keeps the Lua script in

memory during processing single ArchiFrame element. This way the script may keep data in memory as Lua globals between calls to DoCutList ().

18.1.18.1 Textual cut list

Function must return 1-based table of text runs. Each item in the table may contain fields:

- str, mandatory text. Use \n to add line feeds.
- sizemul, multiplication factor for the size, default is 1.0.
- bold, use bold style.
- italic, use italic style.

18.1.18.2 Library part cut list

It is easiest to use these built-in definitions that pick just part of the pieces in the element:

- cutlist_table_coreall
- cutlist_table_corestuds
- cutlist_table_coreothers
- cutlist_table_coremodules
- cutlist_table_insulation
- cutlist_table_extboards
- cutlist_table_extcladstuds
- cutlist_table_intcladstuds
- cutlist_table_extstuds1
- cutlist_table_extstuds2
- cutlist_table_extclad
- cutlist_table_intboards
- cutlist_table_intstuds
- cutlist_table_allplanks
- cutlist_table_acc to list the balk shoes and other accessories in the planks&boards

These are referred for example from own composite element template definitions like below:

```
<group layout="horizontal">
  <cutlist layoutmargins="6.5,0.1,0.1,0.1" settingsref="cutlist_table_corestuds"
layoutminheight="3"></cutlist>
  <cutlist layoutmargins="0,0.1,0.1,0.1" settingsref="cutlist_table_coreothers"></cutlist>
</group>
```

The script fills given new cut list library part object. ac_objectopen(nil) will open the current table GDL-object to process (always new).

<luasettings> tag can also be added for example to add definition gnFloorRoof telling ArchiFrame the type of the element to get correct element role names into the cutlist (gnFloorRoof is checked by the default code in ArchiFrameElements.xml):

```
<cutlist layoutmargins="2,0.1,0.1,0.1" settingsref="cutlist_table_coreall">
  <luasettings>
    <![CDATA[
      gnFloorRoof=1   -- nil=not known or a wall, 1=it is a floor, 2=it is a roof
]]>
  </luasettings>
</cutlist>
```

Cells may have tags before actual text for formatting. Possible tags are:

- <mulheight number>, multiplication factor for text size. Default=1.
- <merge number>, merge next number cells to current cell in this row.
- <nolines>, no lines for the cell.
- <b>, **bold.**
- <i>, *italic.*
- <u>, underline.
- <left>, left align.
- , center align.
- <right>, right align.
- <vtop>, <vcenter>, <vbottom>, vertical alignment.
- <fillrgb r g b>, background fill color. Values r g b between 0…1, red is <fillrgb 1 0 0>. Or if it contains just single value, then it defines the pen number directly.
- <penrgb r g b>, pen color.

Cut list object has parameter *Manage manually* (iManual in xml-file) that can have these values:

- 0, let ArchiFrame update content and position.
- 1, full manual – ArchiFrame will not to touch it.
- 2, keep position but let ArchiFrame update content. This value is useful if the elevation layout needs hand tuning.

For library part cut list function DoCutList() may return values:

- 0, do not create the list (list is empty)
- 1, normal operation (this is the default option if there is no return value from DoCutList)

Default cut list script cutlist_table_an11 supports giving settings from Lua global table gListSettings. It allows customizing the cut lists much further. This customization script gets environment set by the default cut list script, default one gives these:

- gLang, three letter language code: "eng", "fin" etc.

Table gListSettings may have following fields with the default cut list script:

Field	Description
title	Title for the cut list. Following replace-strings may be used: • #id# will be replaced by the element ID
cols_skiptitle	If set to true, the first title line will be skipped.
rowheight	Table row height setting also sets the font size. Default is 0.170 (170 mm).
tablewidth	If given, this value is used. Default is 3.5 m / 4.5 m / 5.0 m depending on column count.
funccollect	Function that defines if a piece should be included into the cut list. Function has single parameter: Data of the collected piece. Content of the data depends on the cut list script – default data is described in the following table.
funcisless	If order or grouping of the planks is different than the default. Takes two planks/boards and returns true, if first one is less than second one. Requires funccollect to be set. The default comparator function is in ArchiFrameElements.xml function CmpPlanks() – that is a good base for a custom one.
funcgetrows	If defined, will tell the whole content of the cut list. Function is called without any parameters, and it must return 1-based array of rows each item having

	another 1-based array cols which contains text for each column. Please see cutlist_table_acc from base data for details.
collectexternal	If given and value is true, ArchiFrame collects even external attached planks for cut list processing. Planks are collected only once per processed element – this must be given in the first used cut list per element. See also next item.
addexternal	If given and value is true, external planks are actually given to to funccollect. Otherwise, they are silently skipped.
resetlist	If set to true, will collect all pieces instead of using cached plank list
collectall	If true, will consider each plank/board separate. Useful to go through every piece in the element.
addmc	Add machinings column to the table: 0/1
addlen	Add length column, for boards this should be 0.
cladaddmat	Add cladding material name to cladding piece ID and include it in grouping
col1str_plank	Replace first column's text with this text. Parts to be replaced: • #id#, plank's ID • #matwidth#, typically smaller measure • #matheight#, typically bigger measure • #len#, plank's length • #typename#, material's type name. ArchiFrame puts both name and id if they are different in form: name (id) • #matname#, just the material name • #matid#, just the material id • #elemgroup#, the role in the element: "Stud", "Bottom plate" etc. • #cutinfo#, cut information for the piece, for example "Angled cut 27 deg" • #pcs#, number of similar planks • #par:xxx#, take plank object's parameter. For example #par:iQuality# By default, the script converts the value to string using default rules. To specify value type, it is possible to give a prefix to the parameter name, for example #par:lendim:iCurrWidth#: o lendim: It is a length, use AC-settings for dimension line dimension points o lencalc: It is a length, use AC-settings for calculation
singlelist	If given and value is true, there will not be subheadings like CORE / CORE OTHERS in the cut list – everything in single list.
sepbylayer	If set to true, similar pieces having the same ID will be split by the owning layer type. This way it is possible to get parts from boarding_ext2 and boarding_ext3 to their own cut lists.
clearelemrole	If set to true, plank's role in the element is cleared. This is useful not to separate for example top and bottom plates that have the same ID.
addinsu	Set to true to collect insulation into the cut list, use with resetlist. Perhaps insulation cut list should be the last one not to include insulation in other cut lists.

Default cut list generator fills these fields for every collected part (can be used in the *funccollect*):

Field	Description
elemtype	Owning element layer's type: core, intstud, extstud etc.

elemtypesort	Internal coding that defines the part's role in the element. Possible values and the default names are: • 001, STUD • 002, OTHERS (just core layer non-vertical piece) • 010, INT STUDDING • 020, EXT STUDDING • 030, CLADDING • 040, BOARD EXT • 050, BOARD INT • 999, OTHERS (unknown layer type core/intstud/extstud etc)
id	Part's ID
plankinfo	af_request ("plankinfo") for the part
more…	Please see function CollectPlanksBoards() for more fields.

It is also possible to define content for the cutlist more completely setting information for all columns like below. Please note that sum of table widths should be 1:

```
gListSettings={}
gListSettings.title="OTHERS"

gListSettings.funccollect=function(plank)
  if not plank.elemtype or not string.match(plank.elemtype, "core") then
    return false
  end
  if plank.elemtypesort~="002" then   -- Non-vertical piece
    return false
  end
  return true
end

gListSettings.cladaddmat=false     -- Must be in every item since this setting is used when
the list is first time created
gListSettings.rowheight=nil        -- Use defaults
gListSettings.tablewidth=nil

-- Custom columns
gListSettings.tablewidth=4.5
gListSettings.cols={}

t={}
t.title="<b><left>TYPE"
t.content="<left>#elemgroup# #typename# (#id#)"
t.width=0.42
gListSettings.cols[1]=t

t={}
t.title="<b>LEN"
t.content="<right>#len#"
t.width=0.10
gListSettings.cols[2]=t

t={}
t.title="<b>PCS"
t.content="<center>#pcs#"
t.width=0.08
gListSettings.cols[3]=t

t={}
t.title="<b><left>CUTS"
```

```
t.content="<left>#cutinfo#"
t.width=0.4
gListSettings.cols[4]=t
```

18.1.19 Lua-scripts to be used later <scripts>

Instructions are in section Lua-scripts with elements.

18.2 Element type <elemtypes> and <elemtype>

Element types are listed with tag <elemtype> inside container tag <elemtypes>. Content of the <elemtype> tag is <elemtype id = "short_id" name = "name for user">. Xml-attributes:

Xlm-attribute	Description
id	Element type ID that remains the same even if the language changes.
name	Element type name for the user. If missing, id is used.
idstamp	Text to put to element stamp at bottom of the elevation.
Tolist	With value idstamp = "0" the element type is not added to the element tool palette.
Density	Defines the insulation, boarding or panelling density in kg/dm3. Default value is 0.050 (50 kg per m3).

After this comes <*settings*>-tag that may contain settings to override common <settings>-part, for this element type.

Also, it is possible to set some options on as default by giving <*options*>-tag like below:

```
      <elemtype class="wall" id="WALL 173 K600" idstamp="42x173">
        <!-- Default settings for this element type -->
        <settings>
          <newelem>
            <elemparam name="pen">1</elemparam>
          </newelem>

          <newelemplank>
          </newelemplank>
          <newelemprojside>
          </newelemprojside>
        </settings>

        <!-- Could set some options on here as default, settings may be given inside
<option>settings</option> and format depends on the option -->
        <options>
          <option id="openingsides">
            <![CDATA[
mat=block
thickness=0.050
height=0.173
]]>
          </option>
        </options>
```

18.2.1 Element weight calculation

It is possible to give the weight definitions also to old style xml-definitions (instead of using own element type definitions which produce this kind of xml-definition):

```
<elemtype class="wall" id="WALL 173+42 PANEL OUTIN" idstamp="173+42">
  …
  <!-- Default element weights for this type -->
```

```
  <elemweight>
    <bools calc="0" door="1" win="1"></bools>
    <floats framingkg="400" boardingkg="1150" insulationkg="35" doorm2kg="10"
doorframekg="4" winm2kg="15" winframekg="4"></floats>
  </elemweight>
```

The attributes for bools are:

Xlm-attribute	Description
calc	Calculate weight for the element 0/1.
door	Calculate doors 0/1.
win	Calculate windows 0/1.
forceframing	Force framing weight to value given in floats-part (instead of using weights given in ArchiFrameBlocks.xml).
forceboarding	Force boarding weight as for framing.

And for floats:

Xlm-attribute	Description
framingkg	Framing weight in kg per cubic meter.
boardingkg	Boarding weight in kg per cubic meter.
insulationkg	Insulation weight in kg per cubic meter. Value 0 is good if it is air space.
doorm2kg	Weight for door area kg per square meter.
doorframekg	Weight for door frame kg per frame meter including bottom.
winm2kg	Weight for window area kg per square meter.
winframekg	Weight for window frame kg per frame meter including bottom.

Default weights set internally inside ArchiFrame are like this:

```
  <elemweight>
    <bools calc="1" door="0" win="0"></bools>
    <floats framingkg="450" boardingkg="1100" insulationkg="30" doorm2kg="15"
doorframekg="5" winm2kg="20" winframekg="5"></floats>
  </elemweight>
```

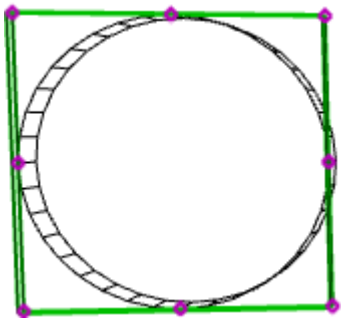
Then the weight settings are searched from each construction layers' *<elemweight>*-tag. Composite structure cannot have weight settings.

18.2.2 Element layers <layers> and <layer>

Now only the core is modeled. Layers can be defined to allow easy anchoring of the element's gypsum board inner surface with Archicad wall side. Layers are listed starting from exterior.

```
<layer name="Core" visible="1" thickness="0.195" type="core" anchorname1="Core ext"
anchorname2="Core int">
```

Xlm-attribute	Description
name	Layer name that is visible in element tool's anchor list.
visible	For future.
thickness	Layer thickness.
type	Core must be specified with setting: type = "core"
anchorname1	Name of the layer's exterior side to the anchor list.
anchorname2	Name of the layer's interior side to the anchor list.
circletorect	Maximum size of an opening that will be converted to bounding rectangle if it has any arcs. For example, here the selected layer is converted and the one

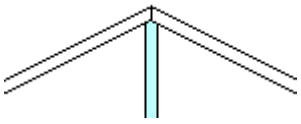
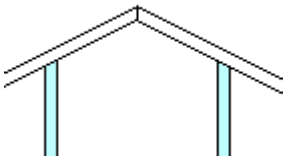
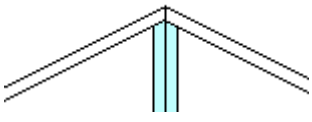
	behind has the line segments following the original circle:  circle_to_rect_max_diameter in own element type settings.

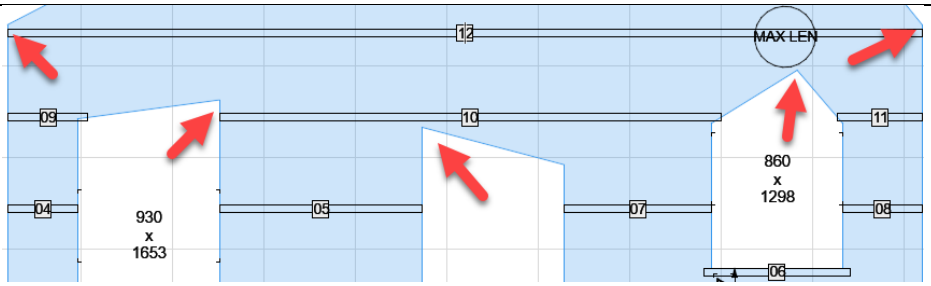
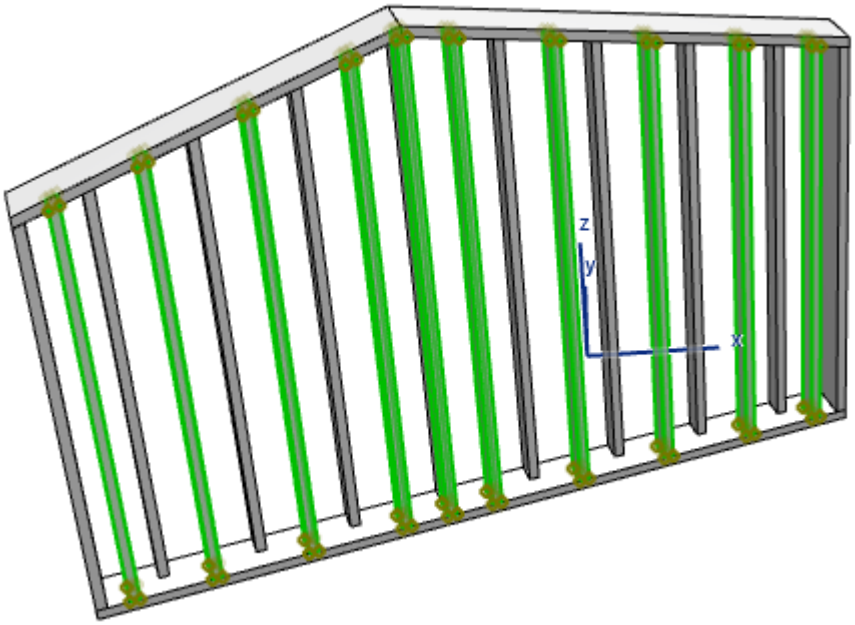
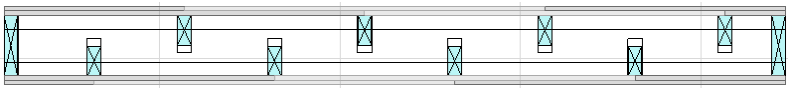
18.2.3 Planks to the layer <planks>

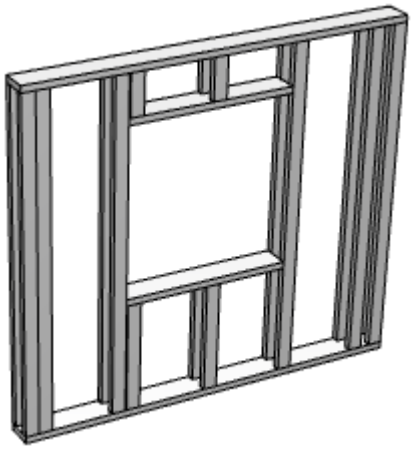
Defines single plank group. Group has a name that is used to for example to adjust plank groups together, creating grooves and other machinings.

```
<!-- Group name will have suffixes: _y=vertical planks, _x=horizontal planks -->
<!-- Group names can be used for example to make grooves for balks and groups may be
referred using wild cards: vertical* -->
<planks group="vertical" axis="y" opening_parallel="1" opening_normal="1" spacing="0.6"
spacingtolerance="0.01" gablehipsplit="0" studs="core" followstagger="50">
  <material id="L123" zoff="0.000" rotangle="0">
    <objparam name="iUsageId">XXX</objparam>
    <elemparam name="pen">8</elemparam >
  </material>
  <force>
    <parallel x1off="0.141"></parallel>
    <parallel x2off="-0.141"></parallel>
  </force>
</planks>
```

Xlm-attribute	Description
group	Group name, ArchiFrame will add postfix at the end of given name (for example "vertical_y"): • _y, plank in Y-axis. • _x, plank in X-axis. • _opening, inclined edges in openings. • _force, forced planks. • _spacing, planks added due to spacing rule.
axis	Planks base direction in element's coordinates, either "x" or "y" or "unused" to process unused contour edges. If used value axis="unused", two more attributes are available: • contour_unused = "1", handle contour line (default = 1). • opening_unused = "0", handle holes (default = 0). Also, value axis="unused_angled" can be used. Then planks will be placed only to edges which are not in x- or y-axis.
skipaxis	With value "1" there will not be planks in the element's base direction.
opening_parallel	• "0", do not add. • "1", add planks to axis-direction edges in openings extending to the polygon.

	• "2", do not extend to the polygon (useful for horizontal studding) but extend by plank's width on the element surface. • "3", do not extend to the polygon and do not extend by plank's width. Please see this.
opening_normal	As previous but axis is rotated 90 degrees from the base direction. Please see this
spacing	Spacing distance in meters.
spacingtolerance	Please see here.
stackingoffset	By default, the spacing will start from the edge of the element object. For example, with spacing value of 600 mm, the first stud will be placed 600 mm from the element edge. Setting this attribute makes the middle line of the first piece to be placed given distance from element's left/right edge. Axis is the base direction, for vertical (axis = "Y") positive starts from left and negative from right, for horizontal (axis = "X") positive starts from top and negative from bottom. This is useful to get evenly spaced pieces for log kind of structures. Also, it is useful if there is a piece with width 90 mm that should be spread evenly using spacing 100 mm.
gablehipsplit	Value is "0" or "1": Default value "0" will add a stud to every corner at the elements top side. With value "1" there will be two studs halving the given spacing from the corner position to left and right. With value "2" there will be two studs touching each other on both sides of the hip. Value 0:  Value 1:  Value 2: 
skipcorners	Avoid getting planks to each polygon corner with the default value 0:

	
create	To be used in element templates. Value is "0" or "1" (default and used if attribute is missing). Value "0" will skip this definition.
studs	If given, tells which layer is followed in current layer. For example, studs="core" means that current layer follows the core layer's studs. Can be used in both framing and boarding layers.
followstagger	With non-zero value defines that pieces are placed staggered to the followed layer. Used in conjunction with studs="xxx". An example: 
zstagger	Another way to make double framed part. Non-zero value gives result like this:  Spacing still defines distance between studs. For example, if spacing of 600 mm is needed to each side, the spacing value for the element must be 300 mm. Example: zstagger="1"

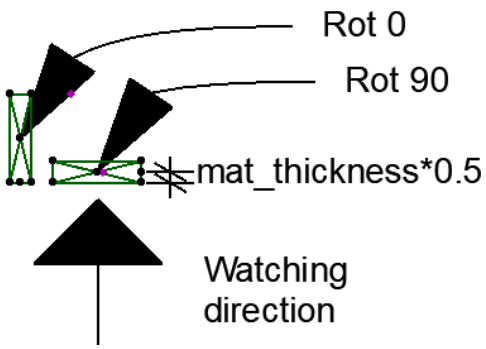
duplicate_zoff	Option to produce double studs for example to be used with SIP-structures:  This is length value how far in the watching direction the secondary piece is placed.
Studs fill pen	The fill pen to use with wall studs: 

18.2.4 Material settings <material>

Similar material settings are used with element and weatherboard settings. It is possible to give settings related to just these planks (see XML-element settings).

```
<material id="block" thickness="0.015" height="0.075" zoff="0.000" rotangle="0">
  <objparam name="iUsageId">XXX</objparam>
  <elemparam name="pen">8</elemparam >
</material>
```

Xlm-attribute	Description
id	Material id, "block" is general type and it also requires giving the material size.
thickness	Material thickness only if id is "block".
height	Material width/height only if id is "block".
xoff,yoff	Move in element's X- and Y-axes.
zoff	Move in element's Z-axis. For example value -0.050 places the plank's reference line 50 mm in front of the current layer.
linexoff	Move to right relative to current line, negative moves to left.
rotangle	Plank rotation angle in degrees. ArchiFrame sets the rotation so that the plank extends towards the watching direction. This value is added to that. For example, plank with size 50 x 100 mm is placed with default settings to the layer's front surface and the plank extends 100 mm in the watching direction.

	Rotation and zoff affect like this:  Please see information about the planks.

Following variables may be used inside <*material*>-tag:

Variable	Description
mat_thickness	Material thickness as given in ArchiFrameBlocks.xml, for example: `<material id="L128RL" name="21x120 VL" thickness="0.021" height="0.120"`
mat_height	Material height/width.

18.2.5 Forced planks <force>

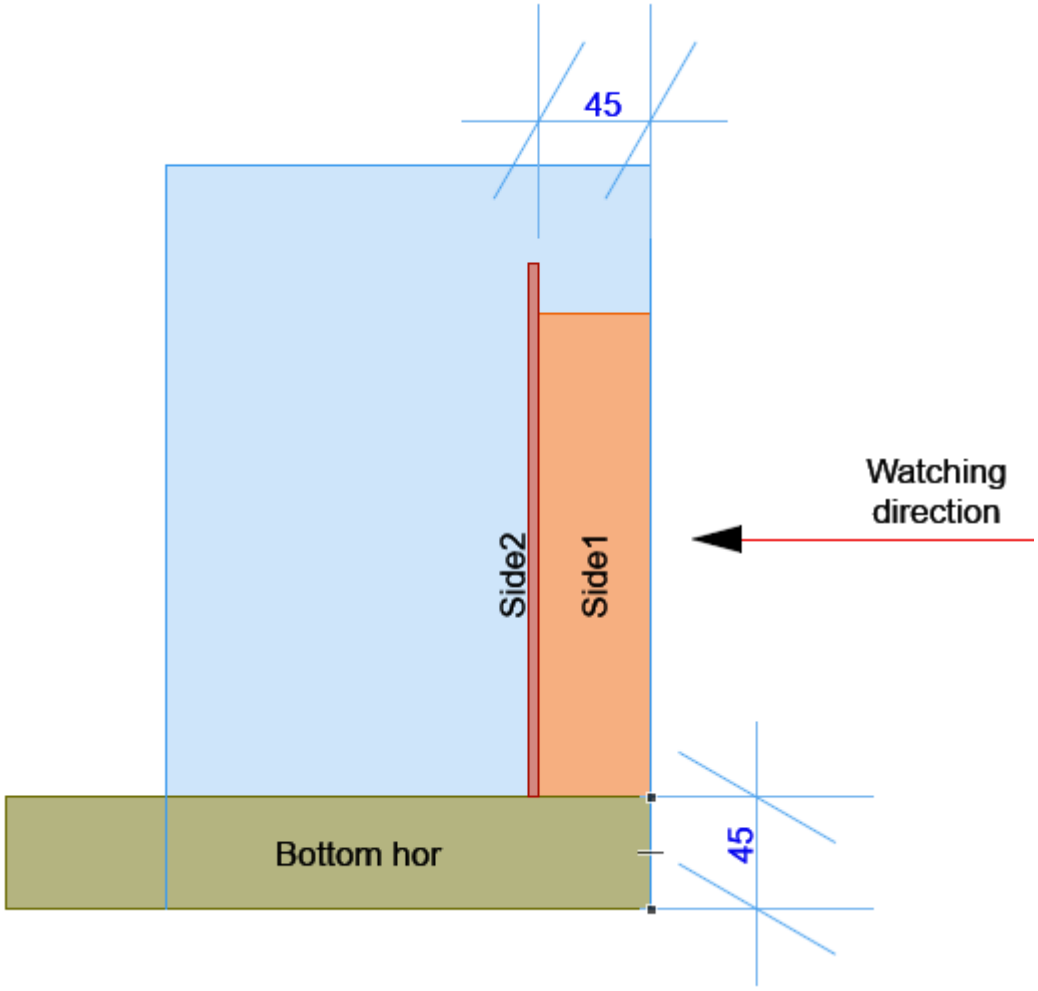
It is possible to add planks in fixed positions, for example double studs at the left and right-hand sides of the element:

```
<force>
  <material id="98x48" zoff="0" rotangle="0"></material>
  <parallel x1off="0.141"></parallel>
  <parallel x2off="-0.141"></parallel>
</force>
```

<parallel> adds planks in element's axis direction and <normal> in the perpendicular direction. Material can be set for the forced planks with <*material*>-tag.

Xlm-attribute	Description
x1off	Plank is placed to the left side of the element relative to current axis. Middle line is moved the given distance to the right. Please note that the offset defines the plank's projected center line – not the base line.
x2off	As x1off, but relative to the right side.
zoff	As in <material>-defintion.
extend	Either "0" or "1". Value "1" extends the plank to the element's contour.

An example of element like this having three planks and the light blue element object on the background vertically oriented as normal walls:




```
<layers>
  <layer name="Core" visible="1" thickness="[thickness;type=length]" type="core"
rotframing="[rotframing;type=real;default=0]">
    <!-- Bottom hor -->
    <planks group="bottom_force" axis="x" skipaxis="1">
      <material id="[matid;type=matid]" zoff="0" rotangle="0"></material>
      <force>
        <parallel x2off="-mat_thickness*0.5"></parallel>
      </force>
    </planks>

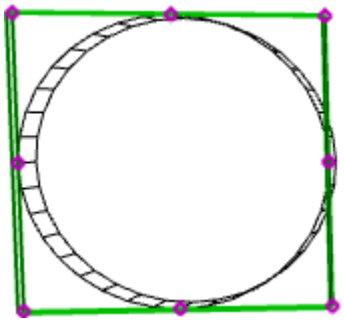
    <!-- side 1 -->
    <planks group="2ndbottom_force" axis="x" skipaxis="1">
      <material id="[matid_vert;default=@matid;type=matid]"
zoff="[zoff;default=mat_thickness*0.5]" rotangle="90"></material>
      <force>
        <parallel x2off="0-([vert1_mid_offset;default=0.045+mat_height*0.5])"></parallel>
      </force>
    </planks>

    <!-- side 2 -->
    <planks group="2ndbottom_force" axis="x" skipaxis="1">
      <material id="[matid_side2;default=@matid;type=matid]" zoff="[zoff_side2;default=0.045
+ mat_thickness*0.5]" rotangle="90"></material>
      <force>
        <parallel x2off="0-([vert2_mid_offset;default=0.045+mat_height*0.5])"></parallel>
      </force>
```

```
    </planks>
  </layer>
</layers>
```

18.2.6 Boards

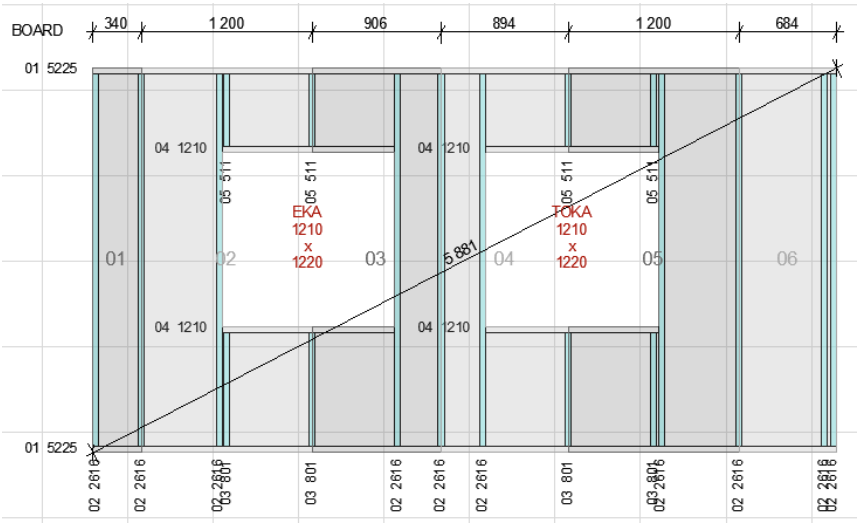
```
<layers>
  <boarding thickness="0.013" overlap="1" boardingtype="1">
    <board id="GYPSUM 13x1200x2700" width="1.2" height="2.7"></board>
  </boarding>
</layers>
```

Xlm-attribute	Description
overlap	Affects multiple layers of boards, values: • 0 or missing, no special processing. Board edges will be at the same place. • 1, avoid board edges at the same stud. If this is set and there is another board layer to process, no other boarding strategies are applied.
circletorect	Maximum size of an opening that will be converted to bounding rectangle if it has any arcs. For example, here the selected layer is converted and the one behind has the line segments following the original circle:  circle_to_rect_max_diameter in own element type settings.

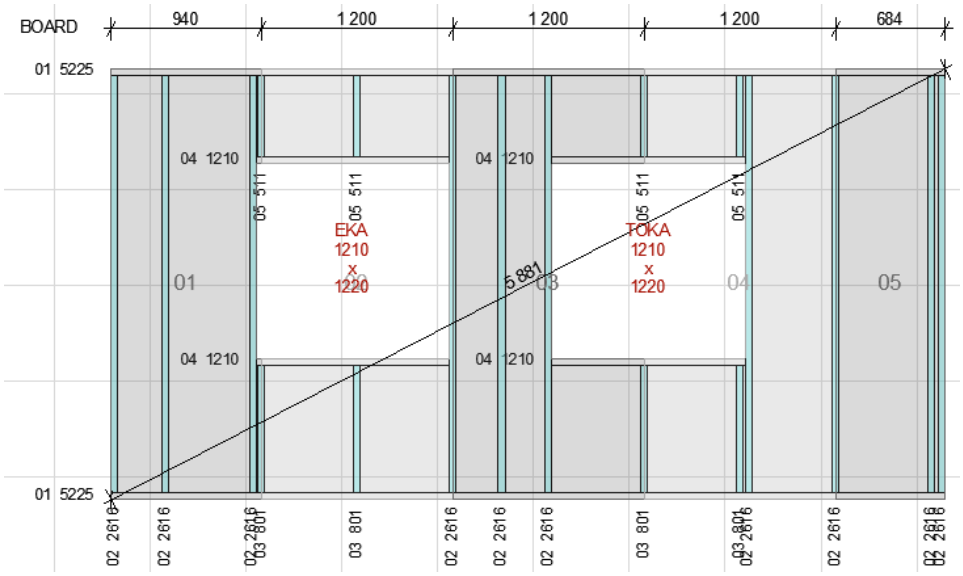
18.2.6.1 Boarding strategies/attribute boardingtype

Studding affects greatly how the boards can be placed. Therefore, the studding layer should have compatible framing rule with the boarding rule.

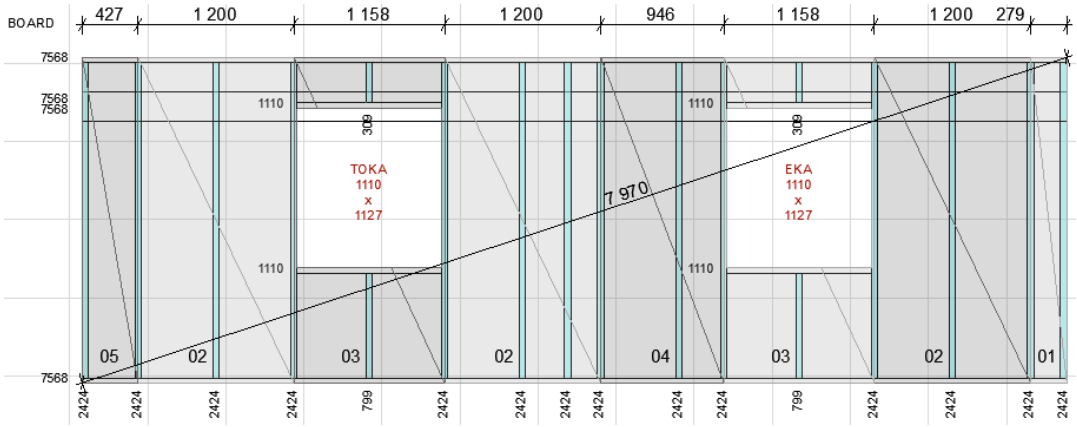
1 or missing (default), no board edge to opening edge.



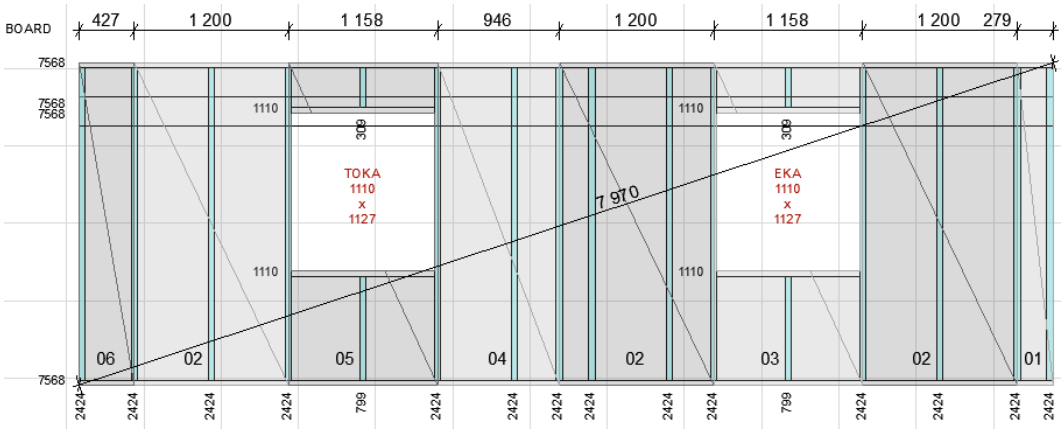
2 starts from left, no special handling for openings.



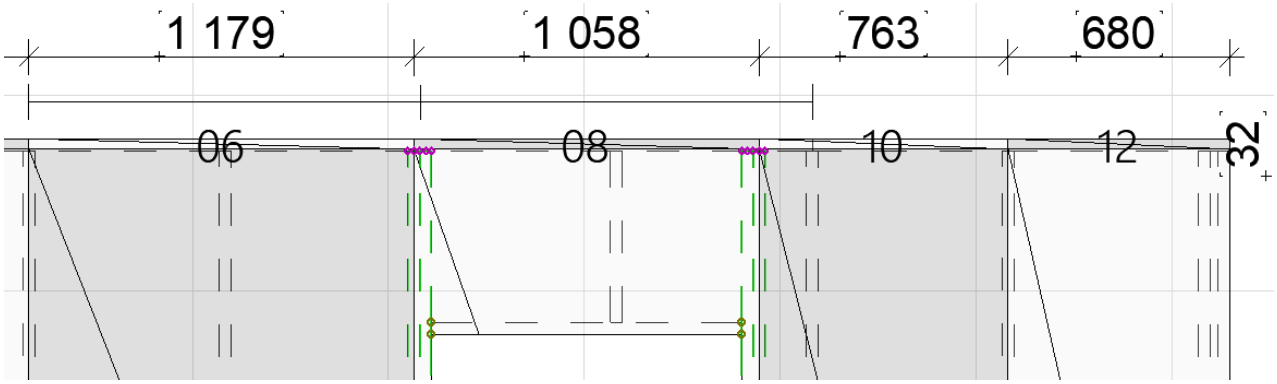
3 start from left and right-hand sides of each opening and then do the openings.



259 (3+256) same as 3 but start processing from right hand side.



Another modifier to be added to the boarding strategy number is 512 – it allows ArchiFrame to place board edge not only to center of a stud but anywhere between touching studs center lines. For example, the result of using boarding strategy 2 (start from beginning):



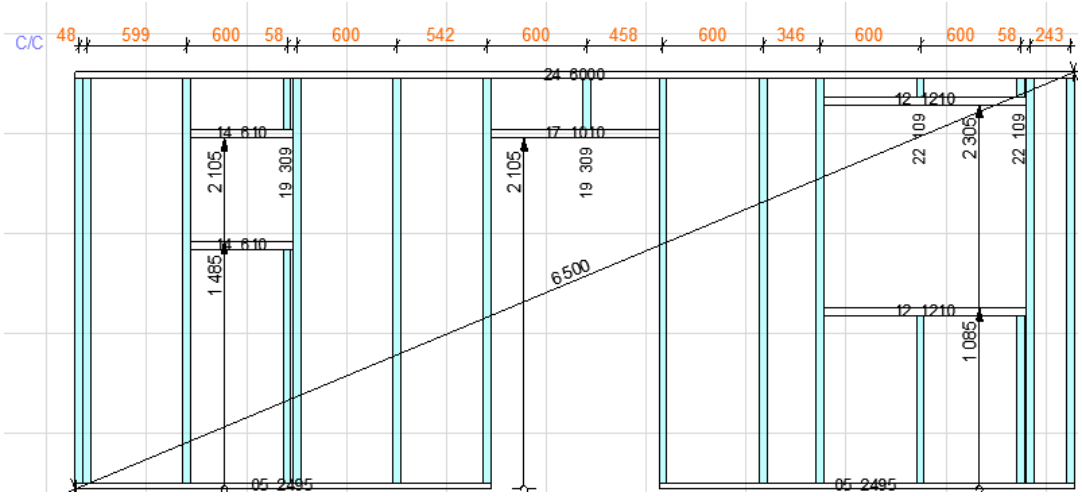
Added 512 using value 514:



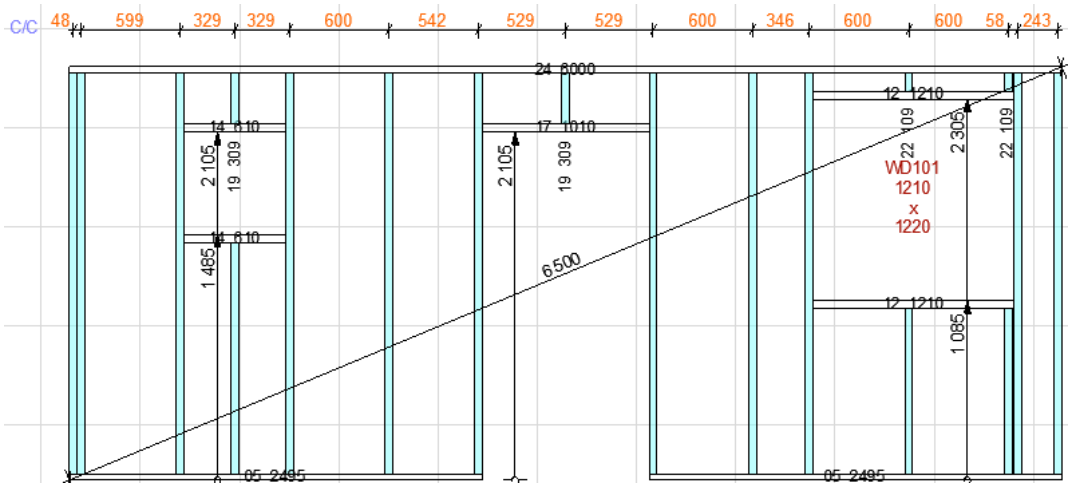
18.2.6.2 Main framing rule/attribute framingrule

Value zero or missing definition uses default rule: Try placing studs from every fixed stud (left & right, opening left & right) and select the one that uses least material.

Value 1, start applying spacing rule from left & right-hand sides of the opening. If there is more than 2 x spacing from the starting point to end of spacing area, the first spacing is spacing – stud width/2:



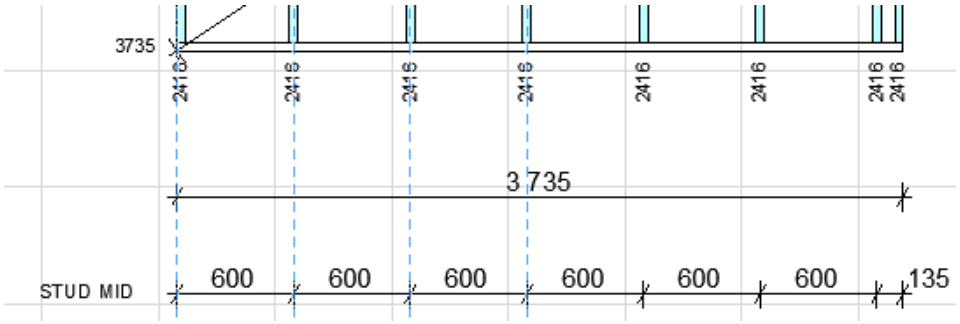
Value 2, same as 1 but place single stud to middle of opening instead of standard spacing if width less than 2 x spacing:



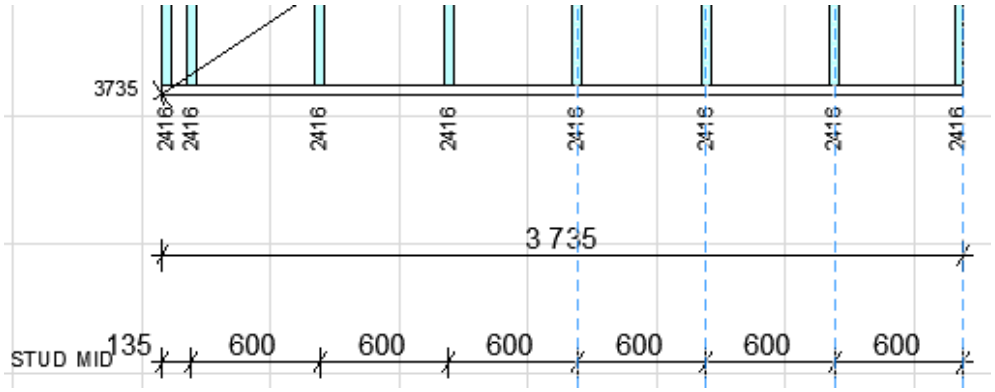
Value 3 is same as 1 but spacing is always from middle of the first stud.

Value 4 is same as 2 but spacing is always from middle of the first stud.

Value 5, start from left and from top for horizontal structures:



Value 6, start from right/bottom:



Value 7 as 5 but spacing starts from middle of the starting piece instead of its edge.

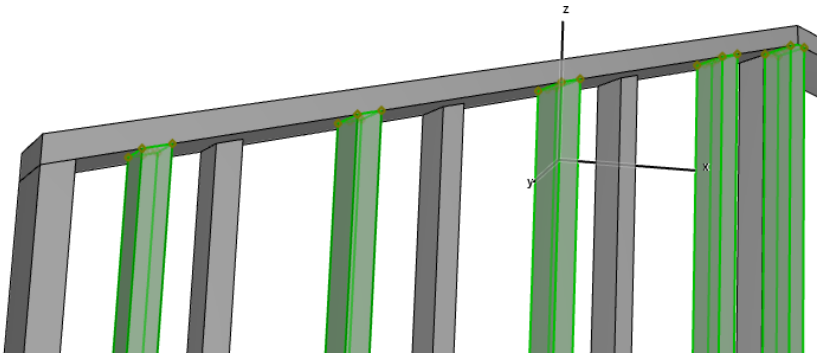
Value 8 as 6 but spacing starts from middle of the starting piece instead of its edge.

18.2.7 Operations between the planks <operations>

In this section the planks are adjusted together, and grooves are added for beams. All operations use plank groups to define operators and targets.

```
<joinends target="horizontal*|inclined*" operator="horizontal*|inclined*">
  <joinends conntype="endtoend" jointgap="0.000"></joinends>
```

```
</joinends>
```

Xlm-attribute	Description
target	Name of the groups that are operation targets. It is possible to give multiple names using separator |. The name may contain wild cards * and ?, * replaces any text and ? single character. Group names may refer to another layer in multilayer element. If that is needed, the layer type tag is given before the group name having dot as a separator. For example, to match another layer to core layer's top&bottom plates: operator="core.bottom*|core.top*|core.vertical_x*|core.contourtilted*" 
operator	Operation operator group names.
targetskip	To be used by scripts: Comma separated lists of plank guids/ptrs to exclude from the group. Also group names with wild characters may be used.
operatorskip	The same for operators.

18.2.7.1 Limiting operation with plank connection type <xxx conntype="yyy>

In many operations it is possible to limit the plank connection types. Attribute conntype may have following values:

Value (yyy)	Description
endtoend	Plank reference lines must be connected from ends.
endtoline	Target plank reference line must connect with its end to middle of the operator plank's reference line. Possible additional definitions: • mindisttoend = "meters" defines target end's minimum distance from end of operator plank ends. • disttoend = "meters", as previous but length must match exactly. Useful for weatherboards.
linetoend	Operator plank reference line must be connected from end to middle of target plank's reference line. Additional definitions as in endtoline-connection.
linex	Target plank's reference line must intersect with operator plank. Note! It is ok if target reference line just intersects the operator – the reference lines may not intersect. Possible additional definitions: • maxdisttoend = "meters" defines maximum distance from intersection to the end of target plank. • maxdisttoendop = "meters" same as previous but distance from operator piece. • mindisttoend="meters", mindisttoendop="meters" as previous but defines required minimum distance instead of maximum.

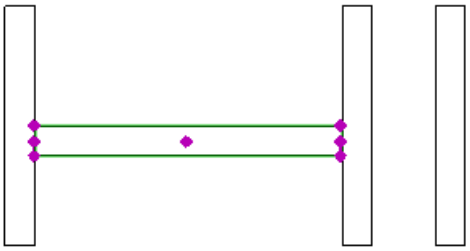
	• absangle="degrees" tells the maximum positive angle between the planks for rule to be true. Value is always between 0…90 deg. Please note that if intersection point is outside the line, the distance is negative and max distance condition is always true.
hasx	The planks have any intersection. Possible additional definitions are the same as for linex above.

18.2.7.2 Adjusting planks to another planks side <jointo>

Straight cut is made with <*cut*>-tag:

```
<jointo target="target_groups" operator="operator_groups">
  <cut jointgap="0.000" extendmaxlen="0.0" expandopfind="0.0" endshape="angled"></cut>
</jointo>
```

Xlm-attribute	Description
jointgap	Gap between the planks, negative value produces overlapping.
skipopface	Bit mask defining operator plank sides to be skipped: 1=top, 2=front, 4=bottom, 8=back, 16=begin, 32=end. Default is 0. To skip all side surfaces and search connection to begin and end only, value 15 is used.
extendmaxlen	Maximum distance between target and operator plank to make the operation.
expandopfind	How much the operator plank is expanded from each side before checking the intersection. Pieces touching just from the corner will not be found unless for example 1 mm (0.001) is given here. 
endshape	Possible additional definition for the cut: • "angled", extend and cut to the operator (default). • "anglednoext", just cut without extending. • "angleddel", as angled but remove all existing cuts. • "angleddelfirst", as previous but delete all existing only for first joint. For next join operations old cuts are preserved (to keep two cuts at hip). • "straightshort", straight cut to first intersection. • "straightlong", straight cut to last intersection.
toend	Cut surface: • 0, trim to first intersecting surface (default). • 1, trim to further surface.
removepart	Remove part: • "up", remove upper part • "down", remove lower part • "between", adjust between first and last operator (if other end is missing, that end will be unmodified) 

	• "keeplonger" to always select the longer part of the piece and stop processing when end is cut away: 
dontcuttarget	Set to 1 if just adding markings – do not cut the piece.
domark	Set to 1 to add marking lines to connection.
markside	Forced marking side, 0=automatic
marklines	0=to sides, 1=to center, -1=no lines
markauto	Set the *Managed by ArchiFrame*-setting for the marking machinings to this value.
marktext	Marking text if any, tags as in the tool palette
markposx	Text position 0 = left, 1 = center, 2 = right
markposy	Text position 0 = bottom, 1 = center, 2 = top
markfontsize	In centimeters, 0 = default
markdir	0=default direction, 1=text in line's direction
conntype	See Limiting operation with plank connection type <xxx conntype="yyy>.

Mortise and tenon joint is done with <*mortisetenon*>-tag:

```
<jointo target="targets" operator="operators/female parts">
  <mortisetenon len="0.056" bordertop="0.02" borderbot="0.005" borderside="0.01"
roundingr="0.0225" endgap="0.001" sidegap="0.001" topgap="0.010"></mortisetenon>
</jointo>
```

Dovetail with <*dovetail*>-tag:

```
<jointo target="targets" operator="operators/female parts">
  <dovetail len="0.038" bordertop="0.00" borderbot="0.050" bottomwidth="0.06"
roundingr="0.0225" endgap="0.000" sidegap="0.000" angle="6" ></dovetail>
</jointo>
```

18.2.7.3 Adjusting ends <joinends>

This operation connects two planks together halving the angle. Both targets and operators are handled the same way.

```
<joinends target="horizontal*|inclined*" operator="horizontal*|inclined*">
  <joinends conntype="endtoend" jointgap="0.000"></joinends>
</joinends>
```

Xlm-attribute	Description
jointagap	Gap between plank ends.
conntype	See Limiting operation with plank connection type <xxx conntype="yyy>.

18.2.7.4 Grooves and balk joint <groove>

This operation is used for example to make grooves to the studs for a beam inside the element.

```
<groove target="vertical_y|vertical_spacing|vertical_force" operator="balktop*">
  <groove overtop="0.000" overbot="0.000" overside="0.000" overlen="0.000"></groove>
</groove>
```

Xlm-attribute	Description
overtop	Oversize at operator plank's top side.
overbot	…bottom side.
overside	…sides, may be given individually with attributes overleft and overright.
overlen	…lengthwise.
forcedepth	To force depth as in groove tool.
force90	To force groove as perpendicular.
forcetargetside	Value 1...6 of forced target surface.
autoupdate	Value 0 makes groove that is not automatically updated and value 1 or missing definition makes automatically updating one.
conntype	See Limiting operation with plank connection type <xxx conntype="yyy>.

Narrowed balk (adds machinings to operator plank too):

```
<groove target="vertical_y|vertical_spacing|vertical_force" operator="balktop*">
  <balkborder bordertop="0.02" borderbot="0.005" borderside="0.01" overtop="0.001"
overbot="0.000" overside="0.001" overlen="0.000"></balkborder>
</groove>
```

Xlm-attribute	Description
bordertop	Narrowing depth at operator-plank's top side.
borderbot	…bottom side.
borderside	…sides.
overlen	How much longer the narrowing part is than the related target plank. Other oversizes as in <groove>-operation.

18.2.8 Projections <projections>

Please see XML-file for details. Attribute layoutmargins defines marginal in order: Left, right, top, bottom.

```
<projections>
  <group layout="horizontal">
    <group layout="vertical">
      <!-- Only planks belonging to the element are included in cutlist -->
      <!-- layoutalign: -1=left/bottom, 0=center, 1=right/top -->
      <cutlist layoutalign="-1" layoutmargins="0.1,0.1,0.1,0.1" layoutminwidth="2.0"
layoutminheight="7.0" settingsref="cutlist"></cutlist>
    </group>
    <group layout="vertical">
      <projection type="front" layoutmargins="2,2,2,2">
        <dimlines>
          <dimline ref="wall_elevation"></dimline>
        </dimlines>
        <elemmarkings>
          <opening ref="mark_opening"></opening>
        </elemmarkings>
      </projection>
      <projection type="top" layoutmargins="2,2,0,2">
        <dimlines>
          <dimline ref="wall_top"></dimline>
        </dimlines>
      </projection>
      <projection type="back" layoutmargins="2,2,0,2">
        <dimlines>
          <dimline ref="wall_elevation"></dimline>
        </dimlines>
      </projection>
    </group>
```

```
  </group>
</projections>
</elemtype>
```

Xlm-attribute	Description
layoutalign	Obsolete.
valign	Align vertically in the parent: • -1, top (default). • 0, center. • 1, bottom.
halign	Align horizontally in the parent: • -1, left (default). • 0, center. • 1, right.
type	For projection tag, possible values: • front • back • top • bottom • topflipx (will mirror X-coordinates of the projection) • bottomflipx (will mirror X-coordinates of the projection) • left • right • frontrot180 (rotated 180 degrees) • backrot180 (rotated 180 degrees) For projections also rotation can be specified with rotate-attribute, for example: `<projection type="front" rotate="90" …`

<*projection*>-tag may contain settings for specific layer's projection plank objects with syntax:

```
<projection type="front" layoutmargins="2,0.5,0,1" exclude="*ext*"
boardpanels="boarding_int*">
  <dimlines>
    <dimline ref="wall_elevation"></dimline>
  </dimlines>

  <projplanksettings type="core" objtype="1">
    <elemparam name="linetype">*dash*</elemparam>
    <objparam name="iFill">0</objparam>
    <objparam name="iShowID">0</objparam>
    <objparam name="iShowLen">0</objparam>
  </projplanksettings>
</projection>
```

This is useful for example, if the core layer should be shown as trace only. Settings may contain anything described in section XML-settings for an Archicad element. Attributes:

- type is the layer type to handle (please give just single type name, to set many types please add multiple projplanksettings-tags): core, intstud, extstud* etc.
- objtype can be added to apply the settings only for planks (objtype="1"), boards (objtype="4") or both (objtype="5").

To make more complicated settings, it is possible to add a Lua script for processing the projection planks. For example, the script at XML-path archiframe/elem/settings/scripts. The function name DoProjSettings() is fixed:

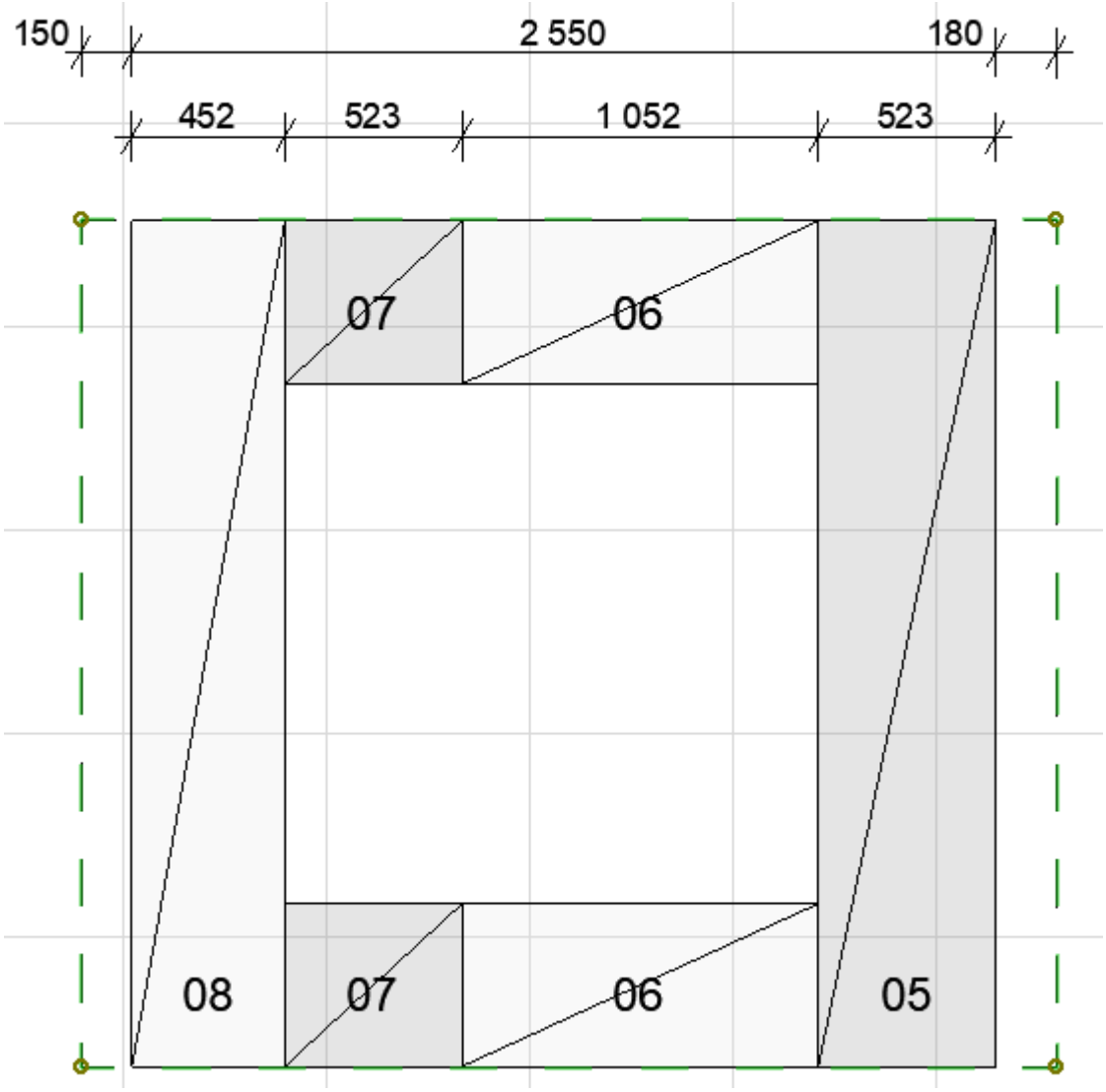
```
        <!-- Sets ID to middle for horizontal planks  -->
        <script id="yit_projplanks" merge="addpre">
          <![CDATA[
function DoProjSettings()
  local z

  for i=1,gnPlanks do
    ac_objectopen(string.format("#%d",i))
    -- Vertical (studs) with different fill
    z=ac_objectget("iEndZ")
    if z and math.abs(z)<0.001 then   -- nil if board
      -- Horizontal piece, ID to middle
      ac_objectset("iIDPlace", 0)
      ac_objectset("iXoffID", 0)
    end
    ac_objectclose()
  end
end
]]>
        </script>
```

And the reference to it:

```
<projection type="front" layoutmargins="2,0.5,1,1" exclude="*boarding*,*stud*,*finish*">
  <projplankscript ref="yit_projplanks">
  </projplankscript>
```

It is possible to place the core layer just by contour lines as reference and include it in offset dim lines:



The xml-syntax for that is (under <projection>-tag):

```
<contourlines create="1" holes="0" hatch="0">
  <elemparam name="pen">1</elemparam>
  <elemparam name="linetype">*dash*|*katkoviiva*</elemparam>
  <layer>xxx</layer>
</contourlines>
```

This will create the dashed lines as in example above on layer xxx. Please note the reference with exclamation mark to include core layer to dimension line even if it is not visible in dimension line xml: includeelem="*boarding*,**!**core"

By default, holes are not drawn – just the contour line. Setting attribute holes="1" will include holes.

Setting attribute hatch="1" will create a hatch instead of lines.

```
<contourlines create="1" holes="1" hatch="1">
  <elemparam name="pen">1</elemparam>
  <elemparam name="linetype">*dash*|*katkoviiva*</elemparam>
  <elemparam name="cutfill">*triangles*</elemparam>
  <elemparam name="cutfillpen">13</elemparam>
  <elemparam name="cutfillbgpen">0</elemparam>
</contourlines>
```

18.2.9 Projections/static texts

It is possible to include title/information texts inside <group>-definitions. For example, to add ID outside the print frame and text LOOKED FROM OUTSIDE to position X = 3, Y = -0.05 from current's group's top & left coordinates.

```
<group layout="vertical">
  <text content="#id#" anchor="7" x="0" y="0.1" update="1">
    <elemparam name="pen">1</elemparam>
    <elemparam name="fontname">arial</elemparam>
    <elemparam name="fontsize">30</elemparam>
    <elemparam name="fontstyle"></elemparam>
  </text>
  <text content="LOOKED FROM OUTSIDE" anchor="1" x="3" y="-0.05">
    <elemparam name="pen">1</elemparam>
    <elemparam name="fontname">arial</elemparam>
    <elemparam name="fontsize">1.5</elemparam>
    <elemparam name="fontstyle"></elemparam>
  </text>
```

Xml-attribute	Description
content	Text to add, following texts are replaced: • #id# by the element id, for example EW01. • #weightkg# by element weight in kilograms. • #weightlb# by element weight in pounds. • #elemtype# by composite element's type ID. • #floornum# by the home floor number. • #floorname# by the home floor name.
anchor	Text anchor point: 1 2 3 4 5 6 7 8 9
x,y	Offset to place text from current position.
update	By default, these static texts are created once – they are not moved, nor is content updated afterwards. By setting this to value 1, the position and content will be updated when element is updated.
script	Refers to xml-path /archiframe/elem/settings/script/script id="xxx". This scrip has environment of a projection's dimension line script like dim_camera is available. ArchiFrame calls fixed name function GetText() The script generates the content of the text returned like below

Attributes to anchor the text to a projection:

Xml-attribute	Description
projanchor_id	Value refers to an <projection id="xx" … >-tag
projanchor_anchor	Anchor point in the projection for text's position like anchor above.
projanchor_xoff, projanchor_yoff	Offset from the anchor point.

A partial example to generate content from Lua-script:

```
-- To be set by luasettings-tag
```

```
gnSide=-1     -- -1=left, 1=right
gsLayer=nil   -- nil=just corner type, str=elem layer to find offsets from

function GetText(sSettingsLua)
  local res

  if sSettingsLua and sSettingsLua~="" then
    -- String contains lua script that fills global settings variables
    local f

    f=loadstring(sSettingsLua)
    if not f then
      ac_environment("tolog", string.format("GetText: Bad settings Lua-script: %s",
sSettingsLua))
    else
      f()
    end
  end

  -- Return text in table for future extensions
  res={}
  res.text=GetSideInfo(gnSide, gsLayer)
  return res
end
```

An example scipt based static text xml-defintion. There is a special <luasettings>-section that allows passing Lua-code to the GetText()-function.

```
<projection id="page1_front" type="front" …>
  …
</projection>

<text content="(UNUSED)" anchor="6" update="1" script="CornerAndOffInfo"
projanchor_id="page1_front" projanchor_anchor="4" projanchor_xoff="-0.1" projanchor_yoff="0">
  <elemparam name="fontname">arial</elemparam>
  <elemparam name="pen">1</elemparam>
  <elemparam name="fontsize">2.5</elemparam>
  <elemparam name="fontstyle"></elemparam>
  <luasettings>
    <![CDATA[
gnSide=-1
gsLayer=nil
]]>
  </luasettings>
</text>
```

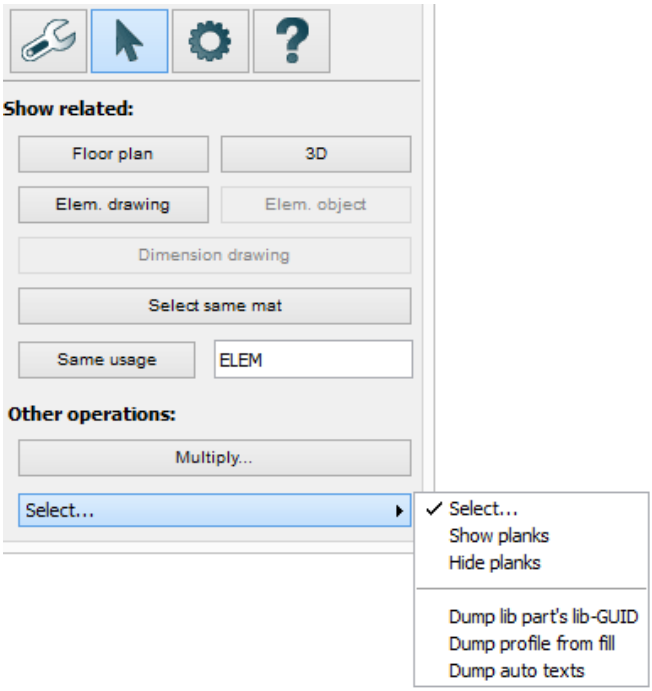
18.2.10 Projections/element stamp <projections>

It is possible to use any kind of stamp object in the element drawings. ArchiFrame library contains at least single stamp that can be slightly customized. As default the stamp looks like:

ID	Type	Project	Project num
A	50x100	AF Sample	123456
	Date	Designer	
	2013-12-17	#CAD Technician	

The field names and values all come from ArchiFrameElements.xml. The default definition is:

```
<elemstamp layoutalign="-1" layoutmargins="-2.2,0,0,0" layoutminwidth="10.0"
layoutminheight="1.0" libname="{9FD03D30-6A12-4B0F-AB3A-01410A529E38}-{D420792D-C066-42CF-
988C-9596DD164EB7}" settingsref="elemstamp">
  <script ref="elemstamp"></script>
</elemstamp>
```

Xlm-attribute	Description
libname	Library part's name, please use GUID that you can get from ArchiFrame main tool palette/Selecion/Other operations: 
settingsref	Refers to common settings xml-tag in elem/settings/presettings.
script ref	Refers to xml-tag elem/settings/scripts. With element scripts multiple references may be given using attributes ref = "first" ref1 = "second" ref3 = "third" etc. To add text from current tag before referenced part(s), attribute currpre="1" is used. It is useful to set global variables before they are used in referenced script.

It is recommended to set the stamp field names in settings xml and the stamp parameters in Lua-script. For example the default settings for field names:

```
<!-- Used for element stamp -->
<elemstamp>
  <elemparam name="pen">1</elemparam>
  <objparam name="iFont">Arial</objparam>
  <objparam name="A">10</objparam>
  <objparam name="B">1</objparam>

  <objparam name="iName1">ID</objparam>
  <objparam name="iName2">Type</objparam>
  <objparam name="iName3">Date</objparam>
  <objparam name="iName4">Project</objparam>
  <objparam name="iName5">Designer</objparam>
  <objparam name="iName6">Project num</objparam>
  <objparam name="iName7"></objparam>
</elemstamp>
```

The default script sets field *Date* and *Designer* only if those are empty. To see available auto texts, open ArchiFrame main tool palette/Selecion/Other operations (picture above). To change the automatic values, select the stamps for example using Archicad Find & Select and open object settings.

Please see default ArchiFrameElements.xml for all the details.

18.2.11 Markings into planks <markings>

```
<markings id="mark_big">
```

```
  <bottomwood setcnc-="0">
    <stud line="3" id="2" idline="1" markxpos="2" markypos="1" marksize="5" marktextdir="1"></stud>
    <opening line="1" windowtext="Vindu [width]x[height]" doortext="Dør [width]x[height]"
marksize="5"></opening>
    <neighbour begtext="&lt;[id]" endtext="[id]&gt;" marksize="10" marktextdir="1"></neighbour>
    <joint char="*" marksize="10" marktextdir="1"></joint>
  </bottomwood>
  <topwood setcnc-="0">
    <stud line="3" id="2" idline="1" markxpos="2" markypos="1" marksize="3" marktextdir="1"
forceside="0"></stud>
  </topwood>
</markings>
```

Specifies markings to the planks in the element. Markings are set whenever the element is updated. Attribute setcnc- controls whether plank's cnc-parameter is set to 1 if there are markings added. Default value is 1. Currently it is possible to mark only the top and bottom wood (connecting elements, studs and openings):



Tag	Description
stud	Stud markings, attributes: • line, 1 = mark left, 2 = mark right, 3 = mark both. • id, 1 = mark whole id, 2 = strip elem id away, just the plank number. • skipid, if given, tells the iElemGroup-values not to add IDs • idline, 1 = left hand side, 2 = right hand side, 3 = let ArchiFrame calculate position not under any stud, 4 = to middle of the stud • skipline, if plank's iElemGroup-value matches this list, no markings are added to the plank at all. • markxpos, justify 1 = left, 2 = middle, 3 = right. • markypos, justify 1 = bottom, 2 = middle, 3 = top. • marksize, text size in centimetres, 0 = default. • marktextdir, 0 = in plank dir, 1 = in line dir. • forceside, 0=automatic target side, 1-6=force marking to given side
opening	Mark openings, attributes: • line, 0 = no line, 1 = line covering the opening. • windowtext, text for window, [width] and [height] are replaced with AC locale aware dimensions. • doortext, the same for doors.
neighbour	Mark connecting elements (only end connections): • begtext, text to connection at plank begin, [id] will be replaced with connecting element id. • endtext, the same for plank end. • mark*, the same as for studs.
joint	Mark joints, attributes: • text, the character to use to mark connecting pieces

18.2.12 Element options, <options>

All element options are defined in <options>-section of the *ArchiFrameElements.xml*–file. Single option definition is like:

```
<option id="openingsides" name="Planks to openings sides" applystate="prespacing"
ordernum="100">
  <script ref="openingsides">
```

```
    <![CDATA[
]]>
  </script>
</option>
```

XIm-attribute	Description
id	Unique id of the option. Not visible to the user.
name	Visible name.
applystate	Specify to alter the phase when the option is applied. Default phase is after creating all element planks. Possible overrides are: • prespacing, after all element edge pieces have been creates but spacing rule has not been applied yet. Used if the option moves studs. • postcreatelayers, after all element layers have been created. Used when option needs information from other layers.
autoreset	Specified if other option(s) need to be reset when this one is set. For example autoreset = “openingsides”.
ordernum	Options may depend on each other. This sets the applying order for the options. Smaller number (default = 100) will be applied first.
reapplyupdate	If set reapplyupdate = “1” and option is checked on, the option is first cleared and then set again when the user clicks Update from the element tools. This is useful for example to add markings to the pieces.

<*reapply*>-tags may be given if some options should be reapplied when current one changes. For example, if double top is changed, grooves at top must be reapplied to the lowest top piece and balk top level may change:

```
<option id="doubletop" name="Double top" name_fin="Tuplayläjuoksu" name_nor="Dobbel
Toppsvill" name_swe="Dubbla hammarband">
  <!-- If there is already grotop for the element, remove existing and set after running
this option -->
  <reapply id="grotop"></reapply>
  <reapply id="balktopint"></reapply>
  <reapply id="balktopext"></reapply>
```

The option itself is defined with *Lua*-script, please see Lua-scripts with elements.

19 Examples of using elements and trusses

These examples use model *ArchiFrameElemDemo.pln* which is installed in folder C:\ArchiFrame\samples by default. Please make sure that ArchiFrame-library is loaded (default folder is C:\ArchiFrame\Lib).

There are more recent videos on this topic:

ArchiFrame elements: an overview, https://vimeo.com/178327076.
https://player.vimeo.com/video/178327076

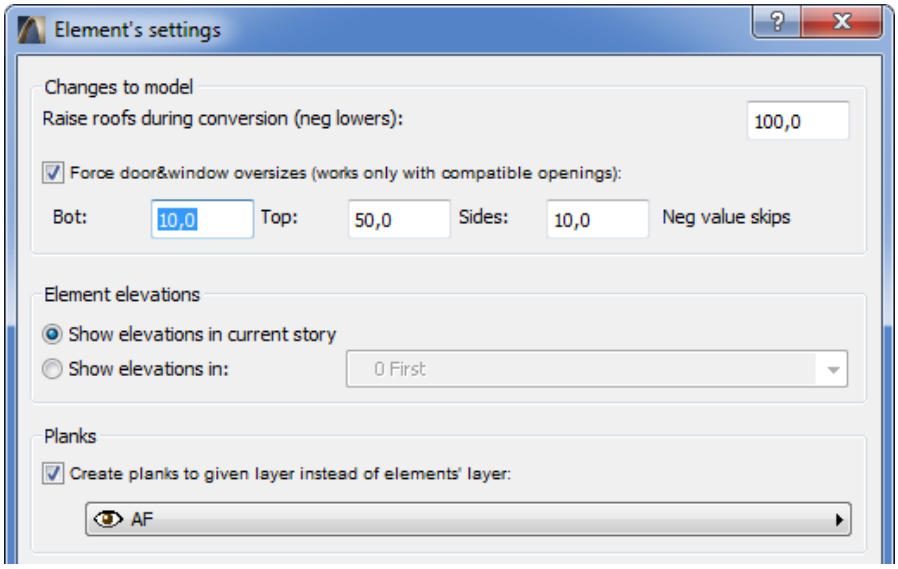
Placing rectangular wall elements, https://vimeo.com/180035580.
https://player.vimeo.com/video/180035580

Placing complex wall elements, https://vimeo.com/181255910.
https://player.vimeo.com/video/181255910

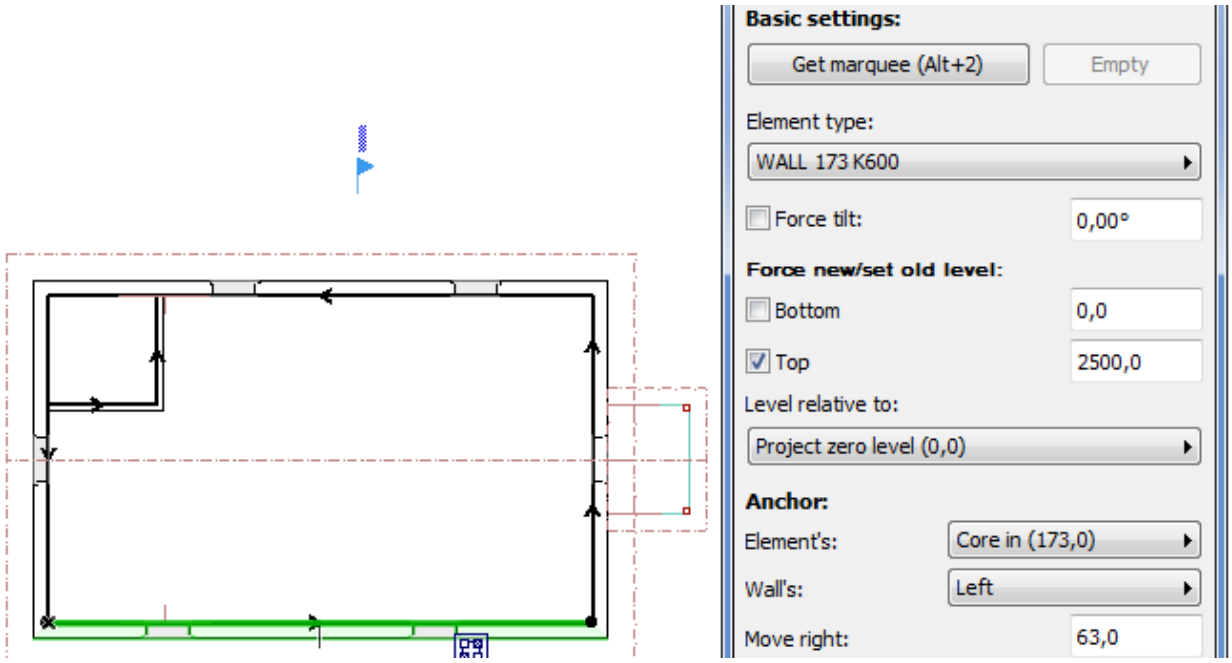
Adding a wall element with steel framing, https://vimeo.com/173754604.
https://player.vimeo.com/video/173754604

19.1 Creating wall elements/frames

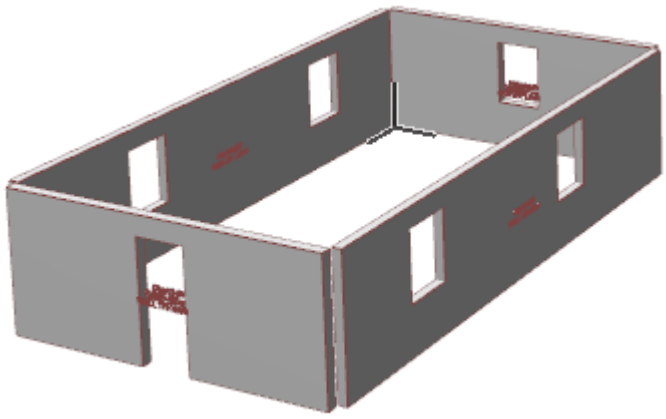
First create elements for the side walls. Specify the settings like below:



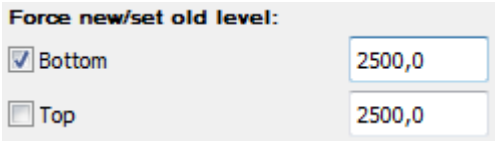
Let's make the side wall elements 2500 mm high (the trusses are placed over those) and adjust core inner side 63 mm outwards from architect's wall (13 mm gypsum board + 50 mm additional studding):



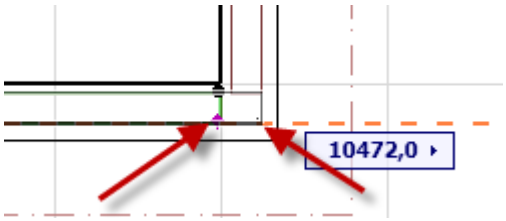
Create new element from selection creates element for the side wall. Do the same for the other side wall. Then make the gable wall elements with *Place new element with line* –operation. Before adding let's make sure that Force tilt has the value 90 degrees, element height is 2500 mm and move to right is 63 mm. Draw the line starting from side wall element's inner corners so that the exterior of the element will be on the right hand side (from bottom to top at right and from top to bottom at left hand side). After that the elements look like:



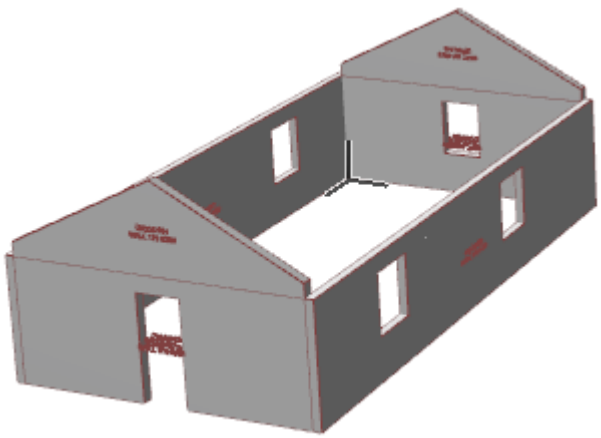
Add angled gable wall elements as previous ones but force bottom level to 2500 and remove forcing top level:



Stretch the side walls to fill the corner gaps:

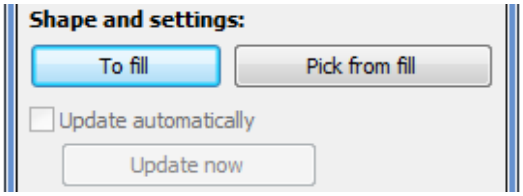


The elements now look like this:



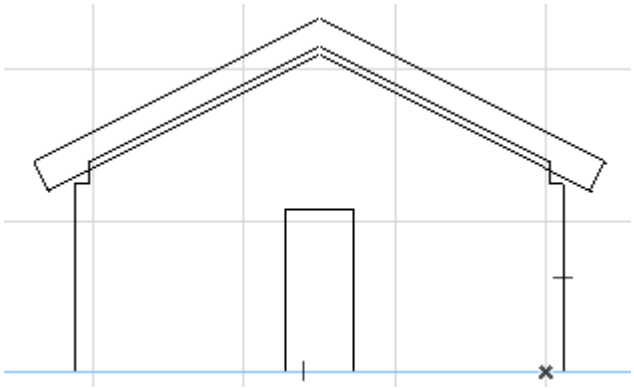
Now we could edit the top gable elements to extend over the side walls with *To fill* and *Pick from fill* –operations. The fill is edited with standard Archicad *fill*-tools and the changed shape is picked back to element with *Pick from fill* button. Fill is deleted after editing.





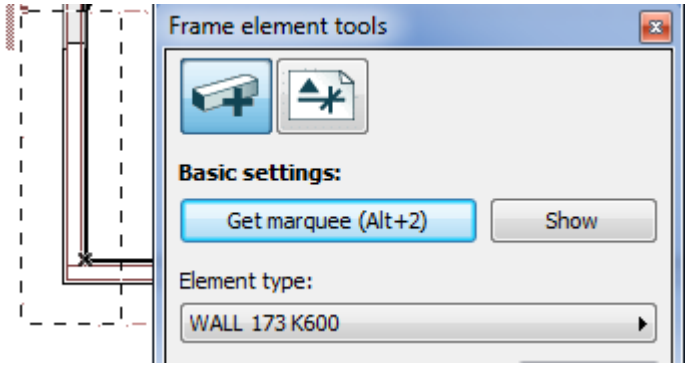
Finally the elements will be given IDs with *Give IDs*-operation.

This section shows that the gable element extends 100 mm inside the roof slope:

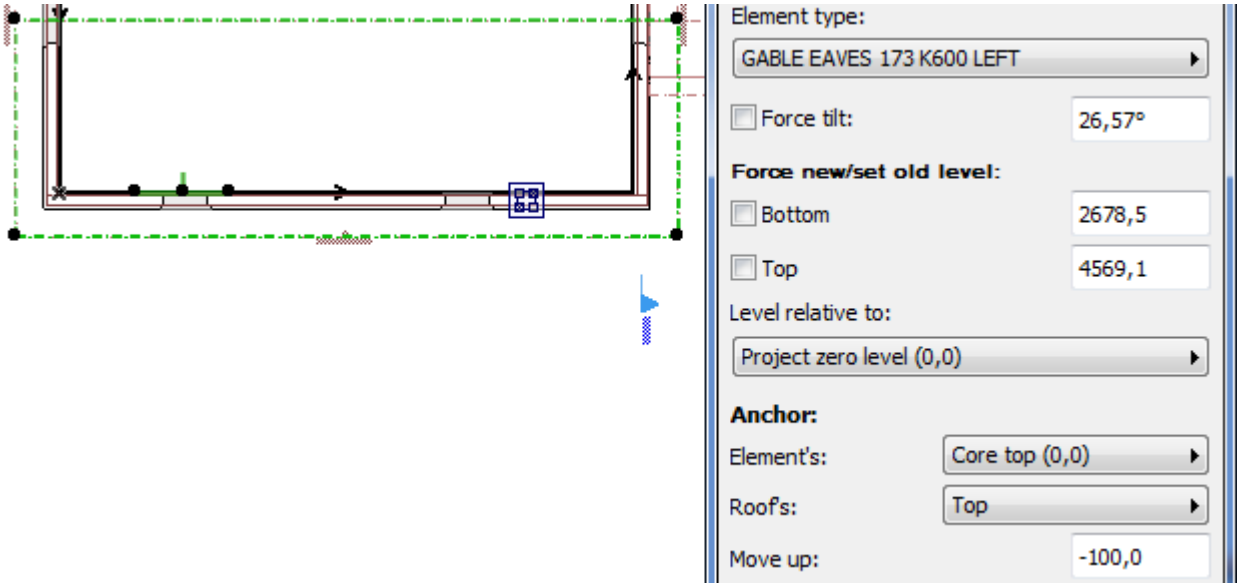


19.2 Creating eaves element from roof

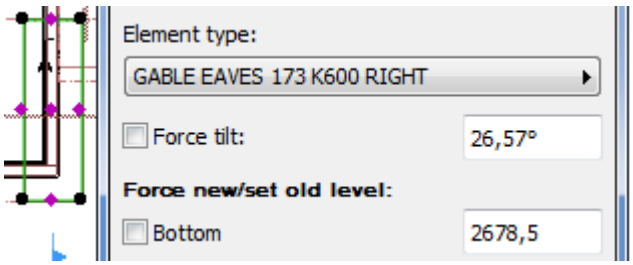
Draw 1200 mm wide marquee to the left hand side of the roof and pick it using the element tool (the marquee can be picked from any polygonal Archicad-element):



Clear the marquee and select the roof and set the values as below to get the element top 100 mm below Archicad roof. Click *Create new element from selection* to create the element:



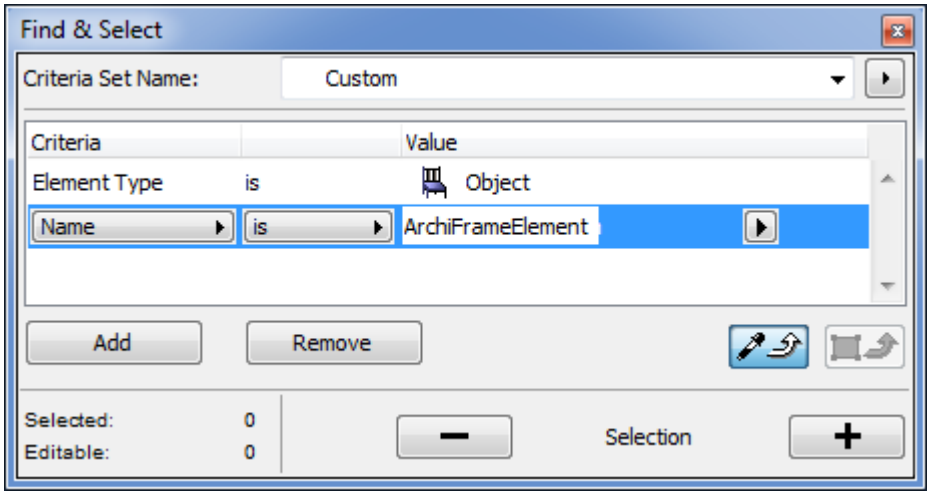
Move and copy the created element to the right side with built in Archicad command (Ctrl+Shift+D) and change element type to right:



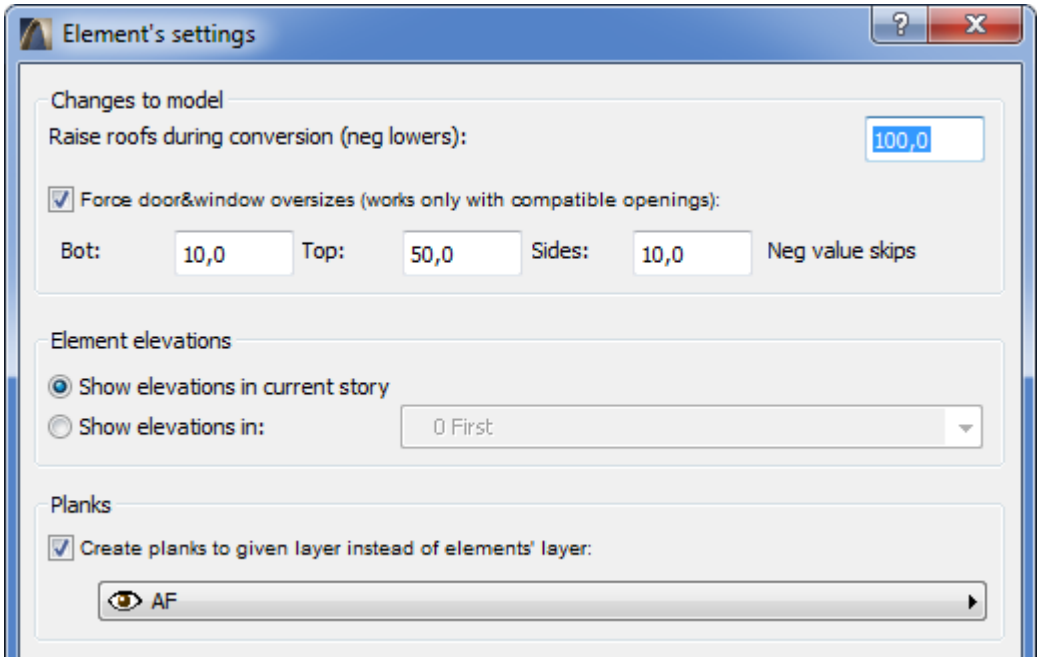
Finally *Give IDs* to new elements. Note that.

19.3 Creating and editing the planks

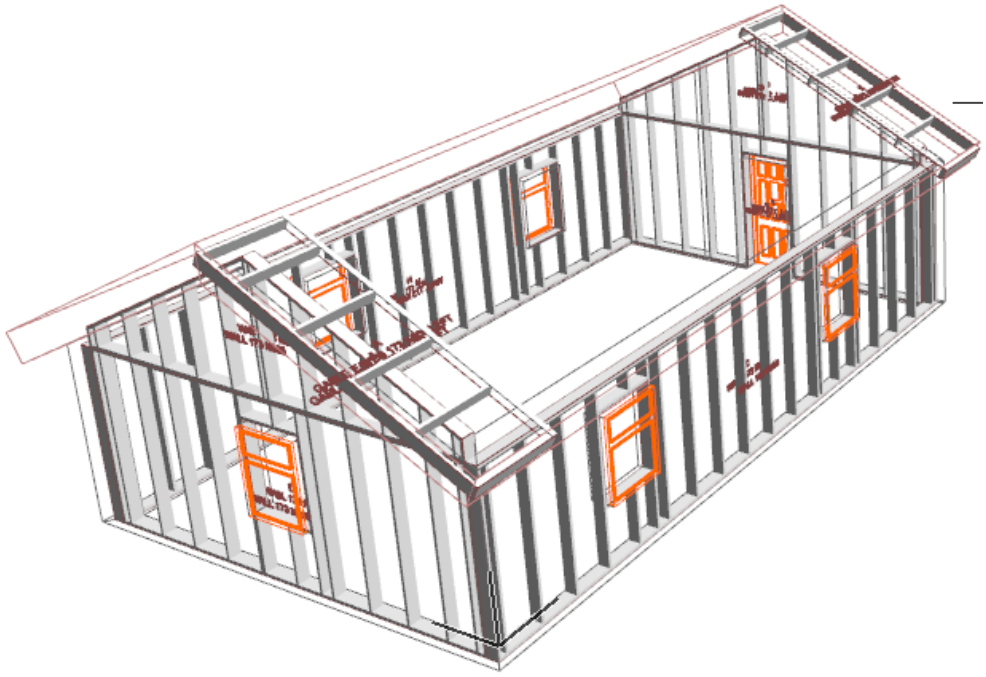
This is a simple phase: Select the element objects to work with. Archicad Find & Select is useful for that:



Then click *Create planks*. Please note that the additional settings affect the planks and the elevations:

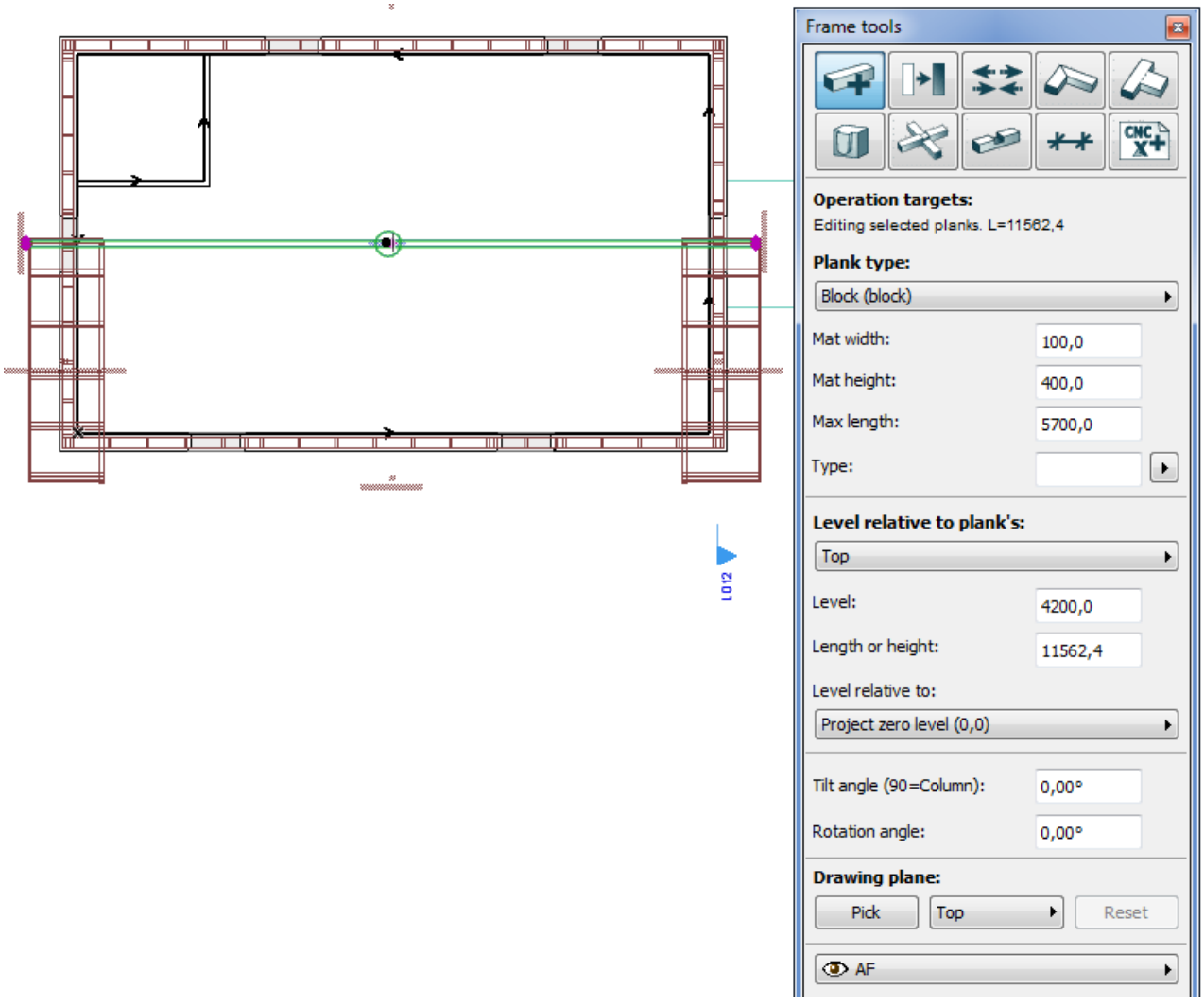


Planks can be edited with standard *Archicad*-tools and ArchiFrame plank tools. Next image is the model after creating planks. The elements are on wireframe layer:

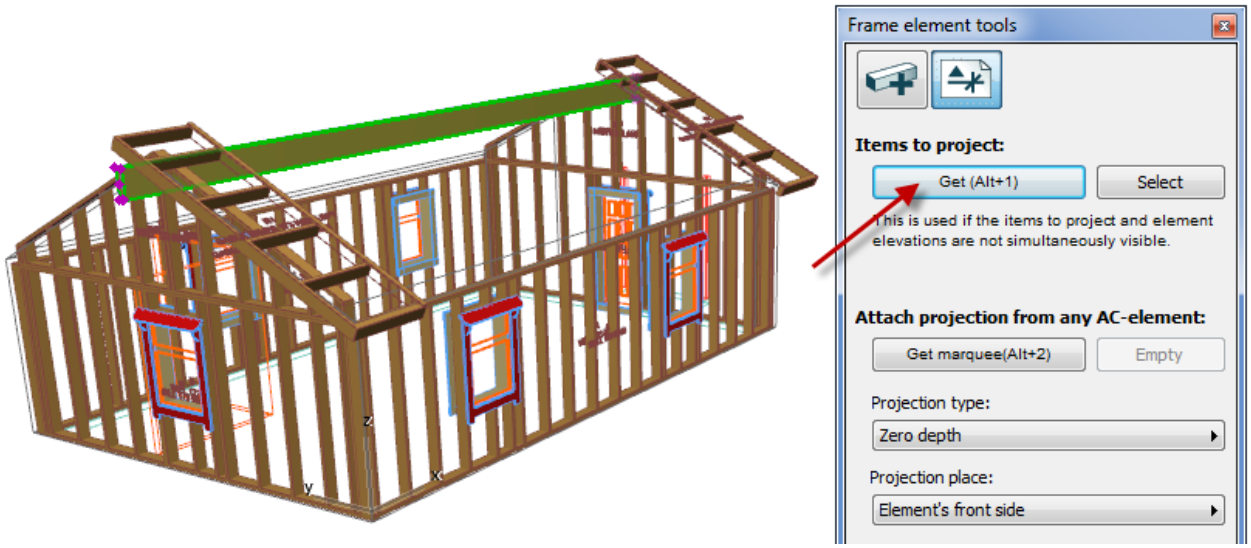


19.4 Adding ridge beam

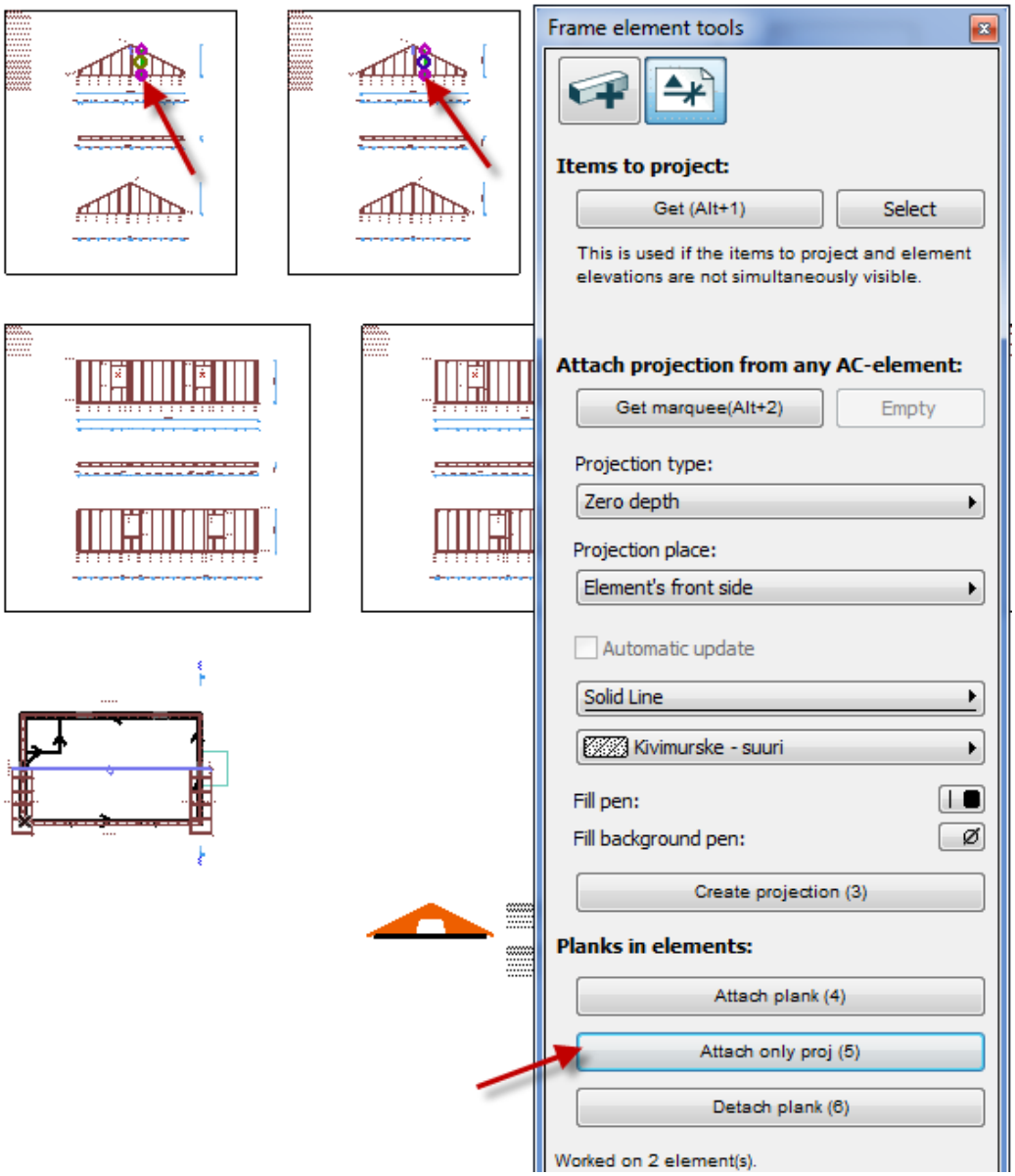
Ridge beam is not part of any element but it is useful to see it in the gable elements. Let's add the ridge beam with the following settings:



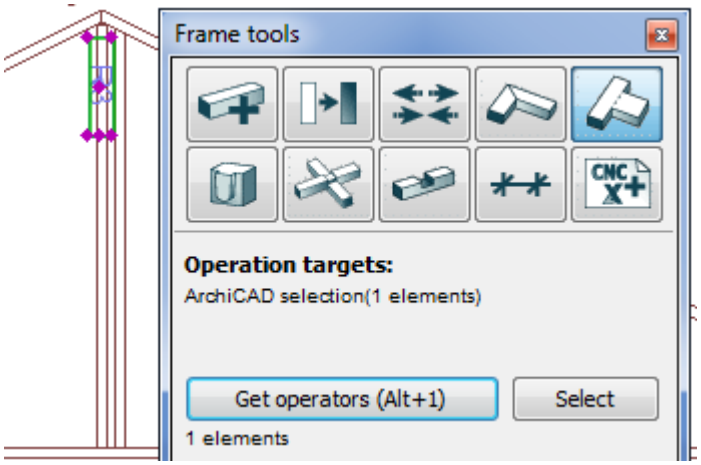
Then add the projections to the gable element. First pick the beam as an item to project from a 3D window:



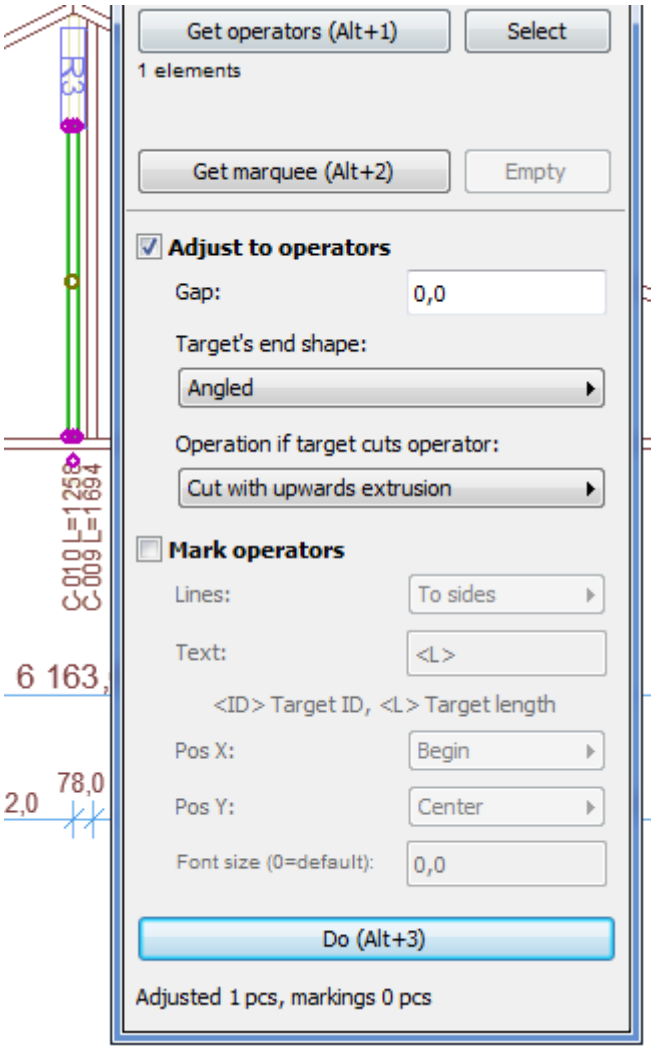
Then select the projections where the ridge beam should be visible and click *Create projection*:



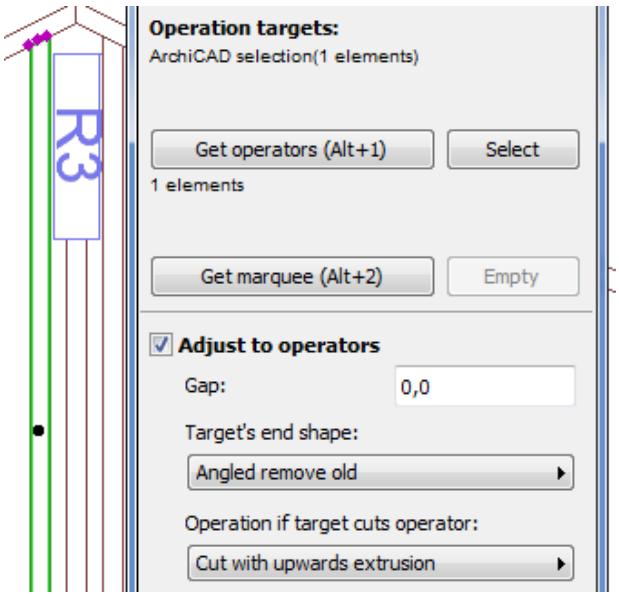
Add load bearing support for the ridge beam with plank tools. Let's select the ridge beam as operator:



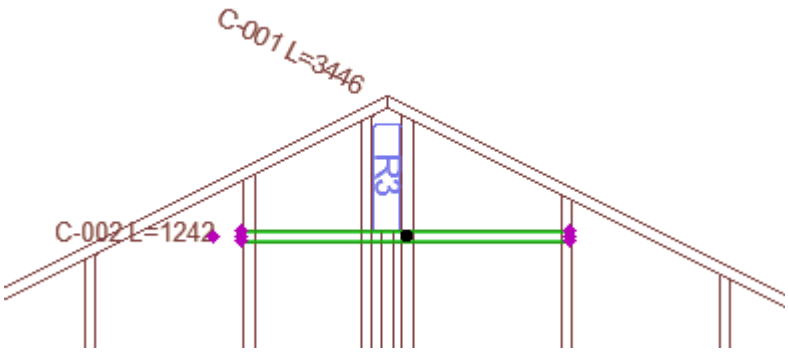
Adjust top of the stud to the beam's bottom side and remove remaining part of the stud above the beam:



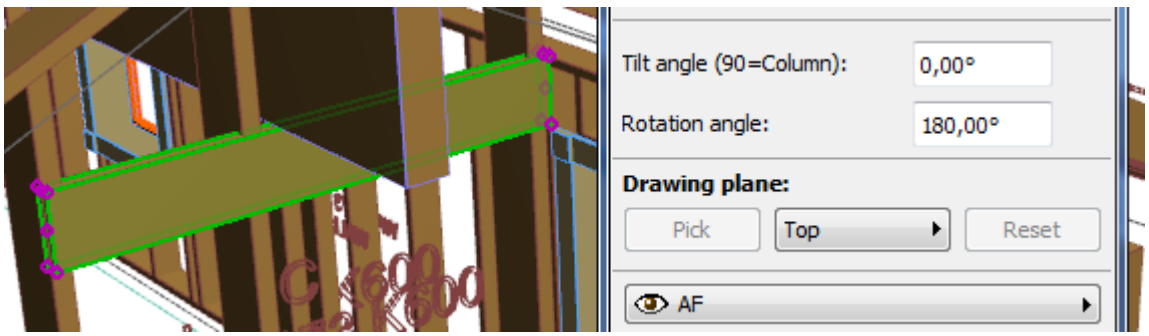
Duplicate the right hand side stud with the Archicad drag & copy operation to the left and adjust it to element's top plank (select it first as operator). This time remove old cuts:



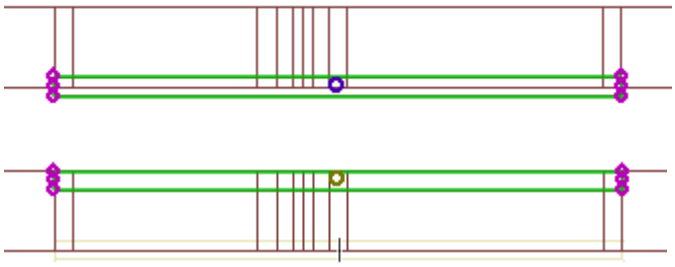
Then add horizontal support by copying the lowest plank of the element and stretching it to fit:



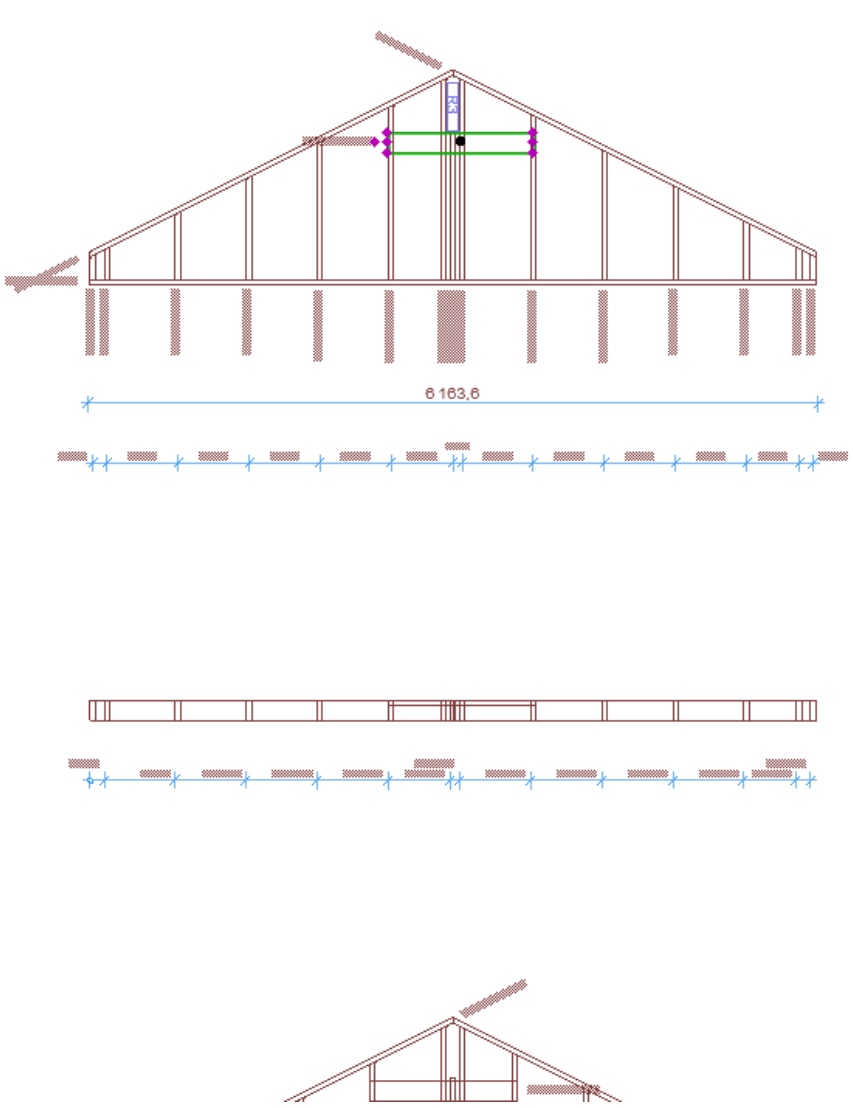
Adjust the rotation angle in a 3D window:

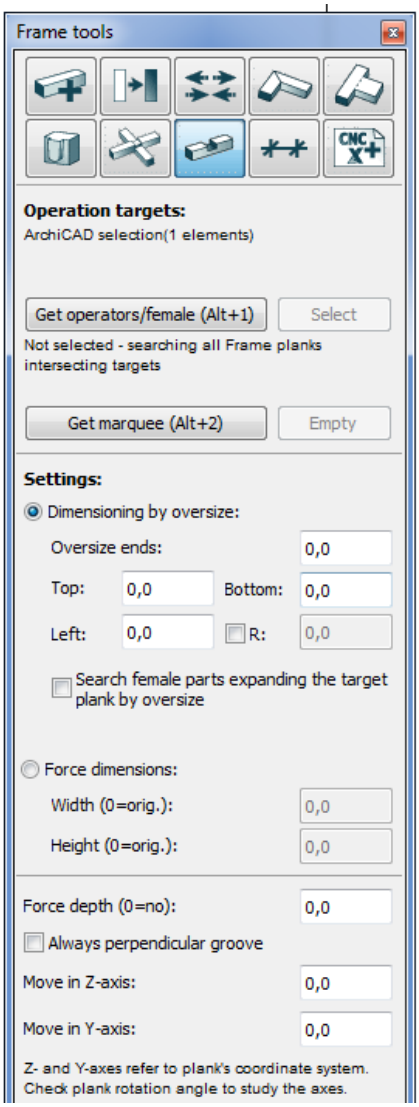


Move the plank to the inner side, either in 3D or using the top projection:

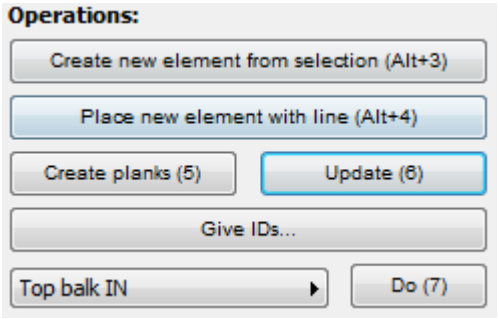


Change grooves to studs:



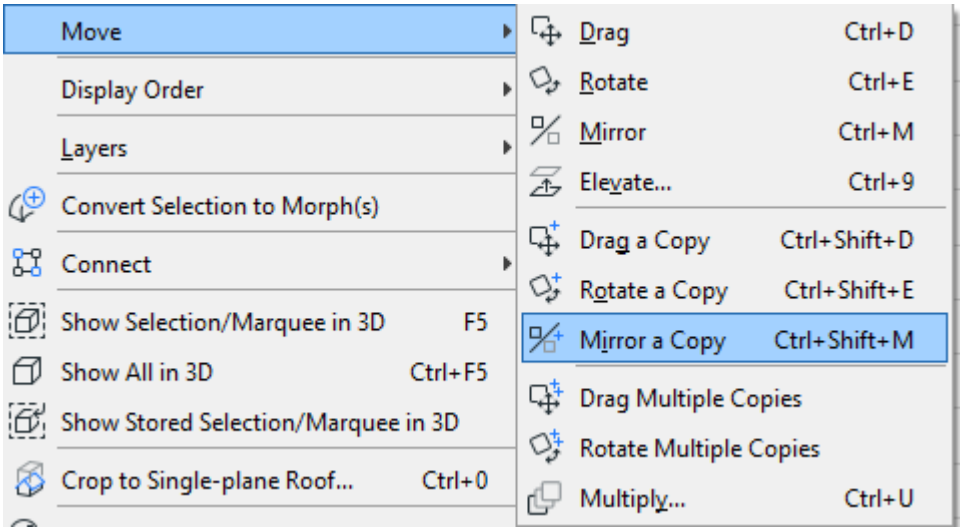


After the changes, update the cut list and dimension lines with element tools *Update* –operation:



19.5 Copying finished element to another place

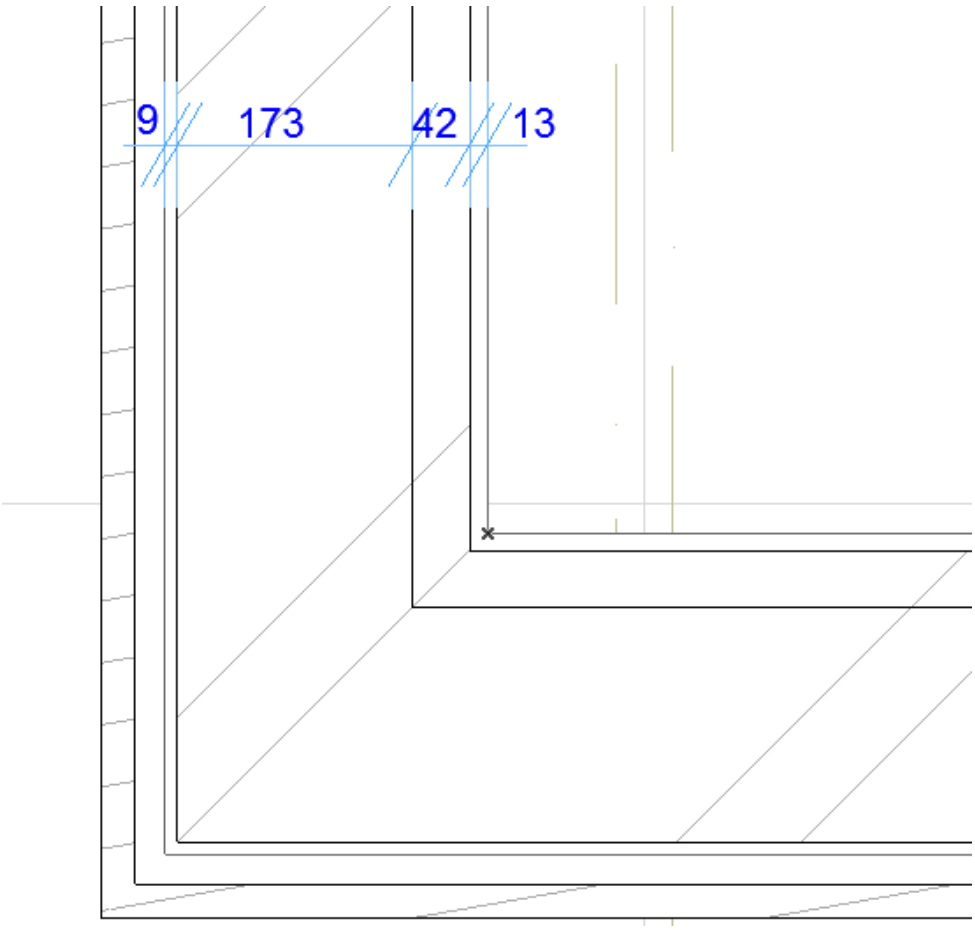
For example, the gable walls may be similar. In this case, only the other end is finished, and it is copied & mirrored to another end. To do that, the related ArchiFrameElement-objects are mirrored & copied with standard Archicad-command:



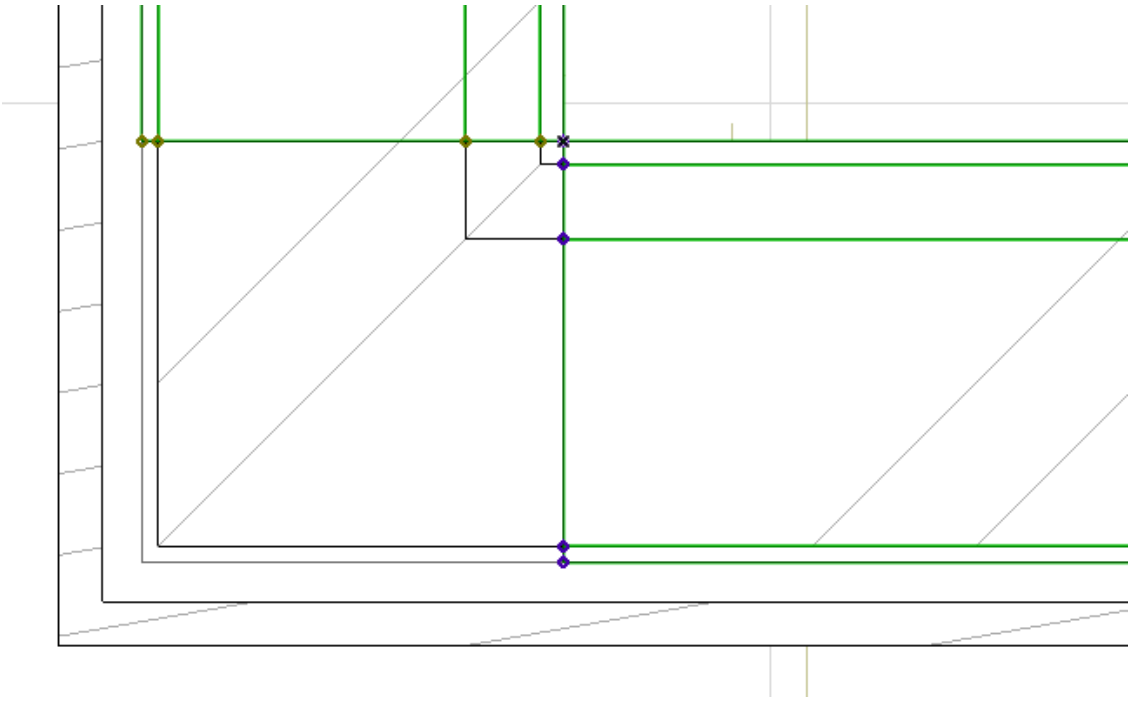
ArchiFrame will mirror & copy the related planks & boards with the ArchiFrameElement-objects. Copied ArchiFrameElements are not linked to Archicad-walls (or slabs/roofs) in the new place. To do that, the ArchiFrameElement-objects must be connected to Archicad-elements using projection tools. Connection is needed to be able to update element openings and for correct opening weight calculation.

19.6 Multilayer elements, corners and boarding

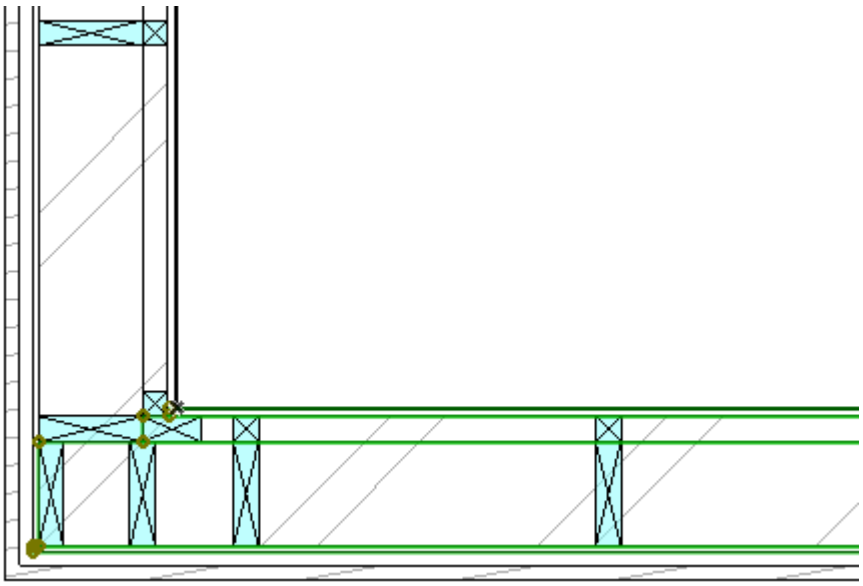
Example structure from default *data*-folder element type *WALL 173+42 VERT*:



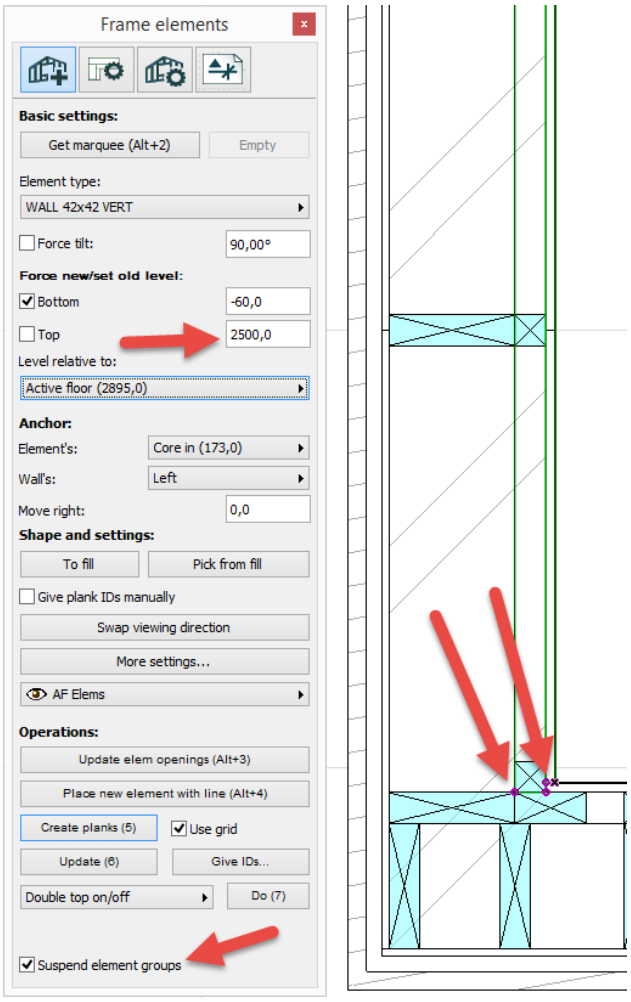
Exterior paneling and related framing are skipped. The elements are placed so that corners touch to allow ArchiFrame to connect the corners together. Corners may overlap if placed for example using the core in as anchor – the corner tool works that way also. IDs are given normally – ArchiFrame detects multilayer element and assigns just a single ID to the group.



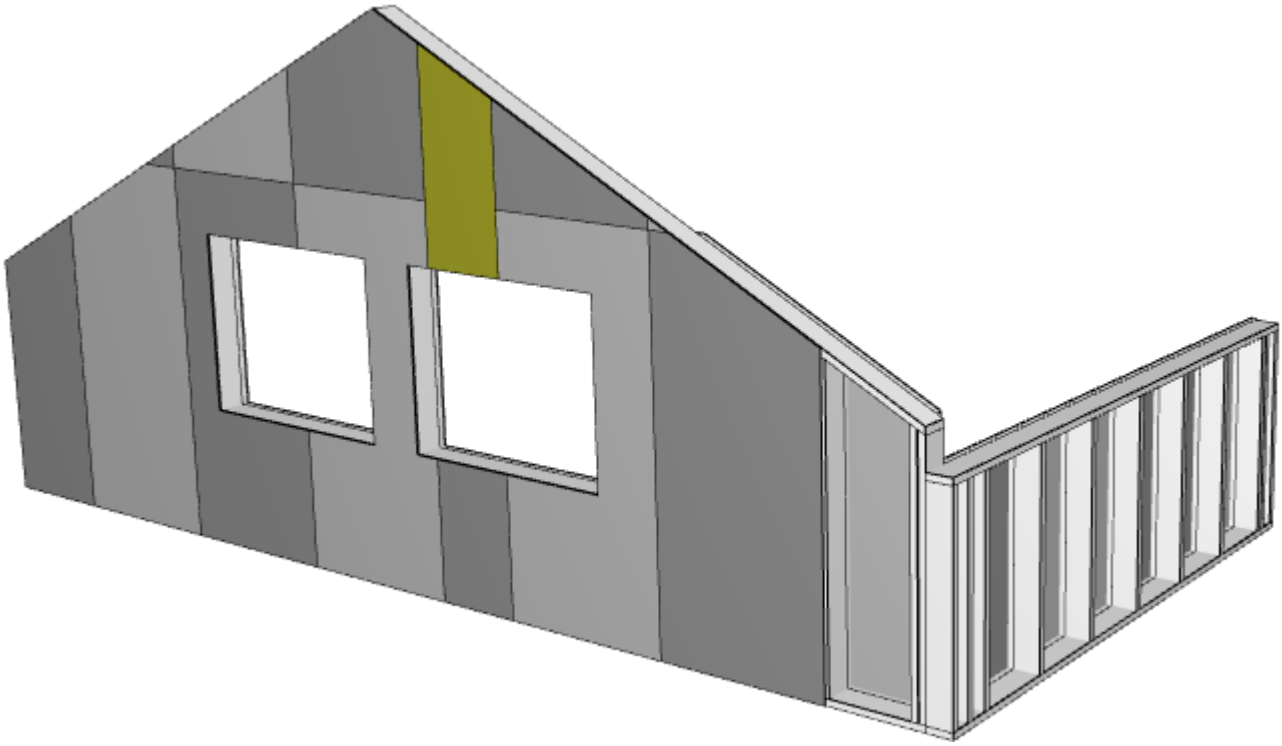
After setting corner type to *Long*, the other wall will be automatically set to *Short*. The result after creating planks:



Then the height of at least the inner boarding and extra studding needs to be limited for gable walls. It is done by selecting the corresponding element objects, suspending element groups to let the main studding and the exterior boarding to fill the whole gable and finally editing the level value. The same could be done with using *To fill*–button, editing the fills and clicking *Pick from fill*. Simple editing is also possible using element object's hotspots in 3D or section.



After creating planks (and deleting some boards to see the framing) the result is:



- ArchiFrame has assigned colours to boards so that they are visible.
- Projections have only exterior boards when looked from outside, interior from the other side.

Miscellaneous notes:

- To find definitions in *ArchiFrameElements.xml* please look for text MULTILAYER.
- To recreate boarding after changes in studding: select just a board belonging to an element and *Create planks* will be changed to *Update boards.*
- Attribute studs = “xxx” means this layer depends on layer xxx. For boards it is where seams can be made and for planks it means that place studs where there is a stud in referenced layer.
- Update element updates colours and RGB-values of adjacent boards to make sure the boards can be seen.

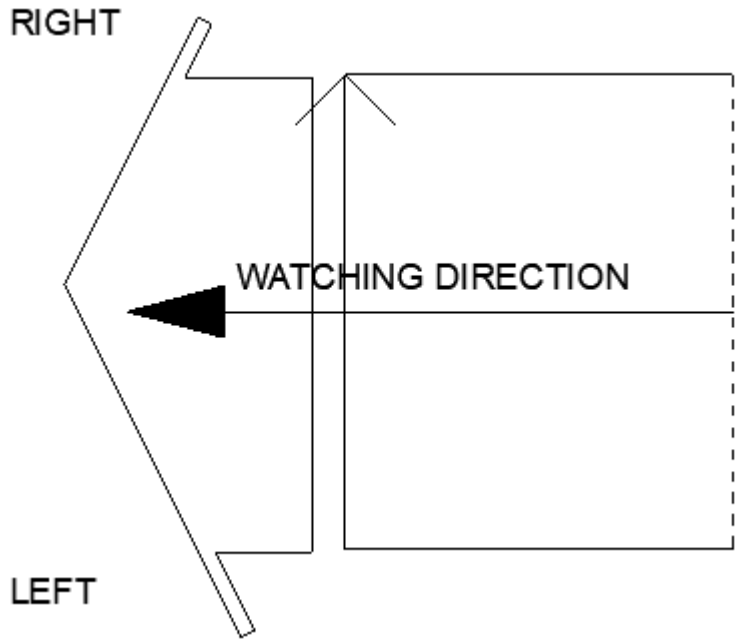
20 Trusses

Trusses are modeled with object ArchiFrameTruss. There are two workflows:

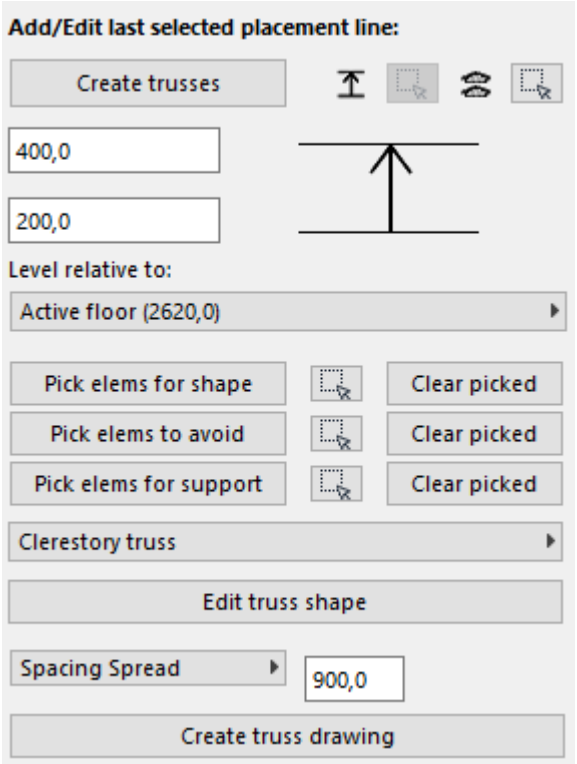
20.1 Truss workflow picking geometry from the model and/or adjusting the truss shape numerically (method A)

- Place ArchiFramePlacementLines-objects to define the begin and end of the lower chord and area where to place the trusses.
- Pick Archicad roofs to define the truss shape basics and other Archicad elements to define support lines and elements to avoid.
- Adjust truss parts numerically and place the trusses
- If necessary, edit individual trusses editing a fill created from it and picking the shape back to the truss.
- Assign IDs to trusses.
- Create truss drawings for manufacturing.

The orientation of the truss related to the placement line is like below (watching direction is to the left from the ArchiFramePlacementLines-object's direction arrow):

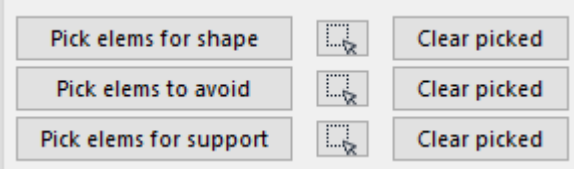


Parts of the palette related to this workflow:



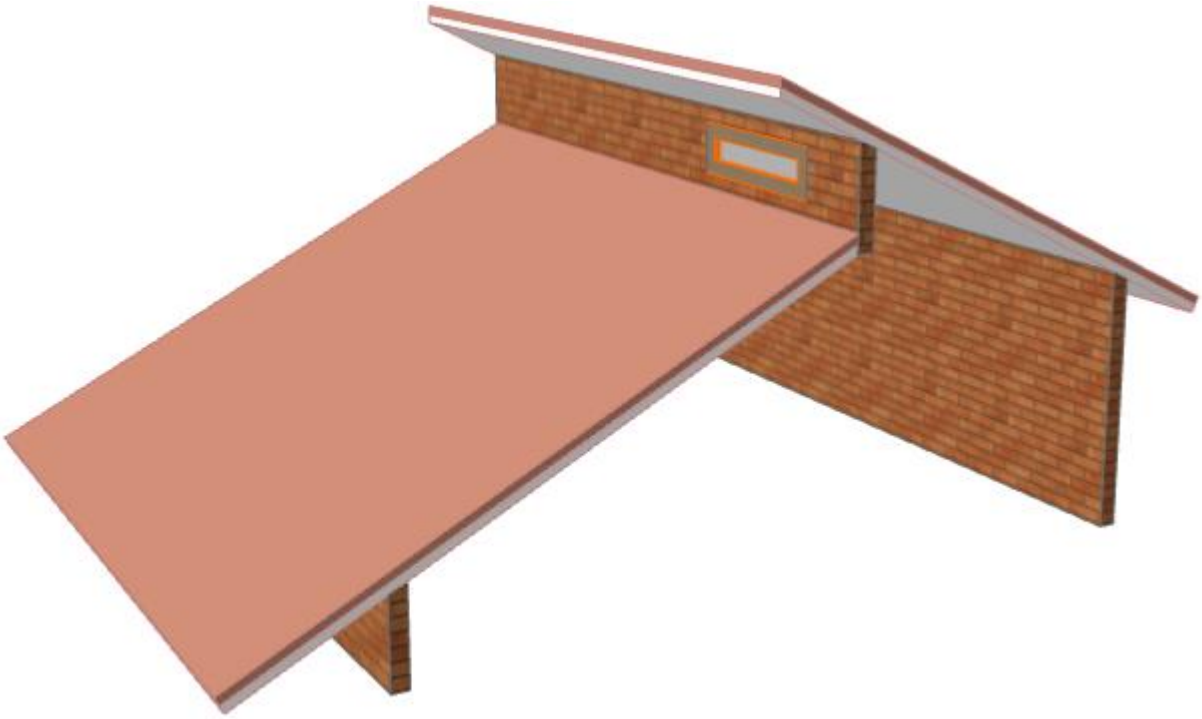
- *Place lines/Create trusses/Update trusses*, without selection this button is used to add new ArchiFramePlacementLines. Having that kind of part selected the button changes to Create trusses that recreates all trusses for selection.
-  selects last edited ArchiFramePlacementLines-object.
-  selects trusses related to currently selected ArchiFramePlacementLines-object.

- Level edits are used to set the levels of ArchiFramePlacementLines- or ArchiFrameTruss-object.
- *Level relative to*, level anchor point.
- The pick- and related buttons:

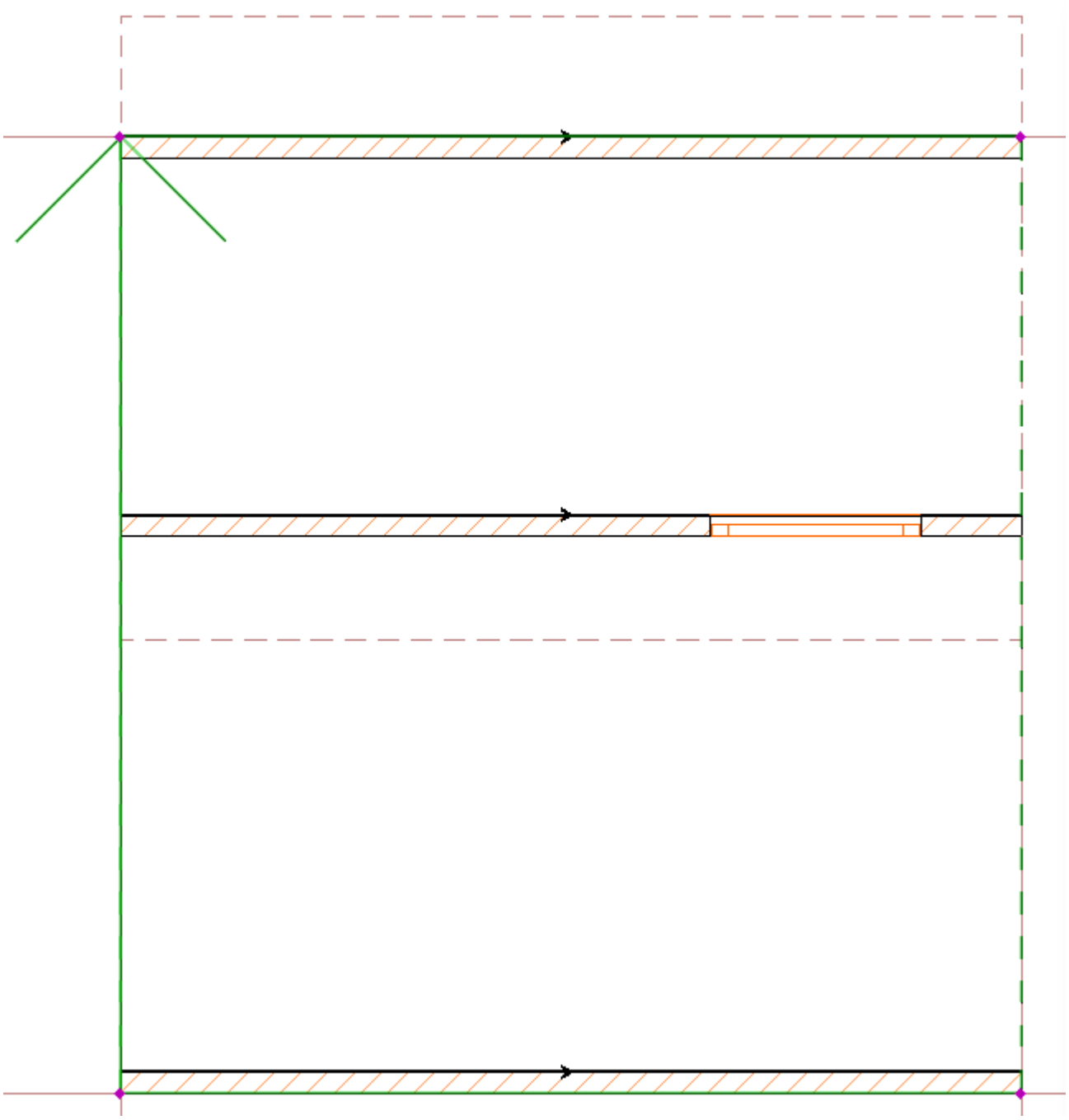


  All these pick-buttons add items to collection. When having the ArchiFramePlacementLines object selected it is possible to show picked elements using the middle button and clear all picked elements with *Clear picked*. Clicking *Pick elems for shape* resets settings edited in the numeric dialogs.
- Truss type popup is used to set the truss shape. After clicking *Pick elems for shape* ArchiFrame detects the shape automatically and the choice can be changed manually.
- *Edit truss shape* opens dialog where truss can be edited numerically. Links to different truss type dialogs:
  - Gable truss
  - Mono truss
  - Clerestory truss
  - Attic truss
  - Scissor truss
- *Spacing spread* places trusses with the same distance between the trusses into selected area not exceeding given maximum distance. *Center to center* uses given value directly.
- *Create truss drawing* creates dimensioned detail drawing of selected trusses. Trusses must have valid IDs (given with *Give IDs*) before this can be done.

Example steps for roof like this:

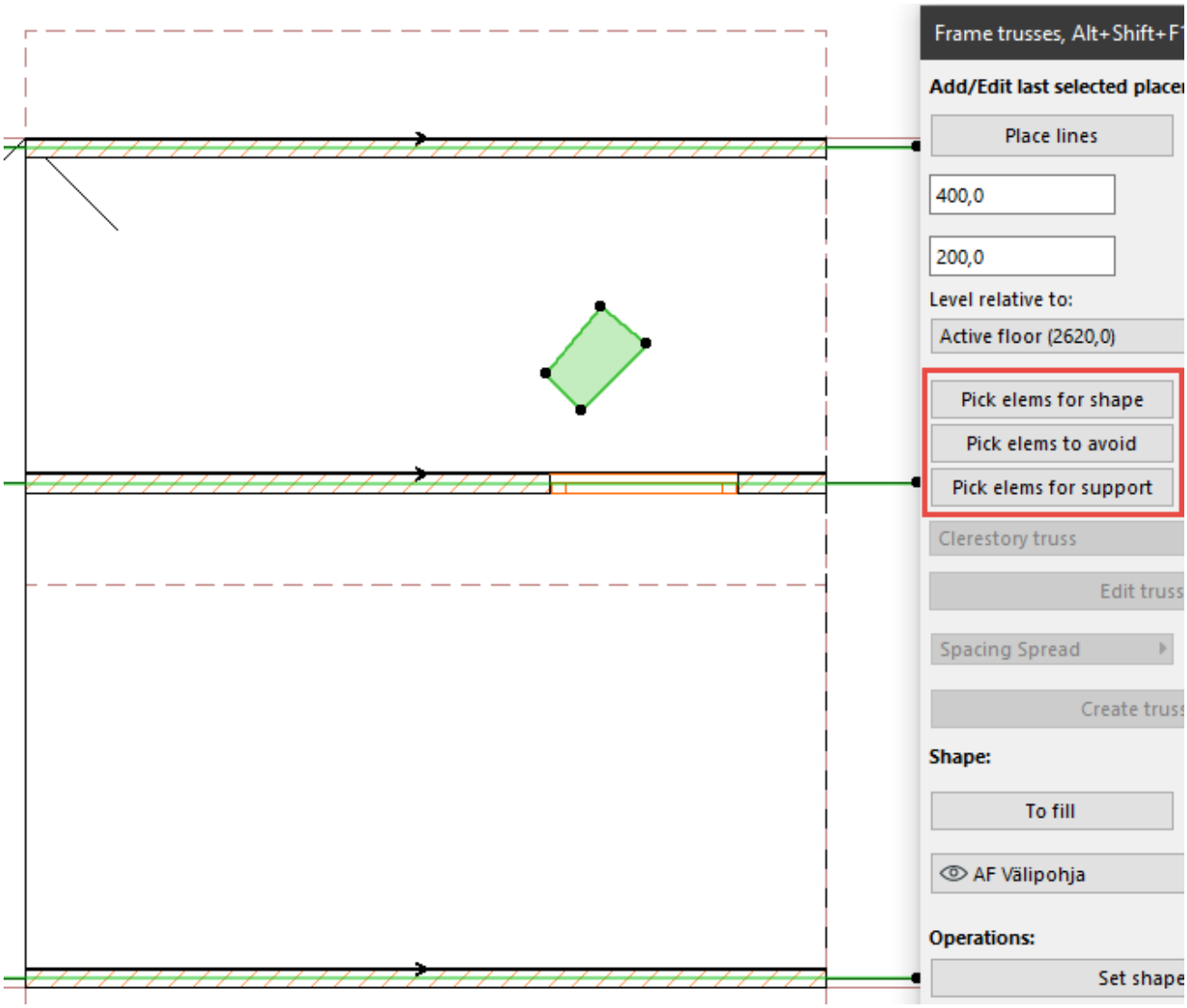


Click *Place lines* and first showing the side of the trusses and then direction:



The arrow shows direction of truss. Arrow end is the right-hand side in all settings dialog and truss drawing.

Then pick the roofs for shape, any 2D-element (fill in this case) to avoid and 2D lines as support center lines:



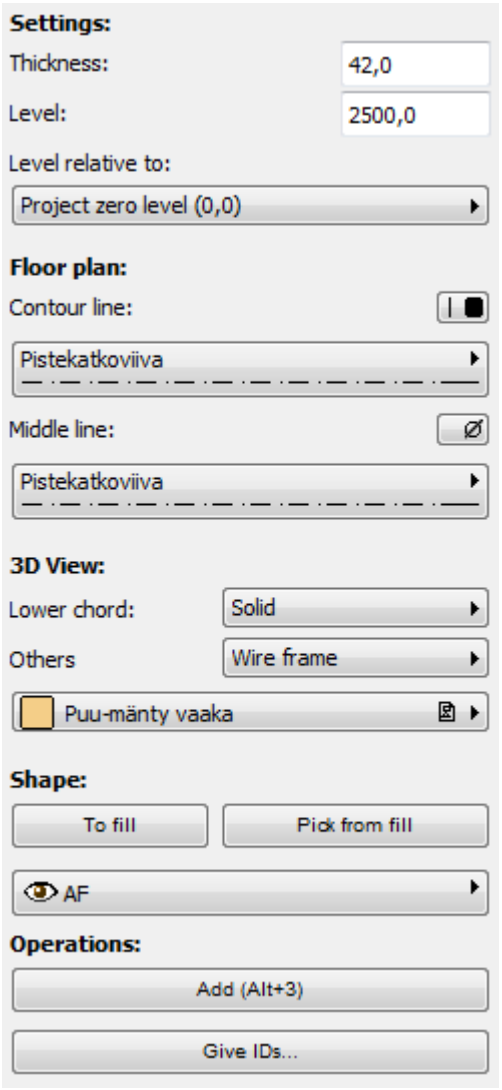
Click *Edit truss shape* to adjust any values and finally create trusses with button *Create/Update trusses*.

Final trusses can be refined more with tools described next for method B.

20.2 Truss workflow when adding trusses using just fills (method B)

- Make a truss section for each truss type and add dimension lines manually if needed.
- Pick the shape from the section to Archicad fill. Use separate fill for the lower chord or other main part to allow getting its projection to elevations without the rest of the truss.
- Pick the truss shape from fill or fills and place the trusses. The truss shape is always adjusted with fill – the truss object cannot be stretched.
- Assign IDs to trusses.

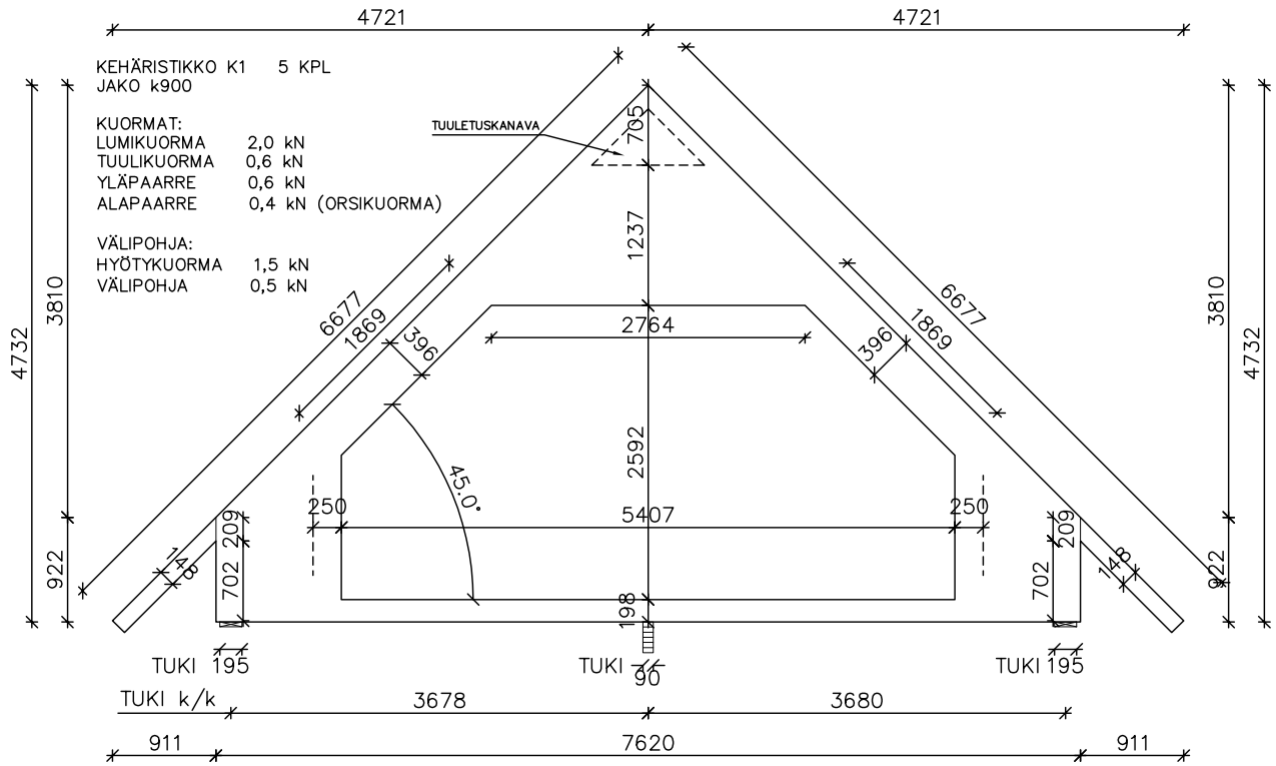
Truss tool parts are:



- *Thickness*, entire truss has the same thickness.
- *Level*, the level of lowest point of the truss.
- *Level relative to*, level anchor point.
- *Floor plan*, the floor plan display.
- *3D View*, it may be useful to show part of the truss as solid and the rest as wire frame.
- *To fill*, creates a fill to be used to edit the truss shape.
- *Pick from fill*, picks the shape from selected fill(s) or placed truss. Picked shape is used to create new trusses or it can be set to existing ones.
- *Add* or *Set shape*, adds new truss to given place or sets shape for selected trusses to shape picked earlier.
- *Give IDs*, gives similar trusses same ID. Can be issued with or without selection.

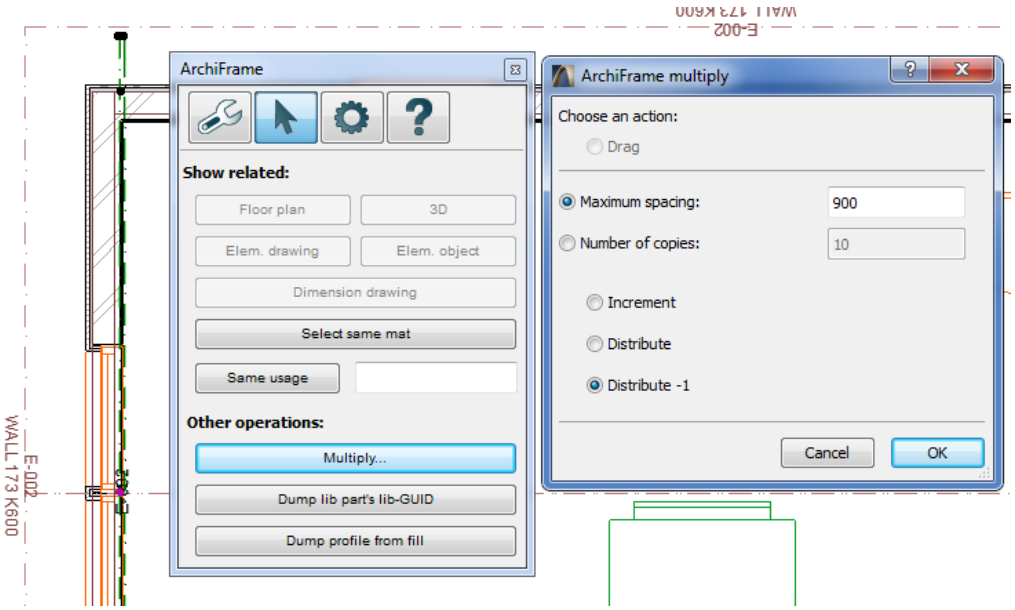
20.3 From truss dimension drawing to fill (method B)

The dimension drawing is possibly easiest to draw with the line or polyline tool. For example, from the illustration below, the outline is picked to Archicad fill by selecting the fill tool and clicking any of the contour edges having space bar pressed. The hole is added afterwards:



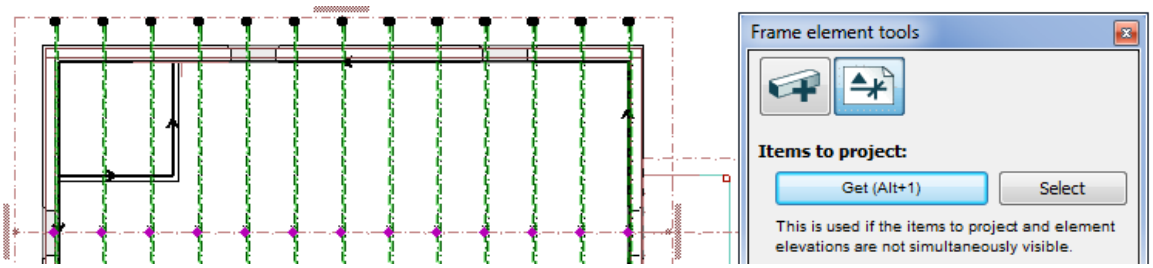
20.4 Placing trusses (method B)

In this example, the shape is first picked and then two trusses are placed next to the gable elements. Use the ArchiFrame *Multiply*-tool to ensure there is maximum 900 mm spacing and that the trusses are placed evenly. Select the first truss and open *Multiply* settings:

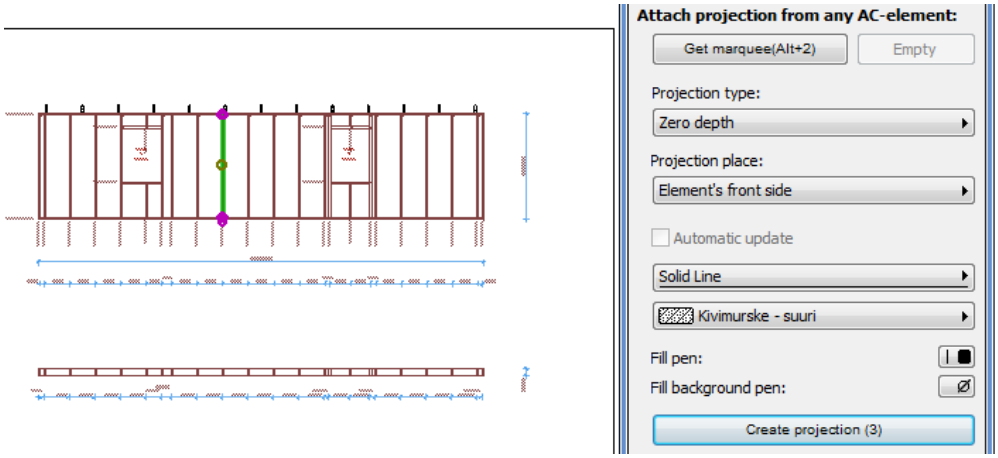


Draw a line from the first truss middle line hotspot, to the other side similar hotspot. After creating every truss, *Give IDs* is used to assign IDs.

The trusses are attached to the element elevations using the element projection tool. First select the trusses and pick them as *Items to project*:



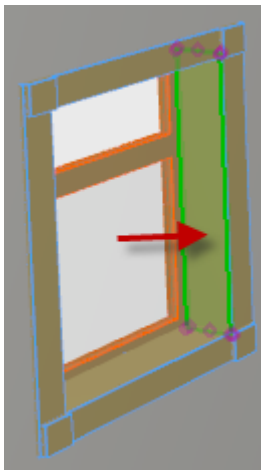
Then select one plank from destination projection and add the trusses with *Create projection*–operation:



Note! Currently the projections are not updated if the truss is changed and vice versa.

21 Weatherboards

Weatherboards are separate *ArchiFramePlank*-objects and they may be used with any door or window. ArchiFrame recognises changes to walls and openings and updates the weatherboards automatically when they are changed. Weatherboards may contain also window jambs:



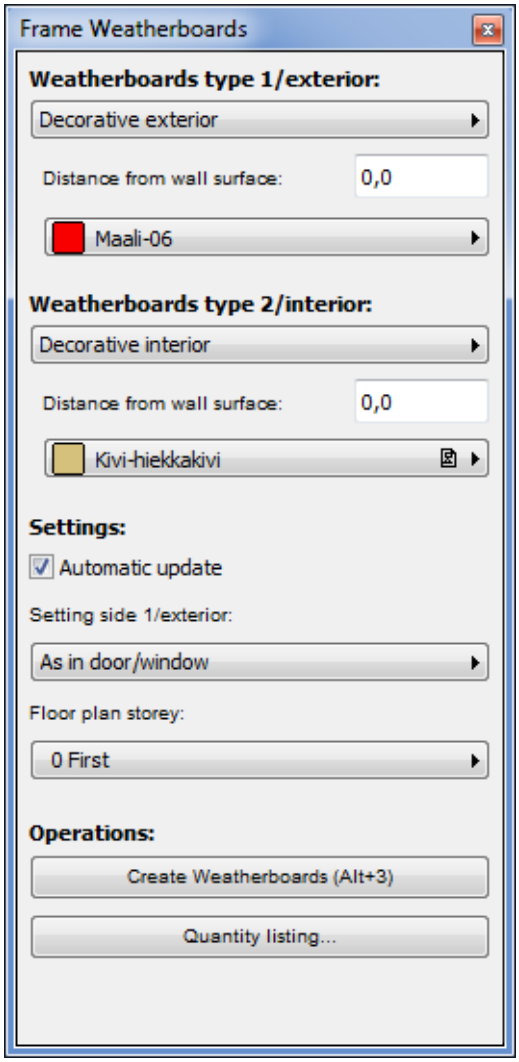
ArchiFrame produces cut list and even cnc-file for the weatherboards. An example of complicated but fully automatic exterior weatherboarding:



This is done by tailoring the manufacturer specific rules into file ArchiFrameCoveringBoards.xml. Please contact us if you are interested to get your own types into ArchiFrame.

When using ArchiFrame weatherboards the openings' own similar parts should be switched off from the door/window settings.

21.1 Weatherboard tools



- *Weatherboard type 1/exterior* defines in exterior walls the exterior type and the type for the other side of the partition walls.
- *Distance from wall surface* is the distance between the weatherboards and the wall. This can be used to make bigger jambs. Negative value places weatherboards inside the wall.

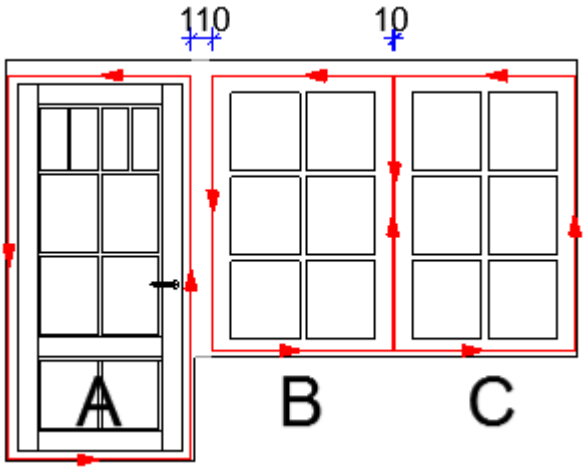
- *Automatic update* allows ArchiFrame to update the weatherboards whenever the wall or the opening changes.
- *Setting side 1/exterior* defines how to detect the exterior side. If the same types are used on both sides then this setting does not matter.
- *Floor plan storey* is the home storey for the weatherboards. Even if the weatherboards are not visible in the floor plan Archicad will use the plank hotspots. This may cause the mouse click to attach to the wrong invisible hotspot when adding dimension lines. It is recommended to create a new helper storey to place all the weatherboards. With default settings ArchiFrame will place the weatherboards to storey having text *constructional storey 1* in its name.
- *Create weatherboards* creates the planks. Selection can be for a single opening, group of openings, wall or multiple walls. ArchiFrame automatically combines together openings which are close enough to each other. This distance is defined in the xml-settings.

Tool palette can also be used to edit existing weatherboards. Selection can contain a plank, opening or one or more walls. Home storey cannot be changed here after creating the weatherboards.

21.2 Weatherboards setting file ArchiFrameCoveringBoards.xml

Because the weatherboards are quite complicated, the rules to create them are not simple. However, everyone who has done GDL-objects should be able to add and edit the rules.

The weatherboards are based on the contour lines made from the frames. For example:



Contour lines are originally oriented counter clockwise. Line between door A and window B is attached to door A's right hand side because it is longer than the window edge. At the same time window B's horizontal edges are extended to the door's vertical line. Lines between windows B and C are combined to form a single line which is centred between the windows. There will also be a special cover plank defined in the settings.



ArchiFrame detects standard rectangular windows. To form the contour line for an angled opening there is a special Lua script. Some of the angled openings in Archicad standard library are already supported by the script, but other openings and libraries require editing the script. Examples and instructions to define the frame edges can be found from the xml-settings under tag <openings_script>.

Weatherboard definitions include rules to handle frame edges <linegroup>, operations between the planks <operations> and *Lua*-scripts <scriptplanks> for example to add supporting planks for fascia board.

21.2.1 Variables in expressions

It is possible to use variables instead of fixed values to allow changing materials without editing the rules to place them.

Variable	Description
jamb_depth	Jamb width, in other words the distance from frame to wall surface.
mat_thickness	Current material size.
mat_height	Current material size.
wall_distance	Distance from wall surface (comes from tool palette).
cosangle	Absolute value of cover plank angle, horizontal=1.0, vertical=0.0.
sinangle	Absolute value of cover plank angle, horizontal=0.0, vertical=1.0.
[parentid]	Can be used to set textual parameter for a plank: Owner opening's ID. For example setting the weatherboard ID: `<objparam name="#id">WB-[parentid]</objparam>`

21.2.2 Settings <settings>

21.2.2.1 Common settings <plank>

```
<plank>
  <misc minlen="0.050" layer="Cover boards" floor="*constructional storey 1*"></misc>
</plank>
```

Xlm-attribute	Description
minlen	Minimum length for a plank. Shorter will not be created.

layer	Forced layer. If not given, the weatherboards are created onto wall's layer. Layer is created and shown automatically but locked layer is not automatically unlocked.
floor	Default home storey for weatherboards. Name can contain wild cards * and ?. See Weatherboard tools.

21.2.2.2 Combining adjacent openings <lines>

```
<lines>
  <combinelines id="cover_out">
    <combine maxdist="0.020" settype="combine20">
      <material id="L990" zoff="jamb_depth-mat_thickness*0.5" linexoff="-mat_height*0.5"
rotangle="90"></material>
    </combine>
    <combine maxdist="0.111">
      <!-- Use type's default material -->
    </combine>
    <combine maxdist="0.136">
      <material id="L149" zoff="-mat_thickness*0.5" rotangle="90"></material>
    </combine>
  </combinelines>

  <!-- The same for inner cover planks -->
  <combinelines id="cover_in">
    <combine maxdist="0.020" settype="combine20">
      <material id="L989" zoff="jamb_depth-mat_thickness*0.5" linexoff="-mat_height*0.5"
rotangle="90"></material>
    </combine>
    <combine maxdist="0.111">
      <!-- Use type's default material -->
    </combine>
    <combine maxdist="0.136">
      <material id="L124" zoff="-mat_thickness*0.5" rotangle="90"></material>
    </combine>
  </combinelines>
</lines>
```

Defines rules for combining edges and special cover planks for narrow, normal and wide gap. In the example, the narrow covering plank is used up to 20 mm gap, weatherboard material up to 111 mm and special wide material from 111 to 136 mm gap. Material settings are given as in element module, see Material settings <material>.

```
<combinelines id="cover_out">
```

Gives the settings ID "cover_out" that is used to refer to these settings in many following weatherboard type definitions.

```
<combine maxdist="0.020" settype="combine20">
  <material id="L990" zoff="jamb_depth-mat_thickness*0.5" linexoff="-mat_height*0.5"
rotangle="90"></material>
</combine>
```

Xlm-attribute	Description
maxdist	Maximum distance for edges to be combined.
settype	Gives the edge a geometry ID that will be used instead of ID coming from geometry rules. Also defines that the piece is narrow type which will not get jambs.
notnarrow	Used with settype=”xxx” to indicate that the piece is not narrow one but normal piece getting jambs: notnarrow="1"

skipdifflen	If given skipdifflen="1" the combine rule is not used if related lines have different lengths.

The main <combinelines>-tag must contain the widest combining rule to be used in different <linegroup>-tags. There the maximum width to combine can be reduced but not the other way round.

Weatherboard Lua-script has these globals set by ArchiFrame:

Variable	Description
gJamb	Distance from wall surface to the window frame.

21.2.2.3 Default settings for new plank <newplank>

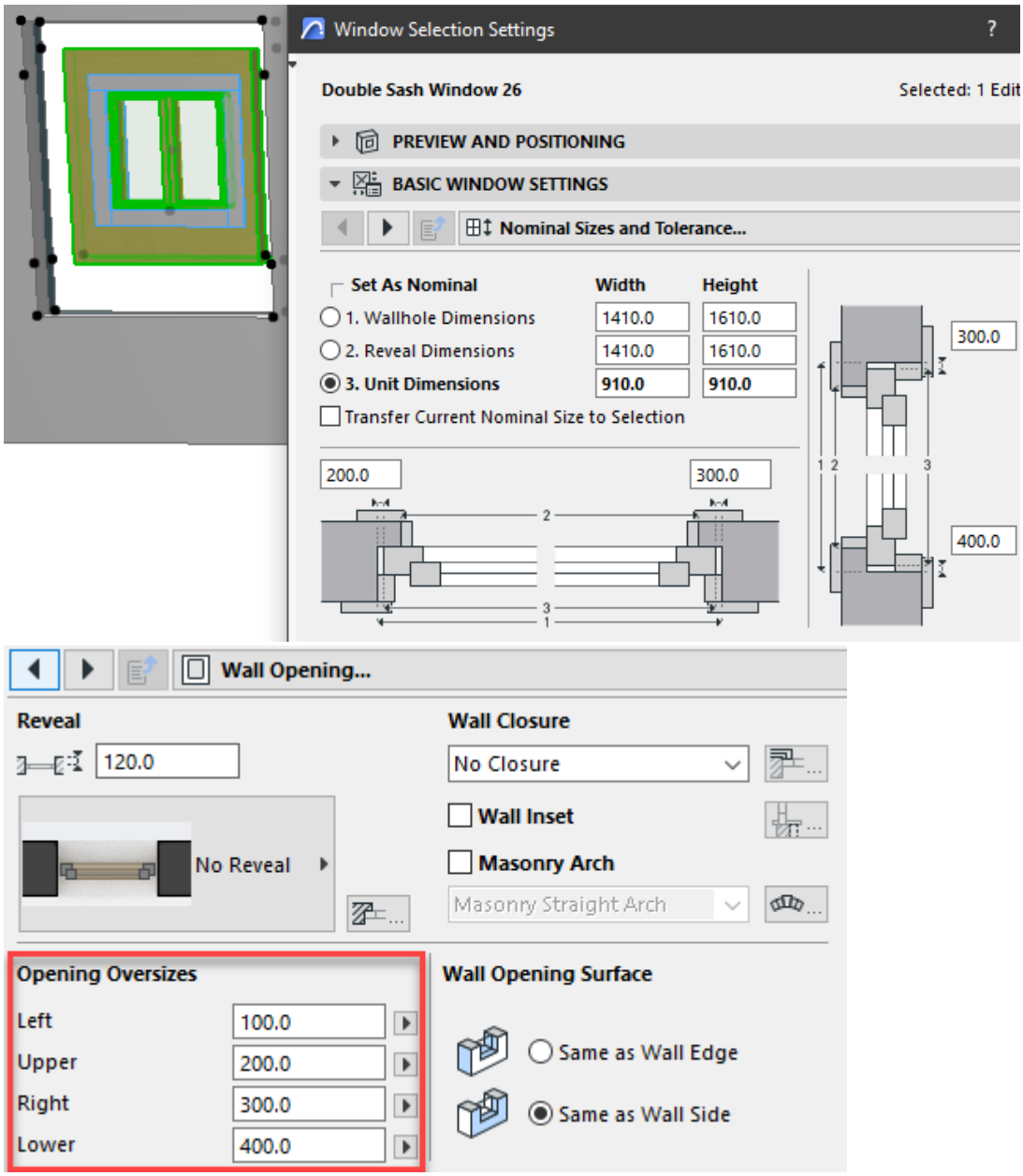
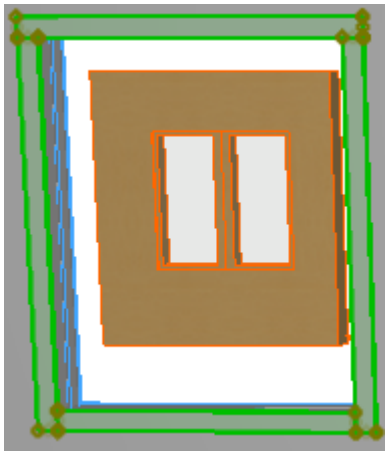
See XML-settings for an Archicad element.

21.2.3 Weatherboard types <covertypes> / <covertype>

In this part we will go through weatherboard type Decorative exterior. For example:

```
<covertype id="xxx_tradisjonell_10" name=" Tradisjonell YV 1.0" idstr="WB-#parentid#-01"
oversize="0">
```

Xlm-attribute	Description
id	Must be present – the internal ID of the weatherboard type
name	The name visible to the user
idstr	This is given if weatherboards should have unique ID. ArchiFrame will use the last part as changing part. Possible replaced strings inside idstr: #parentid” ID of the related door or window

oversize	With the default value 0 the weatherboards are anchored to the window frame:  Value 1 anchors to the full opening edges including the oversizes above: 

21.2.3.1 Settings for the type <settings>

```
<settings>
  <newplank>
```

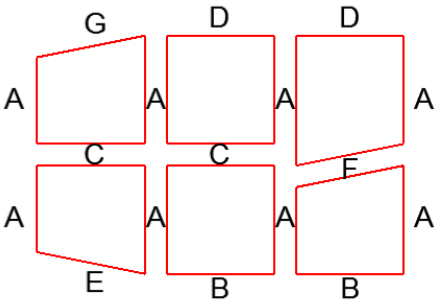
```
  </newplank>

  <combinelines ref="cover_out">
  </combinelines>
</settings>
```

<*newplank*>-tag is used to override settings given at the beginning of the file. Tag <combinelines ref = "cover_out"> refers to settings defined earlier.

21.2.3.2 Weatherboard group <create>/<linegroup>

The <linegroup> defines handling for frame edges. Edges will get geometry ID based on the position:



Letter	Geometry ID
A	vertical, middle ones verticalmid
B	horizontalbot
C	horizontal
D	horizontaltop
E	inclinedbot
F	inclined
G	inclinedtop

In addition to these, the geometry ID can come from Combining adjacent openings <lines>-section, for example <combine maxdist = "0.020" settype = "combine20"> gives geometry ID combine20.

<*linegroup*>-tag may contain attribute type = "jamb". In this case, the definitions are for jambs and the operations will change:

- Frame edges will not be combined but instead every adjacent frame will get the jambs.
- Jamb is created only if there is a weatherboard on top of it or if the weatherboard type defines only jambs (the only <*linegroup*>-rule is for jambs).

Single <combinelines>-definition can be placed inside the <linegroup> to override the main combining rules. Please note that the main combine-rule decided whether the openings are combined or not. For example, to set special piece to middle of the openings:

```
<linegroup>
  <!-- Override for this layer: 014 if distance of the frames < 103 mm -->
  <combinelines>
    <combine maxdist="0.1031" extendtop="-0.200" settype="verticalmid_handled">
      <material id="04.617.014" zoff="mat_height" rotangle="180"></material>
    </combine>
  </combinelines>
```

```
  <planks group="vertical" type="vertical" lineoff="-0.021" extendbot="0.056"
extendtop="0.093" extendt="0">
    <material id="04.617.013" zoff="mat_thickness*0.5" rotangle="90"></material>
  </planks>

  <planks group="verticalmid" type="verticalmid" lineoff="-0.021" extendbot="0.056"
extendtop="0.093" extendt="0">
    <material id="04.617.013" zoff="mat_thickness*0.5" rotangle="90"></material>
  </planks>

  <planks group="verticalmid_handled" type="verticalmid_handled">
    <material>Material set in combinelines-definition, must have material tag here
too</material>
  </planks>
```

21.2.3.3 Weatherboards group planks <create>/<linegroup>/<planks>

<linegroup> contains <*planks*>-tags that define the actual planks.

```
<planks door="0" group="vertical" type="vertical*" lineoff="-0.010" extendtop="0.120"
extendbot="0.195" extendt="0">
  <material id="block" thickness="0.022" height="0.120" zoff="-mat_thickness*0.5"
rotangle="90">
    <objparam name="iUsageId">VUORIL-LEV</objparam>
    <objparam name="#id">VLLEV-[parentid]</objparam>
  </material>
  <script>
  </script>
</planks>
```

<*planks*>-tag may contain attributes:

Xlm-attribute	Description
door="0/1"	Will this <*planks*>-tag be handled with doors. Default is one, will be handled.
window="0/1"	As previous but for windows.
group="name"	Group name for the planks to be used in later <*operations*>-rules that adjusts the planks together.
type="name"	The kind of lines which will be handled here. Name may contain wild cards, for example, horizontal*. Different values can be given separated with vertical bar, for example, horizontaltop|horizontal|inclined*.
lineoff="meters"	Distance to move the edge right/away from the opening. Negative moves inwards. For example, if the weatherboard should cover 10 mm of the frame, the value is -0.010.
extendtop="m"	How much the edge is extended at upper end.
extendbot="m"	How much the edge is extended at lower end.
extendt="0"	Can be used in conjuction with extendtop and extendbot: If edge ends to middle of another line (T-joint), it will not be extended.
extend="m"	How much the edge is extended from both ends.
extendfree="m"	Extend only if the end of the edge is not inside the opening and does not end at the middle of the other edge (T-joint). Corner is extended.
sideout="2"	Side to be outside from the wall 1-4 (1=top, 2=front, 3=bottom, 4=back). Will not rotate the plank, just swap its direction if need. Useful with profile pieces to get the orientation right. This must be given in the child <material>-tag, for example:

	<planks group="top_drip" type="horizontaltop|inclinedtop" lineoff="0.026" extend="0"> <material id="VL 42x60" zoff="-mat_thickness*0.5" rotangle="90" noswap="1" **sideout="2"**></material> </planks>

<*planks*>-tag has <*material*>-tag that defines the material type. If <*material*>-tag is missing, no plank will be created for the edge, but the <*plank*>-definition may be necessary to adjust other edges.

Material settings are given as in element module, see Material settings <material>. In addition, it is possible to specify XML-settings for an Archicad element. Also, with weatherboards it is possible to give attribute create="0" for example to join lines between openings but not to create the plank:

```
<combine maxdist="0.1461" extendtop="-0.200" settype="verticalmid_handled">
  <material id="04.401.015" zoff="-mat_thickness*0.5" linexoff="-mat_height*0.5"
rotangle="90" create="0"></material>
</combine>
```

Group may also have *Lua*-script. An example of the script is included in weatherboard type Decorative exterior. Script has a global variable gPlank having following fields (note that changing these fields will not change the plank - the plank must be edited with ac_objectset()-function):

Field	Description
guid	Ptr of the item
bDoor	true = door edge, false = window edge.
x1,y1,x2,y2	Edge coordinates in local 2D-co-ordinate system.
gx1,gy1,gz1, gx2,gy2,gz2	Plank coordinates in global model coordinates.
len	Plank length.
begConn	Begin joint type: • 0, No connection. • 1, Connected to other edge's end. • 2, Connected to middle of another edge (T-joint) or end is inside the opening.
endConn	End joint type.
bExtR	true = exterior to right, false = plank direction is swapped and exterior to left.
group	Edge geometry ID name, for example "vertical".
sideout	Only when creating planks with script: Side to be outside from the wall. In scriptplanks-section values 11-16 may be used. Those values make the plank's direction to be swapped without swapping the machinings.
xmlsettings	Normal xml-settings to be applied after base xml-settings.

Script can set global variable gnSideOut to value 1…6 which defines which side of the specified plank must be outside from the wall (see About the planks). This is required to allow placing planks with asymmetric profile. In addition the script can set globals gtblCutBeg and gtblCutEnd to define the cutting plane for the plank. These variables may contain fields:

Field	Description
x,y,z	3D-coordinate on the plane.
dx,dy,dz	Plane normal vector.

21.2.3.4 Operations between the planks <create>/<operations>

As in element module, see Operations between the planks <operations>. With weatherboards it is possible to give additional attributes targetex and operatorex, which refer to weatherboard place:

```
<jointo target="vertical*" targetex="verticalmid"  operator="inclined*">
```

21.2.3.5 Script defined planks <create>/<scriptplanks>

First there is a definition of the planks to be handled:

```
<findgroup resulttbl="gtblTop">
  <base name="topbox"></base >
</findgroup>

<findcross resulttbl="gtblWeather">
  <base name="weatherboard"></base >
  <cross name="vertical*" maxdist="0.150" linepos="mid"></cross>
</findcross>
```

This example creates global *Lua*-variables gtblTop and gtblWeather. Table gtblTop will contain planks with matching group = "topbox"–definition. <findcross> will identify all the planks (weatherboard) intersecting with the vertical planks (vertical*). <*base*>- and <*cross*>-tags may have attributes:

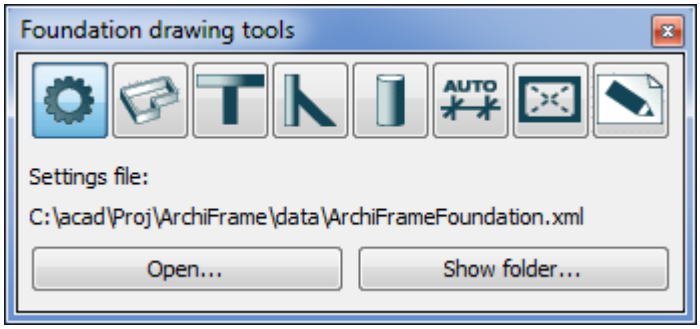
Xlm-attribute	Description
name = "xxx"	Identifies every plank that has matching group name <planks group = "xxx"…>.
maxdist = "m"	How much current line is extended when checking its intersection with other lines.
linepos = "mid"	Use middle lines instead of plank reference lines.

Every plank has same fields as in the plank specific script, see Weatherboards group planks <create>/<linegroup>/<planks>. However, fields' begConn and endConn are always zero here. There is also the field tblCross that contains all crossing planks. It contains fields:

Field	Description
x,y	Intersection point.
tblPlankx	Information about crossing plank (contains all the same fields as the base plank).

22 Foundation drawing

Foundation drawing part is used to produce 2D drawing from the 3D model.

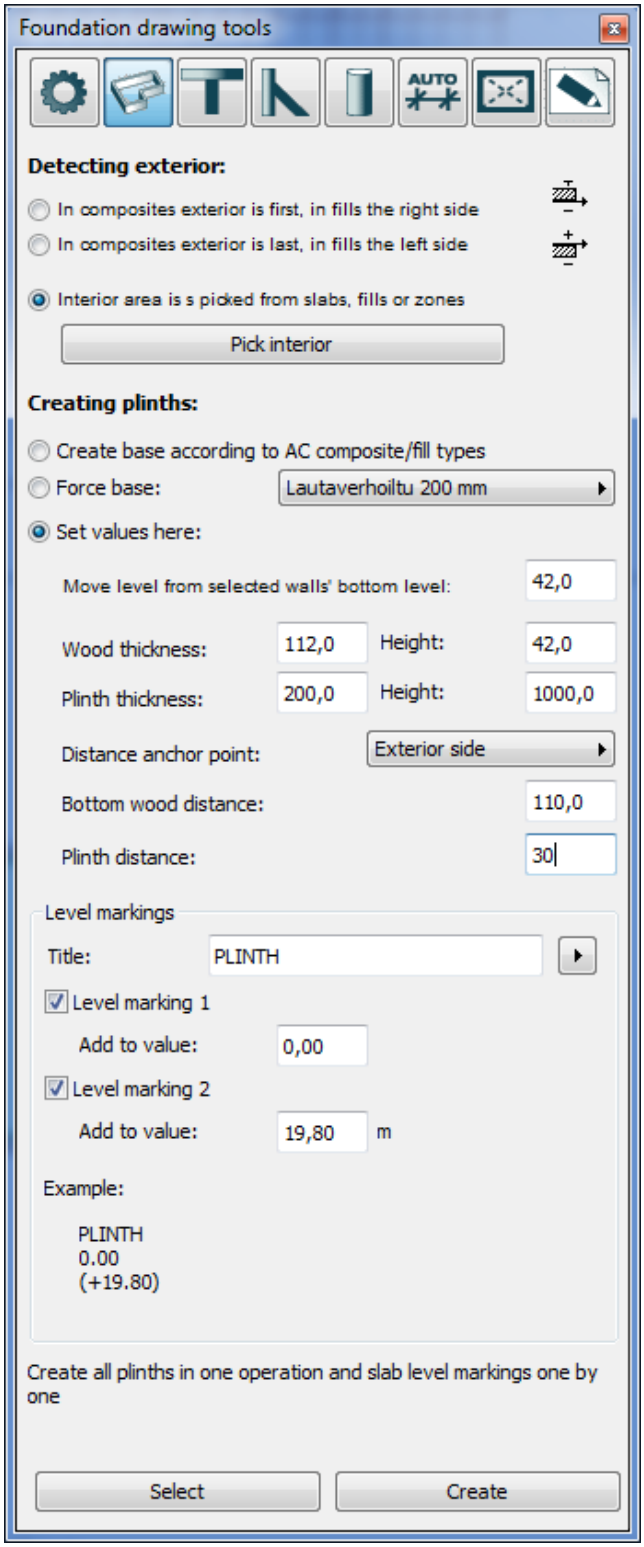


All editable settings are placed into xml setting file. Before editing, the stock file is copied and renamed before changes are made to it. Instructions are included inside the xml-file. For automatic

dimensions, the plinth, sole plate, columns, load bearing and stiffing walls must be placed onto separate layers.

22.1 Plinths and slabs

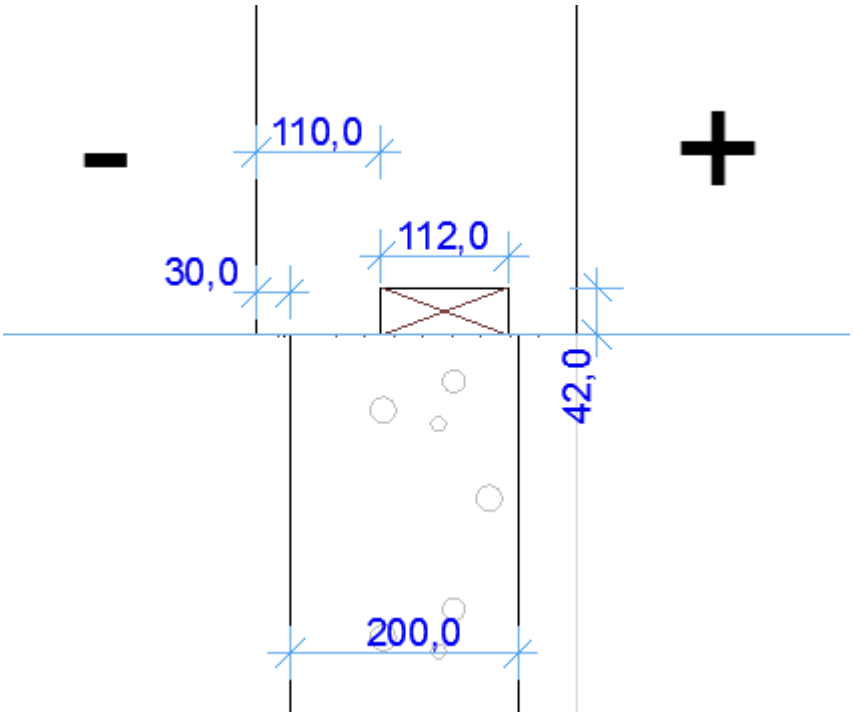
First select the way to detect wall's exterior side. With composite structures the structure defines the exterior side but for walls made with just fills, the interior area is defined by picking slabs, zones or fills that cover an entire interior area. Then Archicad *wall*-tool settings are adjusted to fit the plinth and sole plate before ArchiFrame-settings are edited:



Plinth and sole plate are created by the following settings:

- *Create base according to AC composite/fill type,* which means that the setting file contains the standard composite types and work according to the values entered there.
- *Force base*, use the selected composite type rules for every wall.

- *Set values here,* enter values in the tool palettes. For example, the values entered above is like this (the sole plate will be placed inside the original wall because level offset is equal to the sole plate thickness):



- *Level markings* will be like the example in the tool palette.
- *Select* selects all elements defined in the setting file (need to edit the settings to get meaningful results). Elements can be selected using standard *Archicad*-tools.

To create plinths, select walls that are on top of the plinths. If *Select*-button does not select suitable walls, the walls are selected manually.

If necessary, the plinth is done and finished manually first. After that, the level markings for the porch, terrace and floors etc. are modelled with Archicad *slab*-tool. ArchiFrame extends the slab markings close enough to the plinth to touch the plinth.

22.2 Load bearing and stiffing walls

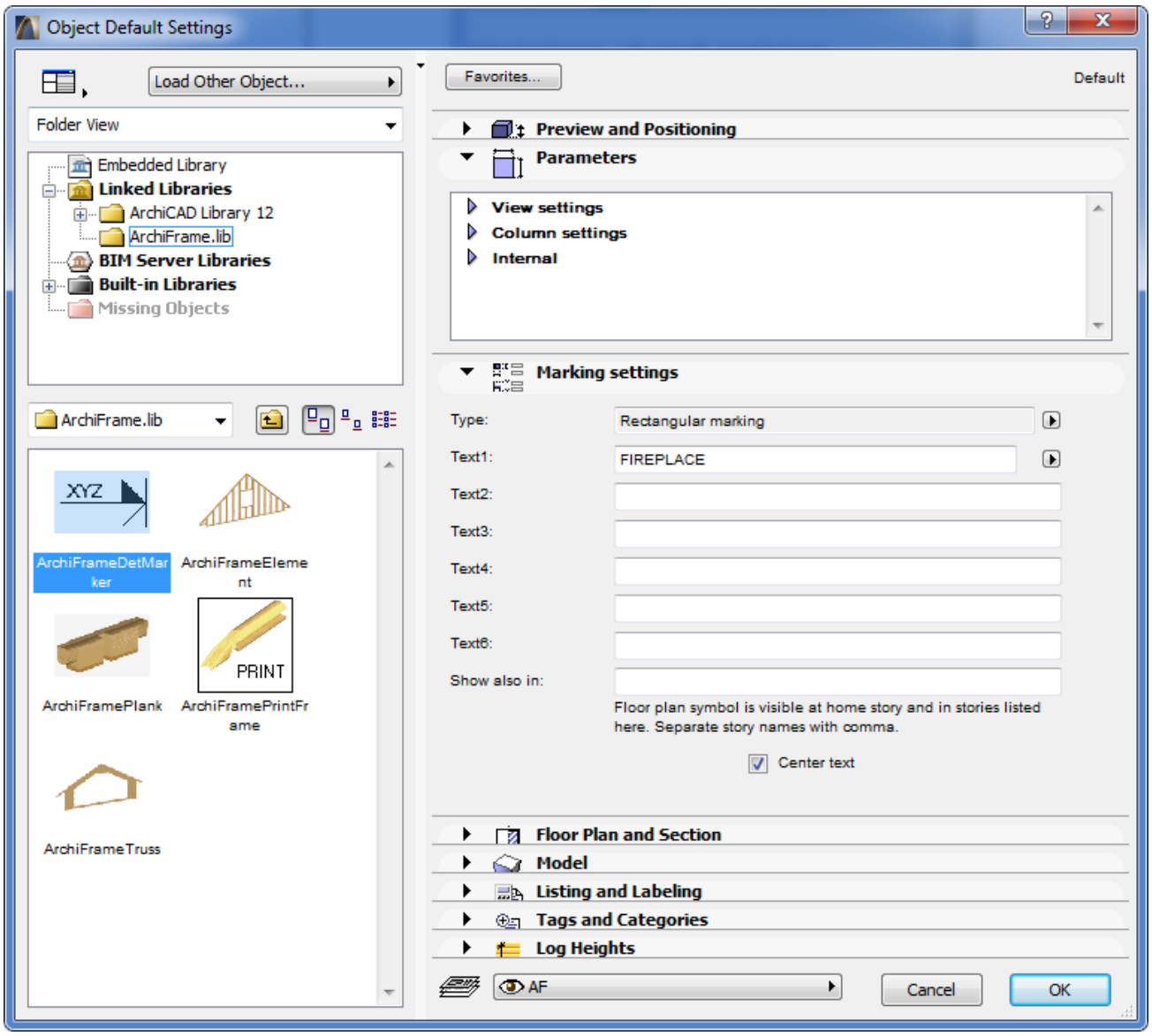
First the walls in question are selected and *Create* will create the markings.

22.3 Columns

This operation adds column markings with *ArchiFrameDetMarker*-object. Columns modeled with Archicad objects may require programming into the settings file.

22.4 Adding other parts manually

For example, fireplaces are added manually using object ArchiFrameDetMarker before automatic dimensions are applied as shown below:



Marking is added to the floor plan and stretched, then moved to the correct place.

Other kinds of marking are added in the same way.

22.5 Automatic dimensions

After the foundation drawing has been completed and any additional items are placed, the dimension lines are added with the *Create*-button. Automatic dimensions require that different items like plinth and load bearing walls are on separate layers. Another requirement is that there must be a plinth to anchor other items too.

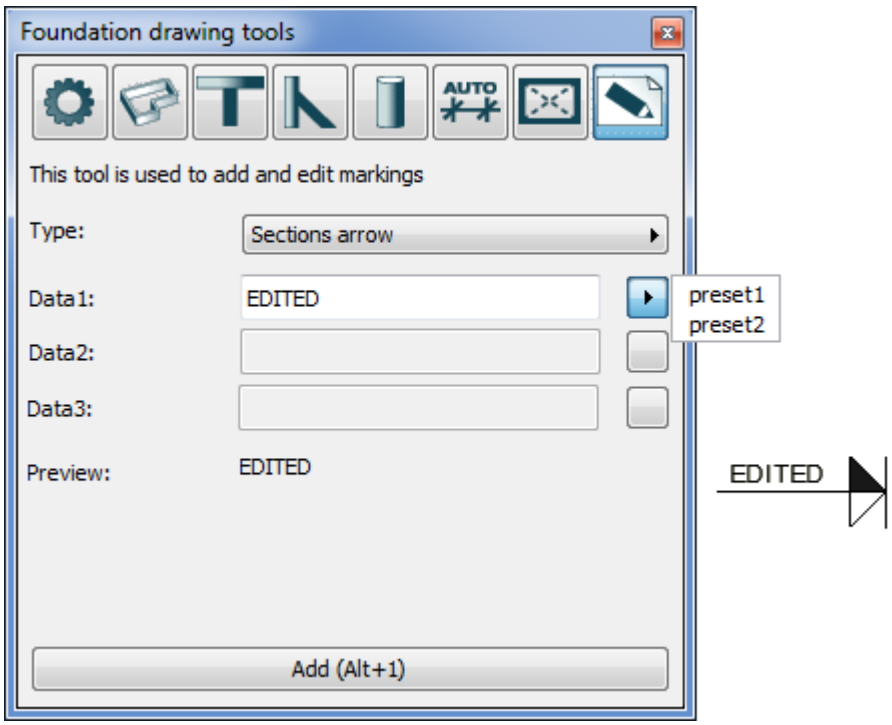
Without any selection, this tool adds dimension lines to every element. For example, if a fire place is added after creating the dimension lines, the tool will only create a dimension line for that item.

22.6 Cross dimensions

This tool adds dimension lines anchored to plinth and slab marking corners. ArchiFrame searches for the closest corner at the clicked place. This makes clicking easier.

22.7 Markings

This tool is used for example to add references to external detail drawings. The preset lists are defined in the settings file and it is very useful to add frequently used detail names there. For example below, the text EDITED is written manually to the Data1 input:



23 XML-settings for an Archicad element

The settings are usually given the same way as in corresponding Archicad setting dialog. All dimensions are given in meters using a full stop as the decimal separator (1.234) and angles as degrees. Sometimes these settings are used to filter elements but usually the settings are for new elements.

Archicad attributes (line type, fill etc) are referenced with number, name or partial name which may contain wild cards?*. When using names it is possible to have many names separated with vertical bar |. This may be useful to support many languages in single setting file. For example, concrete in English and in Finnish: *concrete*|*betoni*.

23.1 <floor>storey</floor >

Storey is numeric or textual as attributes.

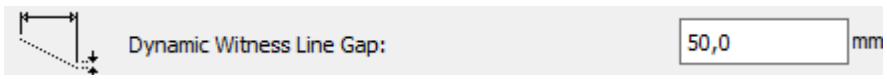
23.2 <layer>layer</layer>

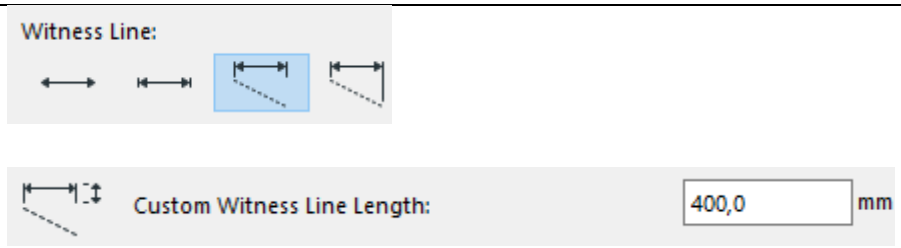
Numeric or textual.

23.3 <elemtype>number</elemtype>

Used when searching elements. The numbers for different element types are found with Google search *API_ElemTypeID*.

23.4 <elemparam name="xxx">value</elemparam>

Name/xxx	Description
fontname	Font name.
fontsize	Font size in current scale
fontsizemodel	Font size in model world. If you specify 100 mm, it will be 100 mm in model world no matter what the scale is.
fontstyle	Font style, list using comma as separator that may contain: bold, italic, underline.
just	Justification: left, center, right, full.
pen	Default pen.
linetype	Line type.
dimensiontype	Dimension line type, possible values: • linear • cumulative
dimlinepen	Pen for dimension lines.
markersize	Marker size for dimension line.
markerpen	Marker pen for dimension line.
markertype	Marker type for dimension line: APIMark_CrossLine, 0 APIMark_EmptyCirc, 1 APIMark_SlashLine, 2 APIMark_OpenArrow30, 3 APIMark_ClosArrow30, 4 APIMark_FullArrow30, ... APIMark_SlashLine45, APIMark_CrossCirc, APIMark_OpenArrow90, APIMark_ClosArrow90, APIMark_FullArrow90, APIMark_FullCirc, APIMark_PepitaCirc, APIMark_BandArrow
witness	Dimension witness line type, numbers 0-3: APIWtn_None = 0. APIWtn_Small = 1. APIWtn_Large = 2. APIWtn_Fix = 3.
dimwitnessgap	From dimension tool settings, if witness = 2:   If witness=3:

	
dimtextpos	These settings:  0=Above 1=In 2=Below
radius	Angle dimension arc radius.
level	Level for wall or object.
height	Height for wall.
thickness	Thickness for wall.
fill	Fill, for walls building material starting from AC17.
cutfill	From Cut surfaces settings. For hatch this is the fill.
cutlinepen	From Cut surfaces settings.
cutfillpen	From Cut surfaces settings. For hatch this is the fill pen.
cutfillbgpen	From Cut surfaces settings. For hatch this is the fill background pen.
material	3D material.
materialrgb	Material in RGB-format. The color is between 0…1. For example 100% green is 0, 1, 0.
libname	Libpart name for searching.
angle	Angle for object in degrees.

23.5 <objparam name="xxx">value</objparam>

Sets library part's parameter. Attributes can be given as text. It is possible to set the attribute type when setting textual value for example to *integer*-type parameter. For example setting fill to *integer*-parameter: <objparam name = "int_par" type = "fill">*concrete*</elemparam>. Attribute type must have one of following values: "linetype", "material", "materialrgb", "fill" or "composite".

These special values can be used as objparam-tag's name attribute:

- #id, object's ID
- #useobjlinetype, set value of *[ ] Use object's line type setting*
- # useobjsectattrs, negation of *[ ] Override Object's Attributes* (1=use object's attributes, 0=nope, use overrides)
- #useobjmat, inverse of value *[ ] Override surfaces* definition (0=use object's materials, 1=nope, use set surface material)

For bit fields attribute bitmask="x" is given. For example, with value 8 only bit 3 (0-based) will be set to target. Value 0 means clear bit and any non-zero value will set the bit.

It is possible to set numeric tables with following syntax, array will be automatically expanded:

```
<objparam name="iMc">
  <row><col>101.00000000</col><col>167.63750758</col><col>90.00000000</col><col>-
0.05560000</col></row>
```

```
  <row><col>305.00000000</col><col>1.00000000</col><col>0</col><col>0</col><col>-
0.15370000</col><col>0.02150000</col></row>
  <row><col>305.00000000</col><col>3.00000000</col><col>0</col><col>0</col><col>-
0.15370000</col><col>0.02150000</col></row>
  <row><col>201.00000000</col><col>167.63750758</col><col>90.00000000</col><col>-
0.05560000</col></row>
</objparam>
```

23.6 <class system="ARCHICAD Classification" system_fin="ARCHICAD-luokitus">

Allows setting element's classification from xml. An example definition:

```
<class system="ARCHICAD Classification" system_fin="ARCHICAD-luokitus">
  <classpath name="Materials" name_fin="Materiaalit"/>
  <classpath name="Wood" name_fin="Puu"/>
</class>
```

This sets the classification to ARCHICAD Classification/Materials/Wood or the similar one used in the Finnish version. The attribute names can be anything, just the values are searched from current classifications using case insensitive search. If the classifications do not exist, operation fails silently without an error message.

24 Lua scripts

Lua is an open source scripting language that is used for various automations. Information about the language is available at: http://www.lua.org.

Archicad 21 and later: Any ArchiFrame *Lua*-script is using western 8-bit character set and strings are converted to/from utf-8. An exception to this are scripts loaded from xml-files – these scripts are using utf8 always internally. Script may change this default behaviour by setting global gScriptUtf8 to values:

- Missing/nil, default.
- 0, script uses western locale.
- 1, script is fully utf8.

Useful code pieces:

```
-- Dumps given variable into string
-- To use: ac_msgbox(dump(v))
function dump(o)
  if type(o) == 'table' then
    local s = '{ '
    for k,v in pairs(o) do
      if type(k) ~= 'number' then k = '"'..k..'"' end
      s = s .. '['..k..'] = ' .. dump(v) .. ','
    end
    return s .. '} '
  else
    return tostring(o)
  end
end
```

24.1 Additions to standard Lua

24.1.1 io.open()

To open a file with utf8 file name in Windows (in Macintosh file names have been utf8-encoded for a long time), use prefix utf8, in io.open(), for example:
local hFile=io.open(strFileName, "utf8,wt")

24.2 Built-in support functions

24.2.1 ac_elemget (strGuid [,strOptions])

strOptions, specify to get more data. Multiple values can be used, and possible values are:

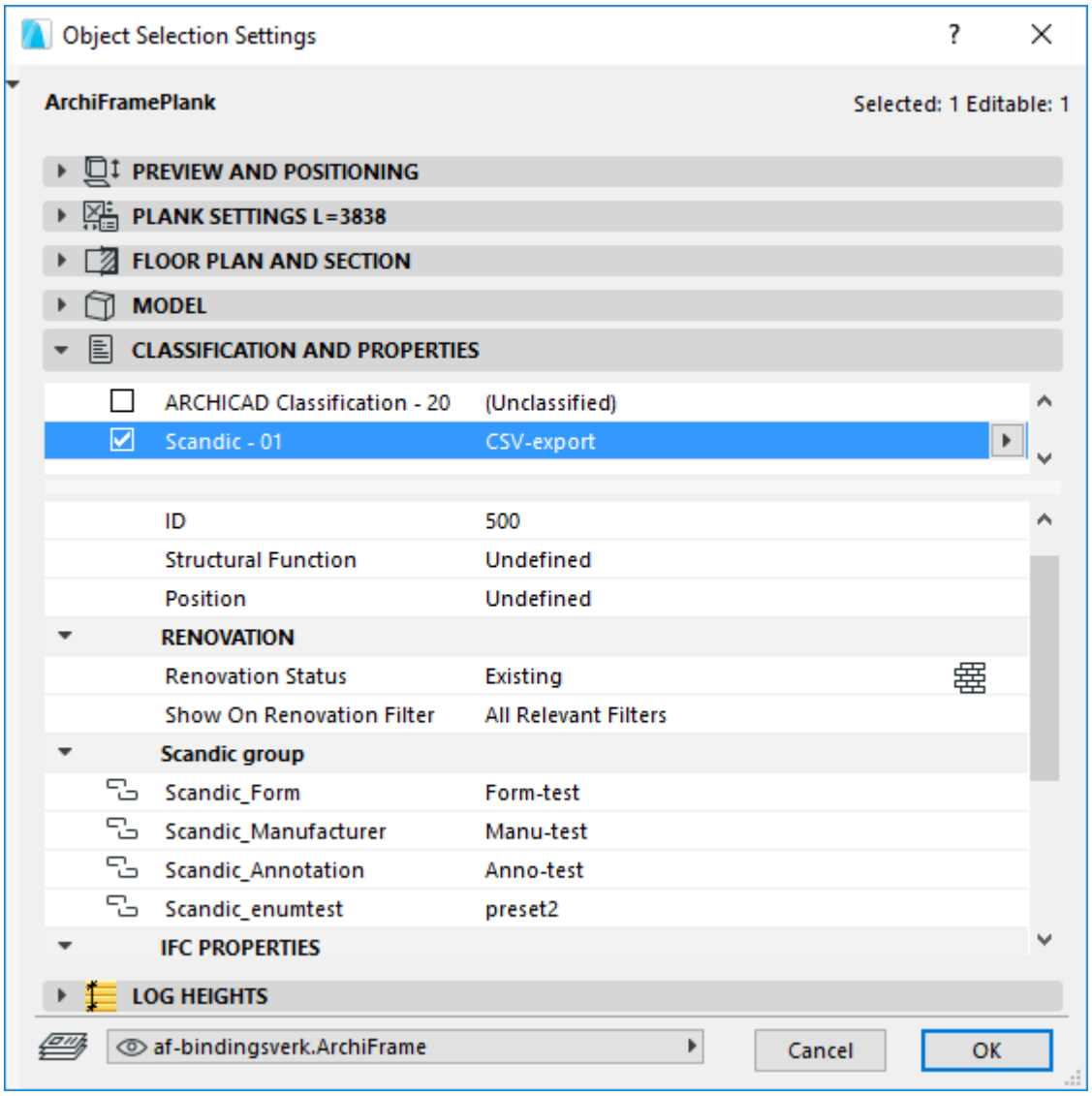
- props=1 Give all set properties in returned table field props (works starting from Archicad 22)
- ifc=1 Give IFC-data

Returns table of given element having fields:

- header.typeID, element type number, 6 = object.
- header.floor, element floor number.
- header.layer, element's layer number.
- header.id, element's ID
- header.libGuid, library part guid if any.

- If requested, field prop which is array of Archicad properties set for the element. Property name is the array key, for example value=prop["propname"].strvalue. ArchiFrame sets property name to lowercase. The fields are:
    - numvalue, set if it is a number
    - strvalue, set if it is a string value
    - Value may be missing if property value type is currently unsupported (like value list)

For example, following properties:



Are returned from expression elem=ac_elemget (elemguid, "props=1") as:

```
elem.props["scandic_form"].strvalue="Form-test"
elem.props["scandic_manufacturer"].strvalue="Manu-test" …
```

24.2.2 ac_findobj (strObjName [, nUseSelection])

Fills table of selected or all object in active storey. Table contains object guids.

- nUseSelection, 0 = get all; 1 = get selected or all if no selection; 2 = just give selected objects.

Returns: The table.

24.2.3 ac_getobjparam (strObjGuid, strParamName [, nRowNum, nColNum])

Gives requested parameter from the object, door or window. Parameters nRowNum and nColNum indexing starts from 1. Value 0 is ignored and -1 returns the table dimension. For example to get number of rows, set nRowNum to -1 and nColNum to 0.

Special *strParamName*-values:

- #id, returns the element ID
- #layer, returns the Archicad layer as number
- #guid, returns element's internal GUID.
- #floor, returns the storey number.
- #pos.x, returns x-co-ordinate of an object or middle pos for opening.
- #pos.y, returns y-co-ordinate of an object or lower pos for opening.
- #level, level relative to object's home storey.
- #levelglob, added also floor zero level.
- #angle, returns object angle in radians or reflected state for opening (0/PI).
- #mirroring, returns 0/1 reflected state for objects and doors & windows.
- #libname, returns library part name.
- #libguid, the same as GUID.
- #pen, default pen.
- #linetype, line type.
- #useobjlinetype, use object's line types.
- # useobjsectattrs, negation of *[ ] Override Object's Attributes* (1=use object's attributes, 0=nope, use overrides)
- #mat, 3D material index
- #matname, 3D material name

For doors and windows:

- #wido_sill, sill value as GDL global WIDO_SILL.
- #oside, 1 if mirrored to x axis, 0 otherwise.
- #refside, 1 if mirrored to x axis same as oSide, 0 otherwise.

For ArchiFrame elements:

Value	Description
"#af_elemdata"	Gives information of the plank related to the element. The table fields are: • x1, y1, z1, x2, y2, z2 Plank's base line coordinates relative to the element surface. • bx1, bx2, bz1, by1, by2, bz2 Plank's bounding box in parent elemen's coordinates. Y-axis is element's watching direction and is calculated ignoring any cuts in the plank unlike x- and y-coordinates. • poly, plank's polygon. • index Plank's 1-based index. • rotangle, rotation angle in degrees from element's watching direction • ptr, see af_request/plankinfo.

	If used from projection scripts (dimension collector or *DoProjSettings()*-function), ptr refers to projection plank's master/3D plank. Also, then following extra fields are available: • typetag, related element layer's type: core, intstud etc. Actually available always for any element related plank when working with element Lua-scripts.

Also, all values described <elemparam name="xxx">value</elemparam> are supported. Character # must be given before the element's parameter name, for example #cutfill. Attribute values in this case are returned as two values: numeric and attribute name. Exceptions:

- #materialrgb returns single string. It is the material's colour in format "RGB" where all color values are between 0…1. Decimal separator is dot, for example "0.5, 0, 0". If the material is GENERAL, the return value is "0".

Returns: The parameter numeric/string or nil = parameter missing.

24.2.4 ac_getattrinfo (strAttrGuid) or ac_getattrinfo (nTypeNum, nAttrIndex)

Gives information of an attribute. Parameter nTypeNum is one of API_AttrTypeID:

```
API_PenID = 1,
API_LayerID = 2,
API_LinetypeID = 3 and so on,
API_FilltypeID,
API_CompWallID,
API_MaterialID,
API_CityID,
API_LayerCombID,
API_ZoneCatID,
API_FontID,
API_ProfileID,
API_PenTableID,
API_DimStandID,
API_ModelViewOptionsID,
API_MEPSystemID,
API_OperationProfileID,
API_BuildingMaterialID,
API_MarkUpStyleID
```

Special value -1 means floor number and return values are: 1. String telling the floor name, 2. Number telling floor global level.

Returns: The table having fields

- type, one of following: "fill", "composite", "type_as_number".
- name, the attribute name.
- thickness, for composite the total thickness and for composite layers, the layer thickness.

  For composites only:
  - num_layers, number of layers.
  - Layers, table of layers each having similar fields as requested for fill, except thickness also set.

24.2.5 ac_objectopen(strGuid)

In special cases strGuid can be nil and the function will open the default object depending on the special usage case. Opens the object for further processing. The guid may also be a C pointer obtained from ArchiFrame. The C pointer start with @ and is followed by 8 (32 bit) or 16 (64 bit) characters. Use pointers with care – misuse will crash the program.

Parameter strGuid may begin with character >. For example, if used in plank observer script, it will work with last saved version skipping ArchiFrame's object cache buffer.

Returns: Nothing, in case of error, Lua error is raised.

24.2.6 ac_objectget (strParamName [, nRowNum, nColNum])

Works as ac_getobjparam but for object opened with call to ac_objectopen.

24.2.7 ac_objectset (strParamName, value [, nRowNum, nColNum])

Sets the parameter only if changed.

Returns: Nothing, in case of error, Lua error is raised.

24.2.8 ac_objectclose ()

Saves changes if any to opened object.

24.2.9 ac_environment (strSel [, additional parameters])

Selector can be one of the following:

24.2.10ac_environment ("lang")

Returns language of theArchicad: "eng", "fin" or "swe".

24.2.11ac_environment ("ntos", nVal, strType, strPrefs)

Converts number to string using current Archicad-settings. Parameter strType can be "length"| "angle" | "area" | "volume". strPrefs is either "work", "dim"or "calc" and it reflects to Archicad working, dimension lines and calculation settings. Parameter nVal is the numeric value. Returns string value.

24.2.12ac_environment ("units", strType, strPrefs)

Returns four values:

- Unit type as number
- Unit string
- Multiplication factor to convert from Archicad's internal format to user preferences, 0=no conversion, for example: 1' 2/32"
- How many decimals to use if numeric

Parameters are as "ntos" but no nVal parameter is not given. First return value numbers are:

```
APIUnit_Metric=0,
APIUnit_Centimetric,
APIUnit_Millimetric,
APIUnit_FootInch,
APIUnit_FootDecInch,
APIUnit_DecFoot,
APIUnit_Inch,
APIUnit_DecInch,
APIUnit_Decimetric,
```

```
APIUnit_DecYard,
APIUnit_Gallon
```

Please note that from AC22 on the values are (decimetre added so check the unit string for imperial check):

```
Meter,
Decimeter,
Centimeter,
Millimeter,
FootFracInch,
FootDecInch,
DecFoot,
FracInch,
DecInch
```

24.2.13ac_environment ("userorigin")

Returns X, Y, Z of the user origin.

ac_environment ("reflevels")
Returns two numeric reference levels defined in AC.

24.2.14ac_environment ("parsetext", strIn)

Parses texts having Archicad automatic texts like <ARCHITECT>. Returns parsed string.

24.2.15ac_environment ("createguid")

Creates new GUID in form {4CABE438-C7F5-4B43-A6DF-B07114D6BDDF}.

ac_environment ("strreplace", strSource, strFind, strReplace)
Does find & replace without any character being pattern.

24.2.16ac_environment ("layer", strName, nShow, nSetMask)

Shows (1) or hides (0) given layer(s). Name may contain wild characters * and ?, for example, “structural*”. If parameter nShow is -1, it will return the state of the layer (if wild card, the state of first matching layer). State is bit combination:

- 1, bit0, is visible: 0 = hidden, 1 = visible.
- 2, bit1, is locked.
- 4, bit2, is mine in TeamWork.
- 16, bit4, is forced to wireframe.
- 32, bit5, the layer belongs to an Xref.

Optional parameter nSetMask defines which bits to apply from nShow to the layer status bits.

Another mode is nShow with value -2. Then the given layer will be created as visible if it is not created already (then its status is not changed). Call this again to set the status of newly created layer if needed. Returns negative error code if creating the layer failed, otherwise positive layer index.

Returns numeric status of the edited layer

24.2.17ac_environment ("rebuild", nRegenerate)

Rebuilds display (for example after showing/hiding layers). With nRegenerate value 1 rebuild is done deeper.

24.2.18ac_environment ("tolog", strText)

Puts given text to a log window (not a dialog).

24.2.19ac_environment ("bittest", value, bitnum)

Returns value 0/1 for given bit in the value. Parameter bitnum must be between 0…31.

24.2.20ac_environment ("getsel")

Returns table of selected element's guids. Processes all element types. Return value nil means no selection.

24.2.21ac_environment ("setsel", tblGuids)

Sets selection to given guid tbl which must start indexing from [1]. Nil = remove selection.

24.2.22ac_environment ("suspendgroups", bSusped)

Sets suspending the groups to given bool value (false = nope, others = suspend). Returns previuous state of suspending: false = not suspended, true = yes.

24.2.23ac_environment ("getcurrwindow")

Returns type of current window:

- 1, floor plan (Google for text APIWind_FloorPlanID to get all values).
- 2, section.
- 3, detail.
- 4, 3D window.

24.2.24ac_environment ("getall", nFilterBits, nElemType1, nElemType2,…, nElemTypeN)

Returns table of all element's guids (never nil but table can be empty).

nFilterBits as in API-calls, for example APIFilt_OnVisLayer|APIFilt_OnActFloor is 2+4 = 6. -1 = default: APIFilt_OnVisLayer|APIFilt_OnActFloor if current window is floor plan; APIFilt_OnVisLayer if current layer is anything else. ArchiFrame sets global gnListingAll to value 1 if selected (*) whole model even if in floor plan.

24.2.25ac_environment ("filterelem", guid, nFilterBits)

Checks if given element fulfils the filter (for example is on visible layer which is number 2). Returns true or false.

24.2.26ac_environment ("filedialog", bSaveAs, strTitle, strFilter, strFileExt)

Runs open or save as file dialog if bSaveAs is true. Parameter strTitle is the title of the dialog box, strFilter is in format "Text files (*.txt)|*.txt|All (*.*)|*.*|", strFileExt is the default file extension without the period. Returns nil if cancelled.

ac_environment ("findattr", nAttrType, strAttrName)
Find given attribute by name, strAttrName may contain wildcards and multiple values may be given with vertical bar. Returns 0 if not found.

```
ac_objectset("iMc", ac_environment("findattr", 6, "*tre-osb*|Tre-
Mahogny"), add, 10)
```

nAttrType must be one of following (number increases by one for each line).

```
API_PenID = API_1,
```

```
API_LayerID,
API_LinetypeID,
API_FilltypeID,
API_CompWallID,
API_MaterialID,
API_CityID,
API_LayerCombID,
API_ZoneCatID,
API_FontID,
API_ProfileID,
API_PenTableID,
API_DimStandID,
API_ModelViewOptionsID,
API_MEPSystemID,
API_OperationProfileID,
API_GraphicOverrideID,
API_BuildingMaterialID,
```

24.2.27ac_environment ("luacomchars", nCodePage)

Sets LuaCOM-interface's global character set. Default is ANSI (0) and other possible value is UTF8 (65001). Returns previous setting that must be restored so that other sripts are not affected.

24.2.28ac_environment ("acver")

Returns current Archicad-version as number: 19=1900, 20=2000, 21=2100 etc.

24.2.29ac_environment ("solidsub", tblTargets, tblOperators)

Makes solid subtraction connection between all targets and operators that have intersecting bounding box. Both tables must be 1-based arrays.

24.2.30ac_environment ("scriptdir")

Returns two values:

1. Path to the script not including the last path character (C:\Folder for example or C: if root folder)
2. Current path sepator character, either "/" or "\"

24.2.31ac_environment ("mkdir", strPath)

Creates given folder. Returns true=succeeded, false=failed

24.2.32ac_environment ("setautotext", strKey, strVal)

Sets given autotext. Please note that strKey is given without <>. Causes an error if fails. Example: ac_environment("setautotext", " PROJECTNAME", "My Project")

24.2.33ac_environment ("gridnext", x, y, dirx, diry)

Calculates next grid position from given position in given direction and returns information about the grid. Returns four values:

- Status: 0=no suitable grid, 1=moved to grid but grid not parallel or in 90 deg angle to the direction vector, 2=grid was parallel to direction vector and the result is good.
- x, y: the coordinates of the calculated position
- Information of the grid as a table having fields:
    - x,y,z Grid origin
    - angle Grid angle in radians
    - mainx,mainy Spacing

24.2.34ac_msgbox (strText)

Shows message box. Useful for debugging.

24.2.35 ac_mbstoutf8 (strIn)

Converts given text encoding into UTF8. Returns converted string.

24.2.36ac_utf8tombs(strIn, codepage)

Converts given utf8 text into requested codepage. Parameter codepage must be ne of the following numbers:

```
CC_Default              = 0,
CC_System               = 1,
CC_User                 = 2,
CC_Application          = 3,
CC_Legacy               = 4,

CC_WestEuropean         = 11,
CC_EastEuropean         = 12,
CC_Baltic               = 13,
CC_Cyrillic             = 14,
CC_Greek                = 15,
CC_Turkish              = 16,

CC_Hebrew               = 21,
CC_Arabic               = 22,

CC_Thai                 = 31,
CC_Japanese             = 32,
CC_TradChinese          = 33,
CC_SimpChinese          = 34,
CC_Korean               = 35,

CC_Symbol               = 40
```

Returns converted string.

24.2.37 ac_geo (strSel [, additional parameters])

Selector can be one of following:

ac_geo ("linex", x1first, y1first, x2first, y2first, x1second, y1second, x2second, y2second)
Gives intersection point of two lines. Lines are infinite length. Returns: No value = no intersection pt, (X, Y) = the intersection point.

ac_geo ("linedist", x1, y1, x2, y2, x, y)
Calculates point distance from line. Return three values: Distance from the line (negative to the left and positive at right hand side), distance from beginning point towards end (negative before line starting point) and length of given line.

ac_geo ("calctranworld3", origfrom, vecxfrom, vecyfrom, veczfrom, origto, vecxto, vecyto, veczto)
Calculates transformation matrix [1...12] for given coordinate worlds. Returns table [12]. To transform a point, use:

Xnew = x*tblTran [1]+ y*tblTran [2]+ z*tblTran [3]+tblTran [4].
Ynew = x*tblTran [5]+ y*tblTran [6]+ z*tblTran [7]+tblTran [8].
Znew = x*tblTran [9]+ y*tblTran [10]+ z*tblTran [11]+tblTran [12].

Due to historical reasons, the offset values at [4], [8] and [12] work only with global coordinate system. Please find function MatInitTran from default ArchiFrameElements.xml to see how to transform from any world to any other world.

ac_geo ("matmul", mat1, mat2)
Multiplies matrix1 with matrix2. Matrices are in format calctranworld3 uses, col 4 is just added.

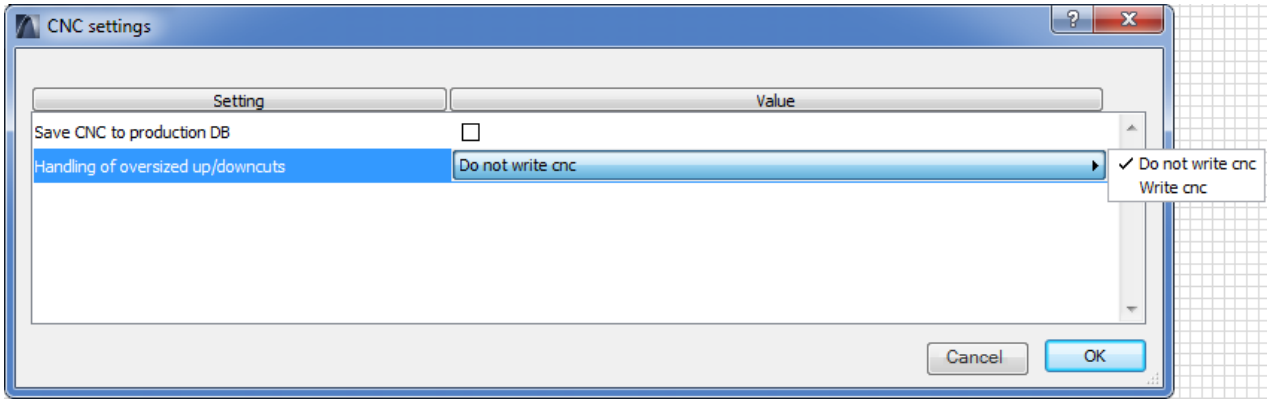
ac_geo ("linepolyx", x1, y1, x2, y2, tblPolys, [extendLine=false, onEdgeInside=true])
Calculates intersection of given line and table of polygons (format as in af_request ("getpoly",…)). TblPolys is 1-based table of separate polygons (to be able to handle split surfaces).

If the two last parameters are missing, the shown default values are used.

Returns 1-based table of line segments each segment having fields x1, y1, x2, y2. Nil if there is no intersection.

24.2.38 ac_optiondlg (strID, strTitle, tblSettings)

Runs setting dialog and saves settings application a specific way – usually to user specific configuration file and to the current pln-file. Dialog looks like:



And code to run this is:

```
gbAskOptions=true
gbCnc-ToDb=true

-- CNC- writing settings, raises error if canceled
function GetCnc-Options()
            local       tblSettings, bRes, sErr, s, n

            -- Ask only once
            if gbAskOptions==false then
                        return true
            end

            tblSettings={}
            tblSettings              [1]         ={}
            tblSettings[1].cfgonly   =1
            tblSettings  [1].type    =2
            tblSettings[1].prompt    ="Save CNC- to production DB"
            tblSettings  [1].key     ="cnc-db"
            tblSettings[1].defvalue  =0

            tblSettings              [2]         ={}
            tblSettings[2].cfgonly   =1
            tblSettings  [2].type    =1
            tblSettings[2].prompt    ="Handling of oversized up/downcuts"
```

```
        tblSettings  [2].key      ="cnc-cut"
        tblSettings[2].valuelist="\"1:Do not write cnc-\",\"2:Write cnc-\""
        tblSettings[2].defvalue  =1

        bRes,sErr=ac_optiondlg("LDCN", "CNC- settings", tblSettings)
        if not bRes then
                if sErr~=nil then
                        RaiseError(sErr)
                else
                        RaiseError("CANCELLED")
                end
        else
                if tblSettings[1].value==0 then
                        gbCnc-ToDb=false
                else
                        gbCnc-ToDb=true
                end
        end

        gbAskOptions=false
end
```

Parameters:

Param	Description
stride	Four letter long ID of the settings. Always start ID with LD not to mix it with application's own settings. For example LDCN.
strTitle	Dialog title, set to nil if you just want to get the saved settings into the table tblSettings. No dialog is shown in this case.
tblSettings	Table of the settings items.

Return values: Boolean bRes, string strErr: bRes is true if dialog was run ok and closed with OK, false in case of internal error or dialog cancelled. In case of internal error strErr is set.

Settings table has fields:

Field	Description
cfgonly	Optional number: • 0 or missing, save option to cfg and pln-file. • 1, save setting only to cfg-file (global to user).
type	Numeric type: • 1, value list/combo box. • 2, check box. • 3, length edit. • 4, any text.
prompt	To setting-column in the dialog.
key	Unique key for the setting inside this setting dialog.
defvalue	Default value to be used if no previously saved settings.
value	If given for input: Value is not loaded from settings but instead this value is used. On output: The actual setting.
valuelist	Given if type is 1. Contains items in format: "Number: Item text". Number is used as unique ID for the item. Items are separated with comma. Note Lua syntax for adding quotation marks in the string: "\"1:Do not write cnc-\", \"2:Write cnc-\"".

24.2 ArchiFrame-specific functions

24.2.1 af_request (strSelector [, additional parameters])

Parameter strSelector defines what is requested:

Value	Description
"singlemat"	TblMat = af_request ("singlemat", "id"), returns table for single material or nil if material does not exist. Fields are: • id, material id. • name, name. • shape, "block" | "plane" | "round". • thickness, height, in meters. • height, in meters. • m3factor, volume factor. • xmlutf8, full xml-definition of the material. Used for example in quantity takeoffs to read extra attributes from ArchiFrameBlocks.xml
"matlist"	Returns table of all materials in material list as Lua hash-table (not 1-based indexed table). They key is the material ID and data is what singlemat gives plus field index telling the 1-based index in the order the materials are defined in ArchiFrameBlocks.xml.
"plankinfo"	tblRes = af_request ("plankinfo", strPlankGuid, plankInfoProj, tblSettings), returns information about given plank. If strPlankGuid is missing, uses currently opened object/plank. If plankInfoProj is given, it contains projection information. tblSettings is optional settings. More info later. It is possible to get geometry information for element objects (in that case returns the framing geometry taking possible rotframing="x" attribute into account) or even Archicad beams, walls, doors, windows and columns too by giving strPlankGuid. Table has fields: • ptr, plank's or board's internal C-pointer (memory address) starting with character @ and followed by 8 or 16 digit hex address. Can be used as plank identifier in plank operations and in ac_objectopen. Ptr is valid only during current operation – it may not be saved for later use and misusing ptr will cause whole program to crash. • guid, plank's guid (nil for new pieces not yet created into Archicad). • type, 1 = plank, 2 = log object, 4 = board. • begc, start point in model coordinates beg. X, Y, Z. • endc, end point in model coordinates end. X, Y, Z. • len, plank length. • width, width. • height, height. • typename, plank/board type name for listings. • matname, the name="xxx" from ArchiFrameBlocks.xml • vecx, vecy, vecz, plank's geometry axes.

	• masterelemguid, guid of the floor plan/3D piece if current part is projection piece
	• ownerelemguid, if belonging to an element, the owning element's guid. Missing if not part of an element. You can use for example "elem_quantities" with this guid.
	• ownerelemid, if belonging to an element, the owning element ID string. Missing if not part of an element.
	• del, true = marked for delete, false = nope.
	• begoff, for log object cnc-writers: Offset from X-coordinate zero point to actual plank start. Positive extends, negative shortens (opposite to plank dir).
	• tblWeather, set if the plank is part of weather boards. Has fields:
	  ○ wbguid, GUID of the weatherboard group, actually GUID of the plank holding the whole group's internal data.
	  ○ pa, pb, pc, pd, plane equation – pa, pb, pc is the normal vector towards the wall.
	  ○ vecy, vertical vector X, Y, Z.
	  ○ vecx, horizontal vector always pointing right looked to the surface from normal vector's opposite direction.
	• tblSides, table of geometry for each side [1]…[6] each having fields:
	  ○ origc, 3D origin.
	  ○ vecx, vecy, Direction vectors for surface axes.
	  ○ vecz, normal vector pointing outside of the piece.
	  ○ dx, dy, surface size in X- and Y-directions.
	• idgenerated, if plank/board has parameter "ID generation string" set, this is the visible ID as parsed in the object.
	
	Quantities if requested:
	• kgpermeter, for planks only. Use with len. Only if tblSettings.quant>=1
	• kgperliter, for boards only. Use with net area or board's gross area. Only if tblSettings.quant>=1
	• acvol, the real GDL-object's volume. Only if tblSettings.quant>=2 and succeeded. Slow to calculate.
	
	If plankInfoProj is given, the plank is projected to given plane. Result is saved to field elemdata and has the same members as with #af_elemdata. Parameter plankInfoProj must contain fields (plankinfo queried for an element is fine):
	• begc, plane's origin.
	• vecx, vecy, vecz, 3D axes
	
	Optional tblSettings may contain following fields:
	
	• idsep Given to override object's parameter iShowIDSep. Useful in cut lists to get real projection visible ID.
	• quant If set to non-zero, ArchiFrame gives quantity information

	Returns nil if failed.

"getpoly"	tblRes = af_request ("getpoly", tblSettings, strPlankGuid), returns polygon information for given strGuid elem (plank, board, element). If strPlankGuid is missing, uses currently opened object/plank. If the given strPlankGuid refers to an ArchiFrameElement-object, this returns the framing polygon taking possible rotframing="x" attribute into account.
	
	tblSettings for boards may be nil. If given, it may contain following fields:
	• side, 1 = top, 2 = front, 3 = bottom, 4 = back. Which side to process.
	• tblguids, instead of processing single part's polygon, unify polygons of all given elements. Useful for example to join exploded cladding pieces.
	• polynum, if surface is split by groove(s), there may be multiple polygons. Number of the polygon to return 1…N. If missing 1st one will be returned.
	• tooldia, milling tool diameter for overbegin and overendin, default = 10 mm.
	• holes, value:
	  ○ Missing, give full polygon with holes.
	  ○ 0, throw holes away.
	  ○ 1, will return list of holes (openings) in the element. If given result field poly is 1-based array of the holes polygons.
	• expand, if given, the value to expand the polygons, positive gives bigger polygons (also if holes requested), negative smaller.
	• fromboard, with value 1 gives polygon list from the boards belonging to the element (same format as for holes). May contain many polygons if surface is split. In owning element's rotated coordinate world.
	With value 2 the polygon extra info is taken from board object's parameter iPolygon. With value 2 parameter strPlankGuid must be a board guid. Result fields overbegin, overendin and cutfromobj come from object instead of calculated values. Use value 2 only if board object's parameter iSawlines is nonzero. NOTE! If holes are requested, they will be returned clockwise instead of standard ccw for contour line. Value 2 implies also givelist = 1.
	• givelist, with value 1 returns always [1]-based polygon list. Otherwise single polygon unless holes or fromboard is given.
	• calcopening, with value 1 calculates openingtype-field using the related element object's links to Archicad-walls/openings.
	• camera, if given contains projection plane as given from "plankinfo"-request:
	  ○ begc, plane's origin.
	  ○ vecx, vecy, vecz, 3D axes.
	
	Returns table having following fields or nil = not supported type:
	• numpolys, number of polygons (in case of split surface).

	• x1, y1, x2, y2, the bounding box of the polygon. Currently polygon is always normalized to have x1, y1 at 0, 0. Not valid if holes or expand given.
	• xoff, yoff, currently always 0, 0. The offset from the lower left corner of the raw board material. Not valid if holes or expand given.
	• area, area of the polygon(s) with holes reduced.
	• poly, if single polygon: [1]-based array of polygon points from the front surface. If givelist was specified, is 1-based array to single polygon's point array. Contour end points are not duplicated to be same as contour begin. Each point has members:
	  ○ x,y, the point position.
	  ○ isbeghole, set only if point is begin of a hole.
	  ○ isendcontour, set only if point is end of a contour.
	  ○ isendpolygon, set only if point is last point of the polygon, always with isendcontour.
	  ○ overbegin, length of line inside the polygon extended by tooldia from next point to current point. Use to check for any overcut in cnc.
	  ○ overendin, length of line inside the polygon extended by tooldia from previous point to current point. Use to check for any overcut in cnc.
	  ○ cutfromobj, if board object had parameter iSawlines set, this tells if the cut from current point to next point is set by user: 0 = no cut, 1 = do the cut.
	  ○ openingtype, if edge from current pt to next pt is related to an opening (set only for element objects):
	  ▪ 0, not related to an opening or related to an empty opening.
	  ▪ 11 window top.
	  ▪ 12 window right.
	  ▪ 13 window bottom.
	  ▪ 14 window left.
	  ▪ 15 window unknown.
	  ▪ 21-25 door similar.
	  ▪ 111-115 empty window similar.
	  ▪ 121-125 empty door similar.
	• err, error message if there was an error in extend operation. nil =ok.

"delplank"	Marks currently opened object/ac_objectopen() to be deleted. Deletes all related projections etc.

"getselplanks"	Returns guid table of selected plank objects. If selection contains projections, its master 3D plank will be returned instead. For historical reasons indexing of this starts from [0] and you need to use Lua pairs instead of ipairs:
	`for i,v in pairs(tblSel) do`
	`  xxx`
	`end`

	To get only boards, give second parameter “board”. With any other value, only planks are collected. tblBoards = af_request ("getselplanks", "board") All planks&boards having parameter’s *Special type* value 1 (part of combined element source) are skipped.

"getsellisting"	Returns table to be used as source elements for listing/cnc-. If anything selected, the lists are limited to the selection. If no selection, the result contains:
	• All visible elements in current floor if in floor plan.
	• All visible elements in the model if in 3D.
	
	In both cases all planks&boards having parameter’s *Special type* value 1 (part of combined element source) are skipped.
	
	Returns 1-based array of tables containing fields:
	• guid
	• ptr
	• id
	• type, element type as:
	  ○ 1 = plank.
	  ○ 2 = log.
	  ○ 4 = board.
	If no elements, returns nil.

"afclick"	t=af_request("afclick", tblSettings)
	Asks user to click a plank or board. Parameter tblSettings has fields:
	• line, true will cause ArchiFrame to ask for two points
	• prompt, the prompt shown at bottom of Archicad-window while waiting the user to click in the model.
	• elems, elements where user can click. Typically selection.
	
	Returns table having fields:
	• begc, first clicked point
	• endc, if line was requested
	• elems, table of clicked elements each having fields
	  ○ guid, guid of the clicked plank/board
	  ○ side1, 1-based side number: 1=top, 2=front, 3=bottom, 4=back, 5=begin, 6=end
	  ○ x1,y1: First clicked point on surface coordinates
	  ○ x2,y2: Set if line was requested
	
	Please see data-folder/ArchiFrameBlocks.xml part <mctool name="Element lifting drillings floor" for an example.

"autocut"	t=af_request("autocut ", tblSettings) Automatically cuts planks with given settings. Parameter tblSettings may have these fields:

	• elemguid, required guid of the element layer to process. ArchiFrame automatically finds the supporting layer under the target. • minlen, piece's minimum length (default is 500 mm) • maxlen, maximum length for a piece, 0=use the maxlen defined for the plank type (default is 0) • minoverlap, minimum distance of cuts between adjacent studs (default is 600 mm) • overlapruledist, maximum distance where minoverlap has effect. With default value 1200 mm, a cut 1200 mm or more from target will not have any effect Does not return any value. The error message is dumped to the log window in case of error.
"flushmc"	Flushes ArchiFrame's internal machining table to the object's parameters. Must be used after calling "adjustbegend" and similar for temporary plank before closing the plank to get all machinings offset correctly.
"getbegendplane"	TblPlane = af_request ("getbegendplane", "beg", nOptions). Returns global cutting plane for current plank's begin or end (replace "beg" with "end"). Parameter tblPlane has fields pa, pb, pc, pd defining the plane equation. With numeric values any side can be queried, replace "beg" with values (see About the planks): • "1", top. • "2", front. • "3", bottom. • "4", back. • "5", begin. • "6", end. If parameter nOptions is missing or is 0, first cut for begin or end will be returned. With value 1, the result is always straight relative to the plank.
"adjustbegend"	af_request ("adjustbegend", "beg", tblPlane). Adjusts currently opened plank's end ("beg" or "end") to given global plane. Parameter tblPlane has fields pa, pb, pc, pd defining the plane equation.
"cutplank"	af_request ("cutplank", tblPlane). Splits currently opened plank into two pieces. Parameter tblPlane defines the cutting plane in global coordinates. Returns values: • None → no valid cut – plank untouched. • strPtr → Return value is the guid/ptr string of the new plank.
"joinplanks"	af_request ("joinplanks", strGuid1, strGuid2). Joins given planks together. The requirements to join are: • Planks must be parallel. Returns: true = ok and strGuid2 marked to be deleted at the end of the undo-scope, nil = nothing done.

"adjusttoside"	af_request ("adjusttoside", tblSettings, strGuidTarget, tblGuidOperators). Adjust to side operation as with adjust to side tool palette. Parameter tblSettings may have fields.
	• doadjust, 0 = no adjusting, 1 = adjust (default).
	• gap, gap in the joint, default = 0.
	• cutop, how to cut:
	  ○ 0 = just cut (default).
	  ○ 1 = remove upwards.
	  ○ 2 = remove downwards.
	  ○ 3 = keep the part closer to set keep origin.
	• endshape, the end shape:
	  ○ 0 = angled, extend and keep existing cuts (default).
	  ○ 1 = as 0 but do not extend.
	  ○ 2 = as 0 but delete any existing cuts.
	  ○ 3 = straight not intersecting with operator piece delete existing cuts.
	  ○ 4 = straight extend to touch operator plane fully delete existing cuts.
	  ○ 5 = as 4 but do not delete existing cuts.
	• planepa, planepb, planepc, planepd, use this plane instead of operators.
	• keeporig, table xyz of origin to define which part to keep.
	• cuttoendplane, adjust to last intersecting plane instead of the first: 0 = no (default), 1 = yes.
	• expandopfind, if given, the operator is expanded with given value to test if there is an intersection between the planks. Useful to handle cases where angled planks touch just with a corner.
	• skipopsides, bit mask to skip checking operator sides. bit0 = top, bit1 = front etc. Bit 1 means skip checking this operator side. Default = 0, do not skip any side.
	• domark, 0 = no marking (default), 1 = markings.
	• marklines, 0 = to sides, 1 = to center.
	• marktext, text for marking, tags as in the tool palette.
	• markposx, text position 0 = left, 1 = center, 2 = right.
	• markposy, text position 0 = bottom, 1 = center, 2 = top.
	• markfontsize, in centimeters, 0 = default.
	• markdir, text in line's direction.
	
	Returns: nil = nothing done, tblNewPlanks = table of edited/new planks from the operation.

"editplank"	af_request ("editplank", tblEdit) Edits single plank, tblEdit has following fields (give only the fields to edit):

	• createtemp, if set to value 1 (please note numeric, not bool), the original plank remains as it is and ArchiFrame creates temporary item which is deleted when current operation ends. • guid, must be given. • mat, plank material id. • elemdata, geometry in element's coordinate world. If begin or end is edited, give all three XYZ-coordinates. • swapdir, if set to value 1 (please note numeric, not bool) current plank will be swapped Returns ptr in form @1234567812345678 to the original or temporary item.
"boardgap"	af_request ("boardgap", tblGuids, tblSettings) Adds specified gap between board edges. Parameters: • tblGuids, 1-based table of boards to process • tblSettings, settings used in the process. Fields: ○ gap, gap to leave between touching edges between the boards
"cmpplanks"	af_request ("cmpplanks", strGuid1, strGuid2). Compares two planks (not comparing the IDs). Comparison can be used for example in listings to make sure that planks having similar ID are similar. Return value: • nil, could not compare – only planks and boards are supported. • <0, item1 < item2. • 0, item1 = item2. • >0, item1 > item2.
"plankx"	af_request ("plankx", strGuidTarget, sideTarget, strGuidCheck) Checks if there is intersection between two planks. Parameters: • strGuidTarget, target plank • sideTarget, intersection bound box is calculated to this side 1-6 • strGuidCheck, the other part of the intersection Returns table with fields: • poly, nil if no intersection to sideTarget. Otherwise has bounding box with fields bx1, by1, bx2, by2 in target side's coordinates • polyclip, as previous but clipped to the actual surface ignoring any machinings in the plank. • plankinfox, target plank shrunk to cover the common area, nil=no intersection.
"mc_getenv"	af_request ("mc_getenv") Retrurns environment information for cnc-writers in a table having fields from ArchiFrameBlocks.xml / cnclimits: • gromaxdepth • nailmindistplank • nailmindistboard • dtangle in degrees

"mc_getnewmcid"	af_request ("mc_getnewmcid") Generates new machining id for the plank opened by ac_objectopen().
"mc_findtextpos"	af_request ("mc_findtextpos", nSide14, dX1, dTextWidthMeter, dFontHeightMeters) Calculates position for plank ID that is as close as possible to dX1. If parameter dFontHeightMeters is given, its value is used instead of the text size in the ArchiFrame marking. Moves the text away from any existing markings and grooves. Returns the best calculated position.
"mc_explodepanel"	af_request ("mc_explodepanel", guidBoardElem, tblSettings). Explodes given *ArchiFrameBoard*-object into separate temporary planks that will not be created into the model (to be used for listings and cnc-). • guidBoardElem, guid of the ArchiFrameBoard-object. • tblSettings, currently not supported. Returns 1-based table of plank @pointers. Nil = no planks created.
"mc_nailings"	af_request("mc_nailings", tblTargets, tblGuidsBehind, tblSettings) Makes nailings to given planks or boards. Parameter tblSettings may contain following fields: • spacing_edge, spacing at board edges. Default = 100 mm. • spacing_mid, spacing at middle of board. Default = 200 mm. • mindist_board, minimum distance from the board edge. Default = 10 mm. • maxdist_board, maximum distance from the board edge. Default = 25 mm. • mindist_plank, minimum distance from the plank edge. Default = 10 mm. • mindist_nails, minimum distance between nails in planks. Default = 25 mm. • spacing_plank_cross, spacing for crossing planks, default = 50 mm. • spacing_plank_parallel, spacing for parallel planks, default = 200 mm.
"mc_sawcuts"	af_request ("mc_sawcuts", guidElem, bViewBack). Finds all placed *ArchiFrameSawCut*-objects that are over any projection of boards owned by given element using default orientation having saw blade on the right side. Parameters: • guidElem, GUID of the element whose saw cuts are returned. • bViewBack, false = View from front (default), true = from back. Returns 1-based table of saw cuts ordered by lineindex (main contour first) each item having fields: • x1,y1,x2,y2, coordinates on given element's coordinate system looked from front. • angledeg, saw blade angle looking in saw line direction positive tilts to left and neg to right. • depth, cut depth, 0 = through the layer. • overbeg, 0 = no overcut at begin, 1 = overcut at begin.

	• overend, as previous for the end. • bladeside, -1 = left from the line, 0 = center, 1 = right. • elemguid, saw blade element guid. • ishole, is it for polygon's hole 0/1. • lineindex, order number inside polygon, holes indexes start from 1000000, if polygon is split, the gap between each is 1000. Successive numbers combine single contour line (for polygon contour or poygon hole).
"mc_hatches"	af_request ("mc_hatches", guidElem, bViewBack, strLayer). Finds all Archicad hatches from given layer that are over any projection owned by given element. Parameters: • guidElem, GUID of the element whose saw cuts are returned. • bViewBack, false = View from front, true = from back and swap coordinates. • strLayer, layer name where to collect the hatches, empty means all layers Returns 1-based table of hatches each item having fields: • elemguid, saw blade element guid. • poly, hatch polygon
"mc_getacc"	af_request ("mc_getacc", guidElem) Finds all placed *ArchiFrameAccessory* and *ArchiFrameElMarkers*-objects that are inside any projection of given element. Scans all projections and related objects. In future may return more library part. Please check ac_elemget.header.libGuid to process the correct library parts and use ac_environment ("filterelem") to get only visible ones if needed. Parameters: • guidElem, GUID of the element Returns 1-based table of resulting guids or nil=no objects related to the element.
"mc_getcutinfo"	af_request ("mc_getcutinfo", tblInfo) Calculates information if a cut projected to one of the plank's sides. Parameters are given in a table which has fields: • guid, required GUID of the element that is processed • mcindex, required 1-based index of the machining to process • cutlineside, optional 1-based side number where to anchor the cut Returns a table having fields (btl-fields refer to btl cut code 010), angles are in radians: • fullcut, does the cut go cut whole plank: 0=no, 1=yes • side1, 1-based best surface to anchor the cut • x1, y1, x2, y2, cur plane's line on the surface • btl_p06, angle between cut edge and reference edge • btl_p07, inclination between face and reference side • btl_p08, lengthwise angle at cut bottom • btl_p11, depth of the cut

	• btl_p12, Length of the cut (from x1,y1 → x2,y2)
"aflang"	Language selected into ArchiFrame: eng/fin/swe/nor/ger/ita.
"afpaths"	Gives current settings paths ending to path separator character (\ in Win and / in Mac) as multiple return values: 1. Current path of the add-on folder 2. Current data folder 3. Current user specific data folder
"dim_section"	af_request("dim_section", strLayer, tblSettings) Available only in dimension line scripts. Parameters: • strLayer, comma separated list of layers to include. Wild card characters *? are accepted. • tblSettings, if given may contain fields: ○ skipbottom, value 1 will remove bottom_force planks from the resulting polygon list if there is at least one 2ndbottom_force plank in the element. Builds table of section polygon bound boxes. Section is made unifying the ArchiFrameElement-objects and the related planks/boards. Each returned table cell has these fields: • x1, y1, x2, y2 Polygon bounding box • ?_minx, ?_miny, ?_maxx, ?_maxy The closest and furthest polygon point polygon's four edges. Replace ? with left/right/top/bottom, for example point(right_minx,right_miny) is the lowest point on the right hand side of the polygon. If there is just single point at the edge, min- and max-points are the same.
"dim_camera"	af_request("dim_camera") Available only in dimension line scripts. Gives current projection camera. Returns table having fields: • begc: plane's origin • vecx, vecy: plane's axes • vecz: plane's normal vector, camera's watching direction • pa, pb, pc, pd: plane'd equation
"elem_automodules"	af_request("elem_automodules ", tblSettings) Groups opening related framing parts with CNC-Module ID. Without any settings each opening gets own ID and IDs start from A. Parameter tblSettings can be nil and if not, it may contain these fields: • id: Starting ID, default="A" • sameid: if given and value is true, all modules will have the same ID

When processing elements, there are these additional requests available:

Value	Description
"elemcorethickness"	Returns the element core thickness.

“elem_getpolyscount”	Returns number of polygons making up the element. Currently always 1.
“elem_getpoly”	af_request ("elem_getpoly ", polyNum). polyNum is currently unused and should be 1. Creates Lua globals that are available at initial create planks phase. Please see Lua-scripts with elements.
“elem_getinfo”	Returns table containing information about current element. Table fields are: • vecx, lengthwise axis. For vertical elements, this has zero z. • vecy, for vertical elements this is 0,0,1. • vecz, the watching direction for the element.
“elem_createplanks”	Issue when script has set up information for new planks. Please see Lua-scripts with elements.
“elem_getstudmat”	Gives ID of random (actually first one sorted by plank’s GUID) vertical plank in the element. If no planks yet, gives ID of first material definition of the element.
“elem_marknoplanks”	af_request ("elem_marknoplanks", x1, y1, x2, y2, nDelExisting), marks give rectangle in the element not to get any planks from spacing rules. • nDelExisting, optional parameter to delete existing planks from given polygon. With value 1 deletes plank intersecting the polygon.

"elem_quantities"	af_request ("elem_quantities", strElemGuid, nOpenLayers [, tblSettings])
	Gives quantities for single element or layered element. Parameters:
	• strElemGuid, nil = use the parent element given from environment, str = guid.
	• nOpenLayers, 0 = Open just given single element, 1 = Open every layer.
	• tblSettings, if given may contain fields:
	  ○ external If present and true, will add also external planks
	  ○ quant Level of quantity calculation to speed up operation if quantities not needed, 0=minimal, 100=all (default). Additional flags to be added to base value 0 or 100:
	  ▪ 256, return openinginfo
	
	Returns table containing information for separate layers and total quantities.
	• tblelems, the elements.
	• quant, sum of quantities for each layer.
	• geo, origin of master element and 3D bounding box. Please see end of this section for description. Not taking rotframing="x" attribute into account.
	• geoframing, as previous but takes possible rotframing="x" attribute into account. Bounding box fields are not set here.
	• comptype, type name of the composite element type, nil = could not get it.

	• templateid, for own element types the *based on* type id, nil = not own element type.
	• xmlutf8, in this level contains composite xml referencing layers and having settings for the whole composite.
	• guidstamp, guid of the related element stamp object, nil if no stamp
	• tblguidcutlist, 1-based table of the related cut lists, nil if no cut lists
	• corners, the corner types for each side or nil=not defined. Fields corners.left, corners.right, corners.bottom and corners.top
	• offedges, edge offsets for the element. Please see next item.
	• offopenings, opening offsets for the element. Fields:
	  ○ id, the preset id. If not matching to existing presets, this is empty string "".
	  ○ layers, 1-based array of tables each having fields:
	  ▪ layertype, finish_ext etc..
	  ▪ layerindex, negative before core, 0=core, positive after core
	  ▪ left, right, top, bottom, the offsets for each side having fields:
	  • is, is it checked (numeric, not Boolean)
	  • off, the offset value positive extending it
	• openinginfo, filled only if requested. Gives information of all Archicad doors and windows related to the core layer or if the given layer if opened just one layer. Fields:
	  ○ guid, the Archicad door/window guid
	  ○ id, the Archicad door/window ID
	  ○ libpart, the related Archicad library part name
	  ○ x1,x2,y1,y2, the bounding box of the opening projected to the core/given layer.
	
	Fields for single element:
	• guid, element's guid.
	• id, element's id.
	• elemtypeid, for example "WALL 173 K600".
	• elemtype, for example "core", "intstud", "boarding_int" etc from type="xxx" xml-attribute.
	• templateid, for own element types the *based on* type id, nil=not own element type.
	• xmlutf8, element's xml-definition in utf8-characters.
	• areanet, element area calculating openings in m2.
	• areagross, openings not reduced in m2.
	• layertypenum:
	  ○ 0 = planks.
	  ○ 1 = boards.
	  ○ 2 = paneling (planks belonging to this layer are paneling/cladding).

	• tblplanks, table of planks belonging to this layer containing fields:
	  ○ guid, the plank guid.
	  ○ ptr, temporary pointer to the plank.
	  ○ external, true=is external plank (only if requested), false/nil=owned by parent element
	
	• tblboards, table of boards containing fields (nil if no boards):
	  ○ guid, the board element.
	  ○ ptr, temporary pointer to the board.
	  ○ id, the board id.
	  ○ thickness, its thickness.
	  ○ width, height, the size of the board.
	  ○ ispanel, 0 = no, 1 = board contains panel/cladding.
	
	The quantity fields for single element or combined for multilayer element as calculated in weight tools:
	• elemvolume, full volume from polygon area * thickness.
	• elemvolumereduced, the planks reduced from element volume.
	• elemweight, total element weight elemweightreduced + plankweight + boardpanelweight plus related openings weight.
	• elemweightreduced, insulation weight. Doors and windows are also calculated if set so in the weight settings for the layer/composite.
	• plankvolume.
	• plankweight.
	• boardpanelvolume.
	• boardpanelweight.
	• openingweight, combined weight of related openings.
	
	The geometry information from geo-table:
	• orig.x, orig.y, orig.z, origin of the core-element. Compared to "plankinfo" origin is at element's front instead of back.
	• vecx, vecy, vecz, direction vectors of the element, vecz is the front projection watching direction. Compared to "plankinfo" vecy and vecz are swapped.
	• x1, x2, y1,y2, z1, z2, minimum and maximum coordinates of any plank or board in element's axes (see Element object). Please note that these are nil if there are no planks or boards related to the element.
	• len, as in "plankinfo", polygon xMax-xMin
	• height, as in "plankinfo", polygon yMax-yMin
	• width, as in "plankinfo", element's thickness

"elem_domarkings"	af_request ("elem_domarkings", strXml). Like the xml-definition, may refer to settings inside <*presettings*>-tag: <markings ref = "mark_big"/>.

24.2.1.1 Element layers

af_request ("elem_openparent", guidElem).

GuidElem can be nil to get current element.

Resulting table contains fields:

Field	Description
id	ID of element type from ArchiFrameElements.xml: • <elemtype class="wall" **id="WALL 173+42 VERT"** idstamp="173+42">.
guid	Guid of the core layer/single layer element object.
ptr	Another temporary ID for the master element.
thickness	Total thickness of the layered elements including any empty space between elements.
tblelems	Table of individual elements.

Tblelems is indexed by *1*-based index in the same order as defined in *ArchiFrameElements.xml*. The table contains these fields:

Field	Description
id	ID of the referenced element or same as parent element if single layer element: • <layer **ref="WALL 173 K600"** anchorname1="Core ext"…
type	Type of the element layer (core for single layer elements): <layer ref="WALL 42x42 VERT" … **type="intstud"**>.
guid	Guid of the element object.
ptr	Another temporary ID for the layer element.
zoff	Offset in watching direction to first surface.
xsize	Maximum width of the element. Lower left corner has always coordinates 0, 0. For walls this is the element length in floor plan.
ysize	Maximum height of the element. For walls this is the height.
thickness	Layer's thickness, logically zsize.
x1off	Offset from core layer's left-hand side: positive extends over core.
x2off	Offset from core layer's right-hand side: positive extends over core.
y1off	Offset from core layer's lower side: positive extends over core.
y2off	Offset from core layer's upper side: positive extends over core.

af_request ("elem_closeparent", nSave).
Closes the current multilayer element.

Parameter	Description
nSave	1 = Save changes, with any other value all changes are discarded.

af_request ("elem_openlayer", guidElem), af_request ("elem_closelayer").
Opens the given layer to edit planks and element polygon shape. Close does not save the changes, it just clears internal information. The changes are finally saved with elem_closeparent.

af_request ("elem_moveside", nCorner, offset).
Moves side of an element layer.

Parameter	Description
nCorner	1 = left, 2 = right, 3 = bottom, 4 = top.
offset	Positive extends, negative makes smaller.

24.2.1.2 Element relations

af_request (“elem_getrelations”, guidElem [, xtolerance}).
Gets the relations for single element layer. GuidElem can be nil to process current element layer.

Parameter	Description
GuidElem	Element layer, nil = use current layer.
xtolerance	How far from the intersection the element’s end may be, default = 1 mm. Use if core is shortened over that in the corner type.

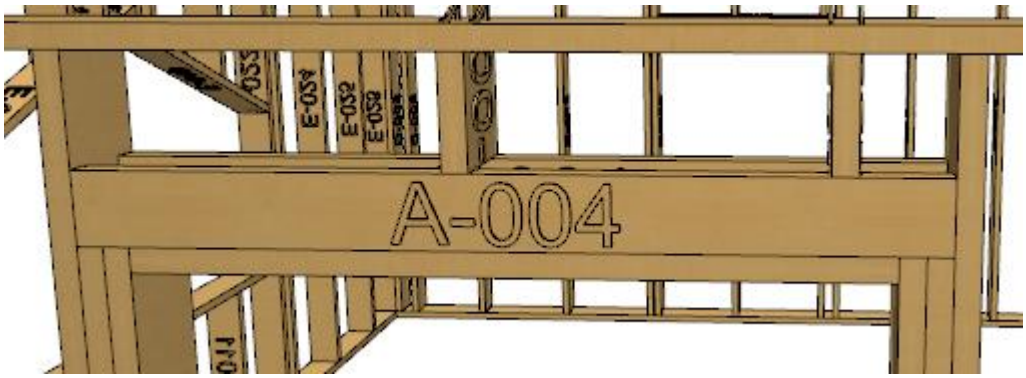
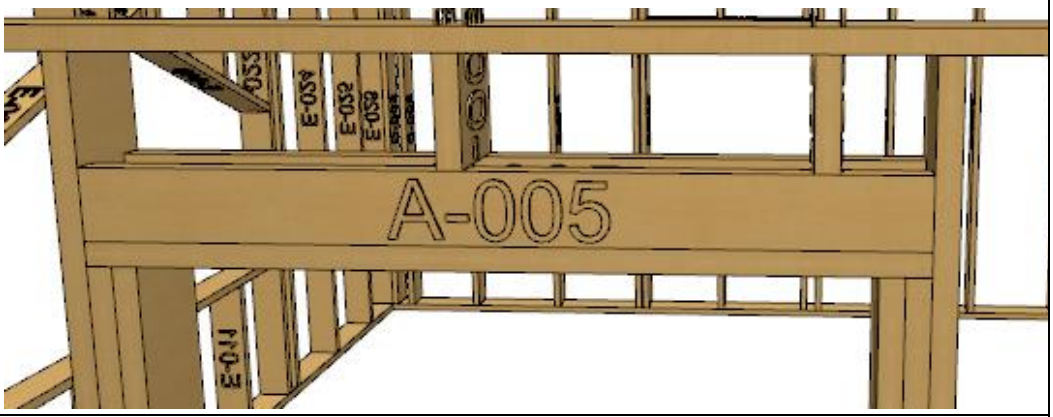
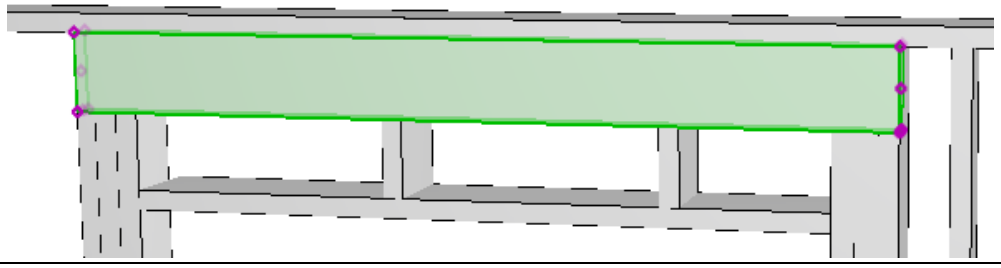
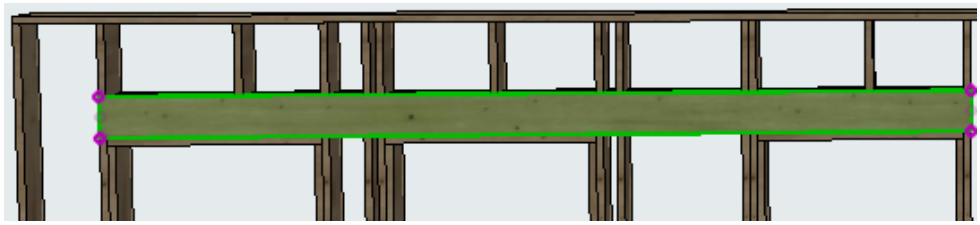
Returns 1-based table of relations. Each relation items which are tables also have following fields:

Field	Description
guidother	Guid of the related element layer object.
typeother	Type of the related element layer: <layer ref="WALL 42x42 VERT" … **type="intstud"**>.
reltype	Numeric value: • 1 to begin. • 2 to end. • 3 to middle (or crossing elements).
x1,x2	Intersection coordinates minimum and maximum values in X-axis. Can be outside the element area, negative before left hand side, bigger than length right from right hand side.
x1off,x2off	Maximum distance from min/max point to the intersection of other plank
angledeg	Angle in degrees between current element’s X-axis and other’s X-axis (which direction may have been swapped). Positive = on the left hand side, negative = on the right hand side.

24.2.1.3 Element lintels

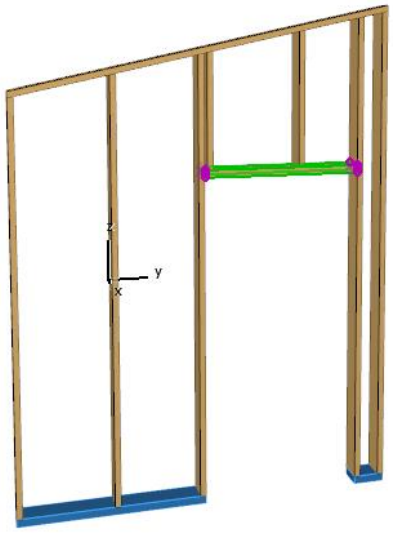
af_request (“elem_lintels”, settings) or af_request (“lintels”, settings).
Creates lintels according to passed setting table. Single lintel settings item contains these fields:

Field	Description
minwidth	Minimum width of the opening to use these rules. Place definition for longest first and continue with shorter ones.
mat	Material ID for the lintel beam.
thickness, height	If material ID is generic without fixed size, these must be given.
matstud	If given and non-empty, the material ID for additional studs. Original is moved further from the opening.
front	Number of lintel beams to front side of the element, 0 = none.
back	Number of lintel beams to back side of the element, 0 = none.
delorg	0 = keep source plank and place beams above it.

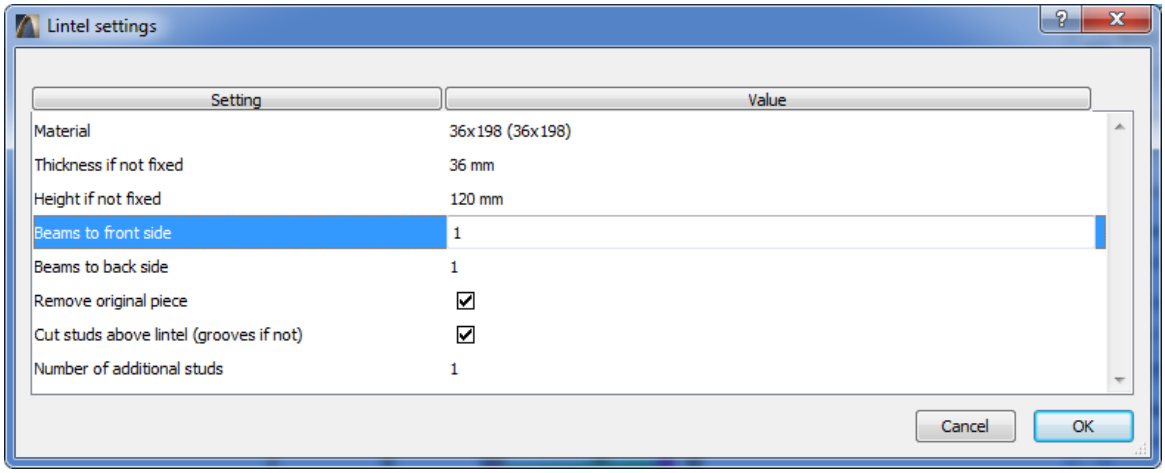
extorg	0:  1: 
totop	0 = place lintel just above the opening (default). 1 = place the lintel below top sill.
cutabove	0 = extend studs above the lintel beam to bottom of the beam and create grooves, 1 = cut studs above lintel beams.
studs	Number of additional studs, 0 = just create grooves to existing studs.
extbeamtosides	0 = Beams end to inner side of the furthest studs (king studs), default 1 = Extend beam to the outer side of the king stud 
joinbeamsdist	Joins newly created lintel beams closer or equal to the given value:  The unit for the value is meters and value -1 or smaller will skip applying this rule.
xmlbeam	Settings for the new beams (studs will have same settings as original ones), for example (also setting the projection pieces' settings): gsXmlBeam="<s><elemparam name=\"material\">Paint-06</elemparam>" ..

	"<newelemprojside><objparam name=\"iFill\">25 %</objparam><objparam name=\"iFillPen\">20</objparam></newelemprojside>" .. "<newelemprojtop><objparam name=\"iFill\">25 %</objparam><objparam name=\"iFillPen\">20</objparam></newelemprojtop>" .. "</s>"

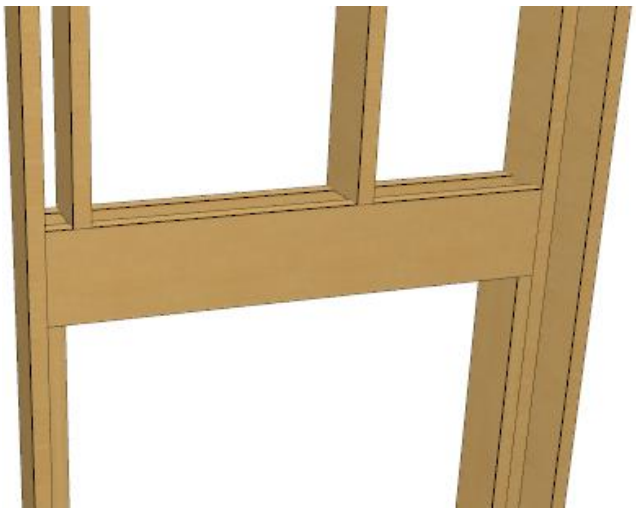
Starting point:



These settings:



The result:



24.3 Lua-scripts with elements

It is possible to create planks to the element with script instead of *xml*-definitions. The script gets information about element contour lines including openings. Contour line is always counterclockwise for both main contour and openings. For the main contour the left-hand side is towards the element and for openings, the right-hand side.

Lua globals set by ArchiFrame when handling elements:

Name	Description
gGuidElem	Element object's GUID to be used for ac_objectopen. Core layer or the owning layer for options scripts.
gGuidElemStamp	Set if handling element stamp, nil = unknown.
gElemIdStamp	The element ID to put into stamp: <elemtype class = "wall" id = "WALL 100 C/C 600" **idstamp="50x100"**>.
gAdjustElems	For options and corner scripts: Set to true if processing setting palette, nil if creating element planks.

Edges are in Lua global gtblEdges and one line contains the following fields:

Field	Description
index	Index to identify the edge.
x1,y1,x2,y2	Edge coordinates in element's coordinate system. Lower left corner is (0, 0).
gx1,gy1,gz1 gx2,gy2,gz2	Edge coordinates in global model 3D-coordinates. Missing if there is no plank related to this edge.
holeindex	0 = contour line, other = opening number 1…N.
used	Is there a plank for this place from earlier <planks>-definitions: • 0, no. • 1, yes. • 2, yes from <*planks axis = "unused" …*>-tag.
tblPlankNums	Indexes to the planks created to this edge in creation order, nil = no planks yet.

Planks to be created are placed into table gtblCreate which has fields:

Field	Description
group	Group name for <operations>-section.

id	Material id.
thickness	Material size if not fixed by material id.
height	Material size if not fixed by material id.
zoff	As in xml-definitions.
rotangle	As in xml-definitions.
x1,y1,x2,y2	Plank's reference line on element's surface.
lineside	Defines reference line alignment: -1 = left, 0 = middle (default), 1= right. This does not work always – if not, the rotangle +90/-90 should be used instead.
fixlen	If given, sets fixed length to the plank.
guidsettings	If given, the plank settings like pens, fills etc. are taken from this existing plank.
copyops	Related to guidsettings: If value = 1 all existing linked machinings (operations) like automatically updating grooves are copied from plank pointed by guidsettings.
xmlsettings	Settings as XML-settings for an Archicad element.
xmlsettingsproj	Settings to be applied to the related projection planks. Under the root tag there may be settings with these tag names: • newelemprojside • newelemprojtop
mayturn	If given and set to true, ArchiFrame may turn the plank to be best in the element elevation.
postlua	Lua-script inside <tag> to be run after the plank has been actually created internally. May refer to base script with attribute ref = “name”.
extendtoelem	Possible values: • Missing or 0, no effect. • 1, given line is extended to cover whole element polygon and split if there is an opening intersecting the line. • 2, extend and cut with polygon edges (gable cut).
force	Controls overlapping pieces, possible values: • false/0 = do not add if there is already a plank (default) • true/1 = keep rather this plank than any existing one • 2 = create even if outside the element • 3 = leave old intersection piece(s) under new forced piece Additional modifier to be added to the force-value: • 256, adjust any existing parallel piece to this added one

Number of existing planks is in global gnPlanks. New planks are appended according to table gtblCreate. Script may define operations for the planks by creating global variable gxmlOperations. Its format is like section Operations between the planks <operations>. For the script planks it is possible to refer to those using index number instead of group name, using format #number (for example target = "#1"). Many numbers may be used by separating numbers with comma, for example target = “#1, 2, 3, 4”. For example adjusting to successive planks together:

```
-- Adjust these two together
strXmlOp = string.format(
    "%s" ..
    "<joinends target=\"#%d\" operator=\"#%d\">\n" ..
    "    <joinends conntype=\"endtoend\" jointgap=\"0\"></joinends>\n" ..
    "</joinends>\n",
```

```
    strXmlOp, gnPlanks+i, gnPlanks+i+1)
```

Please see also ac_getobjparam.

ArchiFrame provides these additional functions to this task:

Function	Description
af_hasedgex (edge, vecx, vecy)	Checks whether an infinite area starting from given edge to given direction, intersects with any other edge. This is used to detect for example top side of the element (vex = 0 and vecy = 1, upwards). Returns. True = intersection found, false = no intersection.

In addition to creating new planks, it is possible to edit planks defined earlier. To do so, ac_objectopen() is called with string format #index, where index is *1*-based index to the plank. In dimension line scripts adding exclamation mark after dash will return the related 3D-piece instead of projection piece (#!%d). For example:

```
<!-- Sets different pen for studs -->
<script id="setcolours">
  <![CDATA[
gnPlankCount=0
gtblCreate = {}

 for i=1,gnPlanks do
  ac_objectopen(string.format("#%d",i))

  -- Vertical (studs) with different pen
  x2=ac_objectget("iEndX")
  y2=ac_objectget("iEndY")
  if x2*x2+y2*y2<0.001 then
    ac_objectset("#pen", 3)
  end
  ac_objectclose()
 end
]]>
</script>
```

24.3.1 Script to set planks' IDs inside an element, <renumscript>, <renumscriptnew>

<renumscriptnew> was added for backwards compatibility. It works just as <renumscript> except when it was added, ArchiFrame allows having an empty definition for the script to be used with element templates.

It is possible to set prefix and postfix to the ID based for example on plank's role in the element. To do that, the recommended way is to include following line to the element definition xml under xml-tag <layer> after all <planks>-definitions:

```
<renumscript ref="kwrenum"></renumscript>
```

This refers to a script in xml-path archiframe/elem/settings/scripts:

```
        <script id="kwrenum">
          <![CDATA[
-- Loaded once per element, called for each plank current plank opened already
-- Returns table having optional fields:
-- idpre  To be added before ID, if starting with !, will not be visible
-- idpost To be added after ID, if starting with !, will not be visible
function GetPlankRenumData()
  local s, t
```

```
  t={}
  s=ac_objectget("iElemGroup")
  if s=="vertical_y" then      -- vertical_y_openingdouble for double opening pieces
    t.idpost="!A"
  end
  return t
end
]]>
        </script>
```

This example script sets the text “!A” to be appeneded after the plank ID. Exclamation mark ! means that the text is actually not appended – it just makes sure that planks classified as opening studs will get their own ID.

Another prefix is #num that tells ArchiFrame that this plank group should be numbered starting from the given number. For example:

```
        <script id="renum_by_role">
          <![CDATA[
-- Loaded once per element, called for each plank current plank opened already
-- Returns table having optional fields:
-- idpre  To be added before ID
-- idpost To be added after ID
-- For both, the special characters:
-- !, will not be visible
-- #num, useful for idpre - will define the starting number for the planks (not visible
either)
function GetPlankRenumData()
  local s, t, res

  t={}
  s=ac_objectget("iElemGroup")

  res="#1"
  if string.match(s, "^vertical_x.*") then
    res="#200"
  elseif string.match(s, "^lintel.*") then
    res="#250"
  elseif string.match(s, "^bottom_force.*") or string.match(s, "^2ndbottom_force.*") then
    res="#300"
  elseif string.match(s, "^top_force.*") or string.match(s, "^2ndtop_force.*") then
    res="#400"
  elseif string.match(s, "^nogging.*") then
    res="#500"
  end

  t.idpre=res
  return t
end
]]>
```

24.3.2 The corner script callback functions

If called from corner tool palette, Lua global gAdjustElems is set to true. Also these globals are present:

- gbCornerFindOther, UI value for check box *Find connecting corner.*
- gbCornerAdjustOther, UI value for check box *Adjust connecting corner.*

Set (sSettings, guidParent, nCorner).

Called to set single corner for given parent element. There are two cases when this function is called:

- When new element is created and corner types are already set.
- When corner type is changed from another type or from default using corner tool palette.

Parameter	Description
sSettings	Previous settings string, “” = default settings.
guidParent	Guid of the master element, use af_request (“elem_getparent”, guidParent) to process.
nCorner	1 = left, 2 = right, 3 = bottom, 4 = top.

OnPlanksCreated (sSettings, guidElem, layerType, nCorner).
Called when planks for element outline and openings are placed but before (stud) spacing rules are applied. This script places the planks specific to the corner type.

Parameter	Description
sSettings	Previous settings string, “”= default settings.
guidElem	Current element’s guid.
layerType	Type of the related element layer: <layer ref = "WALL 42x42 VERT" … **type="intstud"**>.
nCorner	Corner to handle 1 = left, 2 = right.

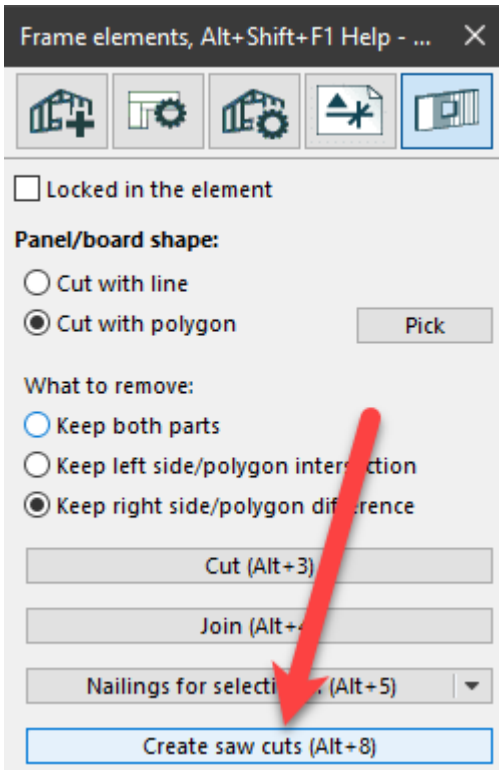
24.3.3 Lua scripts with element options

Element options have fixed function names:

Function	Description
Has()	Called by ArchiFrame to find out if the option is set for options UI. Return: -1 = Not known/no planks, 0 = nope, 1 = has.
Settings(sSettings)	Called by ArchiFrame when user clicks *Settings*-button in UI. Return: nil = canceled, "" = clear settings/use defaults, str = settings str.
Set(sSettings)	Called to set the option on either from options UI (gAdjustElems == true) or when creating element planks.
Reset()	Called to cancel the option from options UI.
GetName(sSettings, baseName)	Called to make special name for the options, for example, when user has set the settings to non-default. BaseName is string of the option name, add it to the front of the result string. Return: string to show in UI.

24.3.4 Lua script to adjust generated saw cuts

ArchiFrame searches for fixed file name ArchiFrameSawCuts.lua from user specific settings folder when creating the saw cuts for selection:



If the script exists, it is loaded and this function is called:

```
-- tblLines is a table having auto-created saw lines for each polygon list edge following
standard polygon rules: contour line ccw, holes cw
-- projWorld is geometry world of the owing ArchiFrameElement
-- guidElem Owner ArchiFrameElement
-- polyList Polygon list of the parts that saw cuts are created to, can be passed
ac_geo("linepolyx",...)
-- Return value: either tblLines as passed here or it adjusted
function AdjustSawLines(tblLines, projWorld, guidElem, polyList)
```

Each item in tblLines has fields:

Field	Description
x1, y1	The starting point of the saw cut
x2, y2	The ending point of the saw cut
overbeg	Boolean: is there overcut at begin
overend	Boolean: is there overcut at end
angledeg	The saw cut angle in degrees
skip	Script can set this item to skip this saw cut
polyind	Polygon index in case of split surface
contourind	0=outline, >0=hole index in current polygon

24.4 Other Lua extensions

Http-requests interface.

Res = curl_httpget ("http://www.google.com").

Makes http-get request. Returns server reply or causes an error if failed.

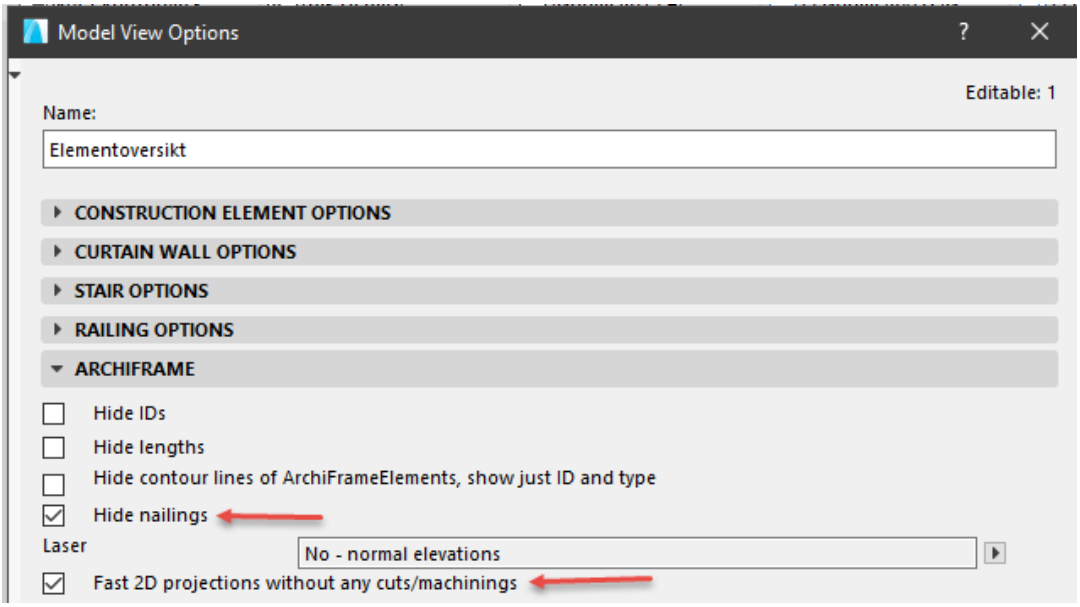
25Tips and tricks

25.1 Huge models

Having a lot of elevations in single storey slows down opening the elevations. It is better to split elevations to multiple storeys if threre is going to be more than ~200 elevations. Another way to split the elevations is to save each storey into separate BIMCould/TeamWork-model having the elevations there in their own storey. Then use hotlinking to connect just the 3D model without elevations to make the 3D master model.

Using non-updating layouts is faster than the automatically updating ones.

There is a model view options to speed up 2D regeneration typically by 25%:



There is an article having good tips here:
https://nordicbim.net/article.php?article=hugemodels

25.2 Copy & paste between Archicads

From 10/2023 ArchiFrame supports copy & pasting between Archicads maintaining the relations between ArchiFrame-elements like ArchiFrameElement to the planks belonging to it. Some notes about copy & pasting:

- To avoid plenty of Archicad attributes, the copy source and target files should be based on the same Archicad template. Otherwise you may end up having two similar building materials etc.
- Source and target files should use the same data folders because the own element type definitions are there, not included in the copy & paste data.
- For the Fascia-tool similar Fascia definitions should exist in the copy source and target files. Please see this.
- ArchiFrame saves the needed data to the copied elements starting from release 10/2023. When copy & pasting earlier models, there may appear a warning: *Originating AC elem of ArchiFrame element "XX" was not found, relation lost.*
  This is because the original AC source elements were not pasted now or copied elements are done with ArchiFrame version prior to 10/2023.
- Element drawings and dimension drawings: With models made prior to 10/2023 version, ArchiFrame cannot maintain the relations inside element drawings (the ones related to

ArchiFrameElements) nor in dimdrawings. To ensure proper copying of data, please regenerate these drawings.

- It is possible to copy & paste the element drawings only if the 3D-parts and the elevation are on the same storey. This is because in Archicad it is possible to copy elements just from a single storey.
- Limitation for dimension drawings: angle dimension cannot maintain connection to the drawings, because of an Archicad bug. That will cause duplicated angle dimensions when dimension drawings are updated.