

# *Implement a new Property Panel*

*The BRAPH 2 Developers*

*August 11, 2023*

This is the developer tutorial for implementing a new property panel. In this Tutorial, we will explain how to create the generator file `*.gen.m` for a new property panel which can be compiled by `braph2genesis`, using the property panel `PanelPropLogical` as an example.

## *Contents*

<i>Implementation of Property Panel (PanelPropLogical)</i>	2
<i>Property Panel for all User Interface Objects</i>	5

### Implementation of Property Panel (*PanelPropLogical*)

We will start by implementing in detail the property panel `PanelPropLogical`, which applies the general concepts of a property panel and is a direct extension of the element `PanelProp`.

**Code 1: `PanelPropLogical` element header.** The header section of generator code for `_PanelPropLogical.gen.m` provides the general information about the `PanelPropLogical` element.

---

```

1 %% iheader!
2 PanelPropLogical < PanelProp (pr, panel property logical) plots the panel of
   a property logical. ①
3
4 %% idescription!
5 PanelPropLogical plots the panel for a LOGICAL property with a checkbox.
6 It works for all categories.
```

---

① The element `PanelPropLogical` is defined as a subclass of `PanelProp`. The moniker will be `pr`.

**Code 2: `PanelPropLogical` element props update.** The `props_update` section of generator code for `_PanelPropLogical.gen.m` updates the properties of the `PanelProp` element. This defines the core properties of the property panel.

---

```

1 %% iprops_update!
2
3 ...
4
5 %% iprop!
6 EL (data, item) is the element. ②
7 %%% idefault!
8 PanelProp()
9
10 %%% iprop!
11 PROP (data, scalar) is the property number. ③
12 %%% idefault!
13 PanelProp.DRAW
14
15 ...
```

---

② It defines the element for this property panel.

③ It defines the property pointer associated with the element for this property panel.

**Code 3: `PanelPropLogical` new props.** The `props` section of generator code for `_PanelPropLogical.gen.m` defines the graphical elements for the `PanelPropLogical` element.

---

```

1 %% iprops!
2
3 %% iprop!
4 CHECKBOX (evanescent, handle) is the logical value checkbox. ④
5 %%% icalculate!
6 el = pr.get('EL');
7 prop = pr.get('PROP');
8
9 checkbox = ucheckbox( ...
10     'Parent', pr.memorize('H'), ... % H = p for Panel
11     'Tag', 'CHECKBOX', ...
12     'Text', '', ...
```

---

④ The panel for a property logical has a `CHECKBOX`.

```

13     'FontSize', BRAPH2.FONTSIZE, ...
14     'Tooltip', [num2str(el.getPropProp(prop)) ' ' el.getPropDescription(prop
15         )], ...
16     'ValueChangedFcn', {@cb_checkbox} ...
17 );
18 value = checkbox;
19 %%% icalculate_callbacks!
20 function cb_checkbox(~, ~) ⑤
21     el = pr.get('EL');
22     prop = pr.get('PROP'); ⑥
23
24     checkbox = pr.get('CHECKBOX');
25     new_value = logical(get(checkbox, 'Value')); ⑦
26
27     el.set(prop, new_value) ⑧
28 end

```

⑤ The panel for a property logical has a callbacks for its CHECKBOX, defining the appropriate behavior of the checkbox.

⑥ The callback firstly extracts the property logical.

⑦ The callback then extracts the value of the CHECKBOX.

⑧ Finally, the callback sets the new value to the logical property.

**Code 4: PanelPropLogical element props update.** The continuing props\_update section of generator code for \_PanelPropLogical.gen.m updates the rest of the properties of the PanelProp element. This defines the panel drawing of the property panel.

```

1 %%% iprops_update!
2
3 ...
4
5
6 %%% iprop!
7 X_DRAW (query, logical) draws the property panel. ⑨
8
9 %%% icalculate!
10 value = calculateValue@PanelProp(pr, PanelProp.X_DRAW, varargin{:}); % also
    warning
11 if value
12     pr.memorize('CHECKBOX')
13 end
14
15 %%% iprop!
16 DELETE (query, logical) resets the handles when the panel is deleted. ⑩
17 %%% icalculate!
18 value = calculateValue@PanelProp(pr, PanelProp.DELETE, varargin{:}); % also
    warning
19 if value
20     pr.set('CHECKBOX', Element.getNoValue())
21 end
22
23 %%% iprop!
24 HEIGHT (gui, size) is the pixel height of the property panel. ⑪
25 %%% idefault!
26 s(4)
27
28 %%% iprop!
29 REDRAW (query, logical) resizes the property panel and repositions its
    graphical objects.
30 %%% icalculate!

```

⑨ X\_DRAW draws the panel. In this case, the panel contains a CHECKBOX.

⑩ DELETES resets the handles when the panel is deleted. In this case, it sets the CHECKBOX to NoValue().

⑪ HEIGHT specifies the height of the panel that contains the CHECKBOX.

```

31 value = calculateValue@PanelProp(pr, PanelProp.REDRAW, varargin{:}); % also
    warning
32 if value
33     w_p = get_from_varargin(w(pr.get('H'), 'pixels'), 'Width', varargin);
34
35     set(pr.get('CHECKBOX'), 'Position', [s(.3) s(.3) .70*w_p s(1.75)])
36 end
37
38 %%% iprop!
39 UPDATE (query, logical) updates the content and permissions of the editfield
    . (12)
40 %%% icalculate!
41 value = calculateValue@PanelProp(pr, PanelProp.UPDATE, varargin{:}); % also
    warning
42 if value
43
44     el = pr.get('EL');
45     prop = pr.get('PROP');
46
47     switch el.getPropCategory(prop)
48         case Category.CONSTANT (13)
49             set(pr.get('CHECKBOX'), ...
50                 'Value', el.get(prop), ...
51                 'Enable', 'off' ...
52             )
53
54         case Category.METADATA (14)
55             set(pr.get('CHECKBOX'), 'Value', el.get(prop))
56
57             if el.isLocked(prop)
58                 set(pr.get('CHECKBOX'), 'Enable', 'off')
59             end
60
61         case {Category.PARAMETER, Category.DATA, Category.FIGURE, Category.
GUI} (15)
62             set(pr.get('CHECKBOX'), 'Value', el.get(prop))
63
64             prop_value = el.getr(prop);
65             if el.isLocked(prop) || isa(prop_value, 'Callback')
66                 set(pr.get('CHECKBOX'), 'Enable', 'off')
67             end
68
69         case {Category.RESULT Category.QUERY Category.EVANESCENT}
70             prop_value = el.getr(prop);
71
72             if isa(prop_value, 'NoValue')
73                 set(pr.get('CHECKBOX'), 'Value', el.getPropDefault(prop))
74             else
75                 set(pr.get('CHECKBOX'), 'Value', el.get(prop))
76             end
77
78             set(pr.get('CHECKBOX'), 'Enable', 'off')
79     end
80 end

```

(12) UPDATE updates the status of the CHECKBOX based on various scenario, influenced by the property's category.

(13) When the property is a CONSTANT, the CHECKBOX is disabled as it cannot be changed.

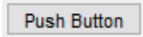
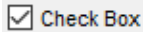

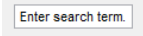
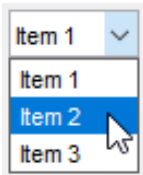
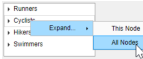
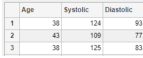
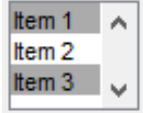

(14) When the property is a METADATA, the CHECKBOX's enabling status corresponds to whether it is locked.

(15) The behavior of the CHECKBOX varies according to different categories, as we have seen two examples until this point. To ensure precise control over the CHECKBOX's functionality, consider the specific cases for behavior.

### *Property Panel for all User Interface Objects*

The concept of implementing `PanelPropLogical`, as shown in the previous section, can be seamlessly extended to all other user interface (UI) elements.

The subsequent table presents a comprehensive overview of various UI objects, each coupled with its corresponding property panel as an illustrative example.

UI Object	Example	PanelProp Element
uibutton		<code>PanelPropItem</code>
checkbox		<code>PanelPropLogical</code>
uitextarea		<code>PanelPropStringList</code>
uieditfield		<code>PanelPropString</code>
uidropdown		<code>PanelPropLogical</code>
uicontextmenu		<code>PanelPropMatrix</code>
uitable		<code>PanelPropItem</code>
uilibox		<code>PanelPropClassList</code>
uislider		<code>PanelPropCell</code>

By referencing the related `PanelProp` elements, the process of creating a new property panel becomes straightforward and efficient.