AddWidgetInPx

Summary

This blocks is used to add a widget in an existing px view.

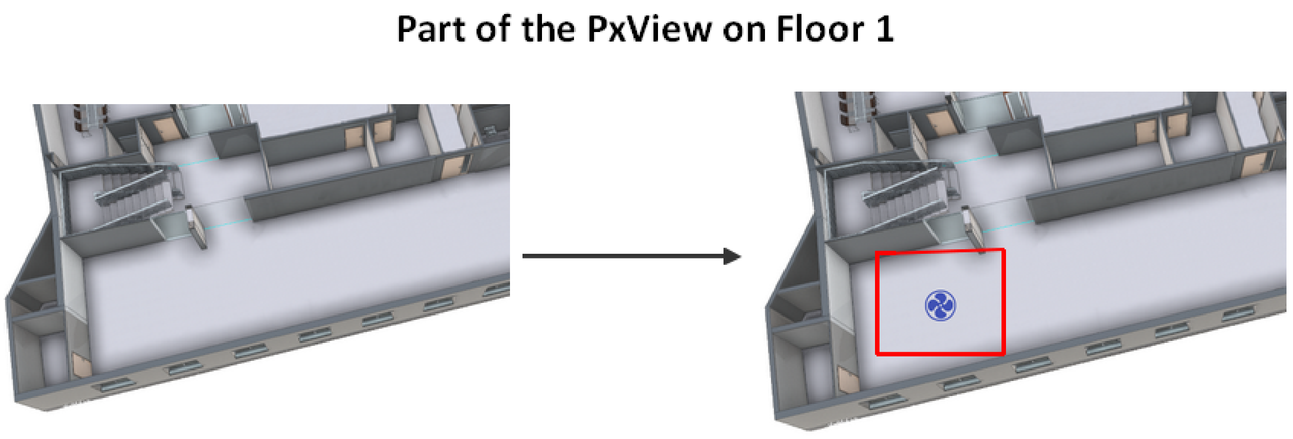
Implementation

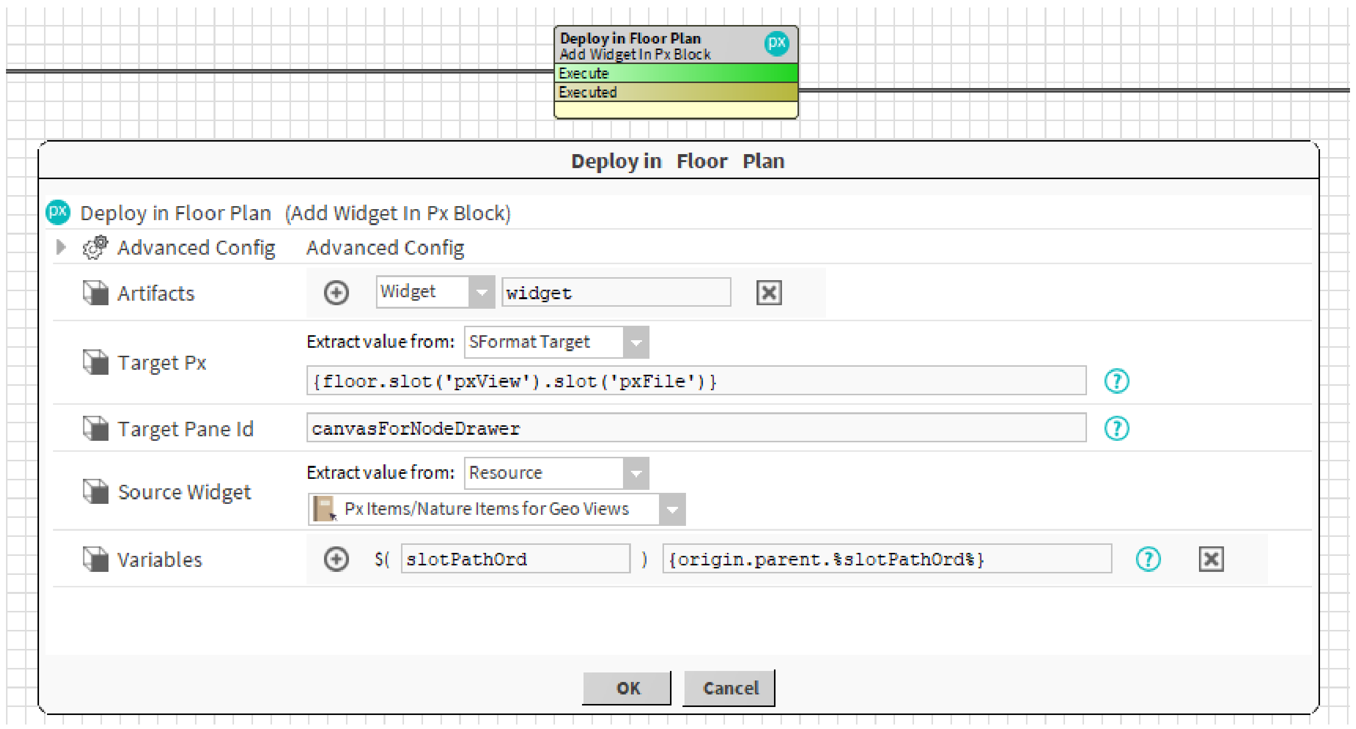
- Drag and drop the block from the palette
- Link the "execute" action from the "executed" topic of the previous block

Example

This block creates a widget that represents a source in the floor plan it is assigned to.

- The origin is a FCU
- The target px is the floor px taken from the floor artifact (we got the floor artifact from an EntityArtifact block)
- The target px is a little bit complex and contains several CanvasPane, so to identify the one, we added to the CanvasPane a btibCore:IdBinding and defined "canvasForNodeDrawer". So the widget will be added into this precise Canvas.
- The sourceWidget is a GridPane or a Label defined in a ResourceDefinition
- We use a variable defined in the source widget to set the slotPathOrd of the FCU





Deploy in Floor Plan

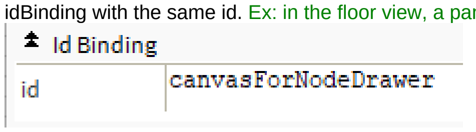
Deploy in Floor Plan (Add Widget In Px Block)

Advanced Config	Advanced Config
Artifacts	Widget — widget
Target Px	Extract value from: SFormat Target `{floor.slot('pxView').slot('pxFile')}`
Target Pane Id	`canvasForNodeDrawer`
Source Widget	Extract value from: Resource Px Items/Nature Items for Geo Views
Variables	$(`slotPathOrd`) `{origin.parent.%slotPathOrd%}`

OK Cancel

Properties

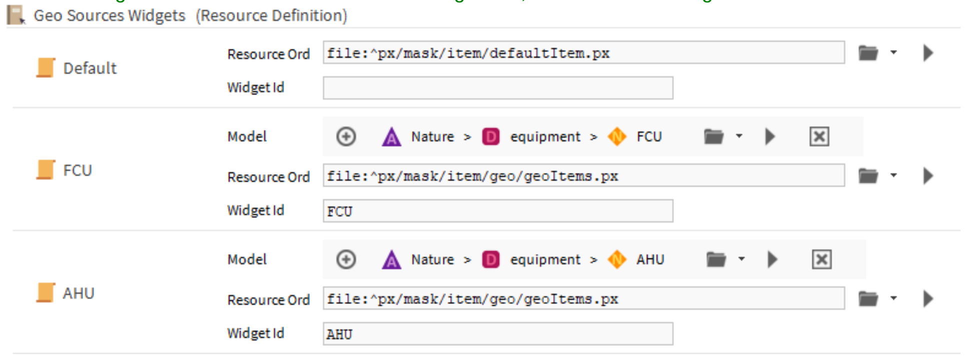
- *Artifacts:* Artifacts created by this block
  - Widget: The created widget. Ex: the fcu widget added in the px
- *TargetPx:* SFormat to define the px where the widget will be added. See General strategy parameters. Ex: floor is an artifact representing the floor the source is assigned to. The targetPx is the view of the floor.
- *TargetPaneId:* (optionnal) SFormat to define the id of the pane where the widget will be added. In the px, the pane should contain a btibCore: idBinding with the same id. Ex: in the floor view, a pane contains the id "canvasForNodeDrawer" as shown below.



Id Binding

id	canvasForNodeDrawer

- *SourceWidget:* Value to define the model of the widget. Ex: the widget depends on the nature of the source. The model is a resource definition and according to the nature node the source is assigned to, the model will change.



Geo Sources Widgets (Resource Definition)

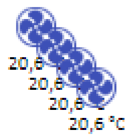
Default	Resource Ord	file:^px/mask/item/defaultItem.px
	Widget Id	
FCU	Model	Nature > equipment > FCU
	Resource Ord	file:^px/mask/item/geo/geoItems.px
	Widget Id	FCU
AHU	Model	Nature > equipment > AHU
	Resource Ord	file:^px/mask/item/geo/geoItems.px
	Widget Id	AHU

- *PxVariables*: some variables can be passed to the widget when it is created and updated. Ex: the ord of the source is passed to the widget (to create an hyperlink)

AdvancedConfig

- *OnCreatePositionPolicy {CopySource, Waterfall}*
  - *CopySource:* The created widget keeps the same position as in the px resource. Useful when you want to create an AHU Model. Each element stays at the same position.

  - *Waterfall:* The created widget is added each below the previous one. Useful when you want to add multiple times the same widget.



- *OnSyncPositionPolicy {CopySource, KeepExisting}*
  - *CopySource:* On redo, the position of the widget will be updated in the target px with the same as the px resource. Useful when you want to match exactly your model (an AHU model for example)
  - *KeepExisting:* On redo, the position of the widget will not be updated if you changed it in the target px. Useful when you positioned widgets in a synoptic or floor plan.
- *OnSyncReplacePolicy*
  - *WholeWidget*: On redo, the widget that was added in the target px will be replaced entirely by the one in the resource. Useful when you want all your widgets to look like the same (an FCU widget for example). It's super easy to update everywhere.
  - *Properties and Variables*: The widget will remain, only its properties will be updated (background, halign etc.)
  - *Ignored properties:* List of properties you don't want to update. Useful when you changed in the target px a property of your widget and you don't want it to revert it back each time the strategy is played such as a background for example.
- *IgnoreErrorOnMissingWidget:* If set to true, doesn't log a warning if the source widget is missing.

Behavior: DO

A widget is added in the target px. In case of REDO, this widget can be relocated or updated if the source widget itself has been updated (depending on the advanced config)

Behavior: UNDO

The widget is removed the from target px.

When a polygon is added for the first time, a hidden id will be added inside (you can see it if you open the text editor view on the px)
This id is composed of two handles, the handle of the origin and the handle of the block.
Then on a second execution of the strategy, the block will build the id and look in the px if there is a widget with this id. If the widget exists, the bloc will update the widget (REDO) otherwise it will create it (DO)
You shouldn't copy paste the strategy, the sourcePx or targetPx to other stations as the handle will be probably different and you might have unexpected results.

Also be careful with the your widgets in your sourcePx, make sure that they don't contain an id.