

# msaSDK Module

## .admin.frontend.components

### Attributes

#### BASE\_DIR module-attribute

```
BASE_DIR = os.path.dirname(os.path.abspath(__file__))
```

### Classes

#### Action

Bases: MSAUINode

##### Attributes

##### actionType class-attribute

```
actionType: str = None
```

[Required] This is the core configuration of the action, to specify the action's role type. Supports: ajax, link, url, drawer, dialog, confirm, cancel, prev, next, copy, close.

##### active class-attribute

```
active: bool = None
```

If or not the button is highlighted.

##### activeClassName class-attribute

```
activeClassName: str = None
```

Add a class name to the button highlighting. "is-active"

##### activeLevel class-attribute

```
activeLevel: str = None
```

The style of the button when it is highlighted, configured to support the same level.

**args** class-attribute

```
args: Union[dict, str] = None
```

Event arguments

**block** class-attribute

```
block: bool = None
```

Use display: "block" to display the button.

**close** class-attribute

```
close: Union[bool, str] = None
```

When action is configured in dialog or drawer's actions, set to true to close the current dialog or drawer after this action.

**componentId** class-attribute

```
componentId: str = None
```

Target component ID

**confirmText** class-attribute

```
confirmText: MSAUITemplate = None
```

When set, the action will ask the user before starting. Can be fetched with \${xxx}.

**disabledTip** class-attribute

```
disabledTip: str = None
```

Popup when mouse hover is disabled, you can also configure the object type: fields are title and content. available \${xxx}.

**icon** class-attribute

```
icon: str = None
```

Set icon, e.g. fa fa-plus.

**iconClassName** class-attribute

```
iconClassName: str = None
```

Add a class name to the icon.

**label** class-attribute

```
label: str = None
```

The text of the button. Can be fetched with \${xxx}.

**level** class-attribute

```
level: LevelEnum = None
```

The style of the button, support: link, primary, secondary, info, success, warning, danger, light, dark, default.

**onClick** class-attribute

```
onClick: str = None
```

Customize the click event by defining the click event as a string onClick, which will be converted to a JavaScript function

**reload** class-attribute

```
reload: str = None
```

Specify the name of the target component to be refreshed after this operation (the component's name value, configured by yourself), separated by ,.

**required** class-attribute

```
required: List[str] = None
```

Configure an array of strings, specifying that the form entry of the specified field name is required to pass validation before the operation can be performed in the form

**rightIcon** class-attribute

```
rightIcon: str = None
```

Set the icon to the right of the button text, e.g. fa fa-plus.

**rightIconClassName** class-attribute

```
rightIconClassName: str = None
```

Add a class name to the right icon.

**script** class-attribute

```
script: str = None
```

Custom JS script code, which can perform any action by calling doAction, and event action intervention through the event object event

size class-attribute

```
size: str = None
```

The size of the button, support: xs, sm, md, lg.

tooltip class-attribute

```
tooltip: str = None
```

popup text when mouse hover, also can configure the object type: title and content. can be \${xxx}.

tooltipPlacement class-attribute

```
tooltipPlacement: str = None
```

If tooltip or disabledTip is configured, specify the location of the tip, you can configure top, bottom, left, right.

type class-attribute

```
type: str = 'button'
```

Specify as Page renderer. button action

## ActionType

### Classes

### Ajax

Bases: Action

### Attributes

actionType class-attribute

```
actionType: str = 'ajax'
```

Click to display a popup box

api class-attribute

```
api: MSA_UI_API = None
```

request address, refer to api format description.

feedback class-attribute

```
feedback: Dialog = None
```

If ajax type, when the ajax returns normal, a dialog can be popped up to do other interactions. The returned data can be used in this dialog. See Dialog for format

messages class-attribute

```
messages: dict = None
```

success: ajax operation success prompt, can not be specified, not specified when the api return prevail. failed: ajax operation failure prompt.

redirect class-attribute

```
redirect: MSAUITemplate = None
```

Specify the path to jump to at the end of the current request, can be fetched with \${xxx}.

## Copy

Bases: Action

### Attributes

actionType class-attribute

```
actionType: str = 'copy'
```

Copy a piece of content to the pasteboard

content class-attribute

```
content: MSAUITemplate
```

Specify the content to be copied. Can be fetched with \${xxx}.

copyFormat class-attribute

```
copyFormat: str = None
```

The format of the copy can be set by copyFormat, default is text/html

## Dialog

Bases: Action

### Attributes

actionType class-attribute

```
actionType: str = 'dialog'
```

Show a popup box when clicked.

**dialog** class-attribute

```
dialog: Union[Dialog, Service, MSAUISchemaNode]
```

Specify the content of the popup box, see Dialog for format

**nextCondition** class-attribute

```
nextCondition: bool = None
```

Can be used to set the next data condition, default is true.

## Drawer

Bases: Action

### Attributes

**actionType** class-attribute

```
actionType: str = 'drawer'
```

Show a sidebar when clicked

**drawer** class-attribute

```
drawer: Union[Drawer, Service, MSAUISchemaNode]
```

Specify the content of the popup box, see Drawer for format

## Link

Bases: Action

### Attributes

**actionType** class-attribute

```
actionType: str = 'link'
```

**link** class-attribute

```
link: str
```

## Url

Bases: Action

### Attributes

**actionType** class-attribute

```
actionType: str = 'url'
```

Jump directly

**blank** class-attribute

```
blank: bool = None
```

false if true will open in a new tab page.

**url** class-attribute

```
url: str
```

When the button is clicked, the specified page will be opened. Can be fetched with \${xxx}.

# Alert

Bases: MSAUINode

alert

## Attributes

**body** class-attribute

```
body: MSAUISchemaNode = None
```

Display content

**className** class-attribute

```
className: str = None
```

class name of the outer Dom

**closeButtonClassName** class-attribute

```
closeButtonClassName: str = None
```

CSS class name of the close button

**icon** class-attribute

```
icon: str = None
```

Custom icon

**iconClassName** class-attribute

```
iconClassName: str = None
```

CSS class name of the icon

level class-attribute

```
level: str = None
```

"info" # level, can be: info, success, warning or danger

showCloseButton class-attribute

```
showCloseButton: bool = None
```

False # whether to show the close button

showIcon class-attribute

```
showIcon: bool = None
```

False # Whether to show icon

type class-attribute

```
type: str = 'alert'
```

Specify as alert renderer

## AnchorNav

Bases: MSAUINode

Anchor Nav

### Attributes

active class-attribute

```
active: str = None
```

The area to be positioned

className class-attribute

```
className: str = None
```

Class name of the outer Dom

direction class-attribute



direction: `str` = None

"vertical" # You can configure whether the navigation is displayed horizontally or vertically. The corresponding configuration items are: vertical, horizontal

linkClassName class-attribute

linkClassName: `str` = None

Class name of the navigating Dom

links class-attribute

links: `list` = None

Contents of links

sectionClassName class-attribute

sectionClassName: `str` = None

Class name of the anchor area Dom

type class-attribute

type: `str` = 'anchor-nav'

Specify as AnchorNav renderer

**Classes**

**Link**

Bases: `MSAUINode`

**Attributes**

body class-attribute

body: `MSAUISchemaNode` = None

area content area

className class-attribute

className: `str` = None

"bg-white b-l b-r b-b wrapper-md" # region member style

href class-attribute

```
href: str = None
```

region identifier

label class-attribute

```
label: str = None
```

name

title class-attribute

```
title: str = None
```

area title

## App

Bases: Page

multi-page app

### Attributes

`__default_template_path__` class-attribute

```
__default_template_path__: str = (
    f"{BASE_DIR}/templates/app.html"
)
```

api class-attribute

```
api: MSA_UI_API = None
```

Page configuration interface, please configure if you want to pull the page configuration remotely. Return the configuration path json>data>pages, please refer to the pages property for the exact format.

asideAfter class-attribute

```
asideAfter: str = None
```

The area under the page menu.

asideBefore class-attribute

```
asideBefore: str = None
```

The area in front of the page menu.

brandName class-attribute

```
brandName: str = None
```

Application name

className class-attribute

```
className: str = None
```

css class name

footer class-attribute

```
footer: str = None
```

The page.

header class-attribute

```
header: str = None
```

header

logo class-attribute

```
logo: str = '/msastatic/img/msa_logo.png'
```

Support image address, or svg.

pages class-attribute

```
pages: List[PageSchema] = None
```

Array specific page configuration. Usually an array, the first layer of the array is grouped, usually only the label set needs to be configured, if you do not want to group, directly not configured, the real pages please start configuring in the second layer, that is, the first layer of children.

type class-attribute

```
type: str = 'app'
```

## Audio

Bases: MSAUINode

audio

### Attributes

autoPlay class-attribute

autoPlay: `bool` = None

False # Whether to play automatically

className class-attribute

className: `str` = None

the class name of the outer Dom

controls class-attribute

controls: `List[str]` = None

"[rates,'play','time','process','volume']" # Internal module customization

inline class-attribute

inline: `bool` = None

True # whether inline mode is used

loop class-attribute

loop: `bool` = None

False # Whether to loop

rates class-attribute

rates: `List[float]` = None

[]" # Configurable audio playback multiplier e.g. [1.0, 1.5, 2.0]

src class-attribute

src: `str` = None

Audio address

type class-attribute

type: `str` = 'audio'

Specify as audio renderer

## Avatar

Bases: `MSAUINode`

## Avatar

### Attributes

className class-attribute

```
className: str = None
```

class name of the outer dom

fit class-attribute

```
fit: str = None
```

"cover" # image scaling type

icon class-attribute

```
icon: str = None
```

Icon

shape class-attribute

```
shape: str = None
```

"circle" # The shape, which can also be square

size class-attribute

```
size: int = None
```

40 # size

src class-attribute

```
src: str = None
```

Image address

style class-attribute

```
style: dict = None
```

The style of the outer dom

text class-attribute

```
text: str = None
```

Text

type class-attribute

```
type: str = 'avatar'
```

## Badge

Bases: `MSAUINode`

corner-icon

### Attributes

animation class-attribute

```
animation: bool = None
```

whether the corner is animated or not

className class-attribute

```
className: str = None
```

class name of outer dom

level class-attribute

```
level: str = None
```

Corner level, can be info/success/warning/danger, different background color after setting

mode class-attribute

```
mode: str = 'dot'
```

Corner type, can be dot/text/ribbon

offset class-attribute

```
offset: int = None
```

corner position, priority is greater than position, when offset is set, position is top-right as the base for positioning  
number[top, left]

overflowCount class-attribute

```
overflowCount: int = None
```

99 # Set the capping number value

**position** class-attribute

```
position: str = None
```

"top-right" # corner position, can be top-right/top-left/bottom-right/bottom-left

**size** class-attribute

```
size: int = None
```

corner size

**style** class-attribute

```
style: dict = None
```

custom style for the corner

**text** class-attribute

```
text: Union[str, int] = None
```

corner text, supports strings and numbers, invalid when set under mode='dot'

**visibleOn** class-attribute

```
visibleOn: MSAUIExpression = None
```

Control the display and hiding of the corner

## Button

Bases: FormItem

Button

### Attributes

**actionType** class-attribute

```
actionType: str = None
```

Set the button type 'button'|'reset'|'submit'|'clear'|'url'

**block** class-attribute

```
block: bool = None
```

option to adjust the width of the button to its parent width

className class-attribute

```
className: str = None
```

Specifies the class name of the added button

disabled class-attribute

```
disabled: bool = None
```

Disable button status

href class-attribute

```
href: str = None
```

Click on the address of the jump, specify this property button behavior and a link consistent

level class-attribute

```
level: LevelEnum = None
```

Set the button style 'link'|'primary'|'enhance'|'secondary'|'info'|'success'|'warning'|'danger'|'light'|'dark'|'default'

loading class-attribute

```
loading: bool = None
```

Show button loading effect

loadingOn class-attribute

```
loadingOn: str = None
```

Show button loading expressions

size class-attribute

```
size: str = None
```

Set the size of the button 'xs'|'sm'|'md'|'lg'

tooltip class-attribute

```
tooltip: Union[str, dict] = None
```

bubble tip content TooltipObject

tooltipPlacement class-attribute



tooltipPlacement: str = None

bubblePlacement 'top'|'right'|'bottom'|'left'

tooltipTrigger class-attribute

tooltipTrigger: str = None

trigger tooltip 'hover'|'focus'

ButtonGroup

Bases: MSAUINode

buttonGroup

Attributes

buttons class-attribute

buttons: List[Action]

Behavior button group

className class-attribute

className: str = None

the class name of the outer Dom

type class-attribute

type: str = 'button-group'

vertical class-attribute

vertical: bool = None

whether to use vertical mode

ButtonToolbar

Bases: MSAUINode

ButtonToolbar

Attributes

buttons class-attribute

```
buttons: List[Action]
```

Behavior button group

type class-attribute

```
type: str = 'button-toolbar'
```

## CRUD

Bases: MSAUINode

add-delete

### Attributes

affixHeader class-attribute

```
affixHeader: bool = None
```

True # Whether to fix the table header (under table)

alwaysShowPagination class-attribute

```
alwaysShowPagination: bool = None
```

whether to always show pagination

api class-attribute

```
api: MSA_UI_API = None
```

The api used by CRUD to get the list data.

autoGenerateFilter class-attribute

```
autoGenerateFilter: bool = None
```

Whether to enable the query area, which will automatically generate a query form based on the value of the searchable property of the column element

autoJumpToTopOnPagerChange class-attribute

```
autoJumpToTopOnPagerChange: bool = None
```

Whether to auto jump to the top when the page is cut.

bulkActions class-attribute

```
bulkActions: List[Action] = None
```

List of bulk actions, configured so that the form can be checked.

className class-attribute

```
className: str = None
```

class name of the table's outer Dom

defaultChecked class-attribute

```
defaultChecked: bool = None
```

Default whether to check all when bulk actions are available.

defaultParams class-attribute

```
defaultParams: dict = None
```

Set the default filter default parameters, which will be sent to the backend together with the query

draggable class-attribute

```
draggable: bool = None
```

Whether to sort by drag and drop

filter class-attribute

```
filter: Union[MSAUISchemaNode, Form] = None
```

Set a filter that will bring the data to the current mode to refresh the list when the form is submitted.

filterDefaultVisible class-attribute

```
filterDefaultVisible: bool = None
```

True # Sets whether the filter is visible by default.

filterToggleable class-attribute

```
filterToggleable: bool = None
```

False # Whether to make the filter visible or invisible

footerToolbar class-attribute

```
footerToolbar: list = None
```

[statistics';pagination] # Bottom toolbar configuration

headerToolBar class-attribute

```
headerToolBar: list = None
```

['bulkActions';pagination] # top toolbar configuration

hideQuickSaveBtn class-attribute

```
hideQuickSaveBtn: bool = None
```

Hide the top quick save prompt

initFetch class-attribute

```
initFetch: bool = None
```

True # Whether to pull data when initializing, only for cases with filter, no filter will pull data initially

interval class-attribute

```
interval: int = None
```

Refresh time (minimum 1000)

itemAction class-attribute

```
itemAction: Action = None
```

Implement a custom action when a row is clicked, supports all configurations in action, such as pop-up boxes, refreshing other components, etc.

itemDraggableOn class-attribute

```
itemDraggableOn: bool = None
```

Use an expression to configure whether draggable is sortable or not

keepItemSelectionOnPageChange class-attribute

```
keepItemSelectionOnPageChange: bool = None
```

True

labelTpl class-attribute

```
labelTpl: str = None
```

Single description template, keepItemSelectionOnPageChange

loadDataOnce class-attribute

```
loadDataOnce: bool = None
```

Whether to load all data at once (front-end paging)

loadDataOnceFetchOnFilter class-attribute

```
loadDataOnceFetchOnFilter: bool = None
```

True # Whether to re-request the api when filtering when loadDataOnce is enabled

messages class-attribute

```
messages: Messages = None
```

Override the message prompt, if not specified, the message returned by the api will be used

mode class-attribute

```
mode: str = None
```

"table" # "table", "cards" or "list"

orderField class-attribute

```
orderField: str = None
```

Set the name of the field used to determine the position, after setting the new order will be assigned to the field.

pageField class-attribute

```
pageField: str = None
```

Set the pagination page number field name. "page"

perPage class-attribute

```
perPage: int = None
```

Set how many data to display on a page. 10

perPageAvailable class-attribute

```
perPageAvailable: List[int] = None
```

[5, 10, 20, 50, 100] # Set how many data dropdown boxes are available for displaying on a page.

**perPageField** class-attribute

```
perPageField: str = None
```

"perPage" # Set the field name of how many data to display on a paginated page. Note: Best used in conjunction with defaultParams, see the following example.

**primaryField** class-attribute

```
primaryField: str = None
```

Set the ID field name.'id'

**quickSaveApi** class-attribute

```
quickSaveApi: MSA_UI_API = None
```

The MSA\_UI\_API used for batch saving after quick editing.

**quickSaveItemApi** class-attribute

```
quickSaveItemApi: MSA_UI_API = None
```

The MSA\_UI\_API used when the quick edit is configured to save in time.

**saveOrderApi** class-attribute

```
saveOrderApi: MSA_UI_API = None
```

The api to save the sorting.

**silentPolling** class-attribute

```
silentPolling: bool = None
```

Configure whether to hide loading animation when refreshing

**source** class-attribute

```
source: str = None
```

Data mapping interface to return the value of a field, not set will default to use the interface to return \${items} or \${rows}, can also be set to the content of the upper-level data source

**stopAutoRefreshWhen** class-attribute

```
stopAutoRefreshWhen: str = None
```

Configure the conditions for stopping the refresh via an expression

**stopAutoRefreshWhenModallsOpen** class-attribute

```
stopAutoRefreshWhenModalIsOpen: bool = None
```

Turn off auto refresh when there is a popup box, and resume when the popup box is closed

**syncLocation** class-attribute

```
syncLocation: bool = None
```

False # Whether to sync the parameters of the filter condition to the address bar, !!! !!! may change the data type after turning on, can't pass fastpi data verification

**syncResponse2Query** class-attribute

```
syncResponse2Query: bool = None
```

True # Sync the return data to the filter.

**title** class-attribute

```
title: str = None
```

"" # can be set to empty, when set to empty, there is no title bar

**type** class-attribute

```
type: str = 'crud'
```

type specifies the CRUD renderer

**Classes****Messages**

Bases: MSAUINode

**Attributes****fetchFailed** class-attribute

```
fetchFailed: str = None
```

prompt when fetch fails

**quickSaveFailed** class-attribute

```
quickSaveFailed: str = None
```

prompt for quick save failure

**quickSaveSuccess** class-attribute`quickSaveSuccess: str = None`

QuickSaveSuccess hint

**saveOrderFailed** class-attribute`saveOrderFailed: str = None`

Hint for failed save order

**saveOrderSuccess** class-attribute`saveOrderSuccess: str = None`

prompt for order success

## Carousel

Bases: MSAUINode

Rotating image

### Attributes

**animation** class-attribute`animation: str = None`

"fade" # Toggle animation effect, default fade, also slide mode

**auto** class-attribute`auto: bool = True`

whether to rotate automatically

**className** class-attribute`className: str = None`

"panel-default" # class name of outer Dom

**controls** class-attribute`controls: list[str] = None`

["dots","arrows"] # Show left and right arrows, bottom dots index

**controlsTheme** class-attribute



```
controlsTheme: str = None
```

"light" # Color of left and right arrows, bottom dot index, default light, dark mode available

duration class-attribute

```
duration: str = None
```

"0.5s" # the duration of the toggle animation

height class-attribute

```
height: str = None
```

"200px" # height

interval class-attribute

```
interval: str = None
```

"5s" # toggle animation interval

itemSchema class-attribute

```
itemSchema: dict = None
```

Custom schema to display data

options class-attribute

```
options: List[Item] = None
```

[] # Rotating panel data

thumbMode class-attribute

```
thumbMode: str = None
```

"cover"|"contain" # default image zoom mode

type class-attribute

```
type: str = 'carousel'
```

Specify as Carousel renderer

width class-attribute

```
width: str = None
```

"auto" # width

Classes

Item

Bases: MSAUINode

Attributes

description class-attribute

description: str = None

Image description

descriptionClassName class-attribute

descriptionClassName: str = None

Image description class name

href class-attribute

href: str = None

link to the image's open URL

html class-attribute

html: str = None

HTML customization, same as Tpl

image class-attribute

image: str = None

Image link

imageClassName class-attribute

imageClassName: str = None

Image class name

title class-attribute

title: str = None

Image title

titleClassName class-attribute

```
titleClassName: str = None
```

Image title class name

## Chart

Bases: `MSAUINode`

chart: <https://echarts.apache.org/zh/option.html#title>

### Attributes

api class-attribute

```
api: MSA_UI_API = None
```

Configuration item interface address

body class-attribute

```
body: MSAUISchemaNode = None
```

Content container

className class-attribute

```
className: str = None
```

class name of the outer Dom

config class-attribute

```
config: Union[dict, str] = None
```

Set the configuration of eschars, when it is string, you can set function and other configuration items

height class-attribute

```
height: str = None
```

Set the height of the root element

initFetch class-attribute

```
initFetch: bool = None
```

Whether to request the interface when the component is initialized

interval class-attribute

```
interval: int = None
```

Refresh time (min 1000)

replaceChartOption class-attribute

```
replaceChartOption: bool = None
```

False # Does each update completely override the configuration item or append it?

SOURCE class-attribute

```
source: dict = None
```

Get the value of a variable in the data chain as a configuration via data mapping

style class-attribute

```
style: dict = None
```

Set the style of the root element

trackExpression class-attribute

```
trackExpression: str = None
```

Update the chart when the value of this expression has changed

type class-attribute

```
type: str = 'chart'
```

Specify as chart renderer

width class-attribute

```
width: str = None
```

Set the width of the root element

## ChartRadios

Bases: Radios

radio box

### Attributes

chartValueField class-attribute

```
chartValueField: str = None
```

"value" # chart value field name

config class-attribute

```
config: dict = None
```

echart chart configuration

showTooltipOnHighlight class-attribute

```
showTooltipOnHighlight: bool = None
```

False # whether to show tooltip when highlighted

type class-attribute

```
type: str = 'chart-radios'
```

## Checkbox

Bases: FormItem

Checkbox

### Attributes

falseValue class-attribute

```
falseValue: Any = None
```

Identifies a false value

option class-attribute

```
option: str = None
```

option description

trueValue class-attribute

```
trueValue: Any = None
```

Identifies the true value

type class-attribute

```
type: str = 'checkbox'
```

## Checkboxes

Bases: `FormItem`

checkboxes

### Attributes

`addApi` class-attribute

```
addApi: MSA_UI_API = None
```

Configure the add options interface

`addControls` class-attribute

```
addControls: List[FormItem] = None
```

Customize the new form item

`checkAll` class-attribute

```
checkAll: bool = None
```

False # If or not checkAll is supported

`columnsCount` class-attribute

```
columnsCount: int = None
```

1 # How many columns to display options by, default is one column

`creatable` class-attribute

```
creatable: bool = None
```

False # New option

`createBtnLabel` class-attribute

```
createBtnLabel: str = None
```

"Add option" # Add option

`defaultCheckAll` class-attribute

```
defaultCheckAll: bool = None
```

False # Whether to check all by default

**deleteApi** class-attribute

```
deleteApi: MSA_UI_API = None
```

Configure the delete option interface

**delimiter** class-attribute

```
delimiter: str = None
```

"," # Splice character

**editApi** class-attribute

```
editApi: MSA_UI_API = None
```

Configure the edit options interface

**editControls** class-attribute

```
editControls: List[FormItem] = None
```

Customize edit form items

**editable** class-attribute

```
editable: bool = None
```

False # Edit options

**extractValue** class-attribute

```
extractValue: bool = None
```

False # extract value

**inline** class-attribute

```
inline: bool = None
```

True # Whether to display as one line

**joinValues** class-attribute

```
joinValues: bool = None
```

True # splice values

**labelField** class-attribute

```
labelField: str = None
```

"label" # option label field

options class-attribute

```
options: MSAOptionsNode = None
```

Options group

removable class-attribute

```
removable: bool = None
```

False # Remove options

source class-attribute

```
source: MSA_UI_API = None
```

dynamic options group

type class-attribute

```
type: str = 'checkboxes'
```

valueField class-attribute

```
valueField: str = None
```

"value" # option value field

## Code

Bases: MSAUINode

Code highlighting

### Attributes

className class-attribute

```
className: str = None
```

Outer CSS class name

editorTheme class-attribute

```
editorTheme: str = None
```



"vs" # theme, and 'vs-dark'

language class-attribute

```
language: str = None
```

The highlighting language used, default is plaintext

name class-attribute

```
name: str = None
```

Used as a variable mapping when in other components

tabSize class-attribute

```
tabSize: int = None
```

4 # Default tab size

type class-attribute

```
type: str = 'code'
```

value class-attribute

```
value: str = None
```

The value of the displayed color

wordWrap class-attribute

```
wordWrap: str = None
```

"True" # whether to wrap the line

## ColumnImage

Bases: `Image`, `TableColumn`

Image Column

## ColumnImages

Bases: `Images`, `TableColumn`

Image collection Column

## ColumnOperation

Bases: `TableColumn`

operationColumn

### Attributes

buttons `class-attribute`

```
buttons: List[Union[Action, MSAUINode]] = None
```

label `class-attribute`

```
label: MSAUITemplate = None
```

"operation"

toggled `class-attribute`

```
toggled: bool = None
```

True

type `class-attribute`

```
type: str = 'operation'
```

## Combo

Bases: `FormItem`

combo

### Attributes

addButtonClassName `class-attribute`

```
addButtonClassName: str = None
```

Add button CSS class name

addButtonText `class-attribute`

```
addButtonText: str = None
```

"Add" # Add button text

addable `class-attribute`

```
addable: bool = False
```

Whether to add

canAccessSuperData class-attribute

```
canAccessSuperData: bool = False
```

Specifies whether the data from the upper level can be automatically fetched and mapped to form items

conditions class-attribute

```
conditions: dict = None
```

Array of rendering types containing all conditions, test in a single array is the judgment condition, items in the array are the schema rendered when the condition is met

deleteApi class-attribute

```
deleteApi: MSA_UI_API = None
```

If configured, an api will be sent before deletion, and the deletion will be completed only if the request is successful

deleteConfirmText class-attribute

```
deleteConfirmText: str = None
```

"Confirm to delete?"

delimiter class-attribute

```
delimiter: str = None
```

"False" # What delimiter to use when flattening is on and joinValues is true.

draggable class-attribute

```
draggable: bool = False
```

whether draggable sorting is possible, note that when draggable sorting is enabled, there will be an additional \$id field

draggableTip class-attribute

```
draggableTip: str = None
```

"Can be reordered by dragging the [swap] button in each row" # Text to indicate draggable

flat class-attribute

```
flat: bool = False
```

Whether to flatten the results (remove the name), only valid if the length of items is 1 and multiple is true.

formClassName class-attribute

```
formClassName: str = None
```

class name of a single group of form items

items class-attribute

```
items: List[FormItem] = None
```

Combined form items to be displayed items[x].columnName: str = None # The class name of the column with which to configure the column width. Default is evenly distributed. items[x].unique: bool = None # Set whether the current column value is unique, i.e. no duplicate selections are allowed.

joinValues class-attribute

```
joinValues: bool = True
```

Defaults to true when flattening is on, whether to send to the backend as a delimiter, otherwise as an array.

maxLength class-attribute

```
maxLength: int = None
```

The maximum number of items to add

minLength class-attribute

```
minLength: int = None
```

The minimum number of items to add

multiLine class-attribute

```
multiLine: bool = False
```

Default is to display a row horizontally, set it to display vertically

multiple class-attribute

```
multiple: bool = False
```

Whether or not to multi-select

noBorder class-attribute

```
noBorder: bool = False
```

Whether to show border for a single group of table items

nullable class-attribute

```
nullable: bool = False
```

Allow nullable, if the validator is configured inside the child form item and it is in single entry mode. Can allow the user to choose to clear (not filled).

placeholder class-attribute

```
placeholder: str = None
```

""" # Show when there is no member

removable class-attribute

```
removable: bool = False
```

If or not it can be removed

scaffold class-attribute

```
scaffold: dict = {}
```

The initial value of a single table item

strictMode class-attribute

```
strictMode: bool = True
```

Default is strict mode, when set to false, when other table items are updated, the table items inside can also be retrieved in time, otherwise it will not.

subFormMode class-attribute

```
subFormMode: str = None
```

"normal" # Optional normal, horizontal, inline

syncFields class-attribute

```
syncFields: List[str] = []
```

Configure sync fields. Only valid if strictMode is false.

type class-attribute

```
type: str = 'combo'
```

typeSwitchable class-attribute

```
typeSwitchable: bool = False
```

whether the condition is switchable, used with conditions

ConditionBuilder

Bases: FormItem

Combined Condition

Attributes

className class-attribute

```
className: str = None
```

outer dom class name

fieldClassName class-attribute

```
fieldClassName: str = None
```

The class name of the input field

fields class-attribute

```
fields: List[Field] = None
```

is an array type, each member represents an optional field, supports multiple layers, configuration example

source class-attribute

```
source: str = None
```

pull configuration items via remote

type class-attribute

```
type: str = 'condition-builder'
```

Classes

Date

Bases: Field

date

Attributes

defaultValue class-attribute

```
defaultValue: str = None
```

default value

format class-attribute

```
format: str = None
```

default "YYYY-MM-DD" value format

inputFormat class-attribute

```
inputFormat: str = None
```

Default "YYYY-MM-DD" date format for display.

type class-attribute

```
type: str = 'date'
```

## Datetime

Bases: Date

datetime

### Attributes

timeFormat class-attribute

```
timeFormat: str = None
```

Default "HH:mm" time format, determines which input boxes are available.

type class-attribute

```
type: str = 'datetime'
```

## Field

Bases: MSAUINode

### Attributes

defaultOp class-attribute

```
defaultOp: str = None
```

default to "equal"

**label** class-attribute

```
label: str = None
```

Name of the field.

**operators** class-attribute

```
operators: list[str] = None
```

Configure to override if you don't want that many.

**placeholder** class-attribute

```
placeholder: str = None
```

Placeholder

**type** class-attribute

```
type: str = 'text'
```

configured as "text" in the field configuration

## Number

Bases: Field

number

### Attributes

**maximum** class-attribute

```
maximum: float = None
```

maximum value

**minimum** class-attribute

```
minimum: float = None
```

minimum

**step** class-attribute

```
step: float = None
```

step length

**type** class-attribute

```
type: str = 'number'
```



## Select

Bases: `Field`

Dropdown selection

### Attributes

`autoComplete` class-attribute

```
autoComplete: MSA_UI_API = None
```

AutoComplete will be called after each new input, and will return updated options according to the interface.

`options` class-attribute

```
options: MSAOptionsNode = None
```

list of options, `Array<{label: string, value: any}>`

`searchable` class-attribute

```
searchable: bool = None
```

If or not searchable

`source` class-attribute

```
source: MSA_UI_API = None
```

Dynamic options, please configure api.

`type` class-attribute

```
type: str = 'select'
```

## Text

Bases: `Field`

text

## Time

Bases: `Date`

time

### Attributes

`type` class-attribute

```
type: str = 'datetime'
```

## Custom

Bases: `MSAUINode`

custom component

### Attributes

`className` class-attribute

```
className: str = None
```

node class

`html` class-attribute

```
html: str = None
```

Initialize node html

`id` class-attribute

```
id: str = None
```

node id

`inline` class-attribute

```
inline: bool = False
```

default use div tag, if true use span tag

`name` class-attribute

```
name: str = None
```

node name

`onMount` class-attribute

```
onMount: str = None
```

"Function" # The function to call after the node is initialized

`onUnmount` class-attribute

```
onUnmount: str = None
```

"Function" # The function called when the node is destroyed

onUpdate class-attribute

```
onUpdate: str = None
```

"Function" # Function to be called when data is updated

type class-attribute

```
type: str = 'custom'
```

## Dialog

Bases: MSAUINode

Dialog

### Attributes

actions class-attribute

```
actions: List[Action] = None
```

If you want to not show the bottom button, you can configure: [] "[Confirm] and [Cancel]"

body class-attribute

```
body: MSAUISchemaNode = None
```

Add content to the Dialog content area

bodyClassName class-attribute

```
bodyClassName: str = None
```

"modal-body" # The style class name of the Dialog body area

closeOnEsc class-attribute

```
closeOnEsc: bool = None
```

False # Whether to close the Dialog by pressing Esc

data class-attribute

```
data: dict = None
```

Support data mapping, if not set will default to inherit data from the context of the trigger button.

disabled class-attribute

`disabled: bool = None`

False # If this property is set, the Dialog is read only and no action is submitted.

`showCloseButton` class-attribute

`showCloseButton: bool = None`

True # Whether to show the close button in the upper right corner

`showErrorMsg` class-attribute

`showErrorMsg: bool = None`

True # Whether to show the error message in the lower left corner of the popup box

`size` class-attribute

`size: Union[str, SizeEnum] = None`

Specify dialog size, supports: xs, sm, md, lg, xl, full

`title` class-attribute

`title: MSAUISchemaNode = None`

popup layer title

`type` class-attribute

`type: str = 'dialog'`

Specify as Dialog renderer

## Divider

Bases: MSAUINode

### Attributes

`className` class-attribute

`className: str = None`

class name of the outer Dom

`lineStyle` class-attribute

`lineStyle: str = None`

The style of the divider line, supports dashed and solid

type class-attribute

```
type: str = 'divider'
```

Divider

## Drawer

Bases: MSAUINode

drawer

### Attributes

actions class-attribute

```
actions: List[Action] = None
```

Can be set without, only two buttons by default. "[Confirm] and [Cancel]"

body class-attribute

```
body: MSAUISchemaNode = None
```

Add content to the Drawer content area

bodyClassName class-attribute

```
bodyClassName: str = None
```

"modal-body" # The style class name of the Drawer body area

closeOnEsc class-attribute

```
closeOnEsc: bool = None
```

False # Whether or not to support closing the Drawer by pressing Esc

closeOnOutside class-attribute

```
closeOnOutside: bool = None
```

False # Whether to close the Drawer by clicking outside the content area

data class-attribute

```
data: dict = None
```

Support data mapping, if not set will default to inherit data from the context of the trigger button.

overlay class-attribute

```
overlay: bool = None
```

True # Whether or not to show the mask

position class-attribute

```
position: str = None
```

'left' # Position

resizable class-attribute

```
resizable: bool = None
```

False # Whether the Drawer size can be changed by dragging and dropping

size class-attribute

```
size: Union[str, SizeEnum] = None
```

Specify Drawer size, supports: xs, sm, md, lg

title class-attribute

```
title: MSAUISchemaNode = None
```

popup layer title

type class-attribute

```
type: str = 'drawer'
```

"drawer" is specified as the Drawer renderer

## Editor

Bases: FormItem

Code Editor

### Attributes

allowFullscreen class-attribute

```
allowFullscreen: bool = None
```

False # switch to show full screen mode or not

language class-attribute

```
language: str = None
```

"javascript" # Language highlighted by the editor, supported by the \${xxx} variable bat, c, coffeescript, cpp, csharp, css, dockerfile, fsharp, go, handlebars, html, ini, java javascript, json, less, lua, markdown, msdax, objective-c, php, plaintext, postits, powershell, pug, python, r, razor, ruby, sb, scss, shell, sol, sql, swift, typescript, vb, xml, yaml

options class-attribute

```
options: dict = None
```

other configurations of monaco editor, such as whether to display line numbers, etc., please refer to here, but can not set readOnly, read-only mode need to use disabled: true

size class-attribute

```
size: str = None
```

"md" # editor height, can be md, lg, xl, xxl

type class-attribute

```
type: str = 'editor'
```

## Flex

Bases: MSAUINode

### Attributes

alignItems class-attribute

```
alignItems: str = None
```

"stretch", "start", "flex-start", "flex-end", "end", "center", "baseline"

className class-attribute

```
className: str = None
```

css class name

items class-attribute

```
items: List[MSAUISchemaNode] = None
```

**justify** class-attribute`justify: str = None`

"start", "flex-start", "center", "end", "flex-end", "space-around", "space-between", "space-evenly"

**style** class-attribute`style: dict = None`

Custom style

**type** class-attribute`type: str = 'flex'`

Specify as Flex renderer

## Form

**Bases:** MSAUINode**Form**

### Attributes

**actions** class-attribute`actions: List[Action] = None`

Form submit button, member of Action

**actionsClassName** class-attribute`actionsClassName: str = None`

Class name of actions

**api** class-attribute`api: MSA_UI_API = None`

The api used by Form to save data.

**asyncApi** class-attribute`asyncApi: MSA_UI_API = None`

After this property is set, the form will continue to poll the interface after it is submitted and sent to the saved interface until the finished property is returned as true.



**autoFocus** class-attribute

```
autoFocus: bool = None
```

If or not autoFocus is enabled.

**body** class-attribute

```
body: List[Union[FormItem, MSAUISchemaNode]] = None
```

Form form item collection

**canAccessSuperData** class-attribute

```
canAccessSuperData: bool = None
```

Specifies whether the data from the upper level can be automatically retrieved and mapped to the form item.

**checkInterval** class-attribute

```
checkInterval: int = None
```

The time interval to poll the request, default is 3 seconds. Set asyncApi to be valid

**className** class-attribute

```
className: str = None
```

The class name of the outer Dom

**clearPersistDataAfterSubmit** class-attribute

```
clearPersistDataAfterSubmit: bool = None
```

Specify whether to clear the local cache after a successful form submission

**columnCount** class-attribute

```
columnCount: int = None
```

How many columns are displayed for the form item

**debug** class-attribute

```
debug: bool = None
```

**finishedField** class-attribute

```
finishedField: Optional[str] = None
```

Set this property if the field name that determines the finish is not finished, e.g. `is_success`

horizontal class-attribute

```
horizontal: Horizontal = None
```

Useful when mode is horizontal.

initApi class-attribute

```
initApi: MSA_UI_API = None
```

The api used by Form to get the initial data.

initAsyncApi class-attribute

```
initAsyncApi: MSA_UI_API = None
```

The api used by Form to get the initial data, unlike `initApi`, it will keep polling the request until the `finished` property is returned as true.

initCheckInterval class-attribute

```
initCheckInterval: int = None
```

After setting `initAsyncApi`, the default time interval for pulling

initFetch class-attribute

```
initFetch: bool = None
```

When `initApi` or `initAsyncApi` is set, the request will start by default, but when set to false, the interface will not be requested from the beginning.

initFetchOn class-attribute

```
initFetchOn: str = None
```

Use an expression to configure

initFinishedField class-attribute

```
initFinishedField: Optional[str] = None
```

When `initAsyncApi` is set, the default is to determine if the request is completed by returning `data.finished`.

interval class-attribute

```
interval: int = None
```

Refresh time (minimum 3000)

messages class-attribute

```
messages: Messages = None
```

Message prompt override, the default message reads the message returned by MSA\_UI\_API, but it can be overridden here.

mode class-attribute

```
mode: DisplayModeEnum = None
```

How the form is displayed, either: normal, horizontal or inline

name class-attribute

```
name: str = None
```

Set a name so that other components can communicate with it

panelClassName class-attribute

```
panelClassName: str = None
```

The class name of the outer panel

persistData class-attribute

```
persistData: str = None
```

Specify a unique key to configure whether to enable local caching for the current form

preventEnterSubmit class-attribute

```
preventEnterSubmit: bool = None
```

Disable carriage return to submit the form

primaryField class-attribute

```
primaryField: str = None
```

Set primary key id, when set, only carry this data when detecting form completion (asyncApi).

promptPageLeave class-attribute

```
promptPageLeave: bool = None
```

whether the form is not yet saved and will pop up before leaving the page to confirm.

redirect class-attribute

```
redirect: str = None
```

When this property is set, the Form will automatically jump to the specified page after a successful save. Supports relative addresses, and absolute addresses (relative to the group).

reload class-attribute

```
reload: str = None
```

Refresh the target object after the operation. Please fill in the name value set by the target component, if you fill in the name of window, the current page will be refreshed as a whole.

resetAfterSubmit class-attribute

```
resetAfterSubmit: bool = None
```

whether to reset the form after submission

rules class-attribute

```
rules: list = None
```

Form combination check rules Array<{rule:string;message:string}>

silentPolling class-attribute

```
silentPolling: bool = False
```

Configure whether to show loading animation when refreshing

stopAutoRefreshWhen class-attribute

```
stopAutoRefreshWhen: str = None
```

Configure the conditions for stopping the refresh via an expression

submitOnChange class-attribute

```
submitOnChange: bool = None
```

The form is submitted when it is modified

submitOnInit class-attribute

```
submitOnInit: bool = None
```

Submit once at the beginning

submitText class-attribute

```
submitText: Optional[str] = None
```

"submit" # The default submit button name, if set to null, the default button can be removed.

target class-attribute

```
target: str = None
```

The default form submission saves the data itself by sending the api, but you can set the name value of another form or the name value of another CRUD model. If the target is a Form, the target Form will retrigger initApi and the api will get the current form data. If the target is a CRUD model, the target model retriggers the search with the current Form data as the argument. When the target is a window, the current form data will be attached to the page address.

title class-attribute

```
title: Optional[str] = None
```

Title of the Form

trimValues class-attribute

```
trimValues: bool = None
```

trim each value of the current form item

type class-attribute

```
type: str = 'form'
```

"form" is specified as a Form renderer

wrapWithPanel class-attribute

```
wrapWithPanel: bool = None
```

Whether to let Form wrap with panel, set to false and actions will be invalid.

## Classes

## Messages

Bases: MSAUINode

### Attributes

**fetchFailed** class-attribute`fetchFailed: str = None`

prompt for fetch failure

**fetchSuccess** class-attribute`fetchSuccess: str = None`

Prompt when fetch succeeds

**saveFailed** class-attribute`saveFailed: str = None`

Prompt for failed save

**saveSuccess** class-attribute`saveSuccess: str = None`

Prompt for successful save

## FormItem

Bases: MSAUINode

FormItemGeneral

### Attributes

**className** class-attribute`className: str = None`

The outermost class name of the form

**copyable** class-attribute`copyable: Union[bool, dict] = None`

Whether copyable boolean or {icon: string, content:string}

**description** class-attribute`description: MSAUITemplate = None`

Form item description

**disabled** class-attribute

```
disabled: bool = None
```

Whether the current form item is disabled or not

disabledOn class-attribute

```
disabledOn: MSAUIExpression = None
```

The condition whether the current form item is disabled or not

inline class-attribute

```
inline: bool = None
```

Whether inline mode

inputClassName class-attribute

```
inputClassName: str = None
```

Form controller class name

label class-attribute

```
label: MSAUITemplate = None
```

Label of the form item Template or false

labelClassName class-attribute

```
labelClassName: str = None
```

Class name of the label

labelRemark class-attribute

```
labelRemark: Remark = None
```

Description of the form item label

name class-attribute

```
name: str = None
```

Field name, specifying the key of the form item when it is submitted

placeholder class-attribute

```
placeholder: str = None
```

Form item description

required class-attribute

```
required: bool = None
```

Whether or not it is required.

requiredOn class-attribute

```
requiredOn: MSAUIExpression = None
```

Expression to configure if the current form entry is required.

submitOnChange class-attribute

```
submitOnChange: bool = None
```

Whether to submit the current form when the value of the form item changes.

type class-attribute

```
type: str = 'input-text'
```

Specify the form item type

validateApi class-attribute

```
validateApi: MSAUIExpression = None
```

Form validation interface

validations class-attribute

```
validations: Union[Validation, MSAUIExpression] = None
```

Form item value format validation, support setting multiple, multiple rules separated by English commas.

value class-attribute

```
value: Union[int, str] = None
```

The value of the field

visible class-attribute

```
visible: MSAUIExpression = None
```

The condition whether the current form item is disabled or not



visibleOn class-attribute

```
visibleOn: MSAUIExpression = None
```

The condition if the current table item is disabled or not

## Grid

Bases: MSAUINode

### Attributes

align class-attribute

```
align: str = None
```

'left'|'right'|'between'|'center' = None # horizontal alignment

className class-attribute

```
className: str = None
```

Class name of the outer Dom

columns class-attribute

```
columns: List[MSAUISchemaNode] = None
```

gap class-attribute

```
gap: str = None
```

'xs'|'sm'|'base'|'none'|'md'|'lg' = None # Horizontal spacing

type class-attribute

```
type: str = 'grid'
```

Specify as Grid renderer

valign class-attribute

```
valign: str = None
```

'top'|'middle'|'bottom'|'between' = None # Vertical alignment

### Classes

### Column

Bases: `MSAUINode`

**Attributes**

ClassName `class-attribute`

ClassName: `str` = None

Column class name

body `class-attribute`

body: `List[MSAUISchemaNode]` = None

lg `class-attribute`

lg: `int` = None

"auto" # Width ratio: 1 - 12

md `class-attribute`

md: `int` = None

"auto" # Width ratio: 1 - 12

sm `class-attribute`

sm: `int` = None

"auto" # Width ratio: 1 - 12

valign `class-attribute`

valign: `str` = None

'top'|'middle'|'bottom'|'between' = None # Vertical alignment of current column content

xs `class-attribute`

xs: `int` = None

"auto" # Width percentage: 1 - 12

**Group**

Bases: `InputGroup`

Form Item Group

**Attributes**

direction `class-attribute`

```
direction: str = None
```

"horizontal" # You can configure whether to display horizontally or vertically. The corresponding configuration items are: vertical, horizontal

gap class-attribute

```
gap: str = None
```

spacing between form items, optional: xs, sm, normal

mode class-attribute

```
mode: DisplayModeEnum = None
```

Display default, same mode as in Form

type class-attribute

```
type: str = 'group'
```

## Hidden

Bases: FormItem

hiddenField

### Attributes

type class-attribute

```
type: str = 'hidden'
```

## Horizontal

Bases: MSAUINode

### Attributes

left class-attribute

```
left: int = None
```

The width of the left label as a percentage

offset class-attribute

```
offset: int = None
```

The offset of the right controller when no label is set

right class-attribute

```
right: int = None
```

The width share of the right controller.

## Html

Bases: MSAUINode

Html Node

### Attributes

html class-attribute

```
html: str
```

html Use MSAUITpl when you need to get variables in the data field.

type class-attribute

```
type: str = 'html'
```

Specify as html component

## Icon

Bases: MSAUINode

icon

### Attributes

className class-attribute

```
className: str = None
```

Outer CSS class name

icon class-attribute

```
icon: str = None
```

icon name, supports fontawesome v4 or use url

type class-attribute

```
type: str = 'icon'
```

Specify the component type

## Iframe

Bases: `MSAUINode`

Iframe

### Attributes

`className` class-attribute

```
className: str = None
```

The class name of the iFrame

`frameBorder` class-attribute

```
frameBorder: list = None
```

frameBorder

`height` class-attribute

```
height: Union[int, str] = None
```

"100%" # iframe height

`src` class-attribute

```
src: str = None
```

iframe address

`style` class-attribute

```
style: dict = None
```

Style object

`type` class-attribute

```
type: str = 'iframe'
```

Specify as iFrame renderer

`width` class-attribute

```
width: Union[int, str] = None
```

"100%" # iframe width

# Image

Bases: MSAUINode

image

## Attributes

className class-attribute

className: str = None

Outer CSS class name

defaultImage class-attribute

defaultImage: str = None

Image to display when no data is available

enlargeAble class-attribute

enlargeAble: bool = None

Support for enlarge preview

enlargeCaption class-attribute

enlargeCaption: str = None

Description of the enlarged preview

enlargeTitle class-attribute

enlargeTitle: str = None

The title of the enlarged preview

height class-attribute

height: int = None

Image thumbnail height

href class-attribute

href: MSAUITemplate = None

External link address

imageCaption class-attribute

```
imageCaption: str = None
```

description

imageClassName class-attribute

```
imageClassName: str = None
```

Image CSS class name

imageMode class-attribute

```
imageMode: str = None
```

"thumb" # Image display mode, optional: 'thumb', 'original' i.e.: thumbnail mode or original image mode

originalSrc class-attribute

```
originalSrc: str = None
```

Original image address

placeholder class-attribute

```
placeholder: str = None
```

Placeholder text

src class-attribute

```
src: str = None
```

Thumbnail address

thumbClassName class-attribute

```
thumbClassName: str = None
```

Image thumbnail CSS class name

thumbMode class-attribute

```
thumbMode: str = None
```

"contain" # Preview mode, optional: 'w-full', 'h-full', 'contain', 'cover'

thumbRatio class-attribute`thumbRatio: str = None`

"1:1" # The ratio of the preview image, optional: '1:1', '4:3', '16:9'

title class-attribute`title: str = None`

title

type class-attribute`type: str = 'image'`

"image" if in Table, Card and List; "static-image" if used as a static display in Form

width class-attribute`width: int = None`

Image scaling width

## Images

Bases: MSAUINode

images collection

### Attributes

className class-attribute`className: str = None`

Outer CSS class name

defaultImage class-attribute`defaultImage: str = None`

Default image to display

delimiter class-attribute`delimiter: str = None`

"," # separator to split when value is a string



**enlargeAble** class-attribute

```
enlargeAble: bool = None
```

Support enlarge preview

**originalSrc** class-attribute

```
originalSrc: str = None
```

The address of the original image, supports data mapping to get the image variables in the object

**SOURCE** class-attribute

```
source: str = None
```

data source

**SRC** class-attribute

```
src: str = None
```

Address of the preview image, supports data mapping to get the image variables in the object

**thumbMode** class-attribute

```
thumbMode: str = None
```

"contain" # preview image mode, optional: 'w-full', 'h-full', 'contain', 'cover'

**thumbRatio** class-attribute

```
thumbRatio: str = None
```

"1:1" # Preview ratio, optional: '1:1', '4:3', '16:9'

**type** class-attribute

```
type: str = 'images'
```

"images" if in Table, Card and List; "static-images" if used as a static display in Form

**value** class-attribute

```
value: Union[str, List[str], List[dict]] = None
```

array of images

## InputArray

Bases: `FormItem`

`arrayInputArray`

### Attributes

`addButtonText` class-attribute

```
addButtonText: str = None
```

"Add" # Add button text

`addable` class-attribute

```
addable: bool = None
```

Whether addable.

`draggable` class-attribute

```
draggable: bool = False
```

whether draggable, note that when draggable is enabled, there will be an extra \$id field

`draggableTip` class-attribute

```
draggableTip: str = None
```

draggable prompt text, default is: "can be adjusted by dragging the [swap] button in each row"

`items` class-attribute

```
items: FormItem = None
```

Configure single-item form type

`maxLength` class-attribute

```
maxLength: int = None
```

Limit the maximum length

`minLength` class-attribute

```
minLength: int = None
```

Limit the minimum length

`removable` class-attribute

```
removable: bool = None
```

Whether removable

type class-attribute

```
type: str = 'input-array'
```

## InputCity

Bases: FormItem

city selector

### Attributes

allowCity class-attribute

```
allowCity: bool = None
```

True # Allow city selection

allowDistrict class-attribute

```
allowDistrict: bool = None
```

True # Allow district selection

extractValue class-attribute

```
extractValue: bool = None
```

True # whether to extract the value, if set to false the value format will become an object containing code, province, city and district text information.

searchable class-attribute

```
searchable: bool = None
```

False # Whether or not the search box is available

type class-attribute

```
type: str = 'location-city'
```

## InputColor

Bases: FormItem

color picker

**Attributes**allowCustomColor class-attribute

```
allowCustomColor: bool = None
```

True # When false, only colors can be selected, use presetColors to set the color selection range

clearable class-attribute

```
clearable: bool = None
```

"label" # whether to show clear button

format class-attribute

```
format: str = None
```

"hex" # Please choose hex, hls, rgb or rgba.

presetColors class-attribute

```
presetColors: List[str] = None
```

"selector preset color values" # default color at the bottom of the selector, if the array is empty, no default color is shown

resetValue class-attribute

```
resetValue: str = None
```

"" # After clearing, the form item value is adjusted to this value

type class-attribute

```
type: str = 'input-color'
```

## InputDate

Bases: FormItem

date

**Attributes**clearable class-attribute

```
clearable: bool = None
```

True # clearable or not

closeOnSelect class-attribute

```
closeOnSelect: bool = None
```

False # Whether to close the selection box immediately after tapping a date

embed class-attribute

```
embed: bool = None
```

False # Whether to inline mode

format class-attribute

```
format: str = None
```

"X" # Date picker value format, see documentation for more format types

inputFormat class-attribute

```
inputFormat: str = None
```

"YYYY-DD-MM" # Date picker display format, i.e. timestamp format, see documentation for more format types

largeMode class-attribute

```
largeMode: bool = None
```

False # Zoom mode

maxDate class-attribute

```
maxDate: str = None
```

Limit the maximum date

minDate class-attribute

```
minDate: str = None
```

Restrict the minimum date

placeholder class-attribute

```
placeholder: str = None
```

"Please select a date" # Placeholder text

**scheduleAction** class-attribute

```
scheduleAction: MSAUISchemaNode = None
```

Custom schedule display

**scheduleClassNames** class-attribute

```
scheduleClassNames: List[str] = None
```

"['bg-warning','bg-danger','bg-success','bg-info','bg-secondary']"

**schedules** class-attribute

```
schedules: Union[list, str] = None
```

Show schedules in calendar, can set static data or take data from context, className reference background color

**shortcuts** class-attribute

```
shortcuts: str = None
```

Date shortcuts

**timeConstraints** class-attribute

```
timeConstraints: dict = None
```

True # See also: react-datetime

**type** class-attribute

```
type: str = 'input-date'
```

**utc** class-attribute

```
utc: bool = None
```

False # Save utc value

**value** class-attribute

```
value: str = None
```

Default value

## InputDateRange

Bases: `InputDatetimeRange`

dateRange

### Attributes

maxDuration class-attribute

```
maxDuration: str = None
```

Limit the maximum span, e.g. 1year

minDuration class-attribute

```
minDuration: str = None
```

Limit the minimum span, e.g. 2days

type class-attribute

```
type: str = 'input-date-range'
```

## InputDatetime

Bases: `FormItem`

date

### Attributes

clearable class-attribute

```
clearable: bool = None
```

True # clearable or not

embed class-attribute

```
embed: bool = None
```

False # Whether to inline

format class-attribute

```
format: str = None
```

"X" # Date time selector value format, see documentation for more format types

inputFormat class-attribute

```
inputFormat: str = None
```

"YYYY-MM-DD HH<sup>ss</sup>" # Date and time picker display format, i.e. timestamp format, see documentation for more format types

maxDate class-attribute

```
maxDate: str = None
```

Limit the maximum date and time

minDate class-attribute

```
minDate: str = None
```

Limit the minimum date and time

placeholder class-attribute

```
placeholder: str = None
```

"Please select the date and time" # Placeholder text

shortcuts class-attribute

```
shortcuts: str = None
```

Date and time shortcuts

timeConstraints class-attribute

```
timeConstraints: dict = None
```

True # See also: react-datetime

type class-attribute

```
type: str = 'input-datetime'
```

utc class-attribute

```
utc: bool = None
```

False # Save utc value

value class-attribute

```
value: str = None
```



Default value

## InputDatetimeRange

Bases: `InputTimeRange`

`DateTimeRange`

### Attributes

`maxDate` class-attribute

```
maxDate: str = None
```

Limit the maximum date and time, use the same as limit range

`minDate` class-attribute

```
minDate: str = None
```

Limit the minimum date and time, use the same as limit range

`ranges` class-attribute

```
ranges: Union[str, List[str]] = None
```

"yesterday,7daysago,prevweek,thismonth,prevmonth,prevquarter" Date range shortcut. optional:  
today,yesterday,1dayago,7daysago,30daysago,90daysago,prevweek,thismonth,thisquarter,prevmonth,prevquarter

`type` class-attribute

```
type: str = 'input-datetime-range'
```

`utc` class-attribute

```
utc: bool = None
```

False # Save UTC value

## InputExcel

Bases: `FormItem`

`Parse Excel`

### Attributes

`allSheets` class-attribute

```
allSheets: bool = None
```

False # whether to parse all sheets

includeEmpty class-attribute

```
includeEmpty: bool = None
```

True # whether to include null values

parseMode class-attribute

```
parseMode: str = None
```

'array' or 'object' parse mode

plainText class-attribute

```
plainText: bool = None
```

True # Whether to parse as plain text

type class-attribute

```
type: str = 'input-excel'
```

## InputFile

Bases: FormItem

FileUpload

### Attributes

accept class-attribute

```
accept: str = None
```

"text/plain" # Only plain text is supported by default, to support other types, please configure this attribute to have the file suffix .xxx

asBase64 class-attribute

```
asBase64: bool = None
```

False # Assign the file as base64 to the current component

asBlob class-attribute

```
asBlob: bool = None
```

False # Assign the file to the current component in binary form

**autoFill** class-attribute

```
autoFill: Dict[str, str] = None
```

After a successful upload, you can configure autoFill to populate a form item with the values returned by the upload interface (not supported under non-form for now)

**autoUpload** class-attribute

```
autoUpload: bool = None
```

True # Automatically start uploading after no selection

**btnLabel** class-attribute

```
btnLabel: str = None
```

The text of the upload button

**chunkApi** class-attribute

```
chunkApi: MSA_UI_API = None
```

chunkApi

**chunkSize** class-attribute

```
chunkSize: int = None
```

5 \* 1024 \* 1024 # chunk size

**delimiter** class-attribute

```
delimiter: str = None
```

"," # Splice character

**downloadUrl** class-attribute

```
downloadUrl: Union[str, bool] = None
```

Version 1.1.6 starts to support post:http://xxx.com/\${value} this way, The default display of the file path will support direct download, you can support adding a prefix such as: http://xx.dom/filename= , if you do not want this, you can set the current configuration item to false.

**extractValue** class-attribute

```
extractValue: bool = None
```

False # Extract the value

fileField class-attribute

```
fileField: str = None
```

"file" # You can ignore this attribute if you don't want to store it yourself.

finishChunkApi class-attribute

```
finishChunkApi: MSA_UI_API = None
```

finishChunkApi

hideUploadButton class-attribute

```
hideUploadButton: bool = None
```

False # Hide the upload button

joinValues class-attribute

```
joinValues: bool = None
```

True # Splice values

maxLength class-attribute

```
maxLength: int = None
```

No limit by default, when set, only the specified number of files will be allowed to be uploaded at a time.

maxSize class-attribute

```
maxSize: int = None
```

No limit by default, when set, files larger than this value will not be allowed to be uploaded. The unit is B

multiple class-attribute

```
multiple: bool = None
```

False # Whether to select multiple.

nameField class-attribute

```
nameField: str = None
```

"name" # Which field the interface returns to identify the file name

**receiver** class-attribute

```
receiver: MSA_UI_API = None
```

Upload file interface

**startChunkApi** class-attribute

```
startChunkApi: MSA_UI_API = None
```

startChunkApi

**stateTextMap** class-attribute

```
stateTextMap: dict = None
```

Upload state text, Default: {init: ', pending: 'Waiting for upload', uploading: 'Uploading', error: 'Upload error', uploaded: 'Uploaded', ready: '}'

**type** class-attribute

```
type: str = 'input-file'
```

**urlField** class-attribute

```
urlField: str = None
```

"url" # The field name of the file download address.

**useChunk** class-attribute

```
useChunk: bool = None
```

The server where msa\_ui is hosted restricts the file upload size to 10M, so msa\_ui will automatically change to chunk upload mode when the user selects a large file.

**valueField** class-attribute

```
valueField: str = None
```

"value" # Which field is used to identify the value of the file

## InputGroup

Bases: `FormItem`

InputGroupGroup

**Attributes**

**body** class-attribute

```
body: List[FormItem] = None
```

collection of form items

**className** class-attribute

```
className: str = None
```

CSS class name

**type** class-attribute

```
type: str = 'input-group'
```

## InputImage

**Bases:** `FormItem`

Image Upload

### Attributes

**accept** class-attribute

```
accept: str = None
```

".jpeg,.jpg,.png,.gif" # Supported image type formats, please configure this attribute as image suffix, e.g. .jpg, .png

**autoFill** class-attribute

```
autoFill: Dict[str, str] = None
```

After successful upload, you can configure autoFill to fill the value returned by the upload interface into a form item (not supported under non-form)

**autoUpload** class-attribute

```
autoUpload: bool = None
```

True # Automatically start uploading after no selection

**crop** class-attribute

```
crop: Union[bool, CropInfo] = None
```

Used to set if crop is supported.

**cropFormat** class-attribute

```
cropFormat: str = None
```

"image/png" # Crop file format

**cropQuality** class-attribute

```
cropQuality: int = None
```

1 # quality of the crop file format, for jpeg/webp, takes values between 0 and 1

**delimiter** class-attribute

```
delimiter: str = None
```

"," # Splice character

**extractValue** class-attribute

```
extractValue: bool = None
```

False # Extract the value

**fileField** class-attribute

```
fileField: str = None
```

"file" # You can ignore this property if you don't want to store it yourself.

**fixedSize** class-attribute

```
fixedSize: bool = None
```

Whether to enable fixed size, if so, set fixedSizeClassName at the same time

**fixedSizeClassName** class-attribute

```
fixedSizeClassName: str = None
```

When fixedSize is enabled, the display size is controlled by this value.

**frameImage** class-attribute

```
frameImage: str = None
```

Default placeholder image address

**hideUploadButton** class-attribute

```
hideUploadButton: bool = None
```

False # Hide the upload button

joinValues class-attribute

```
joinValues: bool = None
```

True # Splice values

limit class-attribute

```
limit: Limit = None
```

Limit the size of the image, won't allow uploads beyond that.

maxLength class-attribute

```
maxLength: int = None
```

No limit by default, when set, only the specified number of files will be allowed to be uploaded at once.

maxSize class-attribute

```
maxSize: int = None
```

No limit by default, when set, file size larger than this value will not be allowed to upload. The unit is B

multiple class-attribute

```
multiple: bool = None
```

False # Whether to select multiple.

receiver class-attribute

```
receiver: MSA_UI_API = None
```

Upload file interface

type class-attribute

```
type: str = 'input-image'
```

## Classes

## CropInfo

Bases: MSABaseUIModel



## Attributes

aspectRatio class-attribute

```
aspectRatio: float = None
```

Crop ratio. Floating point type, default 1 i.e. 1:1, if you want to set 16:9 please set 1.7777777777777777 i.e. 16 / 9.

rotatable class-attribute

```
rotatable: bool = None
```

False # If or not rotatable when cropping.

scalable class-attribute

```
scalable: bool = None
```

False # Whether to scale when cropping

viewMode class-attribute

```
viewMode: int = None
```

1 # View mode when cropping, 0 is no limit

## Limit

Bases: MSABaseUIModel

## Attributes

aspectRatio class-attribute

```
aspectRatio: float = None
```

Limit the aspect ratio of the image, the format is floating point number, default 1 is 1:1. If you want to set 16:9, please set 1.7777777777777777 i.e. 16 / 9. If you don't want to limit the ratio, please set the empty string.

height class-attribute

```
height: int = None
```

Limit the height of the image.

maxHeight class-attribute

```
maxHeight: int = None
```

Limit the maximum height of the image.

**maxWidth** class-attribute`maxWidth: int = None`

Limit the maximum width of the image.

**minHeight** class-attribute`minHeight: int = None`

Limit the minimum height of the image.

**minWidth** class-attribute`minWidth: int = None`

Limit the minimum width of the image.

**width** class-attribute`width: int = None`

Limit the width of the image.

## InputMonth

Bases: `FormItem`

month

### Attributes

**clearable** class-attribute`clearable: bool = None`

True # clearable or not

**format** class-attribute`format: str = None`

"X" # month selector value format, see moment for more format types

**inputFormat** class-attribute`inputFormat: str = None`

"YYYY-MM" # Month selector display format, i.e. timestamp format, see moment for more format types

**placeholder** class-attribute

`placeholder: str = None`

"Please select a month" # Placeholder text

type class-attribute

`type: str = 'input-month'`

value class-attribute

`value: str = None`

Default value

InputMonthRange

Bases: InputDateRange

MonthRange

Attributes

type class-attribute

`type: str = 'input-month-range'`

InputNumber

Bases: FormItem

Number input box

Attributes

kilobitSeparator class-attribute

`kilobitSeparator: bool = None`

thousand separator

max class-attribute

`max: Union[int, MSAUITemplate] = None`

max

min class-attribute

`min: Union[int, MSAUITemplate] = None`

min

precision class-attribute

```
precision: int = None
```

precision, i.e., the number of decimal places

prefix class-attribute

```
prefix: str = None
```

prefix

showSteps class-attribute

```
showSteps: bool = None
```

True # Whether to show the up and down click buttons

step class-attribute

```
step: int = None
```

step size

suffix class-attribute

```
suffix: str = None
```

suffix

type class-attribute

```
type: str = 'input-number'
```

## InputPassword

Bases: InputText

Password input box

### Attributes

type class-attribute

```
type: str = 'input-password'
```

## InputRichText

Bases: `FormItem`

rich-text editor

### Attributes

buttons `class-attribute`

```
buttons: list[str] = None
```

froala specific, configure the buttons to be displayed, tinymce can set the toolbar string with the preceding options

options `class-attribute`

```
options: dict = None
```

Need to refer to tinymce or froala's documentation

receiver `class-attribute`

```
receiver: MSA_UI_API = None
```

' # default image save MSA\_UI\_API

saveAsUbb `class-attribute`

```
saveAsUbb: bool = None
```

whether to save as ubb format

size `class-attribute`

```
size: str = None
```

Size of the box, can be set to md or lg

type `class-attribute`

```
type: str = 'input-rich-text'
```

vendor `class-attribute`

```
vendor: str = None
```

"vendor": "froala", configured to use the froala editor

videoReceiver `class-attribute`

```
videoReceiver: MSA_UI_API = None
```

' # Default video saving MSA\_UI\_API

# InputTable

Bases: `FormItem`

table

## Attributes

addApi `class-attribute`

addApi: `MSA_UI_API` = `None`

The `MSA_UI_API` to submit when adding

addBtnIcon `class-attribute`

addBtnIcon: `str` = `None`

"plus" # Add button icon

addBtnLabel `class-attribute`

addBtnLabel: `str` = `None`

Add button name

addable `class-attribute`

addable: `bool` = `None`

False # Whether to add a row

canAccessSuperData `class-attribute`

canAccessSuperData: `bool` = `None`

False # Whether to access parent data, that is, the same level of data in the form, usually need to be used with `strictMode`

cancelBtnIcon `class-attribute`

cancelBtnIcon: `str` = `None`

"times" # Cancel the edit button icon

cancelBtnLabel `class-attribute`

```
cancelBtnLabel: str = None
```

"" # Cancel the edit button name

columns class-attribute

```
columns: list = None
```

[] # Column information, columns[x].quickEdit: boolean|object = None # Used in conjunction with editable being true,

**columns[x].quickEditOnUpdate: boolean|object = None # can be used to distinguish between new mode and update mode editing configuration**

confirmBtnIcon class-attribute

```
confirmBtnIcon: str = None
```

"check" # Confirm edit button icon

confirmBtnLabel class-attribute

```
confirmBtnLabel: str = None
```

"" # Confirm edit button name

copyBtnIcon class-attribute

```
copyBtnIcon: str = None
```

"copy" # Copy the button icon

copyBtnLabel class-attribute

```
copyBtnLabel: str = None
```

Copy button text

deleteApi class-attribute

```
deleteApi: MSA_UI_API = None
```

MSA\_UI\_API submitted when deleting

deleteBtnIcon class-attribute

```
deleteBtnIcon: str = None
```

"minus" # Delete the button icon

deleteBtnLabel class-attributedeleteBtnLabel: `str` = None

""" # Delete the button name

editBtnIcon class-attributeeditBtnIcon: `str` = None

"pencil" # editBtnIcon

editBtnLabel class-attributeeditBtnLabel: `str` = None

""" # Edit button name

editable class-attributeeditable: `bool` = None

False # Whether to edit

needConfirm class-attributeneedConfirm: `bool` = None

True # whether to confirm the operation, can be used to control the interaction of the form

perPage class-attributeperPage: `int` = None

Set how many data to display on a page. 10

removable class-attributeremovable: `bool` = None

False # Whether to remove

showAddBtn class-attributeshowAddBtn: `bool` = None

True # Whether to show the add button

showIndex class-attribute



```
showIndex: bool = None
```

False # Show serial number

strictMode class-attribute

```
strictMode: bool = None
```

True # For performance, the default value changes of other form items will not update the current form, sometimes you need to enable this in order to synchronize access to other form fields.

type class-attribute

```
type: str = 'input-table'
```

Specify as Table renderer

updateApi class-attribute

```
updateApi: MSA_UI_API = None
```

The MSA\_UI\_API submitted when modifying

## InputText

Bases: FormItem

input-box

### Attributes

addOn class-attribute

```
addOn: MSAUISchemaNode = None
```

Input box add-on, such as with a prompt text, or with a submit button.

autoComplete class-attribute

```
autoComplete: Union[str, MSA_UI_API] = None
```

autoComplete

clearable class-attribute

```
clearable: bool = None
```

Whether to clear or not

creatable class-attribute

```
creatable: bool = None
```

If or not creatable, default is yes, unless set to false which means that only the value in the option can be selected.

delimiter class-attribute

```
delimiter: str = None
```

Splice character ","

extractValue class-attribute

```
extractValue: bool = None
```

extract value

joinValues class-attribute

```
joinValues: bool = None
```

True # Splice values

labelField class-attribute

```
labelField: str = None
```

option label field "label"

maxLength class-attribute

```
maxLength: int = None
```

Limit the maximum number of words

minLength class-attribute

```
minLength: int = None
```

Limit the minimum number of words

multiple class-attribute

```
multiple: bool = None
```

Whether to multi-select

options class-attribute

```
options: Union[List[str], List[dict]] = None
```

option group

prefix class-attribute

```
prefix: str = None
```

prefix

resetValue class-attribute

```
resetValue: str = None
```

Set the value given by this configuration item after clearing.

showCounter class-attribute

```
showCounter: bool = None
```

Whether to show the counter

source class-attribute

```
source: Union[str, MSA_UI_API] = None
```

dynamic options group

suffix class-attribute

```
suffix: str = None
```

suffix

trimContents class-attribute

```
trimContents: bool = None
```

Whether to remove the first and last blank text.

type class-attribute

```
type: str = 'input-text'
```

input-text|input-url|input-email|input-password|divider

valueField class-attribute

```
valueField: str = None
```

option value field "value"

## InputTime

Bases: `FormItem`

time

### Attributes

clearable `class-attribute`

```
clearable: bool = None
```

True # clearable or not

format `class-attribute`

```
format: str = None
```

"X" # Time picker value format, see moment for more format types

inputFormat `class-attribute`

```
inputFormat: str = None
```

"HH:mm" # Time picker display format, i.e. timestamp format, see moment for more format types

placeholder `class-attribute`

```
placeholder: str = None
```

"Please select time" # Placeholder text

timeConstraints `class-attribute`

```
timeConstraints: dict = None
```

True # See also: react-datetime

timeFormat `class-attribute`

```
timeFormat: str = None
```

"HH:mm" # time selector value format, see moment for more format types

type `class-attribute`

```
type: str = 'input-time'
```

value `class-attribute`

```
value: str = None
```

Default value

# InputTimeRange

Bases: `FormItem`

TimeRange

## Attributes

clearable `class-attribute`

```
clearable: bool = None
```

True # clearable or not

embed `class-attribute`

```
embed: bool = None
```

False # Whether inline mode

format `class-attribute`

```
format: str = None
```

"HH:mm" # time range selector value format

inputFormat `class-attribute`

```
inputFormat: str = None
```

"HH:mm" # Time range selector display format

placeholder `class-attribute`

```
placeholder: str = None
```

"Please select a time range" # Placeholder text

timeFormat `class-attribute`

```
timeFormat: str = None
```

"HH:mm" # time range selector value format

type `class-attribute`

```
type: str = 'input-time-range'
```

## InputTree

Bases: `FormItem`

tree selector box

### Attributes

addApi `class-attribute`

```
addApi: MSA_UI_API = None
```

Configure the add options interface

addControls `class-attribute`

```
addControls: List[FormItem] = None
```

Customize the new form items

autoComplete `class-attribute`

```
autoComplete: MSA_UI_API = None
```

auto prompt complement

cascade `class-attribute`

```
cascade: bool = None
```

False # Do not automatically select children when parent is selected.

creatable `class-attribute`

```
creatable: bool = None
```

False # Add options

deferApi `class-attribute`

```
deferApi: MSA_UI_API = None
```

For lazy loading, please configure defer to true, then configure deferApi to complete lazy loading

deleteApi `class-attribute`

```
deleteApi: MSA_UI_API = None
```

Configure the delete option interface

delimiter class-attribute

```
delimiter: str = None
```

"False" # Splice character

editApi class-attribute

```
editApi: MSA_UI_API = None
```

Configure the edit options interface

editControls class-attribute

```
editControls: List[FormItem] = None
```

Customize edit form items

editable class-attribute

```
editable: bool = None
```

False # Edit options

enableNodePath class-attribute

```
enableNodePath: bool = None
```

False # Whether to enable node path mode

extractValue class-attribute

```
extractValue: bool = None
```

False # extract value

hideRoot class-attribute

```
hideRoot: bool = None
```

True # If you want to show a top node, set to false

iconField class-attribute

```
iconField: str = None
```

"icon" # Icon value field

**initiallyOpen** class-attribute

```
initiallyOpen: bool = None
```

True # Set whether to expand all levels by default.

**joinValues** class-attribute

```
joinValues: bool = None
```

True # join values

**labelField** class-attribute

```
labelField: str = None
```

"label" # option label field

**maxLength** class-attribute

```
maxLength: int = None
```

Maximum number of nodes to select

**minLength** class-attribute

```
minLength: int = None
```

Minimum number of selected nodes

**multiple** class-attribute

```
multiple: bool = None
```

False # Whether to multiple select

**onlyChildren** class-attribute

```
onlyChildren: bool = None
```

False # Whether to add only its children to the value when the parent node is selected in multiple selection.

**options** class-attribute

```
options: MSAOptionsNode = None
```

options group

**pathSeparator** class-attribute



```
pathSeparator: str = None
```

"/" # Separator for node paths, takes effect when enableNodePath is true

removable class-attribute

```
removable: bool = None
```

False # Remove options

rootCreatable class-attribute

```
rootCreatable: bool = None
```

False # Whether top-level nodes can be created

rootCreateTip class-attribute

```
rootCreateTip: str = None
```

"Add first-level node" # Hover tip for creating top-level nodes

rootLabel class-attribute

```
rootLabel: bool = None
```

"top" # Useful when hideRoot is not false, to set the text of the top node.

searchable class-attribute

```
searchable: bool = None
```

False # Searchable or not, only works if type is tree-select

selectFirst class-attribute

```
selectFirst: bool = None
```

showIcon class-attribute

```
showIcon: bool = None
```

True # Whether to show the icon

showRadio class-attribute

```
showRadio: bool = None
```

False # Whether to show radio buttons, multiple is valid when false.

**source** class-attribute

```
source: MSA_UI_API = None
```

dynamic options group

**treeContainerClassName** class-attribute

```
treeContainerClassName: str = None
```

tree outermost container class name

**type** class-attribute

```
type: str = 'input-tree'
```

**unfoldedLevel** class-attribute

```
unfoldedLevel: int = None
```

0 # Set the number of levels to be expanded by default, only effective if initiallyOpen is not true.

**valueField** class-attribute

```
valueField: str = None
```

"value" # option value field

**withChildren** class-attribute

```
withChildren: bool = None
```

False # When the parent node is selected, the value will contain the value of the child node, otherwise only the value of the parent node will be kept.

## Json

Bases: MSAUINode

JSON display component

### Attributes

**className** class-attribute

```
className: str = None
```

Outer CSS class name

**displayDataTypes** class-attribute

```
displayDataTypes: bool = None
```

False # Whether to display data types

**jsonTheme** class-attribute

```
jsonTheme: str = None
```

"twilight" # theme, optional twilight and eighties

**levelExpand** class-attribute

```
levelExpand: int = None
```

1 # Default level of expansion

**mutable** class-attribute

```
mutable: bool = None
```

False # whether to modify

**placeholder** class-attribute

```
placeholder: str = None
```

Placeholder text

**source** class-attribute

```
source: str = None
```

Get the value in the data chain by data mapping

**type** class-attribute

```
type: str = 'json'
```

"json" if in Table, Card and List; "static-json" if used as a static display in Form

**value** class-attribute

```
value: Union[dict, str] = None
```

json value, parse automatically if it is a string

**Link**

Bases: `MSAUINode`

link

### Attributes

blank `class-attribute`

```
blank: bool = None
```

whether to open in a new tab

body `class-attribute`

```
body: str = None
```

text inside the tag

disabled `class-attribute`

```
disabled: bool = None
```

Disable hyperlinks

href `class-attribute`

```
href: str = None
```

Link address

htmlTarget `class-attribute`

```
htmlTarget: str = None
```

target of a tag, takes precedence over the blank attribute

icon `class-attribute`

```
icon: str = None
```

hyperlink icon to enhance display

rightIcon `class-attribute`

```
rightIcon: str = None
```

right icon

title `class-attribute`

```
title: str = None
```

the title of a tag

type class-attribute

```
type: str = 'link'
```

"link" if in Table, Card and List; "static-link" if used as a static display in Form

## LocationPicker

Bases: FormItem

Location

### Attributes

ak class-attribute

```
ak: str = Ellipsis
```

Baidu map ak # Register at: <http://lbsyun.baidu.com/>

clearable class-attribute

```
clearable: bool = None
```

False # Whether the input box is clearable

coordinatesType class-attribute

```
coordinatesType: str = None
```

"bd09" # Default is Baidu coordinates, can be set to 'gcj02'

placeholder class-attribute

```
placeholder: str = None
```

"Please select a location" # Default prompt

type class-attribute

```
type: str = 'location-picker'
```

vendor class-attribute

```
vendor: str = 'baidu'
```

map vendor, currently only implemented Baidu maps

## Log

Bases: `MSAUINode`

LiveLog

### Attributes

autoScroll `class-attribute`

```
autoScroll: bool = None
```

True # whether to auto-scroll

className `class-attribute`

```
className: str = None
```

Outer CSS class name

encoding `class-attribute`

```
encoding: str = None
```

"utf-8" # Returns the character encoding of the content

height `class-attribute`

```
height: int = None
```

500 # height of display area

placeholder `class-attribute`

```
placeholder: str = None
```

Text in load

source `class-attribute`

```
source: MSA_UI_API = None
```

support variable, can be initially set to null, so that it will not be loaded initially, but will be loaded when the variable has a value

type `class-attribute`

```
type: str = 'log'
```

# Mapping

Bases: MSAUINode

mapping

## Attributes

className class-attribute

```
className: str = None
```

Outer CSS class name

map class-attribute

```
map: dict = None
```

Mapping configuration

placeholder class-attribute

```
placeholder: str = None
```

Placeholder text

source class-attribute

```
source: MSA_UI_API = None
```

MSA\_UI\_API or data mapping

type class-attribute

```
type: str = 'mapping'
```

"mapping" if in Table, Card and List; "static-mapping" if used as a static display in Form

# Markdown

Bases: MSAUINode

Markdown rendering

## Attributes

className class-attribute

```
className: str = None
```

The outermost class name of the form

**name** class-attribute`name: str = None`

Field name, specifying the key of the form item when it is submitted

**options** class-attribute`options: dict = None`

html, whether html tags are supported, default false; linkify, whether to automatically recognize links, default is true; breaks, whether carriage return is line feed, default false

**src** class-attribute`src: MSA_UI_API = None`

External address

**type** class-attribute`type: str = 'markdown'`**value** class-attribute`value: Union[int, str] = None`

The value of the field

## Nav

**Bases:** MSAUINode**Navigate**

### Attributes

**className** class-attribute`className: str = None`

Class name of the outer Dom

**deferApi** class-attribute`deferApi: MSA_UI_API = None`

Interface used to delay loading of option details, can be left unconfigured, no common source interface configured.



### dragOnSameLevel class-attribute

```
dragOnSameLevel: bool = None
```

Only allows dragging within the same level

### draggable class-attribute

```
draggable: bool = None
```

Whether to support drag and drop sorting

### itemActions class-attribute

```
itemActions: MSAUISchemaNode = None
```

More action related configuration

### itemBadge class-attribute

```
itemBadge: Badge = None
```

corner markers

### links class-attribute

```
links: list = None
```

Collection of links

### saveOrderApi class-attribute

```
saveOrderApi: MSA_UI_API = None
```

api to save sorting

### source class-attribute

```
source: MSA_UI_API = None
```

Navigation can be created dynamically via variables or the MSA\_UI\_API interface

### stacked class-attribute

```
stacked: bool = True
```

Set to false to display as tabs

### type class-attribute

```
type: str = 'nav'
```

Specify as Nav renderer

## Classes

### Link

Bases: `MSAUINode`

#### Attributes

**active** class-attribute

```
active: bool = None
```

Whether to highlight or not

**activeOn** class-attribute

```
activeOn: MSAUIExpression = None
```

condition to highlight or not, leaving blank will automatically analyze the link address

**children** class-attribute

```
children: List[Link] = None
```

child links

**defer** class-attribute

```
defer: bool = None
```

Flag if it is a lazy add-on

**deferApi** class-attribute

```
deferApi: MSA_UI_API = None
```

Can be unconfigured, if configured priority is higher

**icon** class-attribute

```
icon: str = None
```

Icon

**label** class-attribute

```
label: str = None
```

Name

**target** class-attribute`target: str = None`

"Link relationship" #

**to** class-attribute`to: MSAUITemplate = None`

Link address

**unfolded** class-attribute`unfolded: bool = None`

Initially expanded or not

## NestedSelect

Bases: `Select`

Cascading selector

### Attributes

**cascade** class-attribute`cascade: bool = None`

False # When set true, child nodes are not automatically selected when the parent node is selected.

**hideNodePathLabel** class-attribute`hideNodePathLabel: bool = None`

False # Whether to hide the path of the selected node in the selection box label information

**noResultsText** class-attribute`noResultsText: str = None`

"No results found" # Text when no results are found

**onlyChildren** class-attribute`onlyChildren: bool = None`

False # When multi-select, when the parent node is selected, the value will include only its children in the value.

**onlyLeaf** class-attribute

```
onlyLeaf: bool = None
```

False # Only allow leaf nodes to be selected

searchPromptText class-attribute

```
searchPromptText: str = None
```

"Enter content to search" # Search box placeholder text

type class-attribute

```
type: str = 'nested-select'
```

withChildren class-attribute

```
withChildren: bool = None
```

False # When set true, the value of the parent node will contain the value of the child node when selected, otherwise only the value of the parent node will be kept.

## Page

Bases: MSAUINode

page

### Attributes

\_\_default\_template\_path\_\_ class-attribute

```
__default_template_path__: str = (
    f"{BASE_DIR}/templates/page.html"
)
```

aside class-attribute

```
aside: MSAUISchemaNode = None
```

Add content to the sidebar area of the page.

asideClassName class-attribute

```
asideClassName: str = None
```

"w page-aside-region bg-auto" # Aside dom class name

asideMaxWidth class-attribute

```
asideMaxWidth: int = None
```

The maximum width of the page's sidebar area

asideMinWidth class-attribute

```
asideMinWidth: int = None
```

The minimum width of the page's sidebar area

asideResizor class-attribute

```
asideResizor: bool = None
```

Whether the width of the page's sidebar area can be adjusted

body class-attribute

```
body: MSAUISchemaNode = None
```

Add content to the content area of the page

bodyClassName class-attribute

```
bodyClassName: str = None
```

"wrapper" # Body dom class name

className class-attribute

```
className: str = None
```

Outer dom class name

cssVars class-attribute

```
cssVars: dict = None
```

Custom CSS variables, please refer to styles

headerClassName class-attribute

```
headerClassName: str = None
```

"bg-light b-b wrapper" # Header region dom class name

initApi class-attribute

```
initApi: MSA_UI_API = None
```

Page The api used to fetch the initial data. the returned data can be used at the entire page level.

initFetch class-attribute

```
initFetch: bool = None
```

True # Whether to start fetching initApi

initFetchOn class-attribute

```
initFetchOn: MSAUIExpression = None
```

Whether to start fetching initApi, configured by expression

interval class-attribute

```
interval: int = None
```

Refresh time (min 1000)

regions class-attribute

```
regions: List[str] = None
```

remark class-attribute

```
remark: Remark = None
```

A reminder icon will appear near the title, which will prompt the content when the mouse is placed on it.

silentPolling class-attribute

```
silentPolling: bool = None
```

False # Configure whether to show loading animation when refreshing

stopAutoRefreshWhen class-attribute

```
stopAutoRefreshWhen: MSAUIExpression = None
```

Expression to configure the stop refresh condition

subTitle class-attribute

```
subTitle: MSAUISchemaNode = None
```

page sub-title

title class-attribute

```
title: MSAUISchemaNode = None
```

Page title

toolbar class-attribute

```
toolbar: MSAUISchemaNode = None
```

Add content to the top right corner of the page, note that when there is a title, the area is in the top right corner and when there is not, the area is at the top

toolbarClassName class-attribute

```
toolbarClassName: str = None
```

"v-middle wrapper text-right bg-light b-b" # Toolbar dom class name

type class-attribute

```
type: str = 'page'
```

Specify as Page component

Functions

msa\_ui\_html

```
msa_ui_html(  
    template_path: str = "",  
    locale: str = "zh_CN",  
    site_title: str = "Admin",  
    site_icon: str = "",  
)
```

PageSchema

Bases: MSAUINode

Attributes

children class-attribute

```
children: List[PageSchema] = None
```

Submenus

className class-attribute

```
className: str = None
```

The class name of the menu.

icon class-attribute

```
icon: str = 'fa fa-flash'
```

Menu icon, e.g., 'fa fa-file'.

isDefaultPage class-attribute

```
isDefaultPage: Union[str, bool] = None
```

Useful when you need a custom 404 page, don't have more than one of these, because only the first one will work.

label class-attribute

```
label: str = None
```

The name of the menu.

link class-attribute

```
link: str = None
```

If you want to configure an external link menu, just configure link.

redirect class-attribute

```
redirect: str = None
```

Jump, when the current page is hit, jump to the target page.

rewrite class-attribute

```
rewrite: str = None
```

Change to render a page with another path, this way the page address will not be changed.

schemaApi class-attribute

```
schemaApi: MSA_UI_API = None
```

If you want to pull through the interface, please configure it. The return path is json>data. schema and schemaApi can only be one or the other.

schema\_ class-attribute

```
schema_: Union[Page, Iframe] = Field(None, alias='schema')
```

Configuration of the page, please go to the Page page for details



**sort** class-attribute`sort: int = None`

Sort

**url** class-attribute`url: str = None`

The page routing path to enable the current page when the route hits that path. When the path is not /-headed, the parent path is connected. For example, if the parent path is folder and pageA is configured, then the page will be hit when the page address is /folder/pageA. When the path starts with /, e.g. /crud/list, the parent path is not concatenated. There is also support for routes with parameters such as /crud/view/:id, which can be fetched from the page via \${params.id}.

**visible** class-attribute`visible: str = None`

Some pages may not want to appear in the menu, you can configure it to false, in addition to the route with parameters do not need to be configured, directly is not visible.

## Functions

### as\_tabs\_item

```
as_tabs_item(
    tabs_extra: Dict[str, Any] = None,
    item_extra: Dict[str, Any] = None,
)
```

## Panel

Bases: MSAUINode

### Attributes

**actions** class-attribute`actions: List[Action] = None`

Button area

**actionsClassName** class-attribute`actionsClassName: str = None`

"panel-footer" # Class name of the actions region

## affixFooter class-attribute

```
affixFooter: bool = None
```

Whether to fix the bottom container

## body class-attribute

```
body: MSAUISchemaNode = None
```

content container

## bodyClassName class-attribute

```
bodyClassName: str = None
```

"panel-body" # Class name for the body region

## className class-attribute

```
className: str = None
```

"panel-default" # class name of outer Dom

## footer class-attribute

```
footer: MSAUISchemaNode = None
```

footer container

## footerClassName class-attribute

```
footerClassName: str = None
```

"panel-footer bg-light lter wrapper" # Class name of the footer region

## header class-attribute

```
header: MSAUISchemaNode = None
```

header container

## headerClassName class-attribute

```
headerClassName: str = None
```

"panel-heading" # Class name of the header area

## title class-attribute

<code>title: MSAUISchemaNode = None</code>
title
type <small>class-attribute</small>
<code>type: str = 'panel'</code>
Specify as Panel renderer

Picker

Bases: <small>FormItem</small>
List Picker
Attributes
autoFill <small>class-attribute</small>
<code>autoFill: dict = None</code>
AutoFill
delimiter <small>class-attribute</small>
<code>delimiter: bool = None</code>
False # Splice character
embed <small>class-attribute</small>
<code>embed: bool = None</code>
False # Whether to use inline mode
extractValue <small>class-attribute</small>
<code>extractValue: bool = None</code>
False # extract value
joinValues <small>class-attribute</small>
<code>joinValues: bool = None</code>
True # splice values
labelField <small>class-attribute</small>

```
labelField: str = None
```

"label" # option label field

modalMode class-attribute

```
modalMode: Literal[dialog, drawer] = None
```

"dialog" # Set dialog or drawer to configure popup method.

multiple class-attribute

```
multiple: bool = None
```

Whether to be multiple options.

options class-attribute

```
options: MSAOptionsNode = None
```

Options group

pickerSchema class-attribute

```
pickerSchema: Union[CRUD, MSAUISchemaNode] = None
```

"{mode: 'list', listItem: {title: '\${label}'}}" i.e. rendering with List type to display list information. See CRUD for more configuration

size class-attribute

```
size: Union[str, SizeEnum] = None
```

Supports: xs, sm, md, lg, xl, full

source class-attribute

```
source: MSA_UI_API = None
```

Dynamic options group

type class-attribute

```
type: str = 'picker'
```

List picker, similar in function to Select, but it can display more complex information.

valueField class-attribute

```
valueField: str = None
```

"value" # Option value field

## Portlet

Bases: `Tabs`

### Attributes

contentClassName class-attribute

```
contentClassName: str = None
```

Class name of the Tabs content Dom

description class-attribute

```
description: MSAUITemplate = None
```

Information on the right side of the title

divider class-attribute

```
divider: bool = None
```

False # Remove the divider

hideHeader class-attribute

```
hideHeader: bool = None
```

False # Hide the header

style class-attribute

```
style: Union[str, dict] = None
```

Custom style

tabs class-attribute

```
tabs: List[Item] = None
```

Contents of tabs

type class-attribute

```
type: str = 'portlet'
```

Specify as Portlet renderer

**Classes**

**Item**

Bases: `Tabs.Item`

**Attributes**

`toolbar` class-attribute

```
toolbar: MSAUISchemaNode = None
```

toolbar in tabs, changes with tab toggle

**Property**

Bases: `MSAUINode`

Property table

**Attributes**

`className` class-attribute

```
className: str = None
```

the class name of the outer dom

`column` class-attribute

```
column: int = None
```

3 # how many columns per row

`contentStyle` class-attribute

```
contentStyle: dict = None
```

The style of the property value

`items` class-attribute

```
items: List[Item] = None
```

data items

`labelStyle` class-attribute

```
labelStyle: dict = None
```

Style of the property name

mode class-attribute

```
mode: str = None
```

'table' # display mode, currently only 'table' and 'simple'

separator class-attribute

```
separator: str = None
```

',' # separator between attribute name and value in 'simple' mode

source class-attribute

```
source: MSAUITemplate = None
```

Data source

style class-attribute

```
style: dict = None
```

The style of the outer dom

title class-attribute

```
title: str = None
```

title

type class-attribute

```
type: str = 'property'
```

## Classes

### Item

Bases: MSAUINode

#### Attributes

content class-attribute

```
content: MSAUITemplate = None
```

attribute value

**hiddenOn** class-attribute`hiddenOn: MSAUIExpression = None`

Hide the expression

**label** class-attribute`label: MSAUITemplate = None`

property name

**span** class-attribute`span: int = None`

attribute value across several columns

**visibleOn** class-attribute`visibleOn: MSAUIExpression = None`

Show expressions

## QRCode

Bases: `MSAUINode`

QR Code

### Attributes

**backgroundColor** class-attribute`backgroundColor: str = None`

"ffff" # QR code background color

**className** class-attribute`className: str = None`

Class name of the outer Dom

**codeSize** class-attribute`codeSize: int = None`

128 # The width and height of the QR code

**foregroundColor** class-attribute



```
foregroundColor: str = None
```

"#000" # The foreground color of the QR code

level class-attribute

```
level: str = None
```

"L" # QR code complexity level, there are four ('L' 'M' 'Q' 'H')

qrcodeClassName class-attribute

```
qrcodeClassName: str = None
```

class name of the QR code SVG

type class-attribute

```
type: str = 'qr-code'
```

Specified as a QRCode renderer

value class-attribute

```
value: MSAUITemplate
```

The text to be displayed after scanning the QR code, to display a page enter the full url ("http://..." or "https://..."), supports the use of templates

## Radios

Bases: FormItem

RadioBox

### Attributes

autoFill class-attribute

```
autoFill: dict = None
```

AutoFill

columnsCount class-attribute

```
columnsCount: int = None
```

1 # How many columns to display options by, default is one column

inline class-attribute

```
inline: bool = None
```

True # Whether to display as one line

labelField class-attribute

```
labelField: bool = None
```

"label" # option label field

options class-attribute

```
options: List[Union[dict, str]] = None
```

option group

selectFirst class-attribute

```
selectFirst: bool = None
```

False # Whether to select the first by default

SOURCE class-attribute

```
source: MSA_UI_API = None
```

dynamic options group

type class-attribute

```
type: str = 'radios'
```

valueField class-attribute

```
valueField: bool = None
```

"value" # option value field

## Remark

Bases: MSAUINode

marker

### Attributes

className class-attribute

```
className: str = None
```

Outer CSS class name

content class-attribute

```
content: str = None
```

Prompt text

icon class-attribute

```
icon: str = None
```

"fa fa-question-circle" # Icon

placement class-attribute

```
placement: str = None
```

popup position

trigger class-attribute

```
trigger: str = None
```

trigger condition ['hover','focus']

type class-attribute

```
type: str = 'remark'
```

remark

## Select

Bases: FormItem

dropdown box

### Attributes

addApi class-attribute

```
addApi: MSA_UI_API = None
```

Configure the add options interface

addControls class-attribute

```
addControls: List[FormItem] = None
```

Customize the new form item

autoComplete class-attribute

```
autoComplete: MSA_UI_API = None
```

auto prompt complement

autoFill class-attribute

```
autoFill: dict = None
```

AutoFill

checkAll class-attribute

```
checkAll: bool = None
```

False # Whether to support select all

checkAllBySearch class-attribute

```
checkAllBySearch: bool = None
```

False # Only check all items that are hit when there is a search

checkAllLabel class-attribute

```
checkAllLabel: str = None
```

"Select All" # Text to select all

clearable class-attribute

```
clearable: bool = None
```

Whether clearing is supported in radio mode

columns class-attribute

```
columns: List[dict] = None
```

When the display form is table can be used to configure which columns to display, similar to the columns in the table configuration, but only the display function.

creatable class-attribute

```
creatable: bool = None
```

False # Add option

createBtnLabel class-attribute

```
createBtnLabel: str = None
```

"Add option" # Add option

defaultCheckAll class-attribute

```
defaultCheckAll: bool = None
```

False # defaultCheckAll or not

deleteApi class-attribute

```
deleteApi: MSA_UI_API = None
```

Configure the delete option interface

delimiter class-attribute

```
delimiter: Union[bool, str] = None
```

False # Splice character

editApi class-attribute

```
editApi: MSA_UI_API = None
```

Configure the edit options interface

editControls class-attribute

```
editControls: List[FormItem] = None
```

Customize edit form items

editable class-attribute

```
editable: bool = None
```

False # Edit options

extractValue class-attribute

```
extractValue: bool = None
```

False # extract value

**hideSelected** class-attribute

```
hideSelected: bool = None
```

False # Hide the selected option

**joinValues** class-attribute

```
joinValues: bool = None
```

True # splice values

**labelField** class-attribute

```
labelField: str = None
```

"label" # option label field

**leftMode** class-attribute

```
leftMode: str = None
```

Configure the left option set when the display is associated, supports list or tree. default is list. rightMode: str = None # Configure the left option set when the display is associated.

**leftOptions** class-attribute

```
leftOptions: List[dict] = None
```

Used to configure the left set of options when the display is associated.

**menuTpl** class-attribute

```
menuTpl: str = None
```

Support for configuring custom menus

**mobileClassName** class-attribute

```
mobileClassName: str = None
```

Mobile floating class name

**multiple** class-attribute

```
multiple: bool = None
```

False # Multi-select

**options** class-attribute

```
options: MSAOptionsNode = None
```

options group

**removable** class-attribute

```
removable: bool = None
```

False # Remove options

**rightMode** class-attribute

```
rightMode: str = None
```

Used to configure the right option set when the display is associated, optionally: list, table, tree, chained.

**searchResultMode** class-attribute

```
searchResultMode: str = None
```

If not set, the value of selectMode will be used, can be configured separately, refer to selectMode, determine the display of search results.

**searchable** class-attribute

```
searchable: bool = None
```

False # Searchable

**selectMode** class-attribute

```
selectMode: str = None
```

Optional: group, table, tree, chained, associated, respectively: list form, table form, tree select form, tree select form cascade selection form, association selection form (the difference with cascade selection is that the cascade is infinite, while the association is only one level, the left side of the association can be a tree).

**source** class-attribute

```
source: MSA_UI_API = None
```

dynamic options group

**type** class-attribute

```
type: str = 'select'
```

valueField class-attribute

```
valueField: str = None
```

"value" # option value field

# Service

Bases: MSAUINode

Function container

## Attributes

api class-attribute

```
api: MSA_UI_API = None
```

Initial data domain interface address

body class-attribute

```
body: MSAUISchemaNode = None
```

Content container

className class-attribute

```
className: str = None
```

Class name of the outer Dom

data class-attribute

```
data: dict = None
```

dataProvider class-attribute

```
dataProvider: str = None
```

Data fetch function

initFetch class-attribute

```
initFetch: bool = None
```

Whether to pull by default

initFetchSchema class-attribute



```
initFetchSchema: bool = None
```

Whether to pull Schema by default

interval class-attribute

```
interval: int = None
```

polling interval (minimum 3000)

messages class-attribute

```
messages: dict = None
```

Message hint override, the default message reads the toast hint text returned by the interface, but it can be overridden here.

name class-attribute

```
name: str = None
```

node name

schemaApi class-attribute

```
schemaApi: MSA_UI_API = None
```

Used to fetch the remote Schema interface address

silentPolling class-attribute

```
silentPolling: bool = None
```

False # Configure whether to show loading animation when polling

stopAutoRefreshWhen class-attribute

```
stopAutoRefreshWhen: MSAUIExpression = None
```

Configure the conditions for stopping polling

type class-attribute

```
type: str = 'service'
```

Specified as a service renderer

WS class-attribute

```
ws: str = None
```

WebSocket address

## Spinner

Bases: `MSAUINode`

loading

### Attributes

type `class-attribute`

```
type: str = 'spinner'
```

## Static

Bases: `FormItem`

Static display/label

### Attributes

type `class-attribute`

```
type: str = 'static'
```

support for displaying other non-form items by configuring the type as static-xxx component static-json|static-datetime

### Classes

#### Datetime

Bases: `FormItem`

Show date

### Attributes

type `class-attribute`

```
type: str = 'static-datetime'
```

value `class-attribute`

```
value: Union[int, str]
```

support 10-bit timestamp: 1593327764

## Json

Bases: `FormItem`

### Attributes

type `class-attribute`

```
type: str = 'static-json'
```

value `class-attribute`

```
value: Union[dict, str]
```

## Status

Bases: `MSAUINode`

status

### Attributes

className `class-attribute`

```
className: str = None
```

class name of the outer Dom

labelMap `class-attribute`

```
labelMap: dict = None
```

Mapping text

map `class-attribute`

```
map: dict = None
```

Mapping icon

placeholder `class-attribute`

```
placeholder: str = None
```

Placeholder text

type `class-attribute`

```
type: str = 'status'
```

Specify as Status renderer

## Switch

Bases: `FormItem`

switch

### Attributes

falseValue `class-attribute`

```
falseValue: Any = None
```

"false" # Identifies a false value

offText `class-attribute`

```
offText: str = None
```

Text when off

onText `class-attribute`

```
onText: str = None
```

Text when on

option `class-attribute`

```
option: str = None
```

option description

trueValue `class-attribute`

```
trueValue: Any = None
```

"True" # Identifies a true value

type `class-attribute`

```
type: str = 'switch'
```

## Table

Bases: `MSAUINode`

table

## Attributes

affixHeader class-attribute

```
affixHeader: bool = None
```

True # Whether to fix the table header

affixRow class-attribute

```
affixRow: list = None
```

Bottom summary row

autoFillHeight class-attribute

```
autoFillHeight: bool = None
```

Content area adaptive height

checkOnItemClick class-attribute

```
checkOnItemClick: bool = None
```

False # Whether the current row can be checked by clicking on the data row

className class-attribute

```
className: str = None
```

"panel-default" # Outer CSS class name

columns class-attribute

```
columns: List[Union[TableColumn, MSAUISchemaNode]] = None
```

Used to set column information

columnsToggable class-attribute

```
columnsToggable: Union[str, bool] = None
```

"auto" # Show column display switch, auto i.e.: automatically on when the number of columns is greater than or equal to 5

combineNum class-attribute

```
combineNum: int = None
```

Automatically merge cells

footable class-attribute

```
footable: Union[bool, dict] = None
```

When there are too many columns, there is no way to display all the content, so you can let some of the information displayed at the bottom, which allows users to expand to see the details. The configuration is very simple, just turn on the footable property and add a breakpoint property to \* for the columns you want to display at the bottom.

footerClassName class-attribute

```
footerClassName: str = None
```

"Action.md-table-footer" # bottom outer CSS class name

headerClassName class-attribute

```
headerClassName: str = None
```

"Action.md-table-header" # top outer CSS class name

itemActions class-attribute

```
itemActions: List[Action] = None
```

Hover row action button group

itemBadge class-attribute

```
itemBadge: Badge = None
```

Row corner configuration

itemCheckableOn class-attribute

```
itemCheckableOn: MSAUIExpression = None
```

condition to configure whether the current row is checkable, use expression

itemDraggableOn class-attribute

```
itemDraggableOn: MSAUIExpression = None
```

condition to configure whether the current row is draggable or not, use the expression

placeholder class-attribute

```
placeholder: str = None
```

"No data yet" # Text alert when there is no data

prefixRow class-attribute

```
prefixRow: list = None
```

Top summary row

rowClassName class-attribute

```
rowClassName: str = None
```

Add a CSS class name to the row

rowClassNameExpr class-attribute

```
rowClassNameExpr: MSAUITemplate = None
```

Add a CSS class name to the row via a template

source class-attribute

```
source: str = None
```

"\${items}" # data source, bound to the current environment variable

tableClassName class-attribute

```
tableClassName: str = None
```

"table-sqlite\_db table-striped" # Table CSS class name

title class-attribute

```
title: str = None
```

title

toolbarClassName class-attribute

```
toolbarClassName: str = None
```

"Action.md-table-toolbar" # Toolbar CSS class name

type class-attribute

```
type: str = 'table'
```

Specify as table renderer

## TableCRUD

Bases: `CRUD`, `Table`

TableCRUD

## TableColumn

Bases: `MSAUINode`

columnConfiguration

### Attributes

breakpoint class-attribute

```
breakpoint: str = None
```

`*,ls`

copyable class-attribute

```
copyable: Union[bool, dict] = None
```

whether copyable boolean or {icon: string, content:string}

fixed class-attribute

```
fixed: str = None
```

Whether to fix the front row left|right|none

label class-attribute

```
label: MSAUITemplate = None
```

the text content of the table header

name class-attribute

```
name: str = None
```

Data associated by name

popOver class-attribute

```
popOver: Union[bool, dict] = None
```



popup box

quickEdit class-attribute

```
quickEdit: Union[bool, dict] = None
```

QuickEdit

remark class-attribute

```
remark: Remark = None
```

Prompt message

searchable class-attribute

```
searchable: Union[bool, MSAUISchemaNode] = None
```

False # Whether to search quickly boolean|Schema

sortable class-attribute

```
sortable: bool = None
```

False # Whether to sort

tpl class-attribute

```
tpl: MSAUITemplate = None
```

Template

type class-attribute

```
type: str = None
```

Literal['text','audio','image','link','tpl','mapping','carousel','date','progress','status','switch','list','json','operation']

width class-attribute

```
width: Union[str, int] = None
```

Column width

## Tabs

Bases: `MSAUINode`

### Attributes

**addBtnText** class-attribute

```
addBtnText: str = None
```

"add" # Add button text

**addable** class-attribute

```
addable: bool = None
```

False # If or not addable is supported

**className** class-attribute

```
className: str = None
```

The class name of the outer Dom

**closeable** class-attribute

```
closeable: bool = None
```

False # Whether to support delete

**draggable** class-attribute

```
draggable: bool = None
```

False # Whether drag and drop is supported

**editable** class-attribute

```
editable: bool = None
```

False # Whether to make the tag name editable or not

**mode** class-attribute

```
mode: str = None
```

Display mode, can be line, card, radio, vertical, chrome, simple, strong, tiled, sidebar

**mountOnEnter** class-attribute

```
mountOnEnter: bool = None
```

False # Render only when tab is clicked

**scrollable** class-attribute

```
scrollable: bool = None
```

False # whether navigation supports content overflow scrolling, not supported in vertical and chrome modes; chrome mode compresses tabs by default (property deprecated)

showTip class-attribute

```
showTip: bool = None
```

False # Whether to support hints

showTipClassName class-attribute

```
showTipClassName: str = None
```

" " # The class of the prompt

sidePosition class-attribute

```
sidePosition: str = None
```

"left" # sidebar mode, tab position left / right

SOURCE class-attribute

```
source: str = None
```

tabs association data, can repeat tabs after association

tabs class-attribute

```
tabs: List[Item] = None
```

tabs content

tabsClassName class-attribute

```
tabsClassName: str = None
```

The class name of the Tabs Dom

tabsMode class-attribute

```
tabsMode: TabsModeEnum = None
```

display mode, the value can be line, card, radio, vertical, chrome, simple, strong, tiled, sidebar

toolbar class-attribute

```
toolbar: MSAUISchemaNode = None
```

Toolbar in tabs

toolbarClassName class-attribute

```
toolbarClassName: str = None
```

The class name of the toolbar in tabs

type class-attribute

```
type: str = 'tabs'
```

Specify as Tabs renderer

unmountOnExit class-attribute

```
unmountOnExit: bool = None
```

False # Destroy when tab is toggled

## Classes

### Item

Bases: MSAUINode

#### Attributes

className class-attribute

```
className: str = None
```

"bg-white b-l b-r b-b wrapper-md" # Tab area style

closable class-attribute

```
closable: bool = None
```

False # Whether to support deletion, prioritize over component's closable

disabled class-attribute

```
disabled: bool = None
```

False # Whether to disable

hash class-attribute

```
hash: str = None
```

Set to correspond to the hash of the url

icon class-attribute

```
icon: Union[str, Icon] = None
```

Tab's icon

iconPosition class-attribute

```
iconPosition: str = None
```

"left" # Tab's icon position left / right

reload class-attribute

```
reload: bool = None
```

Set the content to be re-rendered every time, useful for crud re-pulling

tab class-attribute

```
tab: MSAUISchemaNode = None
```

Content area

title class-attribute

```
title: str = None
```

Tab title

unmountOnExit class-attribute

```
unmountOnExit: bool = None
```

Destroy the current tab bar every time you exit

## TabsTransfer

Bases: Transfer

Combination shuttle

### Attributes

type class-attribute

```
type: str = 'tabs-transfer'
```

## TabsTransferPicker

Bases: `Transfer`

Combination shuttle selector

### Attributes

type `class-attribute`

```
type: str = 'tabs-transfer-picker'
```

## Tasks

Bases: `MSAUINode`

A collection of task actions

### Attributes

btnClassName `class-attribute`

```
btnClassName: str = None
```

"btn-sm btn-default" # Configure the container button className

btnText `class-attribute`

```
btnText: str = None
```

"Go Live" # Action button text

checkApi `class-attribute`

```
checkApi: MSA_UI_API = None
```

Return a list of tasks, see items for the returned data.

className `class-attribute`

```
className: str = None
```

Class name of the outer Dom

interval `class-attribute`

```
interval: int = None
```

3000 # When there is a task in progress, it will be detected again at regular intervals, and the time interval is configured by this, default 3s.

items `class-attribute`

```
items: List[Item] = None
```

List of tasks

operationLabel class-attribute

```
operationLabel: str = None
```

"operation" # Description of the operation column

reSubmitApi class-attribute

```
reSubmitApi: MSA_UI_API = None
```

This MSA\_UI\_API is used when submitting if the task fails and can be retried

remarkLabel class-attribute

```
remarkLabel: str = None
```

"Remarks" # Remarks column description

retryBtnClassName class-attribute

```
retryBtnClassName: str = None
```

"btn-sm btn-danger" # Configure the container retry button className

retryBtnText class-attribute

```
retryBtnText: str = None
```

"Retry" # Retry action button text

statusLabel class-attribute

```
statusLabel: str = None
```

"Status" # Status column description

statusLabelMap class-attribute

```
statusLabelMap: List[str] = None
```

Configuration of the class name corresponding to the status display ["label-warning", "label-info", "label-success", "label-danger", "label-default", "label-danger"]

statusTextMap class-attribute

```
statusTextMap: List[str] = None
```

"[\"Not Started\", \"Ready\", \"In Progress\", \"Error\", \"Completed\", \"Error\"]\" # Status display corresponding to the text display configuration

submitApi class-attribute

```
submitApi: MSA_UI_API = None
```

The MSA\_UI\_API used to submit the task

tableClassName class-attribute

```
tableClassName: str = None
```

Class name of the table Dom

taskNameLabel class-attribute

```
taskNameLabel: str = None
```

"Task name" # Description of the task name column

type class-attribute

```
type: str = 'tasks'
```

Specify as Tasks renderer

**Classes**

**Item**

Bases: MSAUINode

**Attributes**

key class-attribute

```
key: str = None
```

Task key value, please distinguish it uniquely

label class-attribute

```
label: str = None
```

Task name

remark class-attribute



```
remark: str = None
```

Current task status, supports html

status class-attribute

```
status: str = None
```

Task status: 0: initial, inoperable. 1: ready, operable. 2: in progress, not yet finished. 3: with errors, not retryable. 4: has ended normally. 5: with errors, and retryable.

## Textarea

Bases: FormItem

Multi-line text input box

### Attributes

maxLength class-attribute

```
maxLength: int = None
```

Limit the maximum number of words

maxRows class-attribute

```
maxRows: int = None
```

maximum number of lines

minLength class-attribute

```
minLength: int = None
```

Limit the minimum number of words

minRows class-attribute

```
minRows: int = None
```

minimum number of rows

readOnly class-attribute

```
readOnly: bool = None
```

whether to read only

showCounter class-attribute

```
showCounter: bool = True
```

Whether to show the counter

trimContents class-attribute

```
trimContents: bool = None
```

whether to remove first and last blank text

type class-attribute

```
type: str = 'textarea'
```

## Transfer

Bases: FormItem

shuttle

### Attributes

columns class-attribute

```
columns: List[dict] = None
```

When the display form is table can be used to configure which columns to display, similar to the columns in the table configuration, but only the display function.

delimiter class-attribute

```
delimiter: str = None
```

"False" # Splice character

extractValue class-attribute

```
extractValue: bool = None
```

False # extract value

joinValues class-attribute

```
joinValues: bool = None
```

True # Splice values

leftMode class-attribute

```
leftMode: str = None
```

Configure the left option set when the display is associated, supports list or tree. default is list. rightMode: str = None # Configure the left option set when the display is associated.

leftOptions class-attribute

```
leftOptions: List[dict] = None
```

Used to configure the left set of options when the display is associated.

menuTpl class-attribute

```
menuTpl: MSAUISchemaNode = None
```

Used to customize the option display.

options class-attribute

```
options: MSAOptionsNode = None
```

options group

resultTitle class-attribute

```
resultTitle: str = None
```

"Current selection" # The title text of the right result

rightMode class-attribute

```
rightMode: str = None
```

Use to configure the right option set when the display is associated, options are: list, table, tree, chained.

searchApi class-attribute

```
searchApi: MSA_UI_API = None
```

You can set an api if you want to search through the interface.

searchResultMode class-attribute

```
searchResultMode: str = None
```

If not set will use the value of selectMode, can be configured separately, refer to selectMode, determine the search results display form.

searchable class-attribute

```
searchable: bool = None
```

False # When set to true means that options can be retrieved by partial input.

selectMode class-attribute

```
selectMode: str = None
```

"list" # Optional: list, table, tree, cascaded, associated. respectively: list form, table form, tree selection form, tree selection form cascade selection form, associated selection form (the difference with cascade selection is that cascade is infinite, while associated is only one level, and the left side of associated can be a tree).

selectTitle class-attribute

```
selectTitle: str = None
```

"Please select" # The title text on the left side

sortable class-attribute

```
sortable: bool = None
```

False # Results can be sorted by dragging and dropping

source class-attribute

```
source: MSA_UI_API = None
```

dynamic options group

statistics class-attribute

```
statistics: bool = None
```

True # Whether to display statistics

type class-attribute

```
type: Literal[
    transfer,
    transfer - picker,
    tabs - transfer,
    tabs - transfer - picker,
] = "transfer"
```

valueTpl class-attribute

```
valueTpl: MSAUISchemaNode = None
```

Used to customize the display of values

## TransferPicker

Bases: `Transfer`

shuttlePicker

### Attributes

type `class-attribute`

```
type: str = 'transfer-picker'
```

## TreeSelect

Bases: `InputTree`

Tree Selector

### Attributes

hideNodePathLabel `class-attribute`

```
hideNodePathLabel: bool = None
```

Whether to hide the path of the selected node in the selection box label information

type `class-attribute`

```
type: str = 'tree-select'
```

## Validation

Bases: `MSABaseUIModel`

### Attributes

equals `class-attribute`

```
equals: str = None
```

The current value must be exactly equal to xxx.

equalsField `class-attribute`

```
equalsField: str = None
```

The current value must match the value of the xxx variable.

isAlpha `class-attribute`

```
isAlpha: bool = None
```

Must be a letter.

isAlphanumeric class-attribute

```
isAlphanumeric: bool = None
```

Must be alphabetic or numeric.

isEmail class-attribute

```
isEmail: bool = None
```

Must be Email.

isFloat class-attribute

```
isFloat: bool = None
```

Must be floating point.

isId class-attribute

```
isId: bool = None
```

if it is an ID number, no check

isInt class-attribute

```
isInt: bool = None
```

Must be integer.

isJson class-attribute

```
isJson: bool = None
```

If or not it is a legal Json string. isUriPath: bool = None

isLength class-attribute

```
isLength: int = None
```

If or not the length is exactly equal to the set value.

isNumeric class-attribute

```
isNumeric: bool = None
```

Must be numeric.

isPhoneNumber class-attribute

```
isPhoneNumber: bool = None
```

If or not it is a legitimate phone number

isTelNumber class-attribute

```
isTelNumber: bool = None
```

if it is a legitimate phone number

isUrl class-attribute

```
isUrl: bool = None
```

Must be Url.

isZipcode class-attribute

```
isZipcode: bool = None
```

Whether it is a zip code number

matchRegex class-attribute

```
matchRegex: str = None
```

Must hit some regular. /foo/

maxLength class-attribute

```
maxLength: int = None
```

The maximum length.

maximum class-attribute

```
maximum: int = None
```

The maximum value.

minLength class-attribute

```
minLength: int = None
```

The minimum length.

minimum class-attribute

```
minimum: int = None
```

The minimum value.

## Video

Bases: MSAUINode

video

### Attributes

autoPlay class-attribute

```
autoPlay: bool = None
```

Whether to auto play

className class-attribute

```
className: str = None
```

class name of the outer Dom

isLive class-attribute

```
isLive: bool = None
```

False # whether it is live, need to add on when the video is live, supports flv and hls formats

muted class-attribute

```
muted: bool = None
```

whether to mute

poster class-attribute

```
poster: str = None
```

video cover address

rates class-attribute

```
rates: List[float] = None
```

multiplier in the format [1.0, 1.5, 2.0]



src class-attribute

```
src: str = None
```

video address

type class-attribute

```
type: str = 'video'
```

Specify as video renderer

videoType class-attribute

```
videoType: str = None
```

Specify the format of the live video

## Wizard

Bases: MSAUINode

Wizard

### Attributes

actionClassName class-attribute

```
actionClassName: str = None
```

"btn-sm btn-default" # Button CSS class name

actionFinishLabel class-attribute

```
actionFinishLabel: str = None
```

"Finish" # Finish button text

actionNextLabel class-attribute

```
actionNextLabel: str = None
```

"Next" # Next button text

actionNextSaveLabel class-attribute

```
actionNextSaveLabel: str = None
```

"Save and Next" # Save and Next button text

actionPrevLabel class-attribute

```
actionPrevLabel: str = None
```

"Previous" # Previous button text

api class-attribute

```
api: MSA_UI_API = None
```

The interface saved in the last step.

className class-attribute

```
className: str = None
```

Outer CSS class name

initApi class-attribute

```
initApi: MSA_UI_API = None
```

Initialize the data interface

initFetch class-attribute

```
initFetch: MSA_UI_API = None
```

Initialize whether to pull data.

initFetchOn class-attribute

```
initFetchOn: MSAUIExpression = None
```

Initially pull data or not, configured by expression

mode class-attribute

```
mode: str = None
```

Display mode, choose: horizontal or vertical

redirect class-attribute

```
redirect: MSAUITemplate = None
```

"3000" # Jump after the operation.

reload class-attribute

```
reload: str = None
```

Refresh the target object after the operation. Please fill in the name value set by the target component, if it is window then the whole current page will be refreshed.

startStep class-attribute

```
startStep: int = None
```

"1" # Start default value, from which step to start. Templates can be supported, but only if the template is rendered and the current step is set when the component is created, and when the component is refreshed later. The current step will not change according to startStep

steps class-attribute

```
steps: List[Step] = None
```

Array to configure step information

target class-attribute

```
target: str = None
```

"False" # You can submit the data to another component instead of saving it yourself. Please fill in the name value set by the target component. If you fill in window, the data will be synced to the address bar, and the component that depends on the data will be refreshed automatically.

type class-attribute

```
type: str = 'wizard'
```

Specify as a Wizard component

## Classes

### Step

Bases: MSAUINode

#### Attributes

api class-attribute

```
api: MSA_UI_API = None
```

The current step saves the interface, can be unconfigured.

body class-attribute

```
body: List[FormItem] = None
```

A collection of form items for the current step, see FormItem.

**horizontal** class-attribute

```
horizontal: Horizontal = None
```

When in horizontal mode, used to control the left/right aspect ratio

**initApi** class-attribute

```
initApi: MSA_UI_API = None
```

The current step data initialization interface.

**initFetch** class-attribute

```
initFetch: bool = None
```

Whether the current step data initialization interface is initially pulling.

**initFetchOn** class-attribute

```
initFetchOn: MSAUIExpression = None
```

Whether the current step data initialization interface is pulling initially, using an expression to determine.

**mode** class-attribute

```
mode: str = None
```

Show default, same as mode in Form, choose: normal, horizontal or inline.

**title** class-attribute

```
title: str = None
```

Step title

## Functions

Last update: September 16, 2022

Created: September 16, 2022