# *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 the be compiled by `braph2genesis`, using the property panel `PanelPropLogical` and `PanelPropNet` as examples.

## *Contents*

## *Implementation of Property Panel*

### *Panel of a property logical*

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.

```
%% iheader!
PanelPropLogical < PanelProp (pr, panel property logical) plots the panel of
      a property logical.  (1)

%%% idescription!
PanelPropLogical plots the panel for a LOGICAL property with a checkbox.
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.

```
%% iprops_update!

...

%%% iprop!
EL (data, item) is the element. (2)
%%%% idefault!
PanelProp()

%%% iprop!
PROP (data, scalar) is the property number. (3)
%%%% idefault!
PanelProp.DRAW

...
```

(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.

```
%% iprops!

%%% iprop!
CHECKBOX (evanescent, handle) is the logical value checkbox.  (4)
%%%% icalculate!
el = pr.get('EL');
prop = pr.get('PROP');

checkbox = uicheckbox( ...
    'Parent', pr.memorize('H'), ... % H = p for Panel
```

(4) The panel for a property logical has a `checkbox`.

```
    'Tag', 'CHECKBOX', ...
    'Text', '', ...
    'FontSize', BRAPH2.FONTSIZE, ...
    'Tooltip', [num2str(el.getPropProp(prop)) ' ' el.getPropDescription(prop
     )], ...
    'ValueChangedFcn', {@cb_checkbox} ...
    );

value = checkbox;
%%%% icalculate_callbacks!
function cb_checkbox(~, ~) (5)
    el = pr.get('EL');
    prop = pr.get('PROP'); (6)

    checkbox = pr.get('CHECKBOX');
    new_value = logical(get(checkbox, 'Value')); (7)

    el.set(prop, new_value) (8)
end
```

(5) The panel for a property logical has a `callbacks` for its `checkbox`, defining the appropreate behavior of the checkbox.
(6) The `callbacks` firstly extracts the property logical.
(7) The `callbacks` then extracts the value of the `checkbox`.
(8) Finally, the `callbacks` 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.

```
%% iprops_update!

...


%%% iprop!
X_DRAW (query, logical) draws the property panel. (9)

%%%% icalculate!
value = calculateValue@PanelProp(pr, PanelProp.X_DRAW, varargin{:}); % also
     warning
if value
    pr.memorize('CHECKBOX')
end

%%% iprop!
DELETE (query, logical) resets the handles when the panel is deleted. (10)
%%%% icalculate!
value = calculateValue@PanelProp(pr, PanelProp.DELETE, varargin{:}); % also
     warning
if value
    pr.set('CHECKBOX', Element.getNoValue())
end

%%% iprop!
HEIGHT (gui, size) is the pixel height of the property panel. (11)
%%%% idefault!
s(4)

%%% iprop!
```

(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`.

```
REDRAW (query, logical) resizes the property panel and repositions its graphical objects.
%%%% !calculate!
value = calculateValue@PanelProp(pr, PanelProp.REDRAW, varargin{:}); % also warning
if value
    w_p = get_from_varargin(w(pr.get('H'), 'pixels'), 'Width', varargin);

    set(pr.get('CHECKBOX'), 'Position', [s(.3) s(.3) .70*w_p s(1.75)])
end

%%% !prop!
UPDATE (query, logical) updates the content and permissions of the editfield . (12)
%%%% !calculate!
value = calculateValue@PanelProp(pr, PanelProp.UPDATE, varargin{:}); % also warning
if value

    el = pr.get('EL');
    prop = pr.get('PROP');

    switch el.getPropCategory(prop)
        case Category.CONSTANT (13)
            set(pr.get('CHECKBOX'), ...
                'Value', el.get(prop), ...
                'Enable', 'off' ...
                )

        case Category.METADATA (14)
            set(pr.get('CHECKBOX'), 'Value', el.get(prop))

            if el.isLocked(prop)
                set(pr.get('CHECKBOX'), 'Enable', 'off')
            end

        case {Category.PARAMETER, Category.DATA, Category.FIGURE, Category.GUI} (15)
            set(pr.get('CHECKBOX'), 'Value', el.get(prop))

            prop_value = el.getr(prop);
            if el.isLocked(prop) || isa(prop_value, 'Callback')
                set(pr.get('CHECKBOX'), 'Enable', 'off')
            end

        case {Category.RESULT Category.QUERY Category.EVANESCENT}
            prop_value = el.getr(prop);

            if isa(prop_value, 'NoValue')
                set(pr.get('CHECKBOX'), 'Value', el.getPropDefault(prop))
            else
                set(pr.get('CHECKBOX'), 'Value', el.get(prop))
            end

            set(pr.get('CHECKBOX'), 'Enable', 'off')
    end
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 now. To ensure precise control over the checkbox's functionality, consider the specific cases for behavior.

*Panel of a property net*

We can now use ...