

Ant Technology

Framework and UI
Components
User Guide

Document Version: 20230209

Legal disclaimer

Ant Group all rights reserved©2022.

No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company, or individual in any form or by any means without the prior written consent of Ant Group.

Trademark statement



and other trademarks related to Ant Group are owned by Ant Group. The third-party registered trademarks involved in this document are owned by the right holder according to law.

Disclaimer

The content of this document may be changed due to product version upgrades, adjustments, or other reasons. Ant Group reserves the right to modify the content of this document without notice and the updated versions of this document will be occasionally released through channels authorized by Ant Group. You must pay attention to the version changes of this document as they occur and download and obtain the latest version of this document from Ant Group's authorized channels. Ant Group does not assume any responsibility for direct or indirect losses caused by improper use of documents.

Document conventions

Style	Description	Example
 Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	 Danger: Resetting will result in the loss of user configuration data.
 Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	 Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
 Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	 Notice: If the weight is set to 0, the server no longer receives new requests.
 Note	A note indicates supplemental instructions, best practices, tips, and other content.	 Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click Settings > Network > Set network type .
Bold	Bold formatting is used for buttons, menus, page names, and other UI elements.	Click OK .
Courier font	Courier font is used for commands	Run the <code>cd /d C:/window</code> command to enter the Windows system folder.
<i>Italic</i>	Italic formatting is used for parameters and variables.	<code>bae log list --instanceid</code> <i>Instance_ID</i>
[] or [a b]	This format is used for an optional value, where only one item can be selected.	<code>ipconfig [-all -t]</code>
{ } or {a b}	This format is used for a required value, where only one item can be selected.	<code>switch {active stand}</code>

Table of Contents

1. Framework and UI Components	10
1.1. H5 Based - Kylin framework	10
1.1.1. kylin introduction	10
1.1.2. Quick start guide	10
1.1.2.1. Build front-end development workspace	10
1.1.2.2. Develop and Debug	12
1.1.3. Project structure	13
1.1.3.1. Introduction to Scaffolding	13
1.1.3.2. Page	17
1.1.3.3. Component	19
1.1.3.4. Command line tool	32
1.1.4. Plugin	36
1.1.4.1. mock	36
1.1.4.2. resource	37
1.1.4.3. Expansion capability	38
1.2. H5 based - UI library	40
1.2.1. Introduction to UI component library	40
1.2.2. User guide	40
1.2.3. Basic components	41
1.2.3.1. AButton	41
1.2.3.2. Flexbox	47
1.2.4. Tab components	58
1.2.4.1. Tab	58
1.2.4.2. TabPanel	61
1.2.5. Pop-up component	63
1.2.5.1. ActionSheet	63

1.2.5.2. ADialog	67
1.2.5.3. Filter	80
1.2.5.4. Toast	89
1.2.6. Entry component	95
1.2.6.1. List	95
1.2.7. Input component	112
1.2.7.1. Checkbox	112
1.2.7.2. Input	121
1.2.8. Loading component	128
1.2.8.1. Loading	128
1.2.9. Result page component	132
1.2.9.1. Message	132
1.2.9.2. PageResult	135
1.2.10. Notification component	141
1.2.10.1. Inform	141
1.2.10.2. Notice	143
1.3. Introduction to Native framework	145
1.4. Native based - Android component library	149
1.4.1. Quick start	149
1.4.2. Dialog component	150
1.4.2.1. Card menu	150
1.4.2.2. Cascade picker	153
1.4.2.3. Date picker	156
1.4.2.4. Float menu	159
1.4.2.5. Image dialog	161
1.4.2.6. Input dialog	172
1.4.2.7. List dialog	174
1.4.2.8. Notice dialog	180

1.4.2.9. Operation result dialog	183
1.4.2.10. Pop up menu	185
1.4.2.11. Recording float tip	187
1.4.2.12. Toast	188
1.4.3. Input components	190
1.4.3.1. Amount input box	190
1.4.3.2. Input box	194
1.4.3.3. Numerical keyboard	206
1.4.3.4. Search bar	209
1.4.3.5. Search input box	213
1.4.4. Item component	215
1.4.4.1. Auxiliary description component	215
1.4.4.2. Bank card item component	216
1.4.4.3. Coupons item component	217
1.4.4.4. List item component	217
1.4.5. Result page components	235
1.4.5.1. Progress page	235
1.4.5.2. Net error page	237
1.4.5.3. QR code page	239
1.4.5.4. Result page	242
1.4.6. Loading component	245
1.4.7. Navigation component	247
1.4.7.1. Carousel component	248
1.4.7.2. List component	248
1.4.7.3. Title bar component	252
1.4.8. Other component	260
1.4.8.1. Index component	260
1.4.8.2. Button component	262

1.4.8.3. Operation bar component	267
1.4.8.4. Check icon component	269
1.4.8.5. Icon component	270
1.4.8.6. Refresh component	274
1.4.8.7. Switch tab component	276
1.4.8.8. TabBar item component	279
1.5. Native based - iOS component library	280
1.5.1. Quick start	280
1.5.2. Basic components	281
1.5.2.1. Activity Indicator base class	281
1.5.2.2. Switch base class	281
1.5.2.3. Check box control	281
1.5.2.4. Image base class	284
1.5.2.5. Label base class	284
1.5.2.6. Footer base class	284
1.5.2.7. mPaaS customized loading control	287
1.5.2.8. Button base class	288
1.5.3. Input components	292
1.5.3.1. Image input box	292
1.5.3.2. Paragraph input box	293
1.5.3.3. Simplified amount input box	294
1.5.3.4. Amount input box	296
1.5.3.5. Normal input box	299
1.5.3.6. Search input box	302
1.5.3.7. Search bar component	304
1.5.3.8. Verification code input box	307
1.5.4. Item component	308
1.5.5. Pop-up window component	322

1.5.5.1. Action sheet	322
1.5.5.2. Date picker component	328
1.5.5.3. Menu component	337
1.5.5.4. Recording status layer	346
1.5.5.5. Image dialog	348
1.5.5.6. Input dialog	353
1.5.5.7. Toast component	357
1.5.5.8. Card menu	363
1.5.5.9. Operation result dialog	373
1.5.5.10. Cascade picker	376
1.5.5.11. Notification dialog	381
1.5.5.12. Custom date picker	386
1.5.6. Loading components	390
1.5.6.1. Pull-up refresh control	390
1.5.6.2. Pull-down refresh component	396
1.5.6.3. Loading component	404
1.5.7. Result page component	407
1.5.7.1. Result page component	407
1.5.7.2. Exception page component	409
1.5.8. Numeric keypad component	413
1.5.9. Guidance component	415
1.5.9.1. Prompt component	415
1.5.9.2. Floating layer bar component	416
1.5.10. Pop menu component	417
1.5.11. Navigation components	420
1.5.11.1. Vertical tab	420
1.5.11.2. Double title	423
1.5.11.3. Navigation bar	425

1.5.11.4. Custom navigation bar	428
1.5.12. QR code component	431
1.5.13. Refresh component	433
1.5.14. Other components	436
1.5.14.1. Carousel component	436
1.5.14.2. Segment component	442
1.5.14.3. Icon component	448
1.5.14.4. Index component	451
1.5.14.5. Title bar segment component	457
1.5.14.6. Navigation button	458
1.5.14.7. Adaptation and dependency	460
1.5.14.8. Image picker encapsulation	463

1. Framework and UI Components

1.1. H5 Based - Kylin framework

1.1.1. kylin introduction

Kylin is a wireless front-end solution for mPaaS HTML5 containers, with various advantages such as efficient runtime, consistent development experience, rich R&D support, and perfect UI components. It addresses a series of issues in mobile Hybrid development such as front-end packaging, browser compatibility, and API mocking.

Kylin only provides a view-layer framework based on Vue.js 2.0 for efficient DOM updates with minimal JS loading overhead. Before using Kylin, read and understand the [Vue.js documentation](#).

As a wireless front-end solution, Kylin assures compatibility only with the `Safari`, `UC Webview`, and `Google Chrome` browsers.

Note

mPaaS has provided updated and more stable Mini program components as the main maintenance and development direction in the future, and stopped the maintenance of Kylin. Compared with the Kylin solution, the Mini program component has higher ease of use and stability, and can adapt to a variety of scenarios. Therefore, it is recommended that new users access the Mini program, and users who have already connected to the Kylin framework are advised to switch to the Mini program component when they encounter new function requirements. For more details, see [Mini program](#).

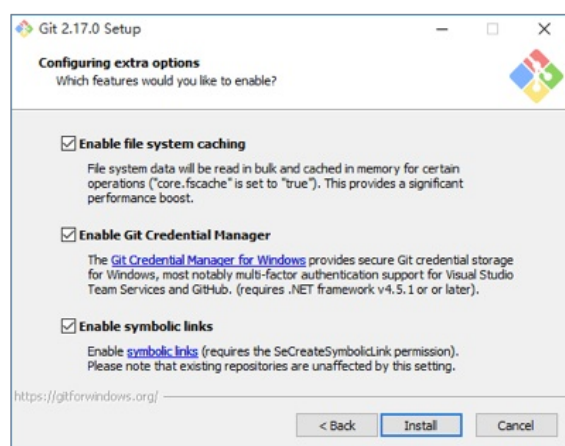
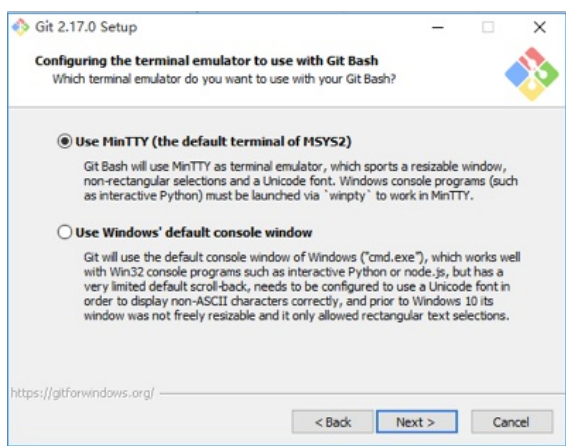
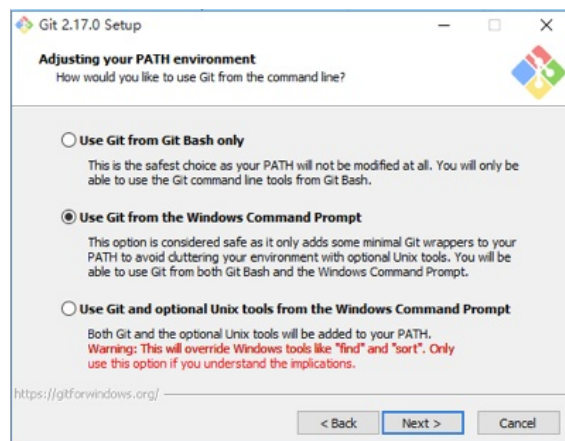
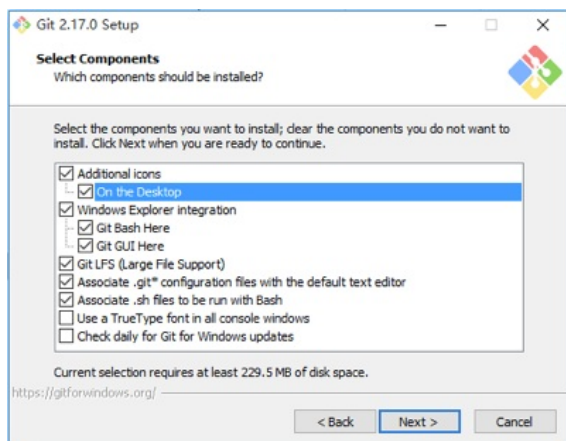
1.1.2. Quick start guide

1.1.2.1. Build front-end development workspace

Node.js and cnpm need to be installed for the front-end development workspace. This topic guides you to build environments for different operating systems. If you use Windows operating system, you need to complete user configuration first.

Build front-end development workspace in Windows

1. Complete Windows user configuration. Follow the figure below to install the `MinGW` command prompt. Download the installation file at [git-scm](#).



2. [Download](#) and install Node.js v8.

3. Install cnpm.

i. In MinGW or on a terminal, run the following command to install `cnpm` :

```
npm install -g cnpm --registry=http://registry.npmmirror.com
```

Note

Do not install `cnpm` in `alias` method that is used for adding the `npm` parameter. For more information about `cnpm`, see [NPM mirror](#).

- ii. After cnpm is installed, run the following command to verify the installation.

```
cnpm -v
cnpm@6.1.0 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\lib\parse_argv.js)
npm@6.11.3 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npm\lib\npm.js)
node@8.11.1 (D:\Program Files (x86)\nodejs\node.exe)
npminstall@3.23.0 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npminstall\lib\index.js)
prefix=C:\Users\wb-wly538545\AppData\Roaming\npm
win32 ia32 10.0.17134
registry=https://r.npm.taobao.org
```

Build front-end development workspace in macOS

1. [Download](#) and install Node.js v8.
2. Install cnpm.

- i. In MinGW or on a terminal, run the following command to install `cnpm` :

```
npm install -g cnpm --registry=http://registry.npmmirror.com
```

Note

Do not install `cnpm` in `alias` method that is used for adding the `npm` parameter. For more information about `cnpm`, see [NPM mirror](#).

- ii. After cnpm is installed, run the following command to verify the installation.

```
cnpm -v
cnpm@6.1.0 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\lib\parse_argv.js)
npm@6.11.3 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npm\lib\npm.js)
node@8.11.1 (D:\Program Files (x86)\nodejs\node.exe)
npminstall@3.23.0 (C:\Users\wb-wly538545\AppData\Roaming\npm\node_modules\cnpm\node_modules\npminstall\lib\index.js)
prefix=C:\Users\wb-wly538545\AppData\Roaming\npm
win32 ia32 10.0.17134
registry=https://r.npm.taobao.org
```

1.1.2.2. Develop and Debug

Development commissioning involves the following steps:

1. Install a dependency
2. Development commissioning
3. Build

Click [Demo](#) to get a code example and complete the following operations.

Install a dependency

Enter the root directory of the project and use `cnpm` to install the `npm` dependency.

```
# Install the npm dependency.  
cnpm install
```

Development commissioning

After installing a workspace, enable the development mode with a command such as the following command:

```
cnpm run dev
```

The preceding command performs the following steps:

- Run `kylin build --dev` to start the following in dev mode:
 - Compression of `css` / `js` without `compress`
 - Automatic change of the `watch` code
 - Hot code updates
- Start a server at `http://localhost:8090/`.

Build

Use the following command to enable the build mode.

```
cnpm run build
```

The preceding command performs the following steps:

- Run the `kylin build` command to compile the source code of the project.
- Export the compilation output to the `./www/` directory for subsequent packaging.

1.1.3. Project structure

1.1.3.1. Introduction to Scaffolding

This topic describes the project initialization structure.

Initialization structure

The project initialization structure is as follows:

```

project
├── mock
│   ├── mock.config.js
│   └── rpc
│       └── test.js
├── package.json
├── www
└── src
    ├── common
    │   ├── components
    │   ├── css
    │   │   └── base.less
    │   ├── img
    │   └── js
    ├── layout
    │   ├── index.html
    │   └── layout.html
    └── pages
        ├── index
        │   ├── components
        │   ├── index.js
        │   └── store
    
```

Subdirectory

- [mock](#)
- [package.json](#)
- [www](#)
- [src/common](#)
- [src/layout](#)
- [src/pages](#)
- [Frequently used parameters](#)

mock

This directory provides a data mocking method. When the startup adopts the `cnpm run dev:mock` method, the data APIs that correspond to the `rpc` and `jsapi` directories are automatically loaded.

package.json

The `kylinApp` field in the `package.json` file contains the metadata about project configurations, including `pages`, `output`, `devPort`, `plugins`, and `dirAlias`.

The following code shows a simple example:

```
{
  "kylinApp": {
    "output": "www",
    "pages": {
      "index": {...}
    },
    "devPort": 8090,
    "dirAlias": {
      "common": "./src/common/",
      "pages": "./src/pages/"
    },
    "plugins": [

  ]
}
}
```

WWW

The output of `cnpm run build` command are automatically exported to the `www` directory.

src/common

This directory stores the `css`, `js`, and `img` files that are used in the project.

src/layout

This directory corresponds to the pages in the `./src/pages/${pageName}` directory. You can configure the HTML template path used by the corresponding page in the `package.json` file. This directory supports the nunjucks syntax.

src/pages

This directory stores pages. Each page is stored in the corresponding `./src/pages/${pageName}/` directory. The directory for each page contains the `components` and `store` directories and the `index.js` file.

- In the `components` directory, each component is a `Vue` component. For information about coding standards, see [Component specifications](#).
- The `store` directory contains a `Vuex.Store` instance. For more information, see [Status injection](#).
- The `index.js` file is the main entry of the current page. The page generates a specific `${pageName}.html` page.

Frequently used parameters

Frequently used parameters refer to all key values other than `pages`, `options`, and `plugins` at the lower level of `kylinApp`. `pages`, `options`, and `plugins` are described separately in the following sections.

Parameter	Type	Default value	Remarks
output	String	dist	The relative directory of output.
devPort	Number	8090	The IPv4 port number used for listening in dev mode. Value: 0.0.0.0:devPort.
dirAlias	Record	{}	Equivalent to the relative path used in <code>webpack.resolve.alias</code> .
pageTemplate	String	-	The public Nunjucks template.

pages

The following code lists the configuration items of the `pages` key-value pair. The `home` in the example indicates that the configurations are valid only for pages whose `pageName` is `home`.

```
{
  "kylinApp": {
    "pages": {
      "home": {
        ... // List fields.
      }
    }
  }
}
```

Field	Type	Default value	Remarks
entry	String	-	The relative path that redirects to the JS packaging entry of the current page.
template	String	-	The relative path that redirects to the HTML packaging path of the current page. If this field is empty, the value of the <code>kylinApp.pageTemplate</code> field will be used.

plugins

The `kylinApp.plugins` field is an array that supports loading plug-ins on demand.

```
{
  kylinApp: {
    plugins: [
      "xxxx",
      ["yyyy", { a: 1 }],
      "zzzz",
      ["6666", { b: 1 }]
    ]
  }
}
```

You can import plug-ins in two modes: **default configuration** and **extended configuration**. The preceding example imports the following plug-ins:

- @ali/kylin-plugin-xxxx, which is loaded in default configuration mode.
- @ali/kylin-plugin-yyyy, which is loaded by using the `{a:1}` option.
- @ali/kylin-plugin-zzzz, which is loaded in default configuration mode.
- @ali/kylin-plugin-6666, which is loaded by using the `{b:1}` option.

Existing plug-ins

Configurable plug-ins include `mock` and `resource`. For more information, see the following topics:

- [mock](#)
- [resource](#)

1.1.3.2. Page

Page is a `Webview` logical abstraction layer and also the root node for component mounting.

Code import

```
import { Page } from '@ali/kylin-framework';
```

Page declaration structure

The API contained in a `Page` is declared in [Page Interface](#) and provides complete control capabilities of `Vue.js` instances. The following shows a simple `Page` usage. `initOptions` is responsible for processing additional `Vue.js` configuration options.

```
import { Page } from '@ali/kylin-framework';
import IndexComponent from './indexComponent.vue';

class IndexPage extends Page {

  initOptions() {
    return {}
  }

  render(h) {
    return <IndexComponent></IndexComponent>
  }

}

new IndexPage('#app');
```

Page Interface

This topic describes the page API, including namespace.

Namespace

ES6 import method:

```
import { Page } from '@ali/kylin-framework';
```

API

Currently, `Page` provides the following member methods for derivation:

- `initOptions`
- `render`

initOptions

```
function initOptions(): VueOptions
```

Return value

The return result must be a valid `Vue.js` input parameter. Generally, we do not recommend that you introduce complex configuration to the `Page` layer. Involved logic can be maintained in `Component`.

render

The function must be a valid `render` function of `Vue.js`.

```
function render(): VNode
```

Return value

The return result must be a valid `VNode` element. Write according to `JSX` specifications.

1.1.3.3. Component

Component is expanded from a `Vue.js` component and provides parameter input capabilities that are equivalent to the `Vue` component. It provides `Decorator` styles for `class` during code writing.

Code import

```
import { Component, Watch } from '@ali/kylin-framework';
```

Structure of component declarations

A component can contain peer configurations of the `options` of `Vue`, such as data, JSX rendering functions, templates, mounting elements, methods, and life cycles. Component declarations include the following parts, which are separately packaged by using the `@Component` and `@Watch` decorators.

- Declarations of classes, which are packaged by using the `@Component` decorator.
- Decorations of class members, which are packaged without using a decorator.
- Declarations of class member methods, which are packaged usually without using a decorator unless the method needs to `watch` another declared variable

The following example shows you how to develop a simple component. For information about specific declaration parameters, see [Component API](#).

*.vue single file component

```

<!-- Hello.vue -->

<template>
  <div>hello {{name}}
    <Child></Child>
  </div>
</template>

<style>
  /* put style here */
</style>

<component default="Child" src="./child.vue" />

<script>
  import { Component } from '@ali/kylin-framework';

  @Component
  class Hello {
    data = {
      name: 'world'
    }
  }

  export default Hello;
</script>

```

For information about the label-style component dependencies in the `<component>` format, see [Component API](#).

Component API

Similar to Vue, the definition of a component is written in the `.vue` file. The following code shows a simple example:

```
<template>
  <div>
    <AButton @click="onClick">Click</AButton>
  </div>
</template>

<style lang="less" rel="stylesheet/less">
  /* less */
</style>

<dependency component="{ AButton }" src="@alipay/antui-vue" lazy/>

<script type="text/javascript">
  import { Component } from '@ali/kylin-framework';
  @Component
  export default class IndexView {
    props = {}
    data = {}
    get comput() { return this.data.c * 2 }
    onClick() {}
    mounted() {}
  }
</script>
```

The preceding example contains four top-level labels including `<template>` , `<style>` , and `<script>` , which are similar to those in Vue, and `<dependency>` .

The following sections describe the usage of the four labels. The definitions of the `<template>` and `<style>` labels are the same as those in `Vue` .

Script

Class structure

Define a Component. Based on the code structure, use Decorators of the class. Available decorators are `@Component` and `@Watch` , which can be imported as shown in the following code:

```
import { Component, Watch } from '@ali/kylin-framework';

@Component
export default class Hello {

}
```

Method type

The Component is declared as a class, and the class must be decorated with a Decorator. Inside a class, member variables are automatically processed by the `babel` plug-in during the building process. Member functions usually do not require a decorator unless `@Watch` is used. The following table describes the properties that the `@Component` decorator can process.

Member type	Name	Usage
get/set property	*	Used to convert into the <code>computed</code> property of <code>Vue</code> . You can call this by using <code>this[varName]</code> .
beforeCreate	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
created	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
beforeMount	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
mounted	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
beforeUpdate	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
updated	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
beforeDestroy	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
destroyed	Life cycle	The life cycle method, which is equivalent to that in <code>Vue</code> .
method	*	A regular member method that is used to convert into the method list <code>methods</code> in <code>Vue</code> .

getter/setter property

```
@Component
export default class Hello {
  get computName() {
    // to sth
  }
}
```

The preceding `getter` declaration is equivalent to the following Vue configuration:

```
HelloOption = {
  computed: {
    computName: {
      get: computName() {
        // to sth
      }
    }
  }
}
```

Likewise, `setter` is also extracted. If both `getter` and `setter` exist, they are extracted together.

Life cycle functions

```
@Component
export default class Hello {
  created() {}
  mounted() {}
}
```

The preceding life cycle functions `created` and `mounted` are extracted as arrays.

```
TestOption = {
  created: [function created() {}],
  mounted: [function mounted() {}],
}
```

The following life cycle methods are supported: `beforeCreate`, `created`, `beforeMount`, `mounted`, `beforeUpdate`, `updated`, `beforeDestroy`, and `destroyed`.

Watch

This decorator exists because `watch` requires the following elements:

- The variables to listen to
- Listening options
- Trigger function

Usage

The following table describes all the parameters for the `@Watch` decorator.

```
function Watch( key: string [, option: Object = {} ] ): PropertyDecorator
```

Parameter		Type	Usage
key		string	The name of the parameter that is listened to. Valid values: <code>computed</code> , <code>data</code> , and <code>props</code> .
option	deep	boolean	Specifies whether an inner data change is triggered when a complex object is listened. Default value: <code>false</code> .
	immediate	boolean	Specifies whether to immediately trigger a callback by using the current value of the expression. Default value: <code>false</code> .

Example

- For a member function that is decorated by `@Watch` , you can configure listening of multiple variables for the member function. In the following sample code, changes to the `a` and `c` parameters are listened. A change to either of the parameters will trigger the member method `OnChangeA` .
- `OnChangeA` is a member method in nature and is extracted to the `methods` block together with other member methods. Therefore, make sure that the name of this member method is unique.
- If there are additional configuration items for Watch, import them by using the `@Watch('a', {deep: false})` method. For more information about configuration items, see [Watch configuration items](#).

```
@Component
class WTest {

    data = {
        a: {
            b: 2
        },
        c: 3
    }

    @Watch('c')
    @Watch('a', {deep: false})
    OnChangeA(newVal, oldVal) {

    }
}
```

Note: The listener for `data.a` is converted. If the `deep: true` option is disabled, the `OnChangeA` method is not triggered when `data.a.b` is changed.

Property types

The build tool automatically applies the `@Component.Property` decorator to member variables. The final merging strategy depends on the identifier name of the decorated `member variable`. The following table describes some members that are built in the framework. Members that are not in the following table are passed through to the `options` object of `VueComponent`.

Member type	Name	Usage
Property	props	The props property of Vue.
Property	data	The data property of Vue. This property is converted into a function and supports the <code>this</code> method. Do not use <code>data() {}</code> .
Property	*	Other unknown properties, which are copied to the corresponding properties in the <code>options</code> object of <code>Vue</code> .

props

```
@Component
export default class Hello {

  props = {
    name: {
      type: String,
      default: 'haha'
    },
    num: Number
  }
}
```

The preceding definition of the member variable `props` is converted into the corresponding `props` property in the `options` object.

```
HelloOption = {
  props: {
    name: {
      type: String,
      default: 'haha'
    },
    num: Number
  }
}
```

For information about the complete definition structure, see the [API-props](#) topic in the Vue documentation.

data

```
@Component
export default class Hello {
  props = {
    name: {
      type: Number,
      default: 1
    },
  },
  data = {
    hello: this.props.name + 2
  }
}
```

The preceding definition of the member variable `data` is converted into a data function. You do not need to manually write a data function.

```
TestOption = {
  props: {
    name: {
      type: Number,
      default: 1
    },
  },
  data: function data() {
    return {
      hello: this.props.name + 2
    }
  }
}
```

dependency

In the preceding example, the definition of `<script>` does not describe how to use component dependencies. We recommend that you use the `<dependency>` label to mark component dependencies in the `.vue` file. In the following example, the `./child.vue` component is referenced.

```
<template>
  <child></child>
</template>

<dependency component="Child" src="./child.vue" />
```

Label properties

default import

The following table describes the properties that are used for introducing the `default` export of `ES6 Module` or the export of the `commonjs Module` object.

Parameter	Type	Default value	Remarks
component	string	Required	The identifier name to be imported to <code>options.components</code> .

Parameter	Type	Default value	Remarks
src	string	Required	The source of the component. The value of this property is the same as the result of the <code>require (src)</code> method.
lazy	boolean	false	Specifies whether to enable lazy loading for the component. If lazy loading is enabled, a module of the component is loaded only when <code>Vue</code> requires this module. Modules are loaded by using <code>require</code> .
style	string	undefined	By default, this property is not used. If you set a string for this property, the system loads the corresponding style file of the component by using the <code>\${src}/\${style}</code> path.

The following code shows an example:

```
<dependency component="name" src="source" lazy />
```

member import

The following table describes the properties that are used for introducing the export of `ES6 module` names.

Parameter	Type	Default value	Remarks
component	string	Required	The multiple naming identifiers to be imported to <code>options.components</code> . Enclose the identifiers in braces <code>{ , }</code> . Otherwise, the identifiers are identified as <code>default</code> import.
src	string	Required	The source of the component. The value of this property is the same as the result of the <code>require (src)</code> method.
lazy	boolean	false	Specifies whether to enable lazy loading for the component. If lazy loading is enabled, a module of the component is loaded only when <code>Vue</code> requires this module. Modules are loaded by using <code>require</code> .

The following code shows an example:

```
<dependency component="{ List, ListItem, AButton }" src="@alipay/antui-vue" lazy />
```

By default, on-demand `babel-plugin-import` loading is supported for the `@alipay/antui-vue` component library.

template

The template content structure is consistent with that of Vue.

```
<template>
  <div>Hello World</div>
</template>
```

style

```
<style lang="less" rel="stylesheet/less">
  /* less */
</style>
```

You can add the property tag `scoped`, so that the style takes effect only for the current component.

```
<style lang="less" rel="stylesheet/less" scoped>
  /* less */
</style>
```

Status injection

We recommend that you use the following `connect` mechanisms to pass through `$store` data:

1. [API declaration](#)
2. Use one of the following methods to pass through data:
 - [mapStateToProps](#)
 - [mapActionsToMethods](#)
 - [mapMethods](#)

If you need to use the properties of `Store` in Kylin components, we recommend that you do not use calculation properties to pass through the properties of the `$store` object. We recommend that you use the following method:

```
@Component
class Hello {
  // Associate the status in the store by using calculation properties.
  get hello() {
    return this.$store.state.hello
  }
}
```

API declaration

```
@Component({
  mapStateToProps: Object|Array,
  mapActionsToMethods: Object|Array,
  mapMethods: Array|Boolean,
  mapEvents: Array
})
class Hello {

}
```

mapStateToProps

This function is used to map specific key values in `state` to those in the `props` property of the current component. This function accepts parameter in the same way as the [auxiliary function `mapState`](#) that is provided by `Vuex`.

You can use one of the following methods to implement this feature:

Function-based method

The following code maps the data named `bbb` in `$store.state` to the `props` property of `aaa`.

```
{
  mapStateToProps: {
    aaa: (state, getters) => state.bbb
  }
}
```

String key-value pair-based method

The following code maps the data named `bbb` in `$store.state` to the `props` property of `aaa`.

```
{
  mapStateToProps: {
    aaa: 'bbb'
  }
}
```

String array-based method

The code below performs the following operation:

- Map the data named `aaa` in `$store.state` to the `props` property of `aaa`.
- Map the data named `bbb` in `$store.state` to the `props` property of `bbb`.

```
{
  mapStateToProps: ['aaa', 'bbb']
}
```

mapActionsToMethods

Similar to the input parameters of the `mapActions` function in `Vuex`, this function allows you to inject the `actions` property in the global `$state` object into the `methods` of the current component. You can complete the injection based on the name mapping of objects or based on the names of arrays.

```
@Component({
  mapActionsToMethods: ['a', 'b']
})
class IndexView {
  async doSomeAction() {
    const ret = await this.a(123);
    // Equivalent to a call.
    // const ret = await this.$store.dispatch('a', 123);
  }
}
```

mapMethods

Implement the `connect` mechanism by adding a middle layer component between a parent component and a child component. When the parent component calls a specific `method` in the child component, the parent component connects to the middle layer component instead of the child component. Use the following configurations to implement the required access:

```
@Component({
  mapMethods: true
})
export default class Child {
  a() {}
}
```

```
<template>
  <div>
    this is parent
    <child ref="child"></child>
  </div>
</template>
<script>
  @Component
  export default class Parent {
    b() {
      // Access is enabled here.
      this.$refs.child.a();
    }
  }
</script>
```

1.1.3.4. Command line tool

Initialize

If you need to add a new page after the project's scaffold is initialized, you can simply copy and paste relevant content or use the following commands to add page definitions and component definitions.

- [init-page](#)
- [init-component](#)

init-page

Command syntax

```
kylin init-page <pageName>
```

Precautions

- `pageName` in the preceding command is a required parameter, and it specifies the English name of the page to be created.
- If the current cwd contains `package.json` with the `kylinApp` field, the new `page` will be automatically added to `kylinApp.pages`.

init-component

Command syntax

```
kylin init-component <componentName>
```

Precautions

- `componentName` in the preceding command is a required parameter, and it specifies the English name of the component to be created.
- If the current cwd contains `package.json` with `kylinApp.pages` specifying more than one page, the system prompts the specific `page/components` directory for the newly created component.

Build

This part describes the command syntax for tool building, and the build prompt for public resource package injection.

Command syntax

```
kylin build # ... args
```

Items

Commonly used parameters

```
kylin build --dev      # dev building and static server
kylin build --server --no-prod --hot # dev building, static server, and hot update enabling
kylin build --server   # prod building and static server
kylin build --no-prod --watch # dev building and listening on file changes
```

Command line input parameters

Parameter	Type	Note
dev	Boolean	Similar to the original build tool, it uses the dev conf and enables the server. If this parameter is enabled, the following settings will be forcibly made: <code>prod=false</code> , <code>server=true</code> , and <code>hot=true</code> .
no-prod	Boolean	When <code>prod</code> is set to <code>true</code> , prod conf is used for compilation. When <code>prod</code> is set to <code>false</code> , dev conf is used for compilation. <code>NODE_ENV</code> is set in the same manner.
server	Boolean	Only the static server is enabled. When this parameter is enabled, the following setting is forcibly made: <code>watch=true</code> .
verbose	Boolean	The webpack output details.
watch	Boolean	Whether to monitor file changes.
no-compress	Boolean	Whether to disable compression. Compression is enabled by default.
no-common	Boolean	Whether to disable <code>CommonsChunkPlugin</code> , which is enabled by default.
hot	Boolean	Whether to enable hot update. Hot update is disabled by default. This parameter is used only when <code>prod</code> is set to <code>false</code> and <code>server</code> is set to <code>true</code> .

Parameter	Type	Note
open [entry]	Boolean, String	This parameter is valid only for <code>--server</code> . It is used to open the <code>entry</code> -specified <code>URL</code> . For <code>--open</code> without an <code>entry</code> specified, the first URL is processed.
mock	Boolean, String	This parameter is used to enable the <code>mock</code> plug-in to read <code>./mock/mock.config.js</code> .

kylinApp configuration options

Parameter	Type	Description
devPort	Number	Default listening IPv4 port 0.0.0.0:8090.
pageTemplate	String	Page template path.
output	String	Relative output directory.
options	Object	Additional options.
dirAlias	Object	Equivalent to <code>webpack.resolve.alias</code> , for example, { common: './src/common/' }.

Build prompt

Public resource package injection

`<script>` / `<link>` labels corresponding to the following `require` / `import` package paths are automatically injected into `HTML`.

Package name	Mapping global object	Mapping path
fastclick	FastClick	as.alipayobjects.com/g/luna-component/luna-fastclick/0.1.0/index.js
vue	Vue	a.alipayobjects.com/g/h5-lib/vue/2.1.6/vue.min.js
es6-promise	Promise	as.alipayobjects.com/g/component/es6-promise/3.2.2/es6-promise.min.js
fetch	fetch	as.alipayobjects.com/g/component/fetch/1.0.0/fetch.min.js
zepto	Zepto	a.alipayobjects.com/amui/zepto/1.1.3/zepto.js

1.1.4. Plugin

1.1.4.1. mock

Kylin-plugin-mock is a data mock plug-in developed for debugging JSAPI errors in the desktop browser (Google Chrome).

Enable the plug-in

Execute the following statement in the scaffold project, which is equivalent to the action of adding `-mock` during command execution:

```
cnpm run dev:mock
```

Use the plug-in

The `./mock/mock.config.js` file of the project contains the following configurations:

```
const config = {};

// The custom mock.
config.call = {
  // The mock rpc API.
  rpc: function (opts, callback) {
    var type = opts.operationType;
    var rpc = require('./rpc/' + type);
    var data = typeof rpc === 'function' ? rpc(opts) : rpc;
    // Prevent the input object from being modified in the service logic.
    data = Object.assign({}, data);
    // Simulate a server/network interface delay. Two logs are recorded. One is for the
    request, and the other contains the return result.
    setTimeout(() => {
      callback && callback(data);
    }, 2000);
  },
}

window.lunaMockConfig = config;
```

The preceding configuration maps data for APIs in `./mock/rpc/*.js`.

Examples

After `cnpm run dev:mock` is executed, the system enters the `mock` mode. In this mode, execute `AlipayJSBridge.call('abc')` in the browser to search for the mock API data in `./mock/jsapi/abc.js`.

1.1.4.2. resource

Kylin-plugin-resource is a resource interception mechanism designed for global offline resource packages on the mPaaS platform.

Use the plug-in

The `package.json` file in the scaffold file contains the following configuration:

```
[ "resource",
  {
    "map": {
      "vue": {
        "external": "Vue",
        "js": "https://gw.alipayobjects.com/as/g/h5-lib/vue/2.5.13/vue.min.js"
      },
      "fastclick": {
        "external": "FastClick",
        "js": "https://as.alipayobjects.com/g/luna-component/luna-fastclick/0.1.0/index.js"
      }
    }
  }
]
```

The above configuration indicates that some actions will be implemented when the following dependency statements appear:

```
import xxx from 'vue';
var xxx = require('vue');
```

When the `vue` dependency is used, the following actions are implemented:

1. Inject the `<script src="https://gw.alipayobjects.com/as/g/h5-lib/vue/2.5.13/vue.min.js" ></script>` script resource into the generated `HTML` template.
2. Redirect the preceding `vue` dependency to the value of `window.Vue`.

1.1.4.3. Expansion capability

The Kylin framework supports extending Webpack, Babel, and Express configurations.

Procedure

Extend the Webpack, Babel, and Express configuration as follows:

1. Create `plugin.js` in the root directory of the project.
2. Add the following configuration item to `kylinApp.plugins` in `package.json`:

```
// In package.json
{
  "kylinApp": {
    "plugins": [
      "module:./plugin.js"
    ]
  }
}
```

3. In `plugin.js`, write the following code. In the `modifyWebpackConfig` and `modifyBabelConfig` functions, modify the original `webpack` and `babel` configuration.

```
// In plugin.js
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
  return {
    webpack: function modifyWebpackConfig(webpackConfig) {
      console.log('webpackConfig', webpackConfig);
      return webpackConfig;
    },
    babel: function modifyBabelConfig(babelConfig) {
      console.log('babelConfig', babelConfig);
      return babelConfig;
    },
    express: function modifyExpress(expressInstance) {
      expressInstance.use(/* some middleware */);
    }
  }
}
```

Examples

Access the httpProxy plug-in

Note

The kylin-build package version should be 0.1.49 or higher.

1. Run the following command to install `http-proxy-middleware`.

```
cnpm install --save-dev http-proxy-middleware
```

2. Modify the `express` of the `httpProxy` plug-in.

```
// In plugin.js
var proxy = require('http-proxy-middleware');
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
  return {
    express: function modifyExpress(expressInstance) {
      expressInstance.use(
        proxy('/api', {target: 'http://www.baidu.com'})
        // For more configuration information, see the http-proxy-middleware module
        documentation.
      );
    }
  }
}
```

Access the px2rem plug-in

1. Run the following command to install `postcss-px2rem`.

```
cnpm install --save-dev postcss-px2rem
```

2. Modify `Webpack` of the `px2rem` plug-in.

```
// In plugin.js
var px2rem = require('postcss-px2rem');
exports.pluginName = '@ali/kylin-plugin-custom';
exports.default = function () {
  return {
    webpack: function modifyWebpackConfig(webpackConfig) {
      webpackConfig.vue.postcss.push(
        px2rem({
          // Settings of this parameter and **font-size** in the template html need
          // to be modified according to the UI interaction text of the project.
          remUnit: 100
        })
      );
      return webpackConfig;
    }
  }
}
```

1.2. H5 based - UI library

1.2.1. Introduction to UI component library

Ant UI is a front-end component library built based on the Vue.js framework and in compliance with visual interaction specifications of Ant Financial Alipay app. Ant UI originates from Ant UI wireless HTML5 style library, with interaction logic improved and configuration items abstracted.

Ant UI comprises the following components:

- @alipay/antui: Style library
- @alipay/antui-vue: Component library based on the style library

Related topics

[Ant UI wireless HTML5 style library](#)

1.2.2. User guide

In the Kylin project, `package.json` contains the `@alipay/antui-vue` dependencies. You can use either of the following methods to use an Ant UI front-end component.

- Use this component in the Kylin project. Import it as a `dependency`, because the Kylin project's scaffold includes dependencies of this component. For more information, see [Kylin](#).
- Use this component in other projects. Import it through ESM module. For more information, see [ESModule](#).

In addition, the API description is provided for each component because the `prop`, `slot`, `method`, and `event` APIs are provided for a standard Kylin component. For example, to use `AButton`, you can define the button size by using `size` in `props` and the button pattern by using `icon` in `slots`, and obtain tapping events based on `tap` in `events`. For specific information, see the API description for each component.

Kylin

The `.vue` component folder of the Kylin project supports the special syntax `<dependency>`. Use the following method to register component dependencies.

```
<dependency component="{ Notice }" src="@alipay/antui-vue" ></dependency>
```

The preceding syntax indicates that the `Notice` component can be used in `template` of the current `.vue` file.

ESModule

In other project structures, ensure the following:

loader

The `.vue` file can be loaded. For example, the file can be loaded by using [vue-loader](#) in `webpack` `.js`.

The loaded `.vue` file is used as a normal component.

```
import { Notice } from '@alipay/antui-vue';

Vue.component(Notice.name, Notice);
```

1.2.3. Basic components

1.2.3.1. AButton

The AButton topic describes different methods of using this component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ AButton }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { AButton } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
type	The type of the button. Valid values: <code>blue</code> , <code>white</code> , <code>warn</code> , <code>bottom</code> , and <code>page-result</code> (only for <code>PageResult</code>).	String	<code>blue</code>
size	The size of the button. Valid values: <code>default</code> and <code>tiny</code> . When it is set to <code>tiny</code> , type must be <code>white</code> and <code>blue</code> (unavailable for <code>[type=page-result]</code>).	String	<code>default</code>
disabled	Disable a button (unavailable for <code>[type=page-result]</code>).	Boolean	<code>false</code>
loading	Set a button icon to loading, applicable only when <code>size = default</code> (unavailable for <code>[type=page-result]</code>).	Boolean	<code>false</code>
success	Set a button icon to success, applicable only when <code>size = default</code> (unavailable for <code>[type=page-result]</code>).	Boolean	<code>false</code>
nativeType	<code>type</code> attribute in DOM	String	<code>button</code>
href	When it is not empty, use the <code>a</code> tag for rendering; otherwise, use the <code>button</code> tag for rendering by default.	String	-
inactiveLoading	When it is set to <code>true</code> and <code>loading</code> is set to <code>true</code> , trigger no <code>tap</code> event and retain the style of <code>button</code> . When it is set to <code>false</code> , retain the original action of <code>loading</code> .	Boolean	<code>false</code>

slots

Name	Description	Scope
-	The default slot for filling in the button text.	-

Name	Description	Scope
icon	The custom node for filling the icon. The default value is filled in when <code>loading</code> / <code>success</code> is set.	-

events

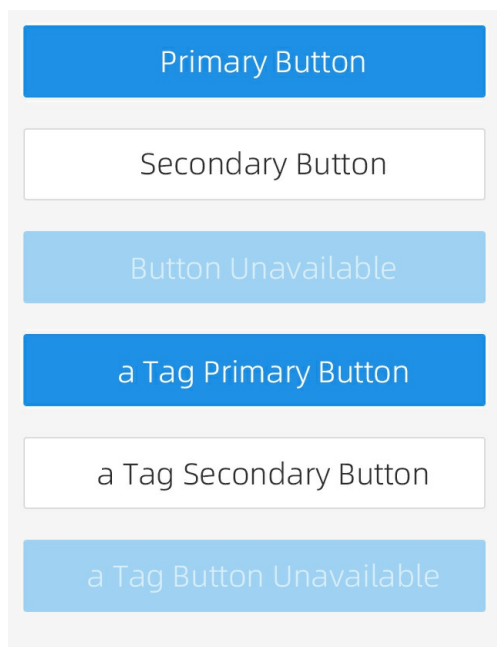
Name	Description	Function
tap	The operation when you press a mouse button.	Function(event : Event): void

Demo

- [Button](#)
- [Auxiliary button](#)
- [Load](#)
- [Warning](#)

Button

Sample image



Sample code

```
<template>

  <div style="padding: 10px;">
    <AButton >Primary Button</AButton>
    <AButton type="white">Secondary Button</AButton>
    <AButton disabled>Button Unavailable</AButton>

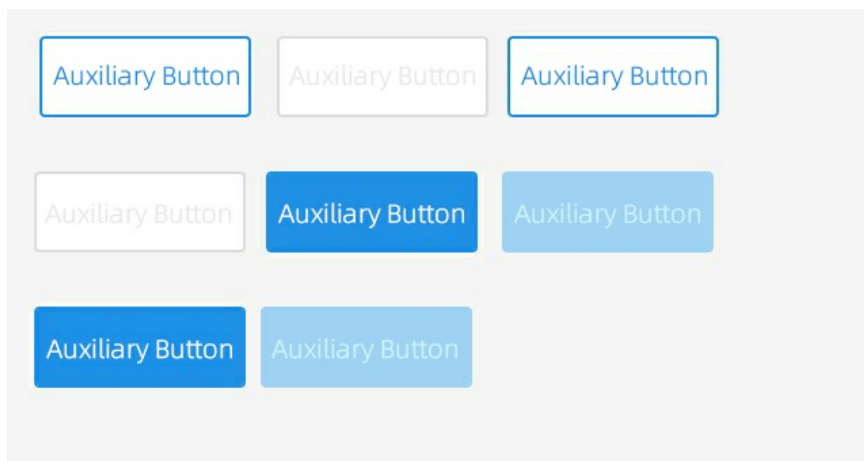
    <AButton href="#">a Tag Primary Button</AButton>
    <AButton href="#" type="white">a Tag Secondary Button</AButton>
    <AButton href="#" disabled>a Tag Button Unavailable</AButton>
  </div>

</template>

<style scoped>
.am-button {
  margin-bottom: 20px;
}
</style>
```

Auxiliary button

Sample image



Sample code

```
<template>

  <div style="padding: 10px;">

    <AButton size="tiny" type="white">Auxiliary Button</AButton>
    <AButton size="tiny" type="white" disabled>Auxiliary Button</AButton>

    <AButton href="#" size="tiny" type="white">Auxiliary Button</AButton>
    <AButton href="#" size="tiny" type="white" disabled>Auxiliary Button</AButton>

    <AButton size="tiny">Auxiliary Button</AButton>
    <AButton size="tiny" disabled>Auxiliary Button</AButton>

    <AButton href="#" size="tiny">Auxiliary Button</AButton>
    <AButton href="#" size="tiny" disabled>Auxiliary Button</AButton>

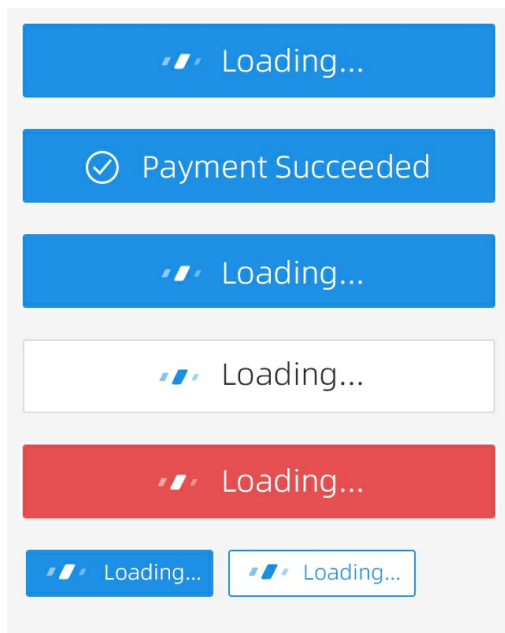
  </div>

</template>

<style scoped>
.am-button {
  margin-bottom: 20px;
}
</style>
```

Load

Sample image



Sample code

```
<template>

  <div style="padding: 10px;">
    <AButton loading>Loading...</AButton>
    <AButton success>Payment Succeeded...</AButton>

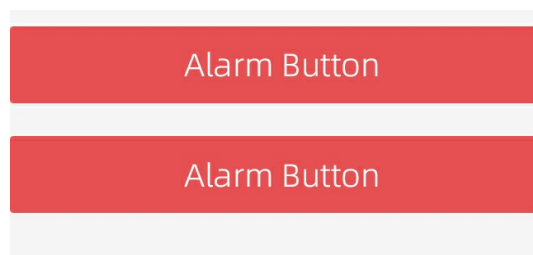
    <AButton loading type="blue">Loading...</AButton>
    <AButton loading type="white">Loading...</AButton>
    <AButton loading type="warn">Loading...</AButton>
    <AButton loading type="bottom">Loading...</AButton>
    <AButton loading type="blue" size="tiny">Loading...</AButton>
    <AButton loading type="white" size="tiny">Loading...</AButton>
  </div>

</template>

<style scoped>
.am-button {
  margin-bottom: 20px;
}
</style>
```

Warning

Sample image



Sample code

```
<template>

  <div style="padding: 10px 0;">
    <AButton type="warn">Alarm Button</AButton>
    <AButton type="warn" disbaled>Alarm Button</AButton>
  </div>

</template>

<style scoped>
.am-button {
  margin-bottom: 20px;
}
</style>
```

1.2.3.2. Flexbox

This topic describes different methods of using the Flexbox component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props and slots.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ Flexbox, FlexboxItem }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Flexbox, FlexboxItem } from '@alipay/antui-vue';
```

API description

Flexbox

props

Property	Description	Type	Default value
direction	The children's arrangement method. Valid values: <code>row</code> , <code>row-reverse</code> , <code>column</code> , and <code>column-reverse</code> .	String	<code>'row'</code>
wrap	The children's line-wrap method. Valid values: <code>nowrap</code> , <code>wrap</code> , and <code>wrap-reverse</code> .	String	<code>'nowrap'</code>
justify	The children's alignment on the main axis. Valid values: <code>start</code> , <code>end</code> , <code>center</code> , <code>between</code> , and <code>around</code> .	String	<code>'start'</code>

Property	Description	Type	Default value
align	The children's alignment on cross axes. Valid values: <code>start</code> , <code>center</code> , <code>end</code> , <code>baseline</code> , and <code>stretch</code> .	String	<code>'center'</code>
alignContent	The alignment on multiple axes. Valid values: <code>start</code> , <code>end</code> , <code>center</code> , <code>between</code> , <code>around</code> , and <code>stretch</code> .	String	<code>'stretch'</code>

slots

Name	Description	Scope
-	Default slot used to populate the content of the layout.	-

FlexboxItem

props

FlexboxItem additionally provides the `flex: 1` style by default, ensuring the same width for all items. A child of Flexbox may not necessarily be a FlexboxItem.

slots

Name	Description	Scope
-	Default slot used to populate the content of the element.	-

Demo

Basic style

Sample image



Sample code

```
<template>
  <div>
    <Flexbox>
      <FlexboxItem>2 Column</FlexboxItem>
      <FlexboxItem>2 Column</FlexboxItem>
    </Flexbox>
    <Flexbox>
      <FlexboxItem>3 Column</FlexboxItem>
      <FlexboxItem>3 Column</FlexboxItem>
      <FlexboxItem>3 Column</FlexboxItem>
    </Flexbox>
    <Flexbox>
      <div class="placeholder">Fixed Width 70px</div>
      <FlexboxItem>Responsive Size</FlexboxItem>
    </Flexbox>
    <Flexbox>
      <FlexboxItem>3</FlexboxItem>
      <FlexboxItem class="am-ft-ellipsis">Customized Settings Customized Settings Customized Settings Customized Settings</FlexboxItem>
      <FlexboxItem>3</FlexboxItem>
    </Flexbox>
  </div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
  .am-flexbox {
    margin: 10px 0;
  }

  .demo-item {
    padding: 6px 0;
    background: #4A89DC;
    border-radius: 4px;
    text-align: center;
    color: #fff;
  }

  .am-flexbox-item {
    .demo-item;
  }

  .placeholder {
    .demo-item;
    width: 70px;
    font-size: 12px;
    margin-left: 8px;
    margin-right: 8px;
  }

</style>
```

Arrangement direction

Sample image



Sample code

```
<template>
  <div>
    <Flexbox>
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
    </Flexbox>

    <Flexbox direction="row-reverse">
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
    </Flexbox>
  </div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
  .am-flexbox {
    margin: 10px 0;
  }

  .demo-item {
    padding: 6px 0;
    background: #4A89DC;
    border-radius: 4px;
    text-align: center;
    color: #fff;
  }

  .am-flexbox-item {
    .demo-item;
  }

  .placeholder {
    .demo-item;
    width: 70px;
    font-size: 12px;
    margin-left: 8px;
    margin-right: 8px;
  }
</style>
```

Element wrap

Sample image



Sample code

```
<template>
  <div>
    <Flexbox>
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
      <div class="placeholder">4</div>
      <div class="placeholder">5</div>
      <div class="placeholder">6</div>
      <div class="placeholder">7</div>
      <div class="placeholder">8</div>
      <div class="placeholder">9</div>
    </Flexbox>

    <Flexbox wrap="wrap">
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
      <div class="placeholder">4</div>
      <div class="placeholder">5</div>
      <div class="placeholder">6</div>
      <div class="placeholder">7</div>
      <div class="placeholder">8</div>
      <div class="placeholder">9</div>
    </Flexbox>
  </div>
</template>
```

```

</Flexbox>

<Flexbox wrap="wrap-reverse">
  <div class="placeholder">1</div>
  <div class="placeholder">2</div>
  <div class="placeholder">3</div>
  <div class="placeholder">4</div>
  <div class="placeholder">5</div>
  <div class="placeholder">6</div>
  <div class="placeholder">7</div>
  <div class="placeholder">8</div>
  <div class="placeholder">9</div>
</Flexbox>
</div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
.am-flexbox {
  margin: 10px 0;
}

.demo-item {
  padding: 6px 0;
  background: #4A89DC;
  border-radius: 4px;
  text-align: center;
  color: #fff;
}

.am-flexbox-item {
  .demo-item;
}

.placeholder {
  .demo-item;
  width: 70px;
  font-size: 12px;
  margin: 8px;
}
</style>

```

Alignment

Sample image



Sample code

```
<template>
  <div>
    <Flexbox>
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
    </Flexbox>

    <Flexbox justify="center">
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
    </Flexbox>
  </div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
.am-flexbox {
  margin: 10px 0;
}

.demo-item {
  padding: 6px 0;
  background: #4A89DC;
  border-radius: 4px;
  text-align: center;
  color: #fff;
}

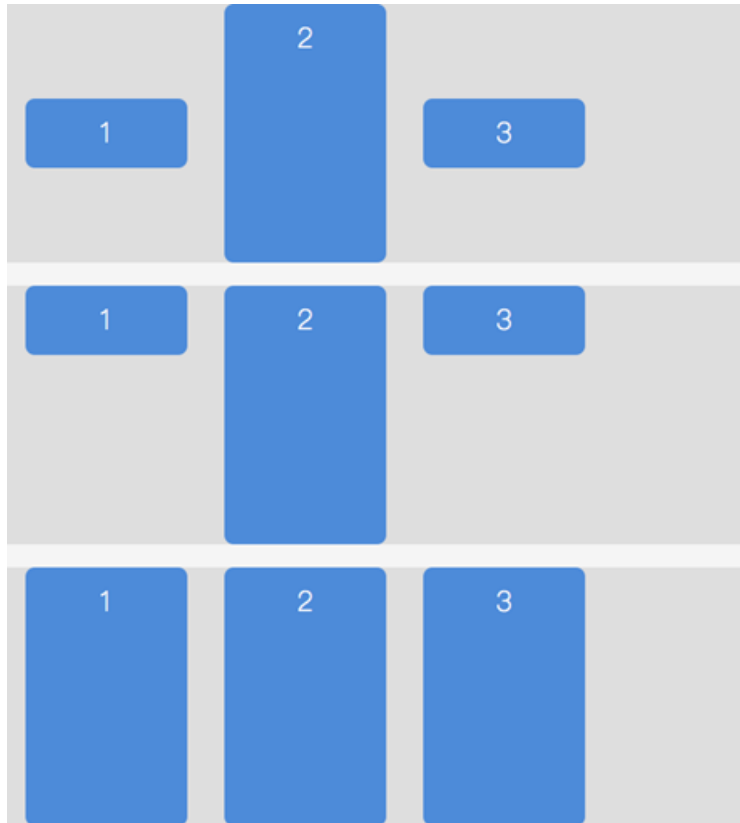
.am-flexbox-item {
  .demo-item;
}

.placeholder {
  .demo-item;
  width: 70px;
  font-size: 12px;
  margin-left: 8px;
  margin-right: 8px;
}

</style>
```

Cross-axis alignment

Sample image



Sample code

```
<template>
  <div>
    <Flexbox>
      <div class="placeholder">1</div>
      <div class="placeholder high">2</div>
      <div class="placeholder">3</div>
    </Flexbox>

    <Flexbox align="start">
      <div class="placeholder">1</div>
      <div class="placeholder high">2</div>
      <div class="placeholder">3</div>
    </Flexbox>

    <Flexbox align="stretch">
      <div class="placeholder">1</div>
      <div class="placeholder high">2</div>
      <div class="placeholder">3</div>
    </Flexbox>
  </div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
  .am-flexbox {
    margin: 10px 0;
    background-color: #dedede;
  }

  .demo-item {
    padding: 6px 0;
    background: #4A89DC;
    border-radius: 4px;
    text-align: center;
    color: #fff;
  }

  .am-flexbox-item {
    .demo-item;
  }

  .placeholder {
    .demo-item;
    width: 70px;
    font-size: 12px;
    margin-left: 8px;
    margin-right: 8px;
  }

  .high {
    height: 100px;
  }

</style>
```

Multi-axis alignment

Sample image



Sample code

```
<template>
  <div>
    <Flexbox wrap="wrap">
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
      <div class="placeholder">4</div>
      <div class="placeholder">5</div>
      <div class="placeholder">6</div>
      <div class="placeholder">7</div>
      <div class="placeholder">8</div>
      <div class="placeholder">9</div>
    </Flexbox>

    <Flexbox wrap="wrap" align-content="start">
      <div class="placeholder">1</div>
      <div class="placeholder">2</div>
      <div class="placeholder">3</div>
      <div class="placeholder">4</div>
      <div class="placeholder">5</div>
    </Flexbox>
  </div>
</template>
```

```

    <div class="placeholder">6</div>
    <div class="placeholder">7</div>
    <div class="placeholder">8</div>
    <div class="placeholder">9</div>
  </Flexbox>
</div>
</template>

<style lang="less" rel="stylesheet/less" scoped>
  .am-flexbox {
    margin: 10px 0;
    height: 200px;
    background-color: #dedede;
  }

  .demo-item {
    padding: 6px 0;
    background: #4A89DC;
    border-radius: 4px;
    text-align: center;
    color: #fff;
  }

  .am-flexbox-item {
    .demo-item;
  }

  .placeholder {
    .demo-item;
    width: 70px;
    font-size: 12px;
    margin: 8px;
  }
</style>

```

1.2.4. Tab components

1.2.4.1. Tab

This topic describes different methods of using Tab component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ Tab, TabItem }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Tab, TabItem } from '@alipay/antui-vue';
```

API description

- [Tab](#)
- [TabItem](#)

Tab

props

Property	Type	Default value	Description
scroll	Boolean	false	Whether to support scrolling.
value	Number, String	null	id of the selected <code>tab-item</code> . v-model is supported.
initial-value	Number, String	null	id of the initially selected <code>tab-item</code> . Non-v-model scenarios are supported.
resistant	Number	0.2	Speed ratio (pixels per second) when sliding is out of the screen.
reverseTime	Number	0.15	Bounce time when sliding is out of the screen, in seconds.
slideTime	Number	0.3	Inertia time after sliding, in seconds.
slideRate	Number	0.1	Inertia distance ratio after sliding (distance = slideRate × speed).

slots

Name	Description	Scope
-	Used for populating TabItem.	-

events

Name	Description	Function
input	Triggered when the selected item is changed.	Function (value: [string, number]): void

TabItem

props

Property	Description	Type
id	ID used to identify an item.	String, Number

slots

Name	Description	Scope
-	Placeholder for populated content.	-

Demo

Tab

Sample image

Option 1 Option 2 Option 3

Sample code

```
<template>
  <Tab v-model="selected">
    <TabItem v-for="i in 3" v-bind:key="i" :id="i">Option {{ i }}</TabItem>
  </Tab>
</template>

<script type="text/javascript">
  export default {
    data() {
      return {
        selected: 1,
      };
    },
  };
</script>
```

1.2.4.2. TabPanel

This topic describes different methods of using the TabPanel component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ TabPanel, TabPanelItem }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { TabPanel, TabPanelItem } from '@alipay/antui-vue';
```

API description

- [TabPanel](#)
- [TabPanelItem](#)

TabPanel

props

Property	Description	Type	Default value
scroll	Whether to support scrolling.	Boolean	true
labels	Labels displayed on the top.	Array	[]
value	Index of the selected tab item, used to support <code>v-model</code> binding.	Number	0
initial-value	Index of the initially selected tab item, used to support higher-performance scenarios that do not require <code>v-model</code> binding.	Number	0
resistant	Speed ratio (pixels per second) when sliding is out of the screen.	Number	0.2

Property	Description	Type	Default value
reverseTime	Bounce time when sliding is out of the screen, in seconds.	Number	0.15
reverseTimingFunction	Bounce easing time when sliding is out of the screen.	String	-
slideTime	Inertia time after sliding, in seconds.	Number	0.3
slideTimingFunction	Easing function after sliding.	String	"cubic-bezier(.86,0,.07,1)"
slideRate	Inertia distance ratio after sliding (distance = slideRate × speed).	Number	0.1

slots

Name	Description	Scope
-	Used for populating TabPanelItem.	-

events

Name	Description	Function
input	Triggered when an item is selected. <code>index</code> indicates the index of the selected item. (It is triggered when the current item is selected again. This issue has been resolved since version <code>0.4.8-open02</code>.)	Function (index: number): void

TabPanelItem

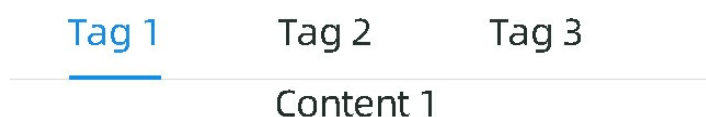
slots

Name	Description	Scope
-	Placeholder for populated content.	-

Demo

Tab-panel

Sample image



Sample code

```
<template>
  <TabPanel
    :labels="['Tag1', 'Tag2', 'Tag3', 'Tag4', 'Tag5', 'Tag6', 'Tag7']"
    v-model="selected"
  >
    <TabPanelItem
      v-for="i in 7"
      style="text-align: center; height: 100px; background: white"
    >
      Content{{ i }}
    </TabPanelItem>
  </TabPanel>
</template>

<script>
  export default {
    data() {
      return {
        selected: 0,
      };
    },
  };
</script>
```

1.2.5. Pop-up component

1.2.5.1. ActionSheet

This topic describes different methods of using the ActionSheet component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).

- Use [Service imperative call](#). [Service documentation](#) provides details about static methods and show options.
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ ActionSheet }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { ActionSheet } from '@alipay/antui-vue';
```

Service imperative call

You can directly use `ActionSheet.show('Display content');` to call a command, instead of writing the command in a template. The `DOM` will be automatically inserted into `document.body`.

```
ActionSheet.show({  
  title: 'Title',  
  items: ['123', '321'],  
  cancelButtonText: 'Cancel'  
}, function (index) {  
  
});
```

Service documentation

static methods

`ActionSheet` provides the following static method.

Name	Description	Function
show	Creates and displays an <code>ActionSheet</code> instance with the parameters in the table below.	Function(option: Object string, callback: Function): vm

show options

The following parameters are accepted for creating an instance.

Property	Description	Type	Default value
title	Description text displayed above the panel.	String	-
items	Text array of options.	Array	-
cancelButtonText	Text of the Cancel button.	String	'Cancel'
destructiveBtnIndex	Index value of a special operation.	Number	-
transitionDuration	Gradient duration.	String	'0.3s'
zIndex	Specifies the <code>zIndex</code> value of the elastic layer.	Number	9999

API description

props

Property	Description	Type	Default value
show	Whether to display the panel.	Boolean	false
title	Description text displayed above the panel.	String	-
items	Text array of options.	Array	-
cancelButtonText	Text of the Cancel button.	String	'Cancel'
destructiveBtnIndex	Index value of a special operation.	Number	-
transitionDuration	Gradient duration.	String	'0.3s'

events

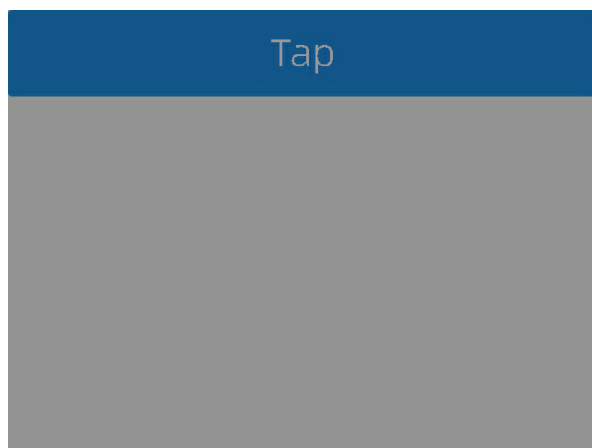
Name	Description	Function
tap	Triggered when an option is tapped. The value is -1 the mask or Cancel button is tapped.	Function(index : number): void

slots

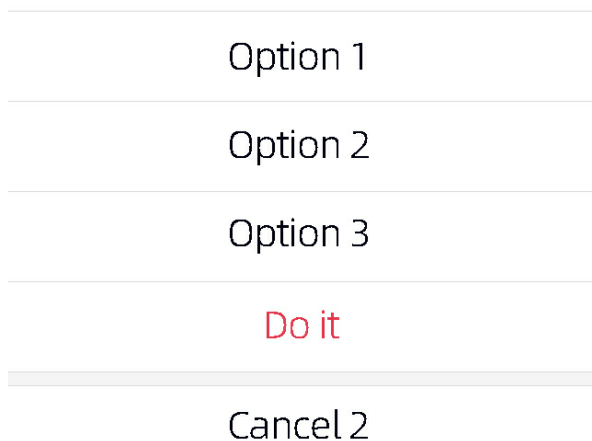
Name	Description	Scope
label	Used for <code>DOM</code> extension of <code>props.items</code>	<code>{item: String}</code>

Demo

Sample image



Here displays at most 2 rows of annotation. Provide clear information to make users understood.



Sample code

```
<template>
  <div>
    <AButton @click="show = true">Tap</AButton>
    <ActionSheet
      :show="show"
      title="Here displays at most 2 rows of annotation. Provide clear information to make
users understood."
      :items="['Option 1', 'Option 2', 'Option 3', 'Do it']"
      cancelButtonText="Cancel 2"
      :destructiveBtnIndex="3"
      transitionDuration="0.3s"
      @click="onClick"
    />
  </div>
</template>

<script>
export default {
  data() {
    return {
      show: false,
    };
  },
  methods: {
    onClick(index) {
      if (index === -1) {
        this.show = false;
      }
    },
  },
};
</script>
```

1.2.5.2. ADialog

This topic describes different methods of using the ADialog component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ ADialog }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { ADialog } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
animation	Whether to enable transition .	Boolean	true
type	The type of a dialog box. Valid values: text and image .	String	'text'
value	Whether to show or hide a dialog box. v-model is supported`.	Boolean	false
closable	Whether to display the x button.	Boolean	false
title	The title of a dialog box.	String	-
content	The content of a dialog box.	String	-
buttons	Pop-up button group that requires the array element to be { [key]:string, [text]:string, [disabled]: boolean } .	Array<{ key:string, text:string, disabled:boolean }>	[]
selection	Tab dialog box used for setting the vertical distribution of buttons.	Boolean	-

Property	Description	Type	Default value
preventMove	Whether to block the touchmove event of document when the pop-up/mask is displayed.	Boolean	true

slots

Name	Description	Scope
image	The area that is used to fill in images.	-
-	Default placeholder as the pop-up content. Custom <code>DOM</code> is supported if required.	-
button	Area for the button group.	<code>buttons</code>

events

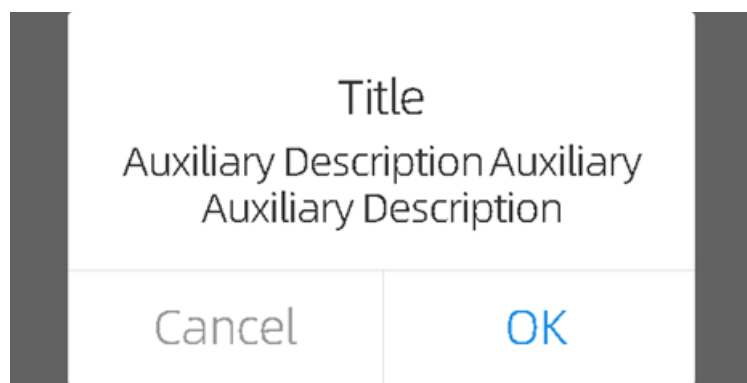
Name	Description	Function
buttonClick	Triggered when a button is tapped.	Function(event: event, buttonKey: string): void
maskClick	Triggered when a mask is tapped.	Function(event: event): void
input	Triggered when <code>value</code> is changed.	Function(value: boolean): void
show	Triggered when <code>value</code> is changed to <code>true</code> .	Function(): void
hide	Triggered when <code>value</code> is changed to <code>false</code> .	Function(): void

Demo

- [Title](#)
- [Untitled text](#)
- [Title + image + content](#)
- [Single-action button: title + image + content](#)
- [Single-action button: title + content](#)
- [Unclickable button: title + content](#)
- [Tab](#)
- [Sending success](#)

Title

Sample image



Sample code

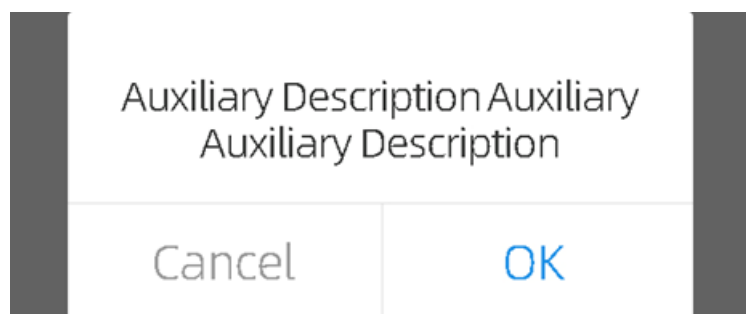
```
<template>
  <ADialog
    title="Title"
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  ></ADialog>
</template>

<script type="text/javascript">
  import Toast from '../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'cancel',
          text: 'Cancel',
        }, {
          key: 'ok',
          text: 'OK',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Untitled text

Sample image



Sample code

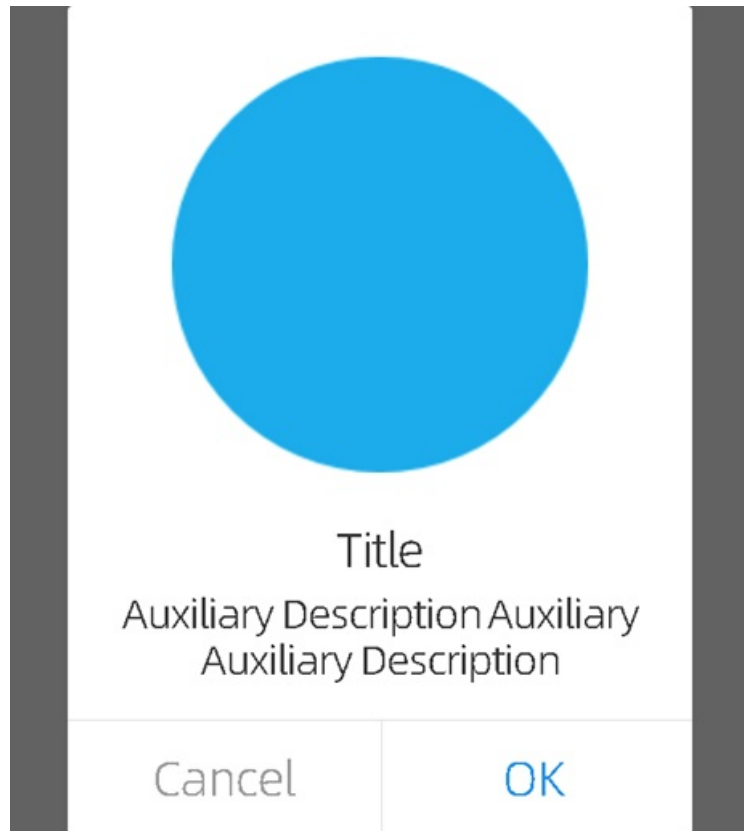
```
<template>
  <ADialog
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  ></ADialog>
</template>

<script type="text/javascript">
  import Toast from '../.../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'cancel',
          text: 'Cancel',
        }, {
          key: 'ok',
          text: 'OK',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Title + image + content

Sample image



Sample code

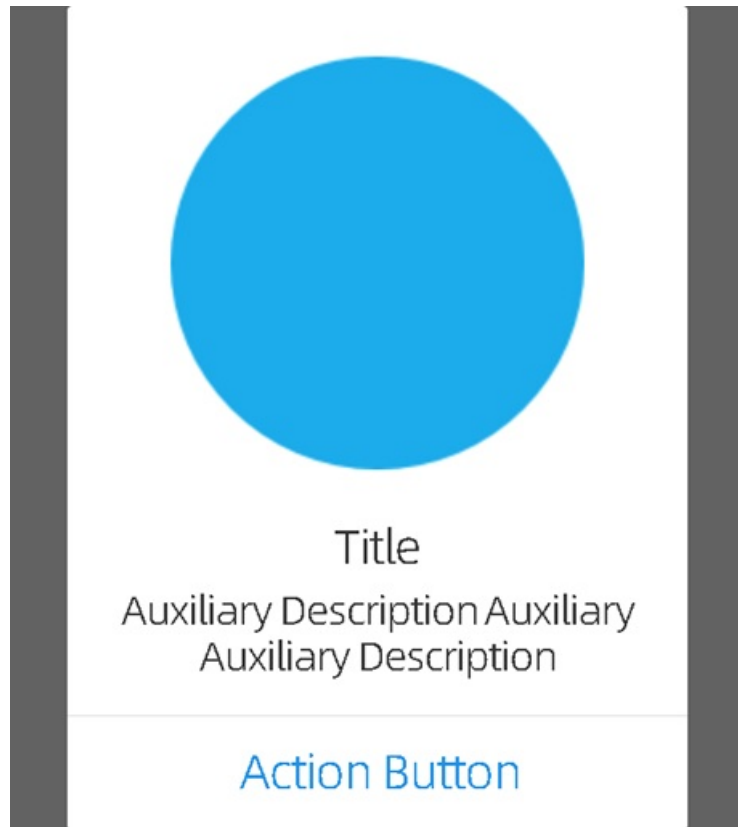
```
<template>
  <ADialog
    title="Title"
    type="image"
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  >
    
  </ADialog>
</template>

<script type="text/javascript">
  import Toast from '../../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'cancel',
          text: 'Cancel',
        }, {
          key: 'ok',
          text: 'OK',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Single-action button: title + image + content

Sample image



Sample code

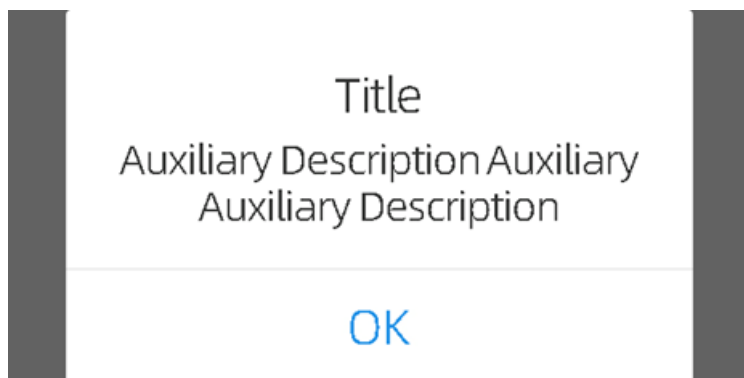
```
<template>
  <ADialog
    title="Title"
    type="image"
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  >
    
  </ADialog>
</template>

<script type="text/javascript">
  import Toast from '../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'action',
          text: 'Action Button',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Single-action button: title + content

Sample image



Sample code

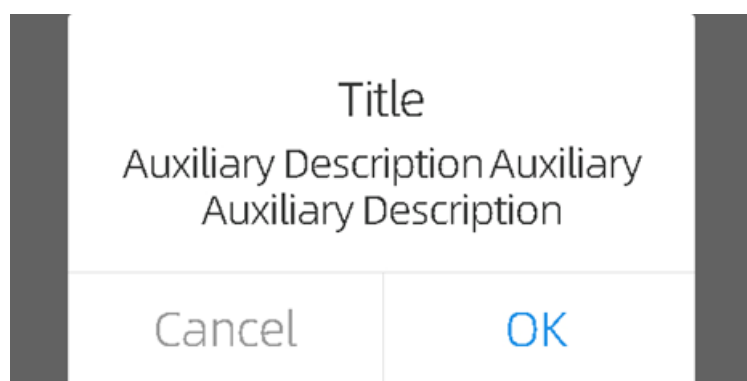
```
<template>
  <ADialog
    title="Title"
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  ></ADialog>
</template>

<script type="text/javascript">
  import Toast from '../.../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'ok',
          text: 'OK',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Unclickable button: title + content

Sample image



Sample code

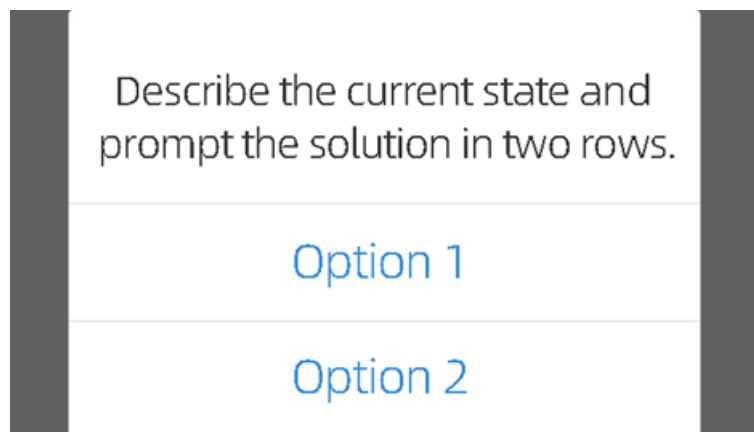
```
<template>
  <ADialog
    title="Title"
    :value="true"
    content="Auxiliary Description Auxiliary Auxiliary Description"
    :buttons="buttons"
    @buttonClick="buttonClick"
  ></ADialog>
</template>

<script type="text/javascript">
  import Toast from '../.../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'cancel',
          text: 'Cancel',
        }, {
          key: 'ok',
          text: 'OK',
          disabled: true,
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Tab

Sample image



Sample code

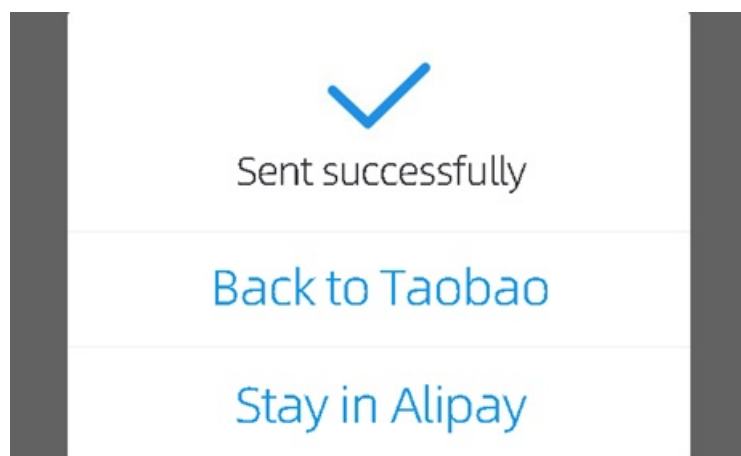
```
<template>
  <ADialog
    :value="true"
    content="Describe the current state and prompt the solution in two rows."
    :selection="true"
    :buttons="buttons"
    @buttonClick="buttonClick"
  ></ADialog>
</template>

<script type="text/javascript">
  import Toast from '../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'action1',
          text: 'Option1',
        }, {
          key: 'action2',
          text: 'Option1',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

Sending success

Sample image



Sample code

```
<template>
  <ADialog
    :value="true"
    :selection="true"
    :buttons="buttons"
    @buttonClick="buttonClick"
  >
    <div class="am-dialog-send-img">
      
    </div>
    <p class="am-dialog-brief">Sent successfully</p>
  </ADialog>
</template>

<script type="text/javascript">
  import Toast from '../lib/toast';

  export default {
    data() {
      return {
        buttons: [{
          key: 'taobao',
          text: 'Back to Taobao',
        }, {
          key: 'alipay',
          text: 'Stay in Alipay',
        }],
      };
    },
    methods: {
      buttonClick(evt, key) {
        Toast.show(`Clicked${key}`);
      },
    },
  };
</script>
```

1.2.5.3. Filter

This topic describes different methods of using the Filter component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ AFilter, FilterList, FilterItem }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { AFilter, FilterList, FilterItem } from '@alipay/antui-vue';
```

API description

AFilter

The `type` is `dialog` (default).

props

Property	Description	Type	Default value
value	Whether to display the content.	Boolean	false
animation	Whether startup animation is enabled.	Boolean	true

slots

Name	Description	Scope
-	Default slot used to populate the content of the layout.	-

events

Name	Description	Function
input	Triggered when you tap the shadow part. The value is <code>false</code> .	Function(value : boolean): void

The `type` is `page`.

props

Property	Description	Type	Default value
value	Whether to display the content.	Boolean	false

Property	Description	Type	Default value
animation	Whether startup animation is enabled.	Boolean	true

slots

Name	Description	Scope
-	Default slot used to populate the content of the layout.	-
footer	Used to change a bottom button.	-

events

Name	Description	Function
reset	Triggered when you tap the Reset button.	Function(): void
confirm	Triggered when you tap the Confirm button.	Function(): void

FilterList

props

Property	Description	Type	Default value
recommend	Recommended option.	Boolean	false
title	Option group title.	String	""

slots

Name	Description	Scope
-	Default slot used to populate the content of the element.	-

FilterItem

Property	Description	Type	Default value
selected	Whether an option is selected. When the value of this property is null, the values of option and value are used.	Boolean, null	null
option	Identifier of this option.	String, Number	-
value	Identifier of the currently selected option. If the value is the same as that of option , the option is selected. If selected is not <code>null</code> , the value of selected is used preferentially.	String, Number	-
disabled	Whether the option is disabled.	Boolean	false

slots

Name	Description	Scope
-	Default slot used to populate the content of the element.	-

events

Name	Description	Function
input	Triggered when the option is selected, and not triggered when the option is disabled.	Function(option: [string, number]): void

Demo

- [Filter box](#)

- [Multi-classification filter](#)
- [Full-screen filter](#)

Filter box

Sample image



Sample code

```
<template>
  <div style="min-height: 200px;">
    <AFilter type="dialog" v-model="show">
      <FilterList>
        <FilterItem v-for="i in 8" :option="i" v-model="selected">
          {{ i }}
        </FilterItem>
      </FilterList>
    </AFilter>
    <div class="button-wrap">
      <button class="am-button white" @click="show = true">Tap to show the filter box</button>
    </div>
  </div>
</template>

<style lang="less" scoped>
  .button-wrap {
    padding: 15px;
  }
</style>

<script>
  export default {
    data() {
      return {
        show: true,
        selected: null,
      };
    },
  };
</script>
```

Multi-classification filter

Sample image

1

2

3

Category Name

1

2

3

4

5

Category Name

1

2

3

4

5

6

Sample code

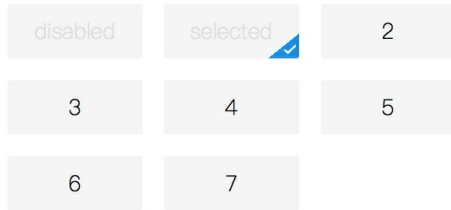
```
<template>
  <div style="min-height: 400px;">
    <AFilter v-model="show">
      <FilterList recommend>
        <FilterItem v-for="i in 3" :option="'recommend-' + i" v-model="selected">
          {{ i }}
        </FilterItem>
      </FilterList>
      <FilterList title="Category Name">
        <FilterItem v-for="i in 5" :option="'a-' + i" v-model="selected">
          {{ i }}
        </FilterItem>
      </FilterList>
      <FilterList title="Category Name">
        <FilterItem v-for="i in 6" :option="'b-' + i" v-model="selected">
          {{ i }}
        </FilterItem>
      </FilterList>
    </AFilter>
    <div class="button-wrap">
      <button class="am-button white" @click="show = true">Tap to show the filter box</button>
    </div>
  </div>
</template>

<style lang="less" scoped>
  .button-wrap {
    padding: 15px;
  }
</style>

<script>
  export default {
    data() {
      return {
        show: false,
        selected: null,
      };
    },
  };
</script>
```

Full-screen filter

Sample image



Sample code

```

<template>
  <div>
    <div class="button-wrap">
      <button class="am-button white" @click="show = true">Tap to show the filter box</button>
    </div>
    <AFilter type="page" v-model="show">
      <FilterList>
        <FilterItem option="disabled" :selected="!!~selected.indexOf('disabled')" @input="onSelect" disabled>
          disabled
        </FilterItem>
        <FilterItem option="selected" :selected="!!~selected.indexOf('selected')" @input="onSelect" disabled selected>
          selected
        </FilterItem>
        <FilterItem v-for="i in 6" :option="i" :selected="!!~selected.indexOf(i)" @input="onSelect">
          {{ i + 1 }}
        </FilterItem>
      </FilterList>
    </AFilter>
  </div>
</template>

<style lang="less" scoped>
  .button-wrap {
    padding: 15px;
  }
</style>

<script>
  export default {
    data() {
      return {
        selected: [],
        show: true
      };
    },
    methods: {
      onSelect(v) {
        if (~this.selected.indexOf(v)) {
          this.selected = this.selected.filter(i => i !== v);
        } else {
          this.selected.push(v);
        }
      },
    },
  };
</script>

```

1.2.5.4. Toast

This topic describes different methods of using the Toast component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- Use [Service imperative call](#). [Service documentation](#) provides details about static methods and show options.
- [API description](#) provides API information for props, slots, events, and methods.

To view the visual effects and code sample of the component, see [Demo](#).

Kylin

```
<dependency component="{ Toast }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Toast } from '@alipay/antui-vue';
```

Service imperative call

You can directly use `Toast.show ('Display content ');` to call a command, instead of writing the command in a template. The `DOM` will be automatically inserted into `document.body`.

```
Toast.show({
  type: 'text',
  text: 'Displayed message'
});
```

Service documentation

static methods

`Toast` provides the following static methods.

Name	Description	Function
show	Creates and displays a <code>Toast</code> instance with the following parameters. You can also directly input a character string as the <code>text</code> .	Function(option: Object string): vm
closeAll	Closes all currently non-closed <code>Toast</code> instances.	Function(void): void

show options

The following parameters are accepted for creating an instance.

Property	Description	Type	Default value
type	Toast type. Value options include <code>text</code> , <code>loading</code> , <code>warn</code> , <code>success</code> , <code>network</code> , and <code>fail</code> .	String	'text'
text	Pure character string text.	String	-
duration	Automatically hides and destructs an instance upon timeout.	Number	3000
zIndex	Specifies the <code>zIndex</code> value of the elastic layer.	Number	9999

instance methods

The `Toast.show` method returns a `vm` instance and exposes the following method.

Name	Description	Function
hide	Proactively hides and destructs the instance within the <code>duration</code> .	Function(void) : void

API description

props

Property	Description	Type	Default value
type	Toast type. Value options include <code>text</code> , <code>loading</code> , <code>warn</code> , <code>success</code> , <code>network</code> , and <code>fail</code> .	String	'text'
value	Displayed or hidden state of the Toast. <code>v-model</code> is supported for triggering scheduled hiding.	Boolean	false

Property	Description	Type	Default value
duration	Upon timeout, <code>\$emit('input', false)</code> is automatically triggered. If the value 0, <code>\$emit('input', false)</code> will not be triggered.	Number	3000
text	Pure character string text. Use <code>slot</code> if <code>DOM</code> is required`.	String	-
onClose	Triggered when the Toast is hidden.	Function	-
multiline	When the value is true, the default pattern supports multi-line text.	Boolean	false

slots

Name	Description	Scope
-	Default placeholder and content, with custom <code>DOM</code> supported`.	-

events

Name	Description	Function
input	Adaptive to <code>v-model</code> .	Function(newV alue: boolean): void

methods

Name	Description	Function
hide	Actively hides the current Toast.	Function(void) : void

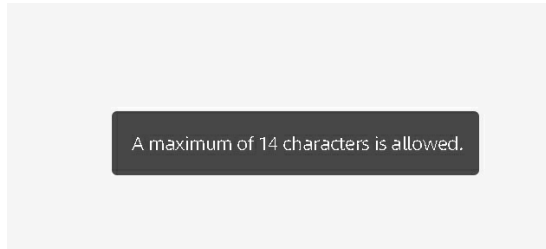
Demo

- [Text](#)

- [Loading](#)
- [Warning](#)
- [Success](#)

Text

Sample image



Sample code

```
<template>

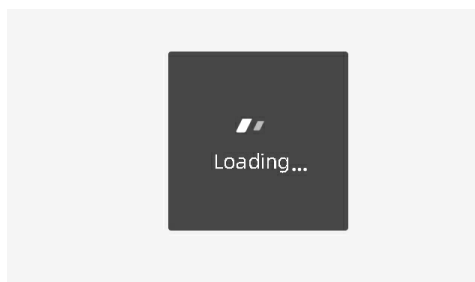
  <div style="padding: 10px;">
    <AButton @click="trigger">{{show?'Close':'Show'}}Toast</AButton>
    <Toast v-model="show">A maximum of 14 characters is allowed.</Toast>
  </div>

</template>

<script>
  import { Toast, AButton } from "@alipay/antui-vue";
  export default {
    data() {
      return { show: true };
    },
    methods: {
      trigger() {
        this.show = !this.show;
      },
    },
    components: {
      Toast,
      AButton,
    },
  };
</script>
```

Loading

Sample image



Sample code

```
<template>

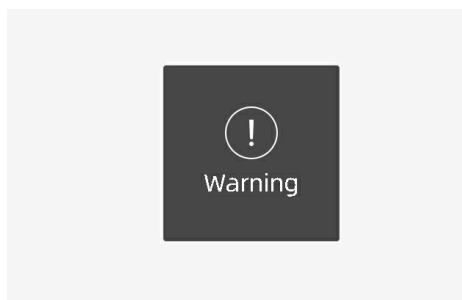
  <div style="padding: 10px;">
    <AButton @click="trigger">{{show?'Close':'Show'}}Toast</AButton>
    <Toast type="loading" v-model="show">Loading...</Toast>
  </div>

</template>

<script>
import { Toast, AButton } from "@alipay/antui-vue";
export default {
  data() {
    return { show: true };
  },
  methods: {
    trigger() {
      this.show = !this.show;
    },
  },
  components: {
    Toast,
    AButton,
  },
};
</script>
```

Warning

Sample image



Sample code

```
<template>

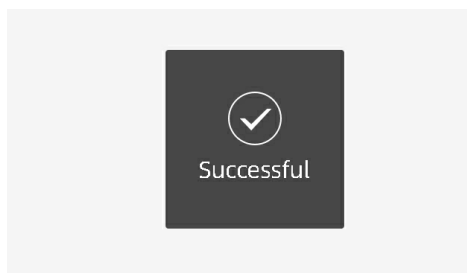
  <div style="padding: 10px;">
    <AButton @click="trigger">{{show?'Close':'Show'}}Toast</AButton>
    <Toast type="warn" v-model="show">Warning</Toast>
  </div>

</template>

<script>
import { Toast, AButton } from "@alipay/antui-vue";
export default {
  data() {
    return { show: true };
  },
  methods: {
    trigger() {
      this.show = !this.show;
    },
  },
  components: {
    Toast,
    AButton,
  },
};
</script>
```

Success

Sample image



Sample code

```
<template>

  <div style="padding: 10px;">
    <AButton @click="trigger">{{show?'Close':'Show'}}Toast</AButton>
    <Toast type="success" v-model="show">Successful</Toast>
  </div>

</template>

<script>
import { Toast, AButton } from "@alipay/antui-vue";
export default {
  data() {
    return { show: true };
  },
  methods: {
    trigger() {
      this.show = !this.show;
    },
  },
  components: {
    Toast,
    AButton,
  },
};
</script>
```

1.2.6. Entry component

1.2.6.1. List

This topic describes different methods of using the List component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

To view the visual effects and code sample of the component, see [Demo](#).

List items are in a complex structure, comprising single-line or multi-line text lists, forms, verification codes, and form elements such as checkbox, radio, and switch. Functions are divided by using `List`, `ListCell`, and `ListItem`.

- `List`: A list packet, equivalent to `am-list`. For details about the supported types, see [API description](#).
- `ListItem`: A one-line list item, which can be used in ordinary list layout. For details about the supported types, see [API description](#).
- `Listcell`: Various blocks in a list item, which can be used in a complex layout through flexible combination. For details about the supported types, see [API description](#).

Kylin

```
<dependency component="{ List, ListItem, ListCell }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { List, ListItem, ListCell } from '@alipay/antui-vue';
```

API description

List

props

Property	Description	Type	Default value
type	The type of the list. Valid values: "form" , "twoline" , "twoline-text" , "twoline-side" , "moreline" , "ptext" , "users" , "users-sm" , "users-lg" , "bank" , "info" , and "delete" .	String	-

slots

Name	Description	Scope
header	Used to fill the list header. <code>ListCell</code> is recommended,	-
footer	Used to fill the list footer. <code>ListCell</code> is recommended.	-

ListCell

props

Property	Description	Type	Default value
----------	-------------	------	---------------

Property	Description	Type	Default value
type	The type of a list element. Valid values: "content", "brief", "extra", "arrow", "thumb", "title", "brief", "sti", "header", "footer", and "header-sp".	String	"content"
noFlex	Whether to remove the flex for the current cell.	Boolean	false
noEllips	Whether the single-line style is not required for the content in a cell.	Boolean	false
alignTop	Whether the content in a cell is aligned to the top.	Boolean	false
twoColumn	Display the content in a cell in two columns.	Boolean	false
title	Inject the <code>ListCell [type = title]</code> node to enter the text.	String	-
brief	Inject the <code>ListCell [type=brief]</code> node to enter the text.	String	-
titleEllips	Whether to enable <code>am-ft-ellipsis</code> for the <code>title</code> injection node.	Boolean	false
briefEllips	Whether to enable <code>am-ft-ellipsis</code> for the <code>brief</code> injection node.	Boolean	false

slots

Name	Description	Scope
-	Used for filling in the child node.	-

events

Name	Description	Function
click	Triggered upon a tap operation.	Function(event: event): void

ListItem

props

Property	Description	Type	Default value
type	The type of list lines. Valid values: <code>""</code> , <code>"twoline"</code> , <code>"moreline"</code> , <code>"check"</code> , <code>"radio"</code> , <code>"more"</code> , <code>"part"</code> , <code>"delete"</code>	String	""
arrow	Whether to display the right arrow. It is enabled by default if the <code>href</code> attribute is configured (<code>href</code> is unavailable when <code>arrow=true</code>).	Boolean	false
href	Whether to enable page redirection upon tapping. If it is set, the <code>A</code> label is added to <code>DOM</code> , otherwise the <code>DIV</code> label is added.	String	-
content	This short term can be used by default when only <code>ListCell[type=content]</code> is filled in a placeholder.	String	-
extra	Placeholder text on the right side	String	-
reddot	Display red dots in the extra part on the right side.	Boolean	-
deletable	Whether to display the Delete button on the left side of a list.	Boolean	-
contentNoFlex	Automatically add the <code>ListCell[noFlex]</code> attribute for <code>content</code> .	Boolean	false

Property	Description	Type	Default value
extraNoFlex	Automatically add the <code>ListCell[noFlex]</code> attribute for <code>extra</code> .	Boolean	false
contentNoEllipsis	Automatically add the <code>ListCell[noEllipsis]</code> attribute for <code>content</code> .	Boolean	false
extraNoEllipsis	Automatically add the <code>ListCell[noEllipsis]</code> attribute for <code>extra</code> .	Boolean	false
cancelText	Display the <code>Cancel</code> text when <code>[type=delete]</code> .	String	"Cancel"
deleteText	Display the <code>Delete</code> text when <code>[type=delete]</code> .	String	"Delete"
indentLine	The indentation of an underscore. Valid values: <code>""</code> , <code>"thumb"</code> , <code>"twoline"</code> .	String	""

slots

Name	Description	Scope
thumb	The placeholder for the thumbnail on the left side.	-
-	Default placeholder. To quickly fill the <code>ListCell[type=content]</code> node, you can directly use the <code>content</code> attribute.	-
extra	If string text is not enough for the placeholder on the right side, you can use the <code>DOM</code> node instead.	-

events

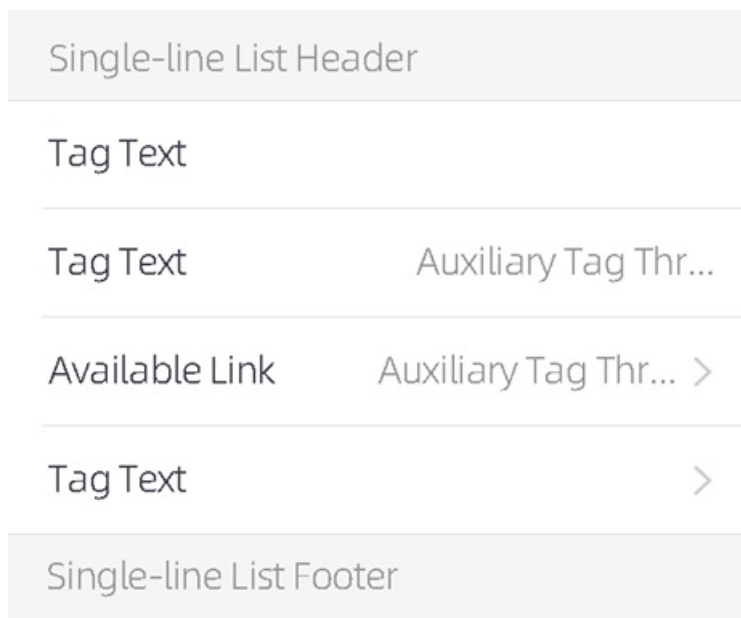
Name	Description	Function
click	Triggered upon a tap operation.	Function(event: event): void
cancel	Tap it to cancel when [type=delete].	Function(): void
delete	Tap it to delete when [type=delete].	Function(): void

Demo

- [List header and footer](#)
- [Single-line text list](#)
- [Extra-long cell](#)
- [Two-line plain text](#)
- [Image-text information](#)
- [Information list](#)
- [Left-right lines](#)

List header and footer

Sample image

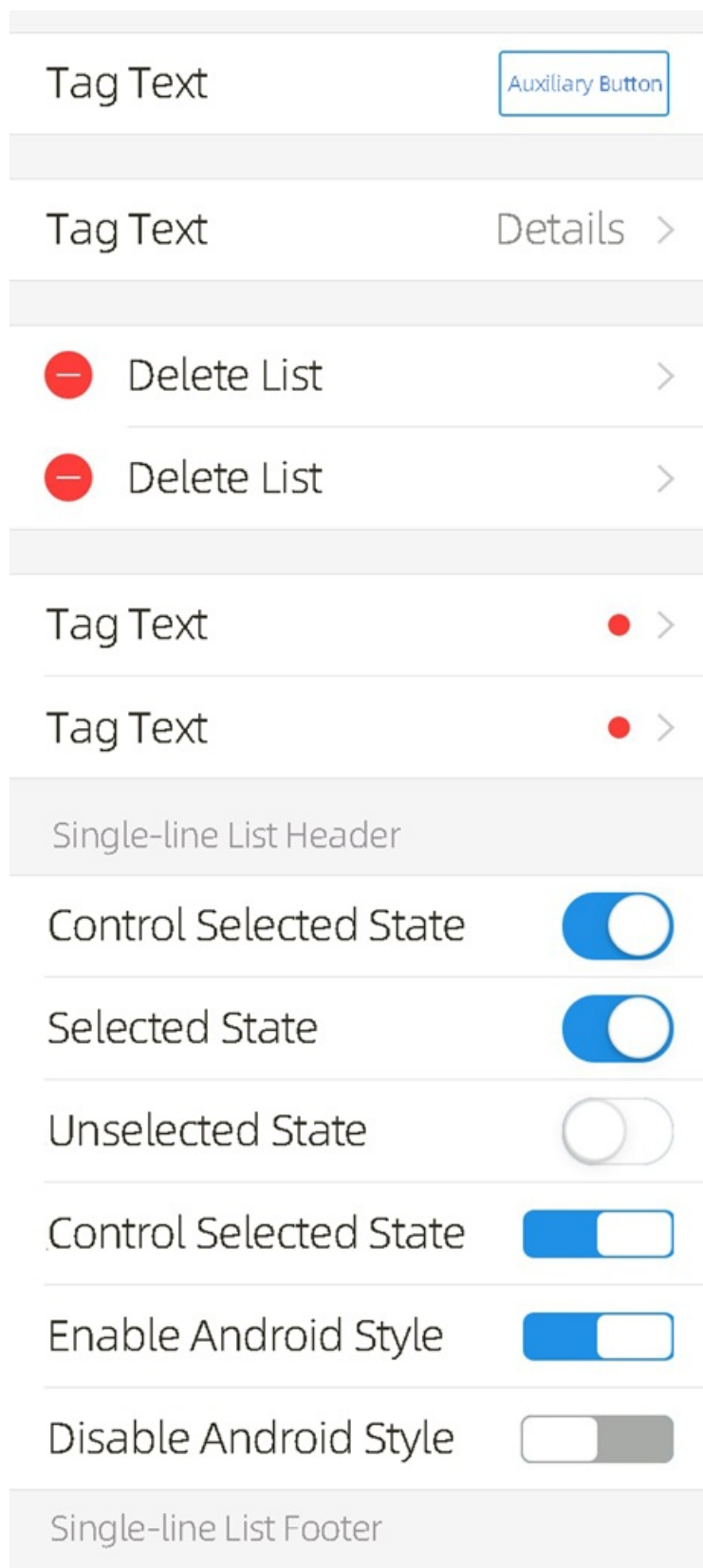


Sample code

```
<template>
  <List>
    <ListCell type="header" slot="header">Single-line list header</ListCell>
    <ListItem content="Tag"></ListItem>
    <ListItem content="Tag" extra="Auxiliary Tag three four five six seven eight nine ten">
  </ListItem>
    <ListItem content="Available link" extra="Auxiliary Tag three four five six seven eight
nine ten" href="#" arrow></ListItem>
    <ListItem content="Tag" arrow></ListItem>
    <ListCell type="footer" slot="footer">Single-line list footer</ListCell>
  </List>
</template>
```

Single-line text list

Sample image



Sample code

```
<template>
```

```
</div>
```

```

</div>

<List>
  <ListItem content="Tag">
    <AButton slot="extra" size="tiny" type="white">Auxiliary button</AButton>
  </ListItem>
</List>

<List>
  <ListItem content="Tag" extra="Details" arrow></ListItem>
</List>

<List>
  <ListItem deletable content="Delete list" arrow></ListItem>
  <ListItem deletable content="Delete list" arrow></ListItem>
</List>

<List>
  <ListItem content="Tag" redden arrow></ListItem>
  <ListItem content="Tag" redden arrow></ListItem>
</List>

<List>
  <ListCell type="header" slot="header">Single-line list header</ListCell>
  <ListItem content="Control selected state">
    <ASwitch slot="extra" v-model="ctrl"></ASwitch>
  </ListItem>
  <ListItem content="Selected state">
    <ASwitch slot="extra" :value="true"></ASwitch>
  </ListItem>
  <ListItem content="Unselected state">
    <ASwitch slot="extra" :value="false"></ASwitch>
  </ListItem>
  <ListItem content="Control selected state">
    <ASwitch platform="android" slot="extra" v-model="ctrl"></ASwitch>
  </ListItem>
  <ListItem content="Enable Android Style
">
    <ASwitch platform="android" slot="extra" :value="true"></ASwitch>
  </ListItem>
  <ListItem content="Disable Android Style">
    <ASwitch platform="android" slot="extra" :value="false"></ASwitch>
  </ListItem>
  <ListCell type="footer" slot="footer">Single-line list footer</ListCell>
</List>

</div>

</template>

<script>
export default {
  data() {
    return {
      ctrl: true
    };
  }
};

```

```

    }
  };
</script>

```

Extra-long cell

Sample image

In case of overly long text

Tag Text Auxiliary Tag >

Tag Text Single-line tag s. >

Tag should be ... Auxiliary Tag >

Tag Text Single-line tag s >

Tag should be ... Auxiliary Tag >

Tag Text Single-line tag s. >

Tag Text Auxiliary Tag Multi-line
Display Auxiliary Tag
Multi-line DisplayAuxiliary >
Tag Multi-line Display

Tag Text Auxiliary Tag Multi-line
Display Auxiliary Tag
Multi-line DisplayAuxiliary >
Tag Multi-line Display

Sample code

```
<template>

  <div>

    <List>
      <ListCell type="header" slot="header">In case of overly long text</ListCell>
      <ListItem content="Tag" extra="Auxiliary Tag" arrow></ListItem>
      <ListItem content="Tag" extra="Auxiliary tag single-line ellipsis display" arrow></ListItem>
    </List>

    <List>
      <ListItem content="The content of tag texts is displayed on the right" extra="Auxiliary Tag" extra-no-flex arrow></ListItem>
      <ListItem content="Tag" content-no-flex extra="The content of auxiliary tags is displayed on the left" arrow></ListItem>

      <ListItem content="The content of tag texts is displayed on the right" arrow>
        <ListCell slot="extra" type="extra" no-flex>Auxiliary Tag</ListCell>
      </ListItem>
      <ListItem extra="The content of auxiliary tags is displayed on the left" arrow>
        <ListCell type="content" no-flex>Tag</ListCell>
      </ListItem>
    </List>

    <List>
      <ListItem
        content="Tag" content-no-flex
        extra="Auxiliary Tag Multi-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line Display" extra-no-ellipsis
        arrow
      ></ListItem>

      <ListItem arrow>
        <ListCell type="content" no-flex>Tag</ListCell>
        <ListCell slot="extra" type="extra" no-ellipsis>Auxiliary Tag Multi-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line DisplayMulti-line Display</ListCell>
      </ListItem>

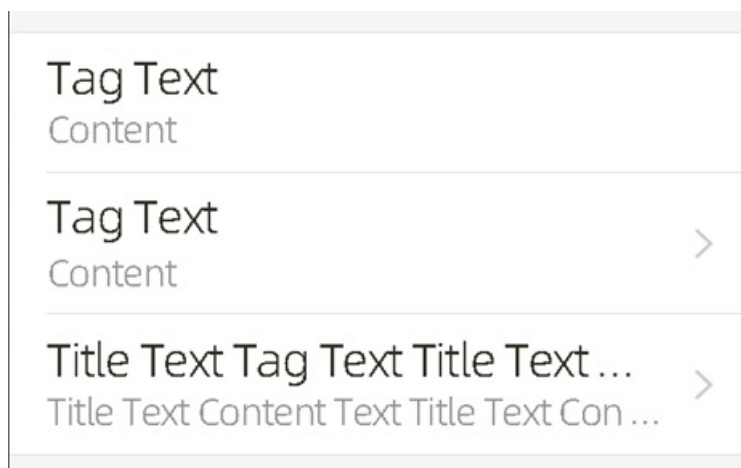
    </List>

  </div>

</template>
```

Two-line plain text

Sample image



Sample code

```
<template>

  <List type="twoline-text">
    <ListItem >
      <ListCell type="content" title="Tag" brief="Content" ></ListCell>
    </ListItem>
    <ListItem arrow>
      <ListCell type="content" title="Tag" brief="Content" ></ListCell>
    </ListItem>
    <ListItem arrow>
      <ListCell
        type="content"
        title="Title Tag Title Tag Title" title-ellips
        brief="Title Content Title Content Title" brief-ellips
      ></ListCell>
    </ListItem>
  </List>

</template>
```

Image-text information

Sample image

Image & Text Pattern with a source attached



Title

"Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder.

Source Time | Other Info

Image & Text Pattern with a source attached



Title

"Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder.



Title

"Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder.

[See More](#)

Sample code

```
<template>
  <div>
    <List type="ptext">
      <ListCell type="header-sp" slot="header">Image & Text Pattern with a source attached
    </ListCell>
    <ListItem content="Tag" >
      <ListCell type="thumb" slot="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
      <ListCell
        title="Title"
        brief=""Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder."
      >
      <ListCell type="sti" slot="after">
        <span>Source Time | Other Info</span>
      </ListCell>
    </ListCell>
    </ListItem>
  </List>

  <List type="ptext">
    <ListCell type="header-sp" slot="header">Image & Text Pattern with a source attached
  </ListCell>
  <ListItem content="Tag" >
    <ListCell type="thumb" slot="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
    <ListCell
      title="Title"
      brief=""Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder."
    </>
  </ListItem>
  <ListItem content="Tag" >
    <ListCell type="thumb" slot="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"/>
    <ListCell
      title="Title"
      brief=""Global Airport Plan" means Alipay will give tourists in overseas airports access to services such as Flight Reminder."
    </>
  </ListItem>
  <ListItem type="more" slot="footer">See More</ListCell>
</List>
</div>

</template>
```

Information list

Sample image

Tag Tag Tag Tag Tag Tag Tag.

Tag Tag Tag Tag Tag Tag Tag. >

Tag Tag Tag Tag Tag Tag Tag. >

Tag Tag Tag Tag Tag Tag Tag

Tag Tag Tag Tag Tag Tag Tag.

Tag Tag Tag Tag Tag Tag Tag

Tag Tag Tag Tag Tag Tag Tag. >

Tag Tag Tag Tag Tag Tag Tag.

Sample code

```
<template>
  <div>

    <List type="info" >
      <ListItem type="oneline" content="TagTag" extra="Tag Tag Tag Tag" />
    </List>

    <List type="info" >
      <ListItem type="oneline" content="TagTag" extra="Tag Tag Tag Tag" arrow/>
    </List>

    <List type="info">
      <ListItem type="more">
        <ListItem type="part" content="Tag" extra="Tag Tag Tag Tag" arrow/>
        <ListItem type="part" :content="false" extra="Tag Tag Tag Tag"/>
      </ListItem>
    </List>

    <List type="info">
      <ListItem>
        <ListItem type="part" content="Tag" extra="Tag Tag Tag Tag" />
        <ListItem type="part" :content="false" extra="Tag Tag Tag Tag"/>
      </ListItem>
    </List>

    <List type="info">
      <ListItem type="more">
        <ListItem type="part" content="Tag" extra="Tag Tag Tag Tag" arrow/>
        <ListItem type="part" content="Tag" extra="Tag Tag Tag Tag"/>
      </ListItem>
    </List>
  </div>

</template>
```

Left-right lines

Sample image

Bank Card List (Round Image 72 x 72(ios),
108 x 108(ios))



CMB
Ending in 7785



Left & Right Text List

Title 1
Content 1

Title 2
Content 2

Title 1
Content 1

Title 2
Content 2



Title 1
Content 1

Title 2
Content 2



Title 1
Content 1

Title 2

Sample code

```
<template>
  <div>

    <List type="bank" >
      <ListCell type="header" slot="header">Bank Card List(Round Image 72 x 72(ios), 108 x 108(ios))</ListCell>
      <ListItem arrow>
        <ListCell type="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png" slot="thumb"/>
        <ListCell title="CMB" brief="Ending in 7785"/>
      </ListItem>
    </List>

    <List type="twoline-side" >
      <ListCell type="header-sp" slot="header">Left & Right text list</ListCell>
      <ListItem >
        <ListCell title="Title 1" brief="Content 1" />
        <ListCell type="extra" title="Title 2" brief="Content 2" slot="extra"/>
      </ListItem>
      <ListItem >
        <ListCell title="Title 1" brief="Content 1" />
        <ListCell type="extra" title="Title 2" brief="Content 2" slot="extra"/>
      </ListItem>

      <ListItem >
        <ListCell type="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png" />
        <ListCell title="Title 1" brief="Content 1" />
        <ListCell type="extra" title="Title 2" brief="Content 2" slot="extra"/>
      </ListItem>
      <ListItem >
        <ListCell type="thumb" src="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png" />
        <ListCell title="Title 1" brief="Content 1" />
        <ListCell type="extra" title="Title 2" slot="extra"/>
      </ListItem>

    </List>

  </div>

</template>
```

1.2.7. Input component

1.2.7.1. Checkbox

This topic describes different methods of using the Checkbox component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).

- [API description](#) provides API information for props, slots, and events.

Kylin

```
<dependency component="{ ListItemCheckbox }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { ListItemCheckbox } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
type	The type of a checkbox. Valid values: <code>checkbox</code> , <code>radio</code> , and <code>agreement</code> .	String	<code>checkbox</code>
value	Return the selected <code>tag</code> field for the <code>radio</code> box, and return <code>true/false</code> for <code>checkbox/agreement</code> .	String, Boolean	-
tag	Provide different options for the <code>radio</code> box under the same <code>name</code> .	String	-
label	The label of a checkbox. To customize <code>DOM</code> , you can use <code>slot[name=label]</code> .	String	-
id	The <code>id</code> attribute of the input.	String	-
Name	The <code>name</code> attribute of the input.	String	-

Property	Description	Type	Default value
disabled	Whether to disable a checkbox.	Boolean	false
brief	Used for an auxiliary label, which is unavailable for the <code>agreement</code> type.	String	-
thumb	Display the URL of an image, which is unavailable for the <code>agreement</code> type.	String	-
thumbAlt	The <code>alt</code> attribute of an image, which is unavailable for the <code>agreement</code> type.	String	-

slots

Name	Description	Scope
label	Used to extend <code>props.label</code> from a string to a <code>DOM node</code> . To customize <code>label</code> , you need to fill the <code>label[for]</code> field with <code>props.id</code> .	<code>{id:String}</code>

events

Name	Description	Function
input	Adapt to <code>v-model</code> . If the type is <code>radio</code> , return the string of the selected <code>tag</code> , otherwise return a Boolean value indicating whether the tag is selected.	Function(value: [string(radio), boolean(other types)]): void

Demo

- [Checkbox](#)

- [Radio button](#)
- [Agreement checkbox](#)

Checkbox

Checkbox without image

Sample image

☐ List Item Check Box - Unselected

☒ List Item Check Box - Selected

☐ List Item Check Box - Unselected

☐ List Item Check Box - Disable

Sample code

```
<template>
  <List>
    <ListItemCheckbox
      id="test1"
      label="List Item Check Box--Unselected"
      v-model="c1"
    />
    <ListItemCheckbox
      id="test2"
      label="List Item Check Box--Selected"
      v-model="c2"
    />
    <ListItemCheckbox
      id="test3"
      label="List Item Check Box--Unselected"
      v-model="c3"
    />
    <ListItemCheckbox id="test4" label="List Item Check Box--Disable" disabled />
  </List>
</template>

<script>
  export default {
    data() {
      return {
        c1: false,
        c2: true,
        c3: false,
      };
    },
  };
</script>
```

Checkbox with image

Sample image



Sample code

```
<template>
  <div>
    <List>
      <ListItemCheckbox
        id="test1"
        label="List Item Check Box--Selected"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        thumbAlt="Picture description"
      />
      <ListItemCheckbox
        id="test2"
        label="List Item Check Box--Disable"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        thumbAlt="Picture description"
        disabled
      />
    </List>
    <List class="towline">
      <ListItemCheckbox
        id="test3"
        label="List Item Check Box--Selected"
        brief="Auxiliary description"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        thumbAlt="Picture description"
      />
      <ListItemCheckbox
        id="test4"
        label="List Item Check Box--Disable"
        brief="Auxiliary description"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        thumbAlt="Picture description"
        disabled
      />
    </List>
  </div>
</template>
```

Radio button

Sample image

Radio Box - Custom



Tag 1



Tag 2



Tag 3

Radio Box - Controlled



Tag 1



Tag 2



Tag 3

Select 2

Sample code

```
<template>
  <div>
    <List>
      <ListCell type="header" slot="header">Radio Box-Custom</ListCell>
      <ListItemCheckbox
        :id="'test'+i"
        type="radio"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        v-for="i in 3"
        :label="'lable'+i"
        :tag="i + ''"
        name="demo"
        @input="onChange"
      />
    </List>
    <List>
      <ListCell type="header" slot="header">Radio Box-Controlled</ListCell>
      <ListItemCheckbox
        :id="'test'+i"
        type="radio"
        thumb="https://os.alipayobjects.com/rmsportal/OhSzVdRBnfwiuCK.png"
        v-for="i in 3"
        :label="'lable'+i"
        :tag="i + ''"
        name="demo2"
        v-model="checked"
      />
      <AButton @click="checked = '2'">
        choice 2
      </AButton>
    </List>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        checked: '1',
      };
    },
    methods: {
      onChange(v) {
        console.log(v);
      },
    },
  };
</script>
```

Agreement checkbox

Sample image



Sample code

```
<template>
  <List>
    <ListItemCheckbox
      type="agreement"
      id="agree"
      label
      v-model="checked"
    >
      <template slot="label">
        <label class="am-ft-md" for="agree">Agree</label>
        <a href="#">《Credit Payment Service Agreement》</a>
      </template>
    </ListItemCheckbox>
    <ListItemCheckbox
      id="test2"
      type="agreement"
      label="List Item Check Box--Disable"
      disabled
    />
  </List>
</template>

<script>
  export default {
    data() {
      return {
        checked: false,
      };
    },
  };
</script>
```

1.2.7.2. Input

This topic describes different methods of using the Input component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

Note

This component is generally used with `List [type = 'form']`.

Kylin

```
<dependency component="{ ListItemInput }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { ListItemInput } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
value	The value of an input box, which can be bound to <code>v-model</code> .	String	-
label	The label on the left side of an input box, which is not displayed when the value is empty.	String	-
labelWidth	The <code>width</code> of the label on the left side of an input box, which can be set to a number (similar to <code>em</code>) or string (in a custom unit).	String, Number	auto
placeholder	The placeholder text of an input box.	String	-
inputType	The type of the input box. Used for the browser to identify the keyboard type.	String	text
clear	The clear button of an input box. Displayed only when the input box is not empty and <code>clear</code> is set to <code>true</code> instead of <code>disabled</code> .	Boolean	false
labelId	The <code>id</code> of the label on the left side of an input box. Used for accessibility,	String	-

Property	Description	Type	Default value
inputId	The <code>id</code> of the label of an input box. Used for accessibility.	String	-
disabled	Whether to disable an input box.	Boolean	false
error	Indicate that the current input value of an input box is improper.	Boolean	false
button	The button text on the right side, which is not displayed when the value is empty.	String	-
buttonDisabled	Whether to disable the button on the right side.	Boolean	false

slots

Name	Description	Scope
-	Enable you to extend <code>props.label</code> from a string to a <code>DOM</code> node.	-

events

Name	Description	Function	Note
input	Trigger an <code>input</code> event when <code>value</code> changes or <code>value</code> is cleared. Support <code>v-model</code> .	Function(value : boolean): void	
click	Triggered upon a tap operation.	Function(event : event): void	
errorClick	Triggered when the red circle on the right side is tapped and <code>error</code> is configured.	Function(void) : void	
buttonClick	Triggered when the button on the right side is tapped and <code>button</code> is configured and enabled.	Function(void) : void	Since <code>0.4.8-open02</code>

Demo

General

Sample image

Regular Item

TagTag ContentContent

Tag Synchronize revised content 

Tag Synchronize revised content 

TagTag Do not synchronize initial value

Sample code

```
<template>

  <div>

    <List type="form">
      <ListCell type="header" slot="header">Regular Item</ListCell>
      <ListItemInput label="Tag" placeholder="Content" clear></ListItemInput>
    </List>

    <List type="form">
      <ListItemInput label="Tag"placeholder="Content" v-model="syncText" clear></ListItemInput>
    </List>
    <ListItemInput label="Tag"placeholder="Content" v-model="syncText" clear></ListItemInput>
    </List>

    <List type="form">
      <ListItemInput label="Tag" placeholder="Content" value="Do not synchronize initial value" ></ListItemInput>
    </List>

  </div>

</template>

<script type="text/javascript">
  export default {
    data() {
      return {
        syncText: 'Synchronize revised content',
      };
    },
  };
</script>
```

Label

Sample image

Regular Item

No tag, dark prompt text

Coupon Enter the coupon name

Only 5 characters are allowed

Coupon Dark prompt text

Coupon Code Dark prompt text

Only 6 characters are allowed

Coupon Dark prompt text

Validity Period Permanent

Sample code

```
<template>

  <div>

    <List type="form">
      <ListCell type="header" slot="header">Regular List</ListCell>
      <ListItemInput label="" placeholder="No tag,dark prompt text" clear></ListItemInput>
      <ListItemInput label="Coupon name" placeholder="Enter the coupon name" clear></ListItemInput>
    </List>

    <List type="form">
      <ListCell type="header" slot="header">Only 5 characters are allowed</ListCell>
      <ListItemInput label="Coupon" label-width="5em" placeholder="dark prompt dark prompt" clear></ListItemInput>
      <ListItemInput label="Coupon Code" label-width="5em" placeholder="dark prompt dark prompt" clear></ListItemInput>
    </List>

    <List type="form">
      <ListCell type="header" slot="header">Only 6 characters are allowed</ListCell>
      <ListItemInput label="Coupon" :label-width="6" placeholder="dark prompt dark prompt" clear></ListItemInput>
      <ListItemInput label="Coupon Validity Period" :label-width="6" value="Permanent validity" ></ListItemInput>
    </List>

  </div>

</template>

<script type="text/javascript">
  export default {
    data() {
      return {
        syncText: 'Synchronize modified content',
      };
    },
  };
</script>
```

Input error

Sample image

In case of any incorrect data

ID Number 3301022017010100 

Note: an incorrect data will invoke a toast prompt.
The toast will disappear after 2 seconds. Tap the red icon to show the toast prompt again.

Sample code

```
<template>

  <div>

    <List type="form">
      <ListCell type="header" slot="header">In case of any incorrect data</ListCell>
      <ListItemInput label="ID Number" value="33010220170101001XX" error @error-click="onClick"></ListItemInput>
      <ListCell type="footer" slot="footer">Note: an incorrect data will invoke a toast prompt. The toast will disappear after 2 seconds. Tap the red icon to show the toast prompt again.</ListCell>
    </List>

  </div>

</template>

<script>
  import Toast from '../.../lib/toast';

  export default {
    methods: {
      onClick() {
        Toast.show('fill in error');
      },
    },
  };
</script>
```

1.2.8. Loading component

1.2.8.1. Loading

This topic describes different methods of using the Loading component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).

- [API description](#) provides API information for props and slots.

Kylin

```
<dependency component="{ Loading, LoadingIndicator }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Loading, LoadingIndicator } from '@alipay/antui-vue';
```

API description

Loading

Packaged loading, including full page loading and footer loading.

props

Property	Description	Type	Default value
type	The loading type. Valid values: <code>""</code> , <code>"page"</code> , <code>"refresh"</code> , and <code>"nomore"</code> .	String	<code>""</code>

slots

Name	Description	Scope
-	Default placeholder and content, with custom <code>DOM</code> supported.	-

LoadingIndicator

Independent three-square animation for loading.

props

Property	Description	Type	Default value
white	Whether to display white squares (blue squares are displayed by default).	Boolean	false
tiny	Whether to display the squares in minimum size.	Boolean	false

Demo

- [Loading animation only](#)
- [Loading at the center of a page](#)
- [Loading at the top of a page](#)
- [Loading at the bottom of a page](#)
- [No more](#)

Loading animation only

Sample image



Sample code

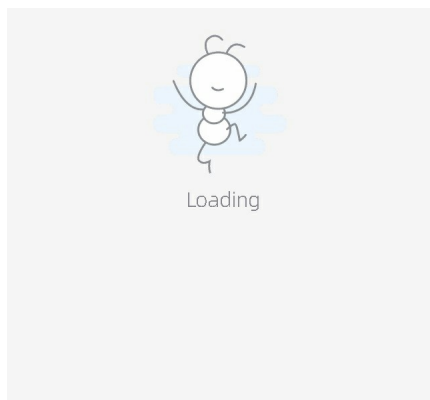
```
<template>

  <div style="text-align: center; background: #999;">
    <LoadingIndicator />
    <br/>
    <LoadingIndicator white />
    <br/>
    <LoadingIndicator tiny />
    <br/>
    <LoadingIndicator tiny white />
  </div>

</template>
```

Loading at the center of a page

Sample image

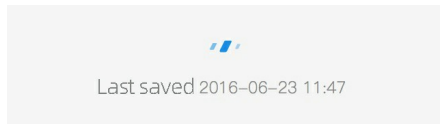


Sample code

```
<template>
  <div style="min-height: 300px;">
    <Loading type="page">Loading</Loading>
  </div>
</template>
```

Loading at the top of a page

Sample image



Sample code

```
<template>

  <Loading type="refresh">Last saved 2016-06-23 11:47</Loading>

</template>
```

Loading at the bottom of a page

Sample image

Content Display Area



Sample code

```
<template>

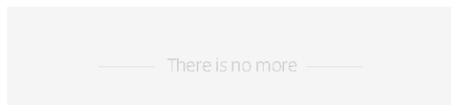
  <div>
    <div style="color:#F4333C;background-color: #fff; text-align: center; height: 300px;">Content Display Area</div>
    <Loading >Loading...</Loading>
  </div>

</template>
```

No more

Sample image

Content Display Area



Sample code

```
<template>

  <div>
    <div style="color:#F4333C;background-color: #fff; text-align: center; height: 300px;">Content Display Area</div>
    <Loading type="nomore">There is no more</Loading>
  </div>

</template>
```

1.2.9. Result page component

1.2.9.1. Message

The Message topic describes different methods of using this component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).

- [API description](#) provides API information for props, slots, and events.

Kylin

```
<dependency component="{ Message }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Message } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
type	Button type. Value options include <code>result</code> , <code>multi</code> , <code>week</code> , and <code>''</code> (empty string).	String	<code>''</code>
icon	Icon type. Value options include <code>success</code> , <code>pay</code> , <code>fail</code> , <code>error</code> , <code>warn</code> , <code>info</code> , <code>wait</code> , and <code>question</code> . You can also input a <code>class</code> to define the icon.	String	<code>success</code>
main	Main content of the pure text prompt. The DOM of the placeholder is not displayed when the main content is specified as <code>false</code> .	String, Boolean	-
sub	Sub content of the pure text prompt. The DOM of the placeholder is not displayed when the main content is specified as <code>false</code> .	String, Boolean	-

slots

Name	Description	Scope
main	Used in the scenario where the DOM needs to be populated into <code>props.main</code> .	-

Name	Description	Scope
sub	Used in the scenario where the DOM needs to be populated into <code>props.sub</code> .	-
-	Used to populate content under sub.	-

events:None

Demo

Success state

Sample image



Successful

Sub-prompt



Prompt

Sub-prompt



Payment Succeeded

Sample code

```
<template>

  <div>
    <Message type="result" icon="success" main="Success" sub="Sub-prompt"></Message>
    <Message type="multi" icon="success" main="Prompt" sub="Sub-prompt"></Message>
    <Message type="" icon="success" main="Payment Succeeded" :sub="false" ></Message>
  </div>

</template>
```

Success result page

Sample image



Successful

Content should be
no more than 2 rows.



Sample code

```
<template>

  <div>
    <Message type="result" icon="successful" main="Success" >
      <template slot="sub">
        Content should be<br />no more than 2 rows.
      </template>
    </Message>
    <div class="am-button-wrap">
      <AButton type="blue">Primary Action</AButton>
      <AButton type="white">Auxiliary Action</AButton>
    </div>
  </div>

</template>
```

1.2.9.2. PageResult

The PageResult topic describes different methods of using this component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

Kylin

```
<dependency component="{ PageResult, AButton }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { PageResult, AButton } from '@alipay/antui-vue';
```

Service imperative call

You can directly use `PageResult.show ('system busy');` to call a command, instead of writing the command in a template. The `DOM` will be automatically inserted into `document.body`.

```
PageResult.show({
  type: 'text',
  text: 'Displayed message'
});
```

Service documentation

static methods

`PageResult` provides the following static methods.

Name	Description	Function
show	Creates and displays a <code>PageResult</code> instance with the following parameters. If the instance is called multiple times, the previous instance will be destructed. You can also directly input a character string as the <code>content</code> .	Function(option: Object string): vm

show options

The following parameters are accepted for creating an instance.

Property	Description	Type	Default value
type	<code>PageResult</code> type. Value options include <code>error</code> , <code>empty</code> , <code>nofound</code> , <code>network</code> , and <code>busy</code> .	String	<code>busy</code>
content	Pure character string text.	String	The system is busy, please try again later.

Property	Description	Type	Default value
title	Title text.	String	''
btns	Array of displayed buttons.	<pre>Array<{ text: string, click: Function }></pre>	<pre>[[{text: 'Reload',click: () => window.location.reload()}]</pre>
zIndex	Specifies the <code>zIndex</code> value of the elastic layer.	Number	9000

API description

props

Property	Description	Type	Default value
type	Icon type. Value options include <code>error</code> , <code>empty</code> , <code>nofound</code> , <code>network</code> , and <code>busy</code> .	String	-
title	The custom business text contains no more than 14 characters.	String	-
brief	The custom business text contains no more than two lines. Auxiliary text is optional.	String	-

slots

Name	Description	Scope
-	Default slot used to place one or more buttons.	-
title	Used to define the <code>DOM</code> .	-
brief	Used to define the <code>DOM</code> .	-

events:None

Demo

Network timeout

Sample image



Network Timeout

The farthest distance in this world.

Refresh

Sample code

```
<template>

  <PageResult type="network"
    title="Network Timeout"
    brief="The farthest distance in this world."
  >
    <AButton type="page-result" >Refresh</AButton>
  </PageResult>

</template>
```

System error

Sample image



Troubleshooting in Progress

We are sorry for the inconvenience

Refresh

Sample code

```
<template>

  <PageResult type="error"
    title="Troubleshooting in Progress"
    brief="We are sorry for the inconvenience."
  >
    <AButton type="page-result" >Refresh</AButton>
  </PageResult>

</template>
```

Empty

Sample image



Title content

Auxiliary content

Action options

Sample code

```
<template>

  <PageResult type="empty"
    title="A maximum of 14 characters is allowed."
    brief="Auxiliary content"
  >
    <AButton type="page-result" >Action options</AButton>
  </PageResult>

</template>
```

Busy

Sample image



System Timeout

System timeout content

Refresh

Sample code

```
<template>

  <PageResult type="busy"
    title="System Busy"
    brief="System timeout content"
  >
    <AButton type="page-result" >Refresh</AButton>
  </PageResult>

</template>
```

1.2.10. Notification component

1.2.10.1. Inform

This topic describes different methods of using the Inform component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

Kylin

```
<dependency component="{ Inform }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Inform } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
operation	Available actions for notifications. Valid values: <code>Go</code> and <code>Button</code> .	String	<code>null</code>
buttonText	The button text that is displayed when <code>operation</code> is set to <code>button</code> .	String	<code>'Got it'</code>
href	Redirection upon tapping. Valid when <code>operation</code> is set to <code>go</code> .	String	<code>null</code>

slots

Name	Description	Scope
-	Notification content	-
button	When <code>operation</code> is set to <code>button</code> , the content of <code>buttonText</code> can be customized as <code>DOM</code> .	-

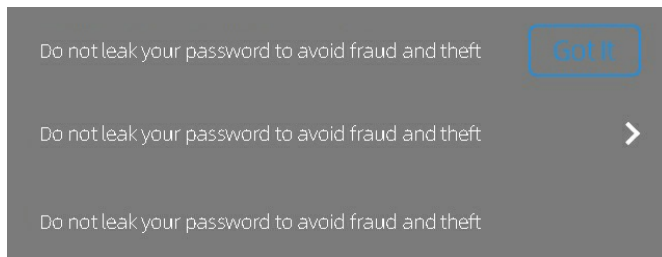
events

Name	Description	Function
buttonClick	Triggered when a button is tapped.	Function(): void

Demo

The following shows a sample for a basic style Inform notice.

Sample image



Sample code

```
<template>
  <div>
    <Inform
      :style="{visibility: visible ? 'visible' : 'hidden'}"
      operation="button"
      buttonText="Got It"
      @buttonClick="visible = false"
    >
      Do not leak your password to avoid fraud and theft
    </Inform>
    <Inform
      operation="go"
      href="https://alipay.com/"
    >
      Do not leak your password to avoid fraud and theft
    </Inform>
    <Inform>
      Do not leak your password to avoid fraud and theft
    </Inform>
  </div>
</template>

<script>
  export default {
    data() {
      return {
        visible: true,
      };
    },
  };
</script>
```

1.2.10.2. Notice

The Notice topic describes different methods of using this component and the API descriptions:

- Use this component in the [Kylin](#) project.
- Use this component in other projects. Import it through [ESModule](#).
- [API description](#) provides API information for props, slots, and events.

Kylin

```
<dependency component="{ Notice }" src="@alipay/antui-vue" ></dependency>
```

ESModule

```
import { Notice } from '@alipay/antui-vue';
```

API description

props

Property	Description	Type	Default value
operation	Available operations for announcements. Valid values: <code>Go</code> and <code>Close</code> .	String	null
href	Redirection upon tapping. Valid when <code>operation</code> is set to <code>go</code> .	String	null

slots

Name	Description	Scope
-	Announcement content	-

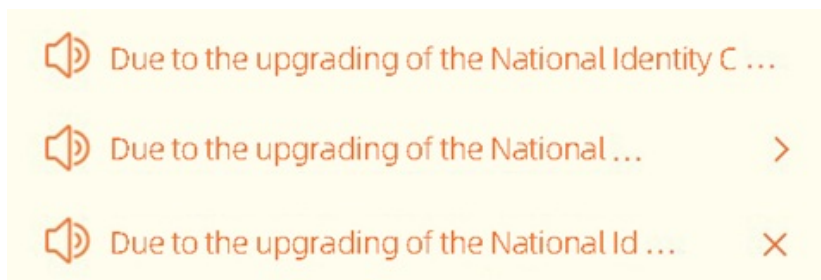
events

Name	Description	Function
close	Triggered upon tapping the close button when <code>operation</code> is set to <code>close</code> .	Function(): void

Demo

The following shows a sample for a basic style notice.

Sample image



Sample code

```
<template>
<div>
  <Notice>
    Due to the upgrading of the National Identity Card System, when adding a bank card.....
  .
</Notice>
<Notice operation="go" href="https://www.alipay.com/">
  Due to the upgrading of the National Identity Card System, when adding a bank card.....
..
</Notice>
<Notice operation="close" @close="show = false" v-if="show">
  Due to the upgrading of the National Identity Card System, when adding a bank card.....
..
</Notice>
</div>

</template>

<script>
export default {
  data() {
    return {
      show: true,
    };
  },
};
</script>
```

1.3. Introduction to Native framework

The mPaaS unified component library (AntUI) converts abstract visual specifications concepts into control entities based on standardized visual specifications. Developers can unify visual specifications on clients upon control access by using the unified component library.

Unified component library architecture

The overall architecture of AntUI is similar to block building. The AntUI unified control system is built from bottom to top:

AntUI



The following table describes the layers of the architecture from bottom to top.

Architecture layer	Description
Foundation	The Foundation layer represents modular visual specifications. It is the foundation for building the AntUI system and consists of atomic resources, atomic widgets, and Iconfont. The Foundation layer is built of the minimum units of visual specifications.
Common	The Common layer is the core unification module of AntUI and the most frequently used unification widget module on the business side. This layer consists of common resources, basic widgets, and the Theme Manager. The Common layer is built based on the combination and visualization of the Foundation layer. This layer can be used in all common scenarios on the client.
Scene	This layer builds a set of scene-oriented widgets, such as the fund widget, business widget, and community widget. Due to the fact that mPaaS is a super app, its volume determines that a large amount of business needs personalized processing. Therefore, AntUI sets up the Scene layer, which builds personalized and scene-oriented widgets for processing business based on the Foundation layer.
Application	The Application layer provides capabilities such as differentiated processing and HTML5 container support. This layer solves the conflict between unification and platform personalization. Atomic resources, combinations, and scenes are the basis of AntUI construction. However, in actual application scenarios, the requirements of Android, iOS, and HTML5 need to be taken into account at the same time. Therefore, the unified component library builds some personalized and differentiated APIs of the platform, that is, the application layers.

Architecture layer	Description
--------------------	-------------

Foundation layer

It represents modular visual specifications. It is the foundation for building the AntUI system and mainly contains the atomic resources, atomic controls, and Iconfont icon.

- **Atomic resources** define the resources that are used by widgets such as color, size, and spacing in an atomized manner to ensure uniqueness. For example, colors include red, yellow, and blue and font sizes include 1, 2, and 3.
- **Atomic widgets** package the built-in widgets of the platform framework to build a basic atomic widget library.
- **Iconfont** collects icons for common scenes and builds the Iconfont format to provide an available widget icon library.

Common layer

The common layer is the core unification module of the AntUI, that is, the unification control module most commonly used by the client. It contains the common resources, basic controls, and style manager.

- **Common resources** are used for secondary definition of atomic resources based on application scenarios, such as title color, content color, and link color.
- **Basic widgets** perform one-to-one visual presentation on the widgets that are defined in the visual draft to ensure consistent naming and implementation on Android and iOS. This facilitates the development and usage on clients.
- The **Theme Manager** defines styles in an abstract manner and manages them in a centralized manner. A specific widget can switch between multiple sets of skins. Style abstraction is implemented in an incremental definition manner. Therefore, you only need to focus on styles of some elements required by business.

Scene layer

This layer builds a set of scene-based controls, for example, fund control, business control, and social control.

Application layer

The Application layer provides capabilities such as differentiated processing and HTML5 container support. This layer solves the conflict between unification and platform personalization.

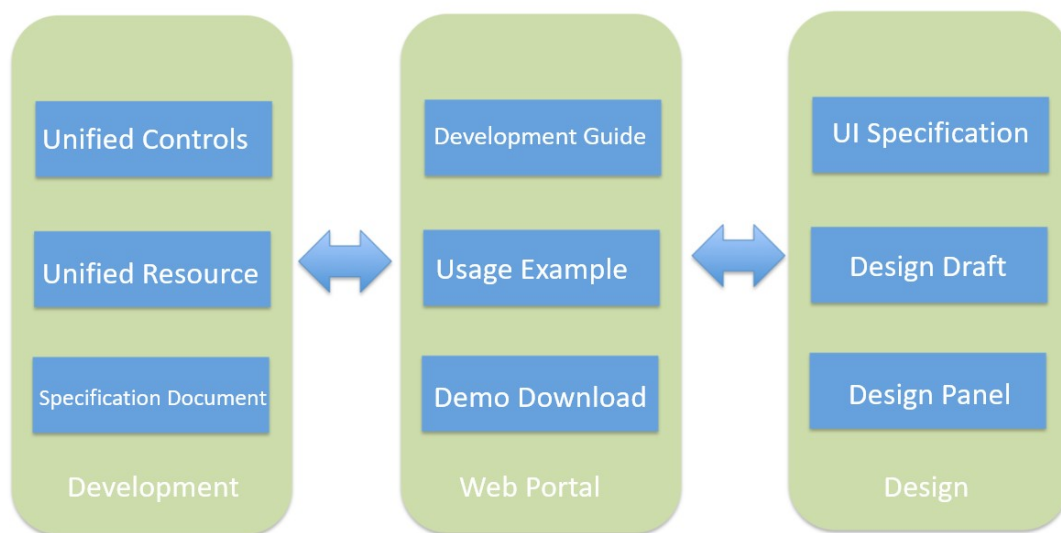
Android and iOS platforms differ in visual specifications. Taking actionsheet as an example, Ant UI performs different processing by platform:

- For the iOS platform, keep the bottom flyout.
- For the Android platform, keep the middle list pop-up window.

There are many different scenes for H5, such as pop-up windows and title bars. To keep platform features in the visual experience with HTML5, Ant UI defines a unified JSAPI for HTML5 containers. This facilitates the evoking of platform widgets for differentiated processing of HTML5 pages on Android and iOS.

Linkage

To reduce the communication cost between designers and developers and avoid repeated control development and visual design, the unified component library (Ant UI) aggregates the development and visual work.



Designers formulate specifications and developers interpret specifications into controls. Complete development guidelines facilitate development implementation and form a one-stop control system.

- Unified naming helps designers and developers achieve unified cognition. For more information, see the [Component specifications and principles](#) section.
- Designers can recognize existing controls through the design panel and build a basic structure of a page through simple drag-and-drop operations.
- Developers can intuitively view visual effects of a control by referring to the portal aggregation development documents and visual specifications and downloading the demo.

Component specifications and principles

• Naming conventions

The same type of widgets must be named the same on Android and iOS. A widget name must be prefixed with AU. The custom properties of widgets must be named in camel case.

 Note

Some widgets may need to be implemented in one platform but not in the other platform due to platform differences.

- **Matching between basic widgets and visual/interaction specifications**

- Controls not specified in the specifications cannot be delivered in standard controls.
- Controls that are not specified in the specifications but are already used in multiple places should be delivered in the candidate control set.
- Single specifications, such as the title bar specifications, are not forcibly required to be implemented as a single control.

- **Ease of use**

- Different from commonui, the unified control library does not simply encapsulate system controls (such as `APIImageView` and `APTextView`). When you need to use system controls, it is recommended that you use native controls.
- The control name must be accurate and clear.
- Similar functions should be consistent in different controls.
- User habits should be respected.

- **Extensibility**

- Do not use hard coding to implement widget features, such as the dynamic modification of the number of labels.
- Some widgets are required to support external modification of their layouts, such as dialog boxes and navigation bars.

- **Novelty**

You can try the latest platform features. For example, you can use `RecyclerView` for Android.

1.4. Native based - Android component library

1.4.1. Quick start

Ant UI supports three types of access: **native AAR**, **mPaaS Inside** and **component-based mode (Portal & Bundle)**.

Prerequisite

- If you want to connect the component to the mPaaS based on the native AAR mode, you need to first complete the prerequisites and the subsequent steps. For more information, see [Add mPaaS to your project](#).
- If you want to connect the component to the mPaaS based on the mPaaS Inside mode, you need to first complete the [mPaaS Inside access procedure](#).

- If you want to connect the component to the mPaaS based on components, you need to first complete the [Component-based access procedure](#).

Add the SDK

Native AAR mode

You can use the **component management** feature to install the Ant UI component in your project. For more information, see [ARR component management](#).

mPaaS Inside mode

Use the **component management** feature to install the Ant UI component in your project.

For more information, see [Manage component dependencies](#).

Component-based mode

In your Portal and Bundle projects, use the **component management** feature to install the Ant UI component.

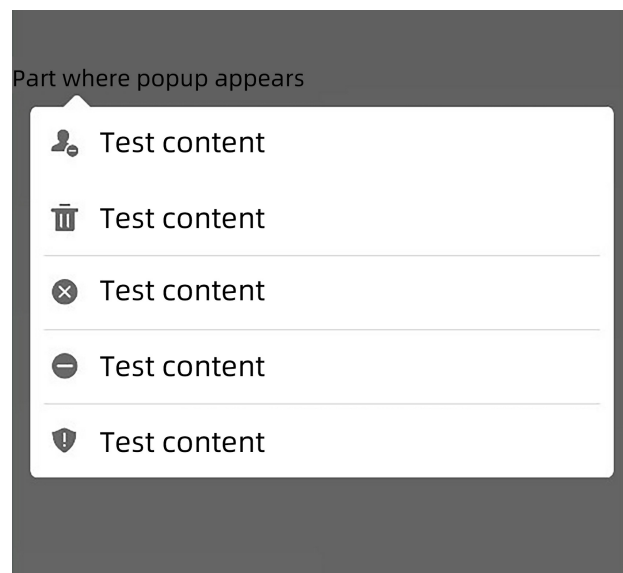
For more information, see [Manage component dependencies](#).

1.4.2. Dialog component

1.4.2.1. Card menu

AUCardMenu is used to display a selection menu when the user taps the card on the homepage of the mPaaS client. It's essentially a dialog, and similar to a pop up window.

Sample image



Dependency

See [Quick start](#).

API description

```

/**
 * show dialog with default width
 * @param view
 * @param popItems
 */
public void showDrop(View view, ArrayList<MessagePopItem> popItems)

/**
 * show dialog with given width
 * @param view
 * @param popItems
 * @param width
 */
public void showDrop(View view, ArrayList<MessagePopItem> popItems, int width) {
    int defaultMarginRight = mContext.getResources().getDimensionPixelSize(R.dimen.AU_SPACE5)/2
    ;
    showDrop(view, popItems, width, defaultMarginRight);
}

/**
 * show dialog with given width & marginRight
 * @param view
 * @param popItems
 * @param width
 */
public void showDrop(View view, ArrayList<MessagePopItem> popItems, int width, int marginRi
ght)

/**
 * show dialog with given ViewLoc
 * @param location
 * @param popItems
 */
public void showDropWithLocation(ViewLoc location, ArrayList<MessagePopItem> popItems)

If an image is in the hyperlink form, you need to download it.
public void setOnLoadImageListener(OnLoadImageListener onLoadImageListener)

public interface OnLoadImageListener {

/**
 * show dialog with given width & marginRight
 * @param url URL of the image.
 * @param imageView Target view of the image.
 * @param defaultDrawable The default image.
 */
public void loadImage(String url, AUImageView imageView ,Drawable defaultDrawable);
}

```

Custom properties

No customized property is available. The XML layout is not supported.

Sample code

```
ArrayList<MessagePopItem> menuList = new ArrayList<MessagePopItem>();

MessagePopItem item1 = new MessagePopItem();
IconInfo info = new IconInfo();
info.type = IconInfo.TYPE_DRAWABLE;
info.drawable = getResources().getDrawable(R.drawable.menu_del_reject);
item1.icon = info;
item1.title = "Test content";
menuList.add(item1);

MessagePopItem item2 = new MessagePopItem();
IconInfo info2 = new IconInfo();
info2.type = IconInfo.TYPE_DRAWABLE;
info2.drawable = getResources().getDrawable(R.drawable.menu_delete);
item2.icon = info2;
item2.title = "Test content";
menuList.add(item2);

MessagePopItem item3 = new MessagePopItem();
IconInfo info3 = new IconInfo();
info3.type = IconInfo.TYPE_DRAWABLE;
info3.drawable = getResources().getDrawable(R.drawable.menu_ignore);
item3.icon = info3;
item3.title = "Test content";
menuList.add(item3);

MessagePopItem item4 = new MessagePopItem();
IconInfo info4 = new IconInfo();
info4.type = IconInfo.TYPE_DRAWABLE;
info4.drawable = getResources().getDrawable(R.drawable.menu_reject);
item4.icon = info4;
item4.title = "Test content";
menuList.add(item4);

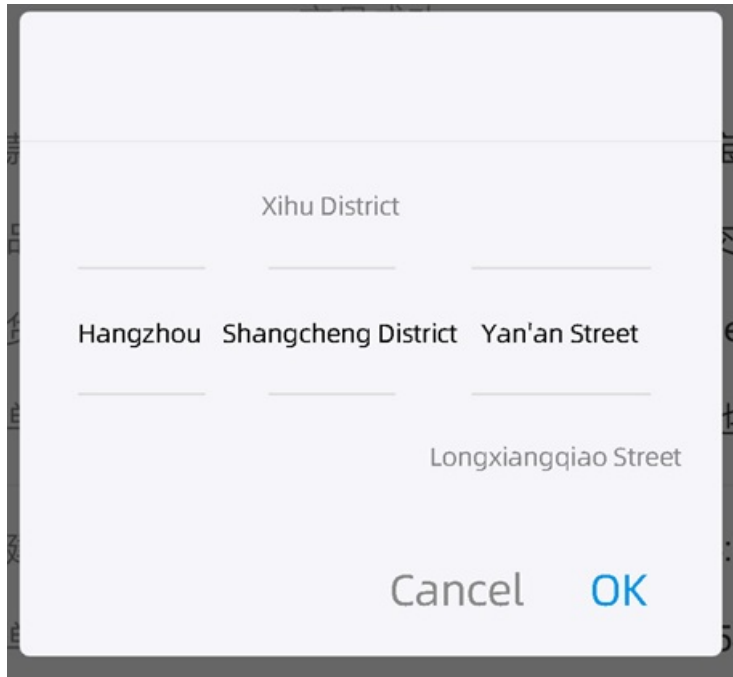
MessagePopItem item5 = new MessagePopItem();
IconInfo info5 = new IconInfo();
info5.type = IconInfo.TYPE_DRAWABLE;
info5.drawable = getResources().getDrawable(R.drawable.menu_report);
item5.icon = info5;
item5.title = "Test content";
menuList.add(item5);

final AUCardMenu popMenu = new AUCardMenu(CardMenuActivity.this);
int id = v.getId();
if(id == R.id.showCardMenu1) {
    popMenu.showDrop(textView1, menuList);
} else if(id == R.id.showCardMenu2) {
    popMenu.showDrop(textView2, menuList);
}
```

1.4.2.2. Cascade picker

AUCascadePicker provides a multi-level cascade selector that supports selection at a maximum of three levels.

Sample image



API description

```
/**
 * Set the selected list.
 */
public void setDateData(List<PickerDataModel> strList)

/**
 * Start the selected items.
 * @param model
 */
public void setSelectedItem(PickerDataModel model)

/**
 * Set listeners for the selected items.
 * @param model
 */
public void setOnLinkagePickerListener(OnLinkagePickerListener listener)
```

JSAPI description

API

ant.UIGetCascadePicker

API usage

```
AlipayJSBridge.call('antUIGetCascadePicker',
{
  title: 'nihao',// The cascade option title.
  selectedList:[{"name":"Hangzhou",subList:[{"name":"Shangcheng District"}]},
  list: [
    {
      name: "Hangzhou",// The entry name.
      subList: [
        {
          name: "Xihu District",
          subList: [
            {
              name: "Gucui Street"
            },
            {
              name: "Wenxin Street"
            }
          ]
        },
        {
          name: "Shangcheng District",
          subList: [
            {
              name: "Yan'an Street"
            },
            {
              name: "Longxiangqiao Street"
            }
          ]
        }
      ]
    }
  ]// The cascade sub-data list.
}
]// The cascade data list.
},
function(result){
  console.log(result);
});
```

Input parameters

Name	Type	Description	Required	Default value	Version
title	String	The cascade control title.	No	-	10.1.2

Name	Type	Description	Required	Default value	Version
selectedList	JSON	Selected state, specifying the selected sub-item and in a format the same as that of the input parameter (<code>{{ "name" : "Hangzhou City", subList: [{ "name" : "Shangcheng District" }] }}</code>)	No	-	10.1.2
List	JSON	The selector data list.	Yes	-	10.1.2
name (a name in a list)	String	The entry name.	Yes	-	10.1.2
subList (a sublist in a list)	JSON	The sub-entry list.	No	-	10.1.2
fn	function	The callback function after selection is complete.	No	-	10.1.2

Output parameters

Name	Type	Description	Version
success	bool	Whether selection is complete. If selection is canceled, false is returned.	10.1.2

Name	Type	Description	Version
result	JSON	The selection result, for example, [{"name": "Hangzhou City", subList: [{"name": "Shangcheng District"}]}] .	10.1.2

Sample code

```
AUCascadePicker datePicker = new AUCascadePicker(PickerActivity.this);
datePicker.setDateData(datas);
datePicker.setOnLinkagePickerListener(new AUCascadePicker.OnLinkagePickerListener() {
    @Override
    public void onLinkagePicked(PickerDataModel msg) {
        PickerDataModel model = msg;
        AuiLogger.info("onLinkagePicked", "onLinkagePicked:"+msg.name+ model);

        StringBuilder sb = new StringBuilder();
        while (msg != null){
            sb.append(msg.name+" ");
            if(msg.subList != null && msg.subList.size() > 0) {
                msg = msg.subList.get(0);
            }else {
                msg = null;
            }
        }
        box3.getInputEdit().setText(sb);
    }
});
datePicker.show();
```

1.4.2.3. Date picker

AUDatePicker is a date selection control and is essentially a pop-up window.

Sample image



Dependency

See [Quick start](#).

API description

```
/**
 * Instantiates a new Date picker.
 *
 * @param activity the activity
 * @param mode     the mode
 * @see #YEAR_MONTH_DAY #YEAR_MONTH_DAY#YEAR_MONTH_DAY
 * @see #YEAR_MONTH #YEAR_MONTH#YEAR_MONTH
 * @see #MONTH_DAY #MONTH_DAY#MONTH_DAY
 */
public ADatePicker(Activity activity, @Mode int mode)

/**
 * Set the date range.
 *
 * @param startYear the start year
 * @param endYear   the end year
 */
public void setRange(int startYear, int endYear)

../**
 * Select the specified year, month, and date.
 *
 * @param year  the year
 * @param month the month
 * @param day   the day
 */
public void setSelectedItem(int year, int month, int day)

/**
 * Select the specified date.
 *
 * @param yearOrMonth the year or month
 * @param monthOrDay  the month or day
 */
public void setSelectedItem(int yearOrMonth, int monthOrDay)
```

```

public void setSelectedItem(String yearOrMonth, String monthOrDay) {

    /**
     * Set the date selection listener.
     *
     * @param listener the listener
     */
    public void setOnDatePickListener(OnDatePickListener listener) {
        this.onDatePickListener = listener;
    }

    /**
     * The interface on year month day pick listener.
     */
    public interface OnYearMonthDayPickListener extends OnDatePickListener {

        /**
         * On date picked.
         *
         * @param year the year
         * @param month the month
         * @param day the day
         */
        void onDatePicked(String year, String month, String day);
    }

    /**
     * The interface On year month pick listener.
     */
    public interface OnYearMonthPickListener extends OnDatePickListener {

        /**
         * On date picked.
         *
         * @param year the year
         * @param month the month
         */
        void onDatePicked(String year, String month);
    }

    /**
     * The interface On month day pick listener.
     */
    public interface OnMonthDayPickListener extends OnDatePickListener {

        /**
         * On date picked.
         *
         * @param month the month
         * @param day the day
         */
        void onDatePicked(String month, String day);
    }

```

```
}
```

Custom properties

No customized property is available. The XML layout file is not supported.

Sample code

```
AUDatePicker datePicker = new AUDatePicker(DatePickActivity.this,AUDatePicker.YEAR_MONTH_DAY);

        datePicker.setRange(1949,2050);
        datePicker.setOnDatePickListener(new AUDatePicker.OnYearMonthDayPickListener() {

            @Override
            public void onDatePicked(String year, String month, String day) {
                Toast.makeText(DatePickActivity.this,year + "-" + month + "-" + day
,Toast.LENGTH_LONG).show();
            }
        });
        datePicker.show();
```

Application of AUDatePicker on a build-in page:

```
AUDatePicker picker = new AUDatePicker(this);
picker.show();
picker.dismiss();
View view = picker.getOutterView();
LinearLayout layout = (LinearLayout) findViewById(R.id.layout);
layout.removeAllViews();
if(view != null) {
    ((ViewGroup) view.getParent()).removeAllViews();
    layout.addView(view);
}
```

For more information, please search for group number 41708565 with DingTalk to join DingTalk group to contact mPaaS after-sales support.

1.4.2.4. Float menu

AUFloatMenu provides a menu that contains icons and option lists.

Sample image

Dependency

See [Quick start](#).

API description

```

/**
 * The constructor.
 *
 * @param context Activity context including the antui-build dependency.
 */
public AUFloatMenu(Context context)
    /**
     * Align with the right by default.
     * @param view Display-based view.
     * @param popItems The list display model.
     */
    @Override
    public void showDrop(View view, ArrayList<MessagePopItem> popItems);

    /**
     * Align with the left.
     * @param view Display-based view.
     * @param popItems The list display model.
     */
    public void showAsDropDownLeft(View view, ArrayList<MessagePopItem> popItems);

    /**
     * Display in the center of the screen.
     * @param parent Display-based view.
     * @param title The title of the display list.
     * @param popItems The list display model.
     */
    public void showAsDropDownTitleCenter(View parent, String title, ArrayList<MessagePopItem> popItems);

    /**
     * Add the entry tapping event of the display list.
     * @param listener
     */
    public void setOnClickListener(AdapterView.OnItemClickListener listener)

```

Sample code

```

ArrayList<MessagePopItem> menuList = new ArrayList<MessagePopItem>();

MessagePopItem item1 = new MessagePopItem();
IconInfo info = new IconInfo();
info.icon = getResources().getString(R.string.iconfont_add_user);
item1.icon = info;
item1.title = "Add contacts";
menuList.add(item1);

MessagePopItem item2 = new MessagePopItem();
IconInfo info2 = new IconInfo();
info2.icon = getResources().getString(R.string.iconfont_group_chat);
item2.icon = info2;
item2.title = "Group chat";
menuList.add(item2);

MessagePopItem item3 = new MessagePopItem();
IconInfo info3 = new IconInfo();
info3.icon = getResources().getString(R.string.iconfont_scan);
item3.icon = info3;
item3.title = "Scan";
menuList.add(item3);

MessagePopItem item4 = new MessagePopItem();
IconInfo info4 = new IconInfo();
info4.icon = getResources().getString(R.string.iconfont_collect_money);
item4.icon = info4;
item4.title = "Payment";
menuList.add(item4);

MessagePopItem item5 = new MessagePopItem();
IconInfo info5 = new IconInfo();
info5.icon = getResources().getString(R.string.iconfont_help);
item5.icon = info5;
item5.title = "Help";
menuList.add(item5);

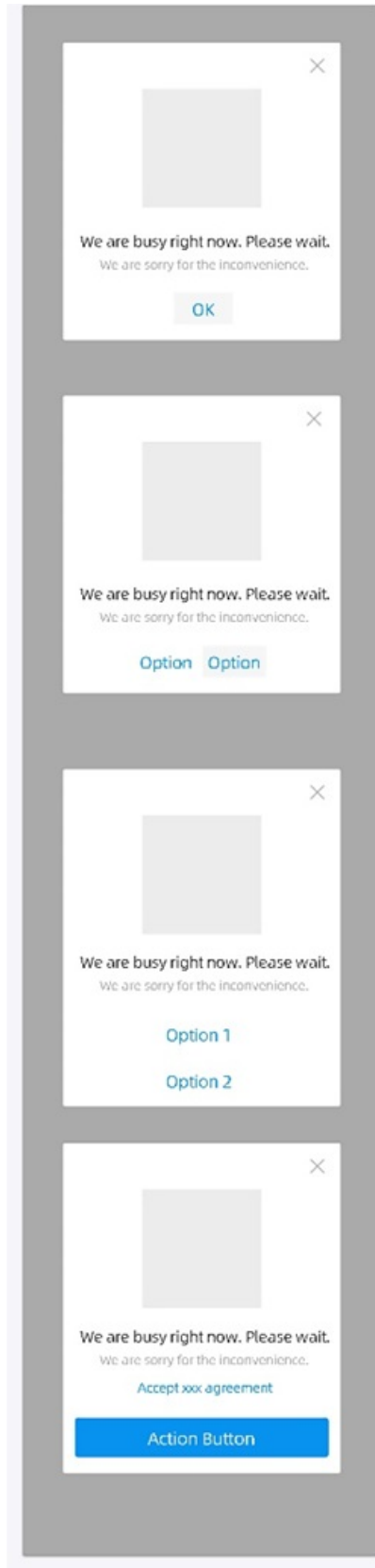
final AUFloatMenu floatMenu = new AUFloatMenu(ScrollTitleBarActivity.this);
floatMenu.showDrop(v, menuList);
floatMenu.setOnClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(ScrollTitleBarActivity.this, String.valueOf(position), Toast.LENGTH_SHORT).show();
        floatMenu.hideDrop();
    }
});

```

1.4.2.5. Image dialog

AUImageDialog (formerly SalesPromotionLimitDialog) provides a dialog box containing a title, three-level text, one confirm button or two buttons (left and right) at the bottom, and an UIImageView in the middle. This component can be used to display message in throttling scenarios.

Sample image



Dependency

See [Quick start](#).

API description

```
public interface OnItemClickListener {
    void onItemClick(int index);
}

/**
 * Obtain an AUIImageDialog instance.
 *
 * @param context The context object.
 * @return Return an AUIImageDialog instance.
 */
public static AUIImageDialog getInstance(Context context)

/**
 * Disable the listener.
 *
 * @param mCloseBtnClickListener
 */
public void setCloseBtnClickListener(View.OnClickListener mCloseBtnClickListener)

/**
 * Set the first-level title text.
 */
public void setTitle(CharSequence title)

/**
 * Set the font size (in sp) of the first-level title.
 *
 * @param size
 */
public void setTitleTextSize(float size)

/**
 * Set the visibility of the first-level title.
 *
 * @param visibility
 */
public void setTitleTextVisibility(int visibility)
}

/**
 * Set the visibility of the second-level title.
 *
 * @param visibility
 */
public void setSubTitleTextVisibility(int visibility)

/**
 * Set the color of the first-level title.
 *
 * @param color
 */
```

```
public void setTitleTextColor(int color)

/**
 * Set the second-level title text.
 *
 * @param title
 */
public void setSubTitle(CharSequence title)

/**
 * Set the font size (in sp) of the second-level title.
 *
 * @param size
 */
public void setSubTitleTextSize(float size)

/**
 * Set the color of the second-level title.
 *
 * @param color
 */
public void setSubTitleTextColor(int color)

/**
 * Set the third-level title text.
 *
 * @param text
 */
public void setThirdTitleText(String text)

/**
 * Set the color of the third-level title.
 *
 * @param color
 */
public void setThirdTitleTextColor(int color)

/**
 * Set the background of ImageView.
 *
 * @param drawable
 */
public void setLogoBackground(Drawable drawable)

/**
 * Set the background of ImageView.
 *
 * @param resid
 */
public void setLogoBackgroundResource(int resid)

/**
 * Set the background color of ImageView.
 *
 */
```

```

    * @param color
    */
    public void setLogoBackgroundColor(int color)

    /**
     * Set the background transparency of the dialog box.
     *
     * @param alpha
     */
    public void setBackgroundTransparency(float alpha)

    /**
     * Indicates whether to return animation.
     */
    public boolean isUsdAnim()

    /**
     * Indicates whether to display animation when a dialog box is displayed or disappears. This
     * parameter is set to true by default.
     *
     * @param usdAnim
     */
    public void setUsdAnim(boolean usdAnim)

    /**
     * Set the visibility of the Close button.
     *
     * @param visibility
     */
    public void setCloseButtonVisibility(int visibility)

    /**
     * Set the text of the Confirm button.
     *
     * @param text
     */
    public void setConfirmBtnText(String text)

    /**
     * Return the Confirm button.
     */
    public Button getConfirmBtn()

    /**
     * Set the Confirm button tapping listener.
     *
     * @param clickListener
     */
    public void setOnConfirmBtnClickListener(View.OnClickListener clickListener)

    /**
     * Display a dialog box without animation.
     */

```

```

    */
    public void showWithoutAnim()

    /**
     * Set the countdown.
     * @param seconds Countdown seconds.
     * @param tickColor
     * @param action
     * @param clickListener
     * @param timerListener
     */
    public void showWithTimer(int seconds, String tickColor, String action, View.OnClickListener clickListener, TimerListener timerListener)

    public void showWithTimer(int seconds, View.OnClickListener clickListener, TimerListener timerListener)

    /**
     * Obtain the default countdown color.
     * @return
     */
    public String getDefaultTimeColorStr()

    /**
     * Dismiss a dialog box without animation.
     */
    public void dismissWithoutAnim()

    @Override
    public void dismiss()

    public boolean isCanceledOnTouch() {
        return canceledOnTouch;
    }

    /**
     * Indicates whether a dialog box is automatically canceled when a user taps the middle image.
     *
     * @param canceledOnTouch
     */
    public void setCanceledOnTouch(boolean canceledOnTouch)

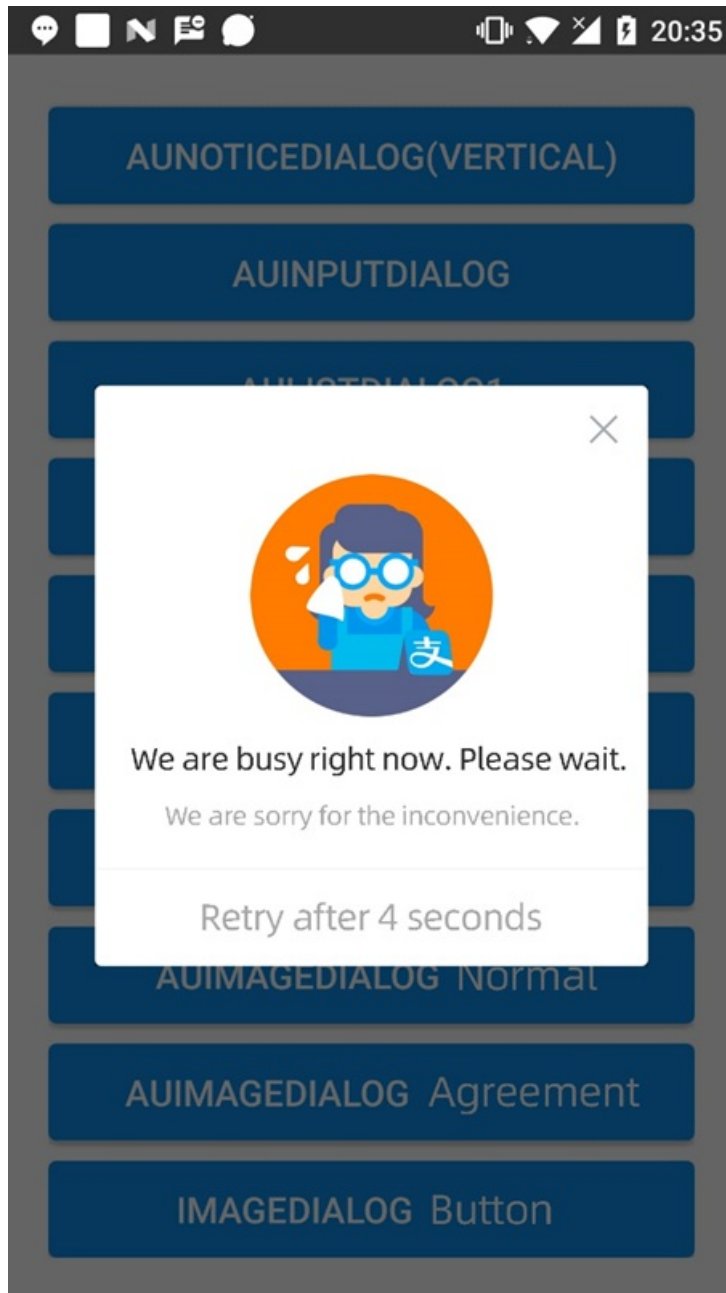
    /**
     * Set the list button.
     * @param buttonListInfo
     * @param listener
     */
    public void setButtonListInfo(List<String> buttonListInfo, OnItemClickListener listener)

    public ImageView getLogoImageView() {
        return bgImageView;
    }
}

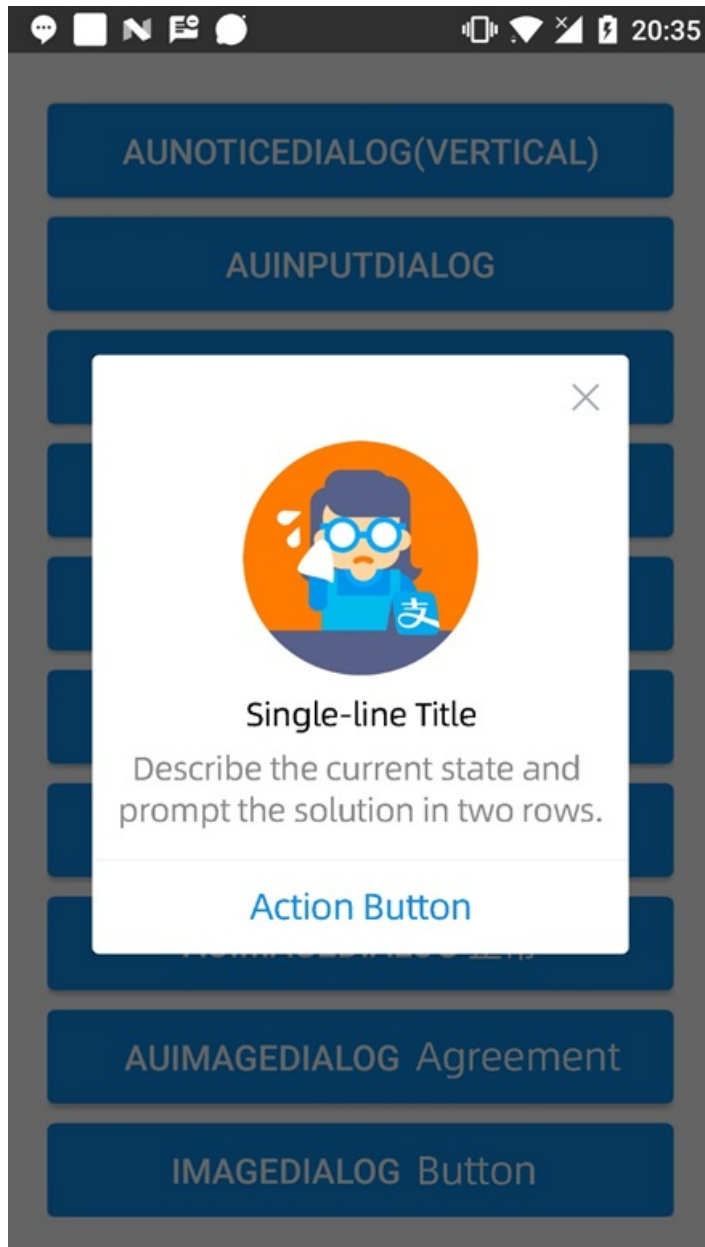
```

```
    },  
  
    public TextView getTitleTextView() {  
        return titleTextView_1;  
    }  
  
    public TextView getSubTitleTextView() {  
        return titleTextView_2;  
    }  
  
    public TextView getThirdTitleTextView() {  
        return titleTextView_3;  
    }  
  
    public ImageView getBottomLine() {  
        return bottomLine;  
    }  
}
```

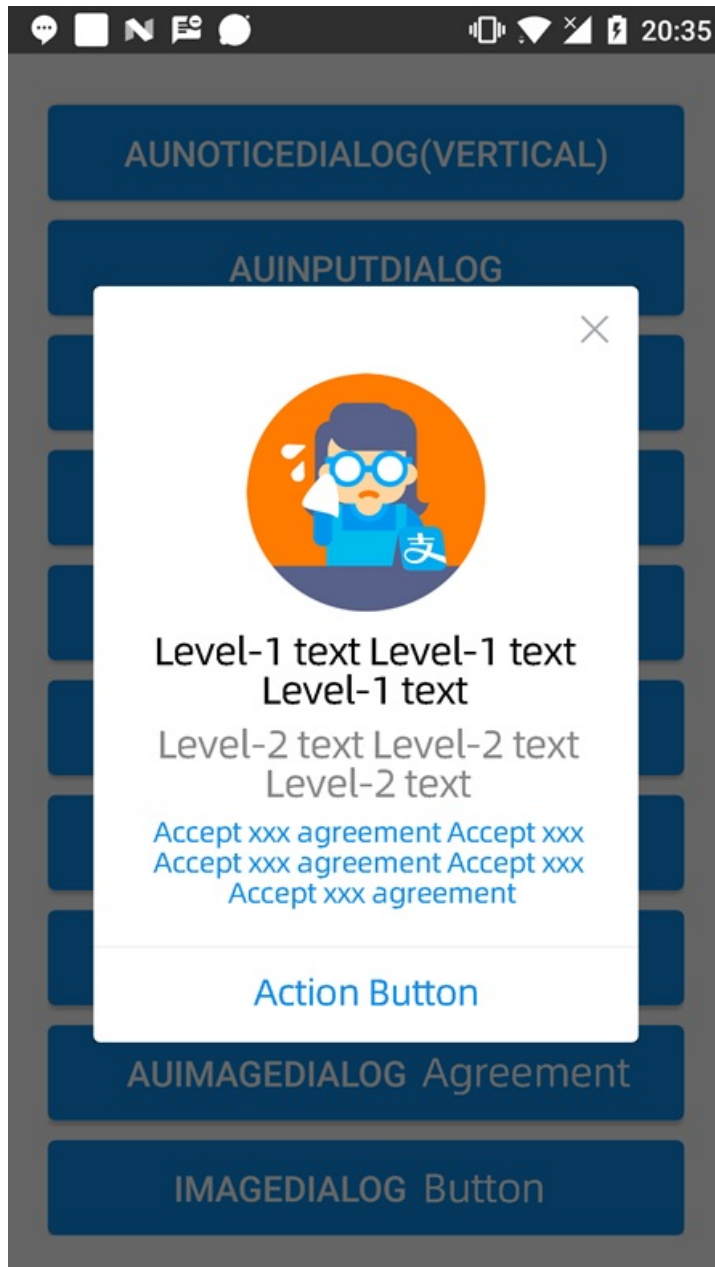
Sample code



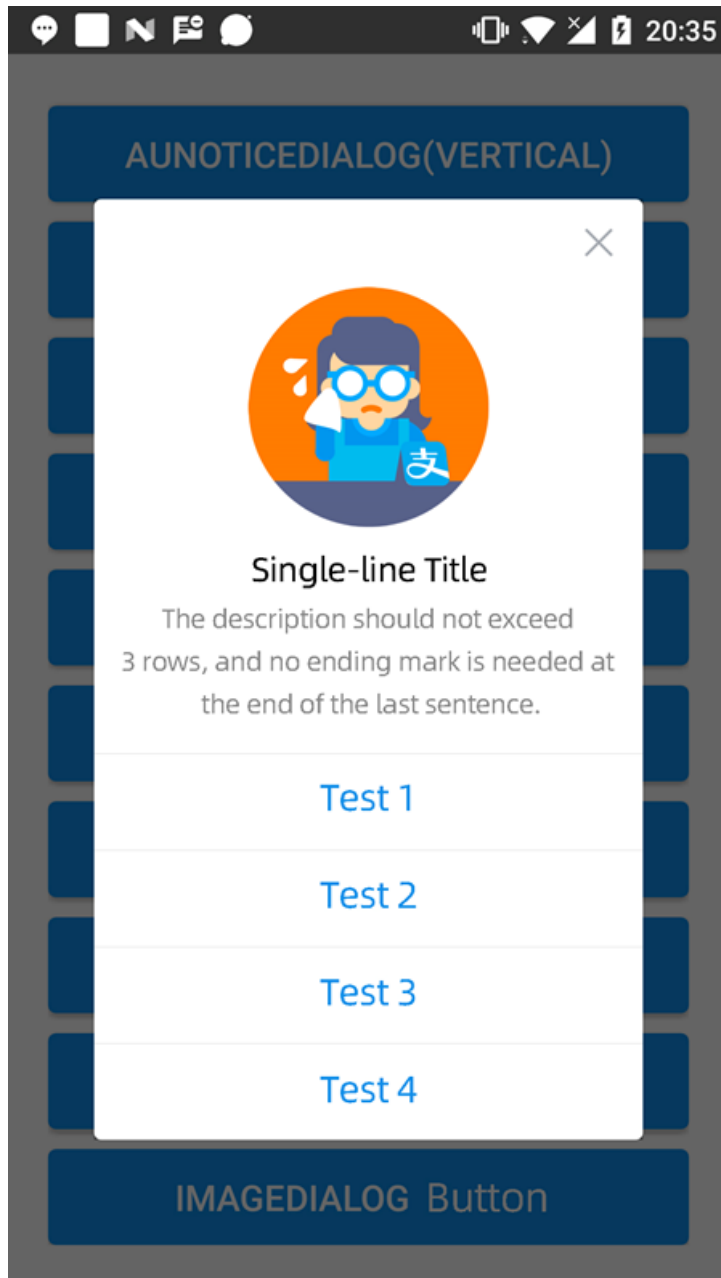
```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.showWithTimer(5, null, null);
```



```
AUIImageDialog dialog = AUIImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("Single-line Title");
dialog.setSubTitle("Describe the current state and prompt the solution in two rows.");
dialog.setConfirmBtnText("Action Button");
dialog.showWithoutAnim();
```



```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("Level-1 textLevel-2 textLevel-2 textLevel-2 textLevel-2 text");
dialog.setSubTitle("Level-2 textLevel-2 textLevel-2 textLevel-2 textLevel-2 textLevel-2 textLevel-2 text");
dialog.setThirdTitleText("Accept xxx agreementAccept xxx agreementAccept xxx agreementAccept xxx agreementAccept xxx agreementAccept xxx agreementAccept xxx agreement");
dialog.setConfirmBtnText("Action Button");
dialog.showWithoutAnim();
```



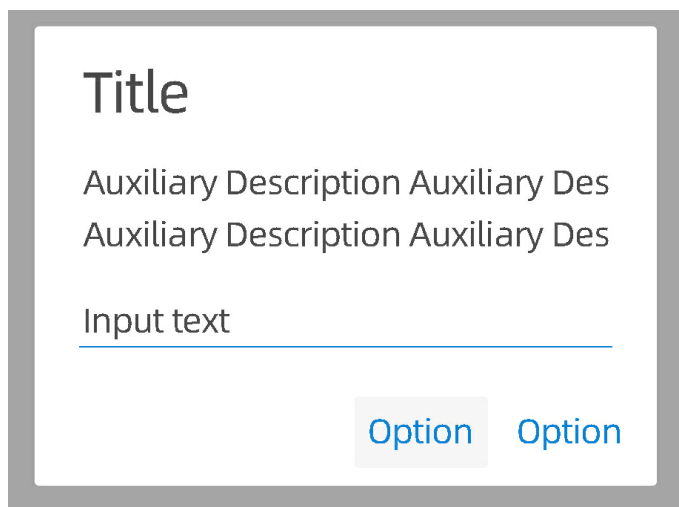
```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setTitle("Single-line Title");
dialog.setSubTitle("The description should not exceed 3 rows, and no ending mark is needed
at the end of the last sentence.");
dialog.setButtonListInfo(getData(), new AUImageDialog.OnItemClickListener() {
    @Override
    public void onItemClick(int index) {

    }
});
dialog.showWithoutAnim();
```

1.4.2.6. Input dialog

AUInputDialog (formerly AInputDialog) provides a dialog box containing a title, body, Confirm and Cancel buttons, and an input box.

Sample image



Dependency

See [Quick start](#).

API description

```
/**
 * Construct AUInputDialog based on Input parameters.
 *
 * @param context The context object.
 * @param title The title.
 * @param msg The message.
 * @param positiveString Confirm button text.
 * @param negativeString Cancel button text.
 * @param isAutoCancel Indicates whether to automatically cancel actions in the area outside the pop-up window.
 */
public AUInputDialog(Context context, String title, String msg, String positiveString,
    String negativeString, boolean isAutoCancel)

/**
 * Obtain the Cancel button.
 */
public Button getCancelBtn();

/**
 * Obtain the Confirm button.
 */
public Button getEnsureBtn();

/**
 * Obtain title TextView.
 */
public TextView getTitle();
```

```
/**
 * Obtain message TextView.
 */
public TextView getMsg();

/**
 * Obtain LinearLayout of the bottom button.
 */
public LinearLayout getBottomLayout();

/**
 * Obtain RelativeLayout at the outermost layer of the pop-up window.
 */
public RelativeLayout getDialogBg();

/**
 * Set the Confirm button listener.
 */
public void setPositiveListener(OnClickPositiveListener listener);

/**
 * Set the Cancel button listener.
 */
public void setNegativeListener(OnClickNegativeListener listener);

/**
 * Obtain EditText of the input box.
 */
public AUEditText getInputContent() {
    return inputContent;
}

/**
 * Starts and display the dialog.
 */
public void show();
```

Sample code

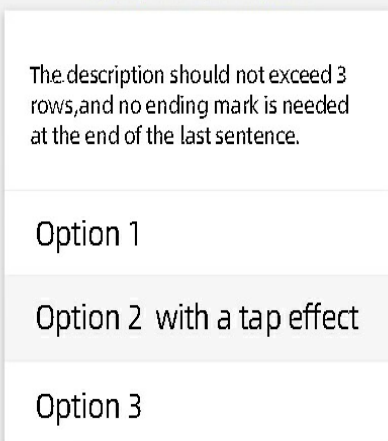
```
AUInputDialog dialog = new AUInputDialog(this, "Title", "Auxiliary Description",
    "OK", "Cancel", true);
dialog.show();
```

1.4.2.7. List dialog

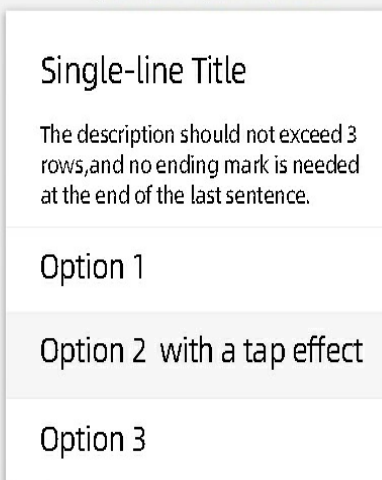
AUListDialog (formerly APListPopDialog) provides a list dialog box containing a title, an option list, a Confirm button, and a Cancel button. Each option is a PopMenuitem with icon, option name, and selected state.

Sample image

Android / Popover/Multiple actions with description



Android / Popover/Multiple actions with description



Dependency

See [Quick start](#).

API description

```
public interface OnItemClickListener {
    void onItemClick(int index);
}

/**
 * Create AUListDialog based on the input list, in which the item contains only text but no
 * image.
 *
 * @param context The context object.
 * @param list The string list with only ItemName instead of any image.
 */
public AUListDialog(Context context, ArrayList<String> list)

/**
 * Create AUListDialog based on the input list.
 *
 * @param list The PopMenuItem list.
 * @param context The context object.
 */
public AUListDialog(ArrayList<PopMenuItem> list, Context context)

/**
 * Create AUListDialog based on the input list.
 *
 * @param title The title.
 * @param list PopMenuItem The object list. Icons are allowed.
 * @param context The context object.
 */
public AUListDialog(String title, ArrayList<PopMenuItem> list, Context context)
```

```

/**
 * Create AUListDialog based on the input list.
 *
 * @param title The title.
 * @param message The message main body.
 * @param list PopMenuItem The object list. Icons are allowed.
 * @param context The context object.
 */
public AUListDialog(String title, String message, ArrayList<PopMenuItem> list, Context context)

/**
 * Create AUListDialog based on the input list.
 *
 * @param title The title.
 * @param list The PopMenuItem list.
 * @param showSelectionState Whether to display the icon in the selected state.
 * @param positiveString Confirm button text.
 * @param positiveListener The Confirm button listener.
 * @param negativeString Cancel button text.
 * @param negativeListener The Cancel button listener.
 * @param context The context object.
 */
public AUListDialog(String title, ArrayList<PopMenuItem> list, boolean showSelectionState,
                    String positiveString, View.OnClickListener positiveListener,
                    String negativeString, View.OnClickListener negativeListener, Context context)

/**
 * Create AUListDialog based on the input list.
 *
 * @param title The title.
 * @param message The message main body.
 * @param list The PopMenuItem list.
 * @param showSelectionState Whether to display the icon in the selected state.
 * @param positiveString Confirm button text.
 * @param positiveListener The Confirm button listener.
 * @param negativeString Cancel button text.
 * @param negativeListener The Cancel button listener.
 * @param context The context object.
 */
public AUListDialog(String title, String message, ArrayList<PopMenuItem> list, boolean showSelectionState,
                    String positiveString, View.OnClickListener positiveListener,
                    String negativeString, View.OnClickListener negativeListener, Context context)

/**
 * Set the list option tapping event listener.
 */
public void setOnItemClickListener(OnItemClickListener listener) {
    this.listener = listener;
}

/**

```

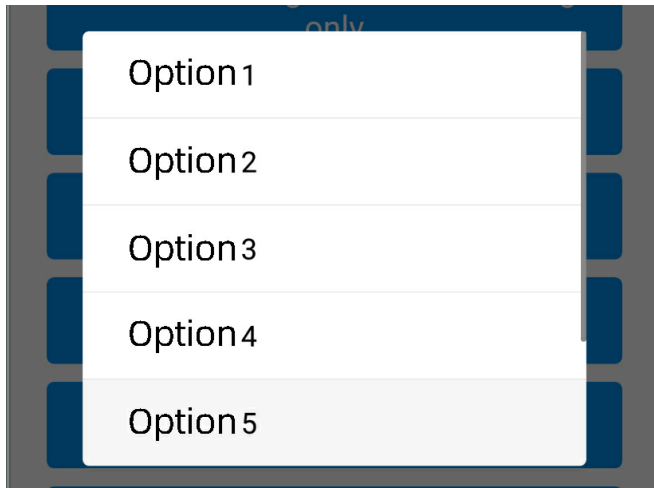
```

    * Dynamic data refreshing API.
    *
    * @param list
    */
    public void updateData(ArrayList<PopMenuItem> list)

```

Sample code

- List dialog with options only



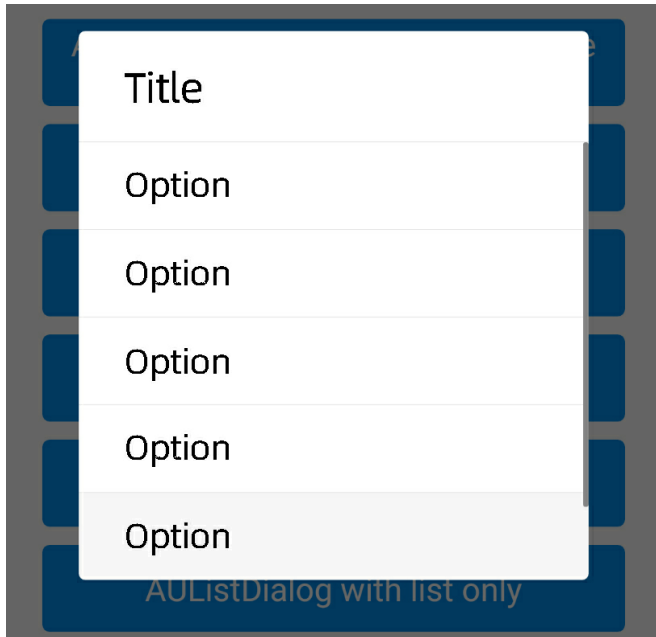
```

new AUListDialog(this, getData(7)).show();

private ArrayList<String> getData(int size){
    ArrayList<String> data = new ArrayList<String>();
    for (int i= 1 ; i<= size; i++){
        data.add("Option"+ String.valueOf(i));
    }
    return data;
}

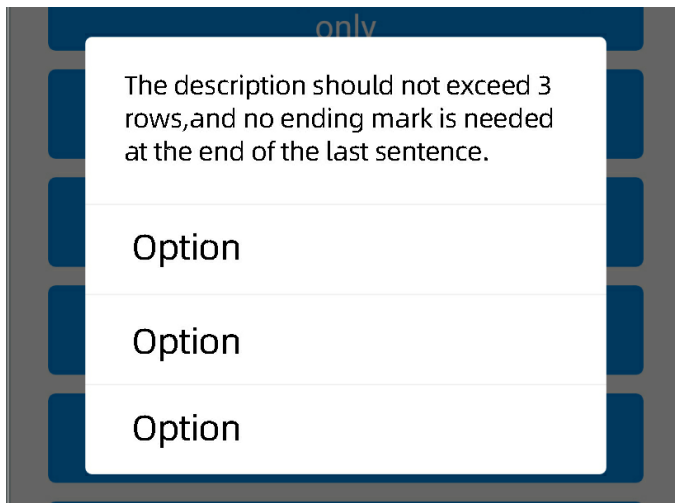
```

- List dialog with a title



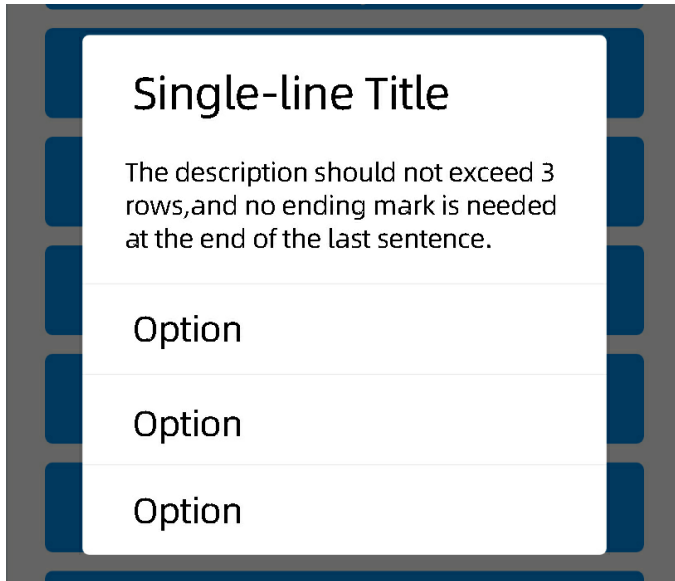
```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
new AUListDialog("Title", items, this).show();
```

- List dialog with description text



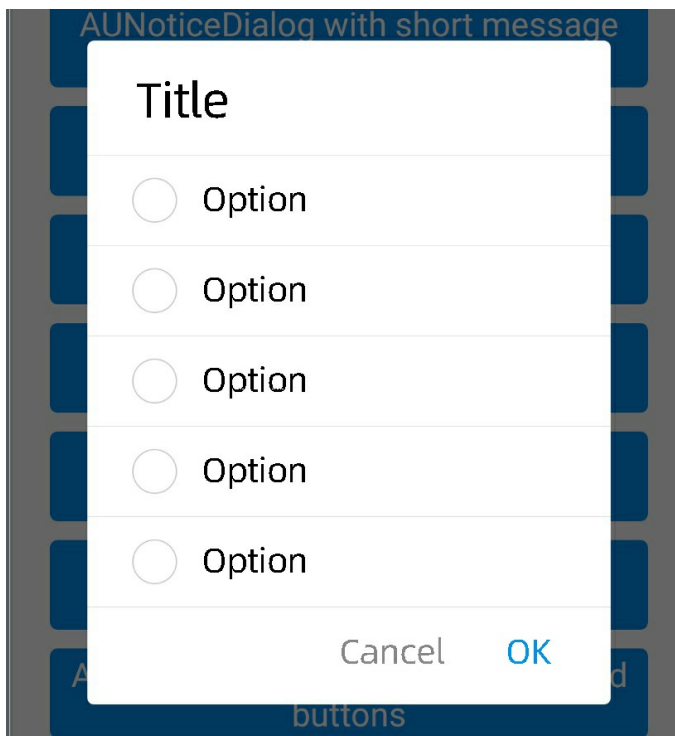
```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
new AUListDialog("", "The description should not exceed 3 rows, and no ending mark is needed at the end of the last sentence.", items, this).show();
```

- List dialog with a title and description text



```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
new AUListDialog("Single-line Title", "The description should not exceed 3 rows, and no ending mark is needed at the end of the last sentence.", items, this).show();
```

- List dialog with checkboxes

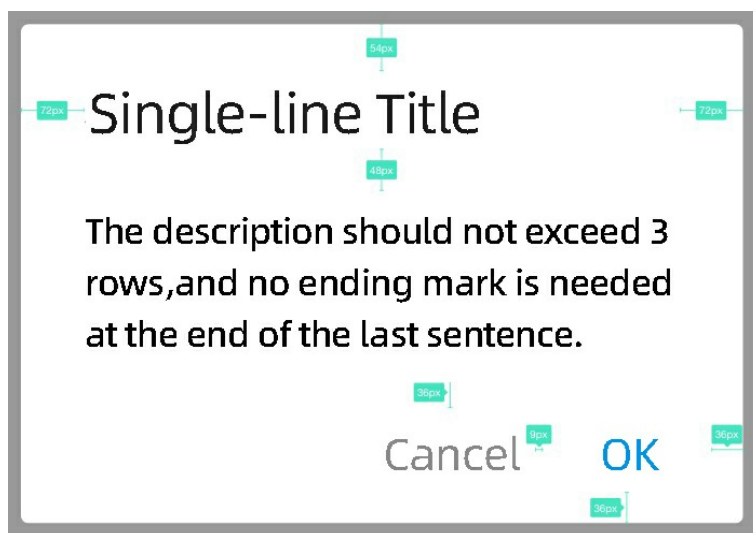


```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
PopMenuItem item = new PopMenuItem("Option", null);
item.setType(AUCheckIcon.STATE_UNCHECKED);
items.add(item);
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
items.add(new PopMenuItem("Option", null));
new AUListDialog("Title", items, true, "OK", null, "Cancel", null, this).show();
```

1.4.2.8. Notice dialog

AUNoticeDialog (formerly APNoticePopDialog) provides a dialog box containing a title, body, a Confirm button, and a Cancel button. It can be used to display common notices.

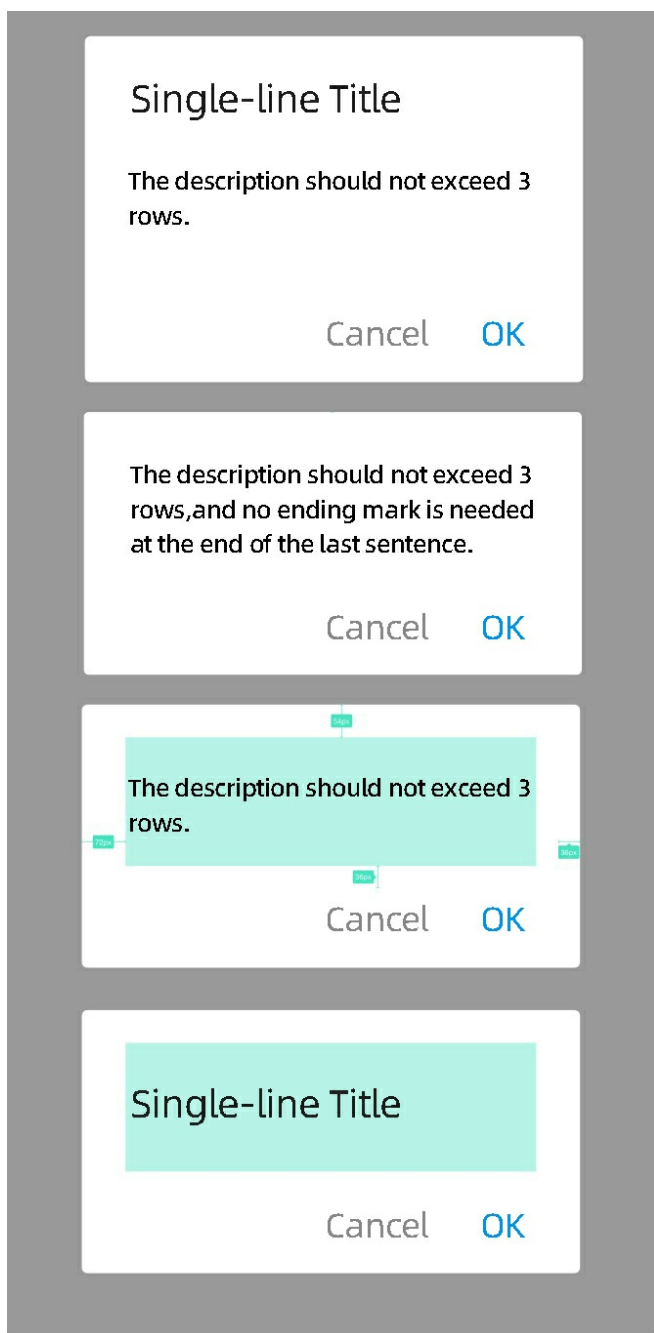
Sample image



```
AUNoticeDialog dialog = new AUNoticeDialog(this, "Single-line Title",
    "The description should not exceed 3 rows, and no ending mark is needed at the end of the last sentence.",
    "OK", "Cancel", true);
dialog.show();
```

Basic rules

- A pop-up window has a minimum height.
- When the window contains only a title or description, it is vertically centered and displayed at the minimum height.



- The text of the **Confirm** and **Cancel** buttons should contain a maximum of four words. Otherwise, the text may not be fully displayed on a phone with a small screen (such as VIVO Y23L).

Dependency

See [Quick start](#).

API

```
public AUNoticeDialog(Context context, CharSequence title, CharSequence msg,
    String positiveString, String negativeString);

public AUNoticeDialog(Context context, CharSequence title, CharSequence msg,
    String positiveString, String negativeString, boolean isAutoCancel) ;
```

```
/**
 * Create AUNoticeDialog based on the Input parameters.
 *
 * @param context The context object.
 * @param title The title.
 * @param msg The message.
 * @param positiveString Confirm button text.
 * @param negativeString Cancel button text.
 * @param isAutoCancel Indicates whether to automatically cancel actions in the area outside the pop-up window.
 */
public AUNoticeDialog(Context context, CharSequence title, CharSequence msg, String positiveString, String negativeString, boolean isAutoCancel);

/**
 * Set the color of the Confirm button text.
 *
 * @ Param c The color value.
 */
public void setPositiveTextColor(ColorStateList c);

/**
 * Set the color of the Cancel button text.
 *
 * @ Param c The color value.
 */
public void setNegativeTextColor(ColorStateList c);

/**
 * Obtain the Cancel button.
 */
public Button getCancelBtn();

/**
 * Obtain the Confirm button.
 */
public Button getEnsureBtn();

/**
 * Obtain title TextView.
 */
public TextView getTitle();

/**
 * Obtain message TextView.
 */
public TextView getMsg();

/**
 * Set the Confirm button tapping listener.
 *
 * @param listener
 */
public void setPositiveListener(OnClickPositiveListener listener);
```

```
/**
 * Set the Cancel button tapping listener.
 *
 * @param listener
 */
public void setNegativeButton(OnClickNegativeButton listener);

/**
 * Obtain RelativeLayout at the outermost layer of the pop-up window.
 */
public RelativeLayout getDialogBg();

/**
 * Start the dialog and display it on screen.
 */
public void show();
```

Sample code

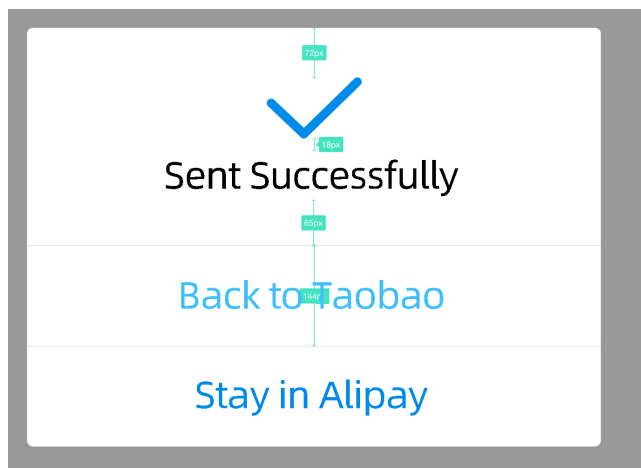
```
//Without a title.
AUNoticeDialog dialog = new AUNoticeDialog(this, "",
    "The description should be within three lines, and punctuation should not be at the rightmost side of a single line.",
    "Confirm", "Cancel", true);
dialog.show();

//Without description.
AUNoticeDialog dialog = new AUNoticeDialog(this, "Single-line title",
    "",
    "Confirm", null, true);
dialog.show();
```

1.4.2.9. Operation result dialog

AUOperationResultDialog provides a pop-up window containing a title and an option list. It is mainly used for displaying social sharing and payment results.

Sample image



Dependency

See [Quick start](#).

API description

```
/**
 * Create AUListDialog based on the input list.
 *
 * @param title      The title.
 * @param list       The PopMenuItem object list. Icons are allowed.
 * @param context    The context object.
 */
public AUOperationResultDialog(Context context, String title, List<String> list)

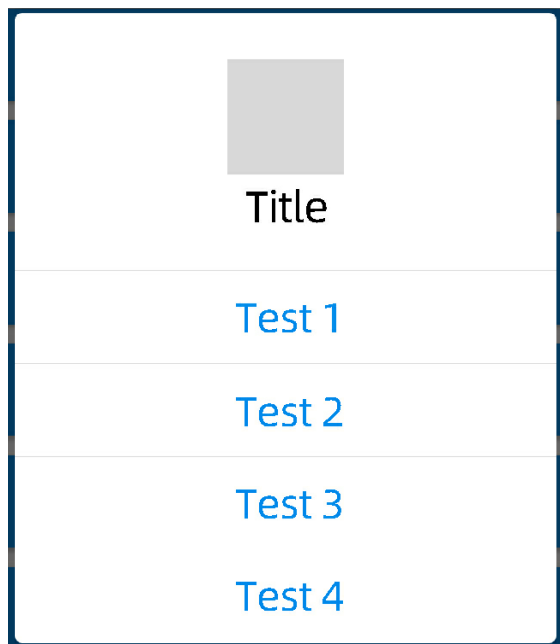
/**
 * Set the list option tapping event listener.
 */
public void setOnItemClickListener(OnItemClickListener listener)

/**
 * Dynamic data refreshing API.
 *
 * @param list
 */
public void updateData(ArrayList<PopMenuItem> list)

/**
 * Obtain imageView.
 * @return
 */
public ImageView getIconView()

/**
 * Set the visibility of the separation line.
 * @param visibility
 */
public void setDivierViewVisibility(int visibility)
```

Sample code



```
public void clickAUOperationResultDialog(View view) {  
    AUOperationResultDialog dialog = new AUOperationResultDialog(this, "Title", getData()  
);  
    dialog.getIconView().setImageDrawable(getResources().getDrawable(R.drawable.image))  
;  
    dialog.show();  
}
```

1.4.2.10. Pop up menu

AUPopMenu provides a pop-up menu displayed when the user taps an option card on the navigation bar, which is essentially a pop-up window.

Different from AUFloatMenu, AUPopMenu has no bottom layer mask but has menu outlines, and all menu texts are centered.

Basic features

- Whether the menu pops upwards or downwards is controlled by the service system.
- The string list passed by the service system shall use the default style, or pass adapter directly.

Sample image

Dependency

See [Quick start](#).

API description

```
/**
 * The data structure. Use the default style.
 * @param context
 * @param itemArrayList
 */
public AUPopMenu(Context context, ArrayList<MessagePopItem> itemArrayList)

/**
 * The adapter structure. Use the custom style.
 * @param context
 * @param listAdapter
 */
public AUPopMenu(Context context, BaseAdapter listAdapter)

/**
 * tip toast down
 * @param anchorView
 */
public void showTipView(View anchorView)

/**
 * tip toast with direction
 * @param anchorView
 * @param isDown
 */
public void showTipView(View anchorView, boolean isDown)

/**
 * Make the window disappear.
 */
public void dismiss()

/**
 * Set the option tapping listener.
 * @param listener
 */
public void setOnItemClickListener(AdapterView.OnItemClickListener listener)
```

Custom properties

No customized property is available. The XML layout is not supported.

Sample code

```
final AUPopMenu popMenu = new AUPopMenu(ScrollTitleBarActivity.this, getItemList());
popMenu.showTipView(view);
popMenu.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {

    }
});
```



```
/**
 * Display the floating layer.
 */
public void show();

/**
 * Make the floating layer disappear.
 */
public void dismiss();

/**
 * Obtain floating layer text view.
 *
 * @return
 */
public AUITextView getTipTextView();

/**
 * Obtain floating layer icon view.
 *
 * @return
 */
public AUIImageView getIconView();
```

Sample code

```
if (!isShowAurecordFloatTip) {
    ((AUIButton) view).setText("Close AurecordFloatTip");
    if (mAurecordFloatTip == null) {
        mAurecordFloatTip = new AurecordFloatTip(this, "Recording");
    }
    mAurecordFloatTip.show();
    isShowAurecordFloatTip = true;
} else {
    ((AUIButton) view).setText("Show AurecordFloatTip");
    if (mAurecordFloatTip != null) {
        mAurecordFloatTip.dismiss();
    }
    isShowAurecordFloatTip = false;
}
```

1.4.2.12. Toast

AUToast is the page pop-up indicator, and the AUProgressDialog is a progress indicator.

When the Toast methods of BaseFragmentActivity and BaseActivity are used, the mPaaS framework modifies the pop-ups in a unified manner.

For activity developed based on BaseFragmentActivity and BaseActivity, the system uses AUToast by default.

Sample images

AUToast

AUProgressDialog

Dependency

See [Quick start](#).

API description

```
/**
 * Instantiate Toast.
 *
 * @param context      The context. Use activities on the current page.
 * @param drawableId   Image resources.
 * @param tipSrcId     The text prompt ID.
 * @param duration     Display duration Toast.Long/Toast.Short.
 * @return Toast
 */
public static Toast makeToast(Context context, int drawableId, int tipSrcId, int duration) {
    CharSequence tipSrc = context.getResources().getText(tipSrcId);
    return makeToast(context, drawableId, tipSrc, duration);
}

/**
 * Create Toast.
 *
 * @param context      The context. Use activities on the current page.
 * @param tipSrcId     The prompt information.
 * @param duration     The duration.
 * @return toast
 */
public static Toast makeToast(Context context, int tipSrcId, int duration) {
    CharSequence tipSrc = context.getResources().getText(tipSrcId);
    return makeToast(context, 0, tipSrc, duration);
}

/**
 * Make a toast that just contains a image view and a text view.
 *
 * @param context      The context. Use activities on the current page.
 * @param drawableId   The image resourceid.
 * @param tipSrc       The text to show. Can be formatted text.
 * @param duration     How long to display the message. Either or
 * @return
 */
public static Toast makeToast(Context context, int drawableId, CharSequence tipSrc, int duration)
```

Code sample

```
// Success.
AUTOast.makeToast (ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_ok,
"Success prompt", Toast.LENGTH_SHORT).show();

// Failure.
AUTOast.makeToast (ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_false, "Failure prompt", Toast.LENGTH_SHORT).show();
}

// Warning.
AUTOast.makeToast (ToastActivity.this, com.alipay.mobile.antui.R.drawable.toast_warning, "Warning prompt", Toast.LENGTH_SHORT).show();
}

// Text.
AUTOast.showToastWithSuper (ToastActivity.this, 0, "The text contains up to 14 words", Toast.LENGTH_SHORT);

// Loading.
AUProgressDialog dialog = new AUProgressDialog(this);
dialog.setMessage("Loading");
dialog.show();
}
```

1.4.3. Input components

1.4.3.1. Amount input box

The AUAmountInputBox component provides an input box for the capital chain, and the number in the input box is in a special digit font. The input box includes two parts: edit box (AUAmountEditText) and notes (AUAmountFootView). AUAmountFootView has two styles, editable input box and text display, which can be used as needed.

In addition, the component provides AUAmountLabelText for special digit font display.

Sample images

Note: The following figure shows the amount input rules.

Dependency

See [Quick start](#).

API description

AUAmountInputBox

```
/**
 * Get the edit box.
 * @return
 */
public AUEditText getEditText()

/**
 * Get the edit box.
 * @return
 */
public AUAmountEditText getEditLayout()

/**
 * Get footView of the capital chain.
 * @return
 */
public AUAmountFootView getFootView()

/**
 * Get the title bar of the output box.
 * @return
 */
public AUTextView getTitleView()

/**
 * Set properties of HeadView.
 * @param style EDIT_STYLE and TEXT_STYLE.
 */
public void setFootStyle(int style)

/**
 * Set the FootView edit box prompt.
 * @param hint
 */
public void setFootHint(String hint)

/**
 * Set the FootView text.
 * @param text
 */
public void setFootText(String text)
```

AUAmountEditText

```

/**
 * Get EditText.
 * @return
 */
public AUEditText getEditText()

/**
 * Get input box information.
 * @return
 */
public Editable getEditTextEditable()

/**
 * Set to show or hide the separation line.
 * @param visible
 */
public void setDividerVisible(boolean visible)

/**
 * Set the prompt.
 * @param hint
 */
public void setHint(String hint)

/**
 * Specify whether to display the Delete button.
 * @param isShow
 */
public void isShowClearIcon(boolean isShow)

/**
 * Add focus listening.
 * @param listener
 */
public void addOnFocusChangeListener(OnFocusChangeListener listener)

/**
 * Bind the external AUNumberKeyboardView ScrollView.
 * @param keyboardView
 * @param scrollView
 */
public void setKeyBoardView(AUNumberKeyboardView keyboardView, ScrollView scrollView)

/**
 * Bind the external AUNumberKeyboardView.
 * @param keyboardView
 */
public void setKeyBoardView(AUNumberKeyboardView keyboardView)

```

Custom properties

Property	Description	Type
footStyle	Type of header view.	editStyle, textStyle
amountTitleText	Edit box title.	String, Reference
amountHintText	Prompt for edit box.	String, Reference

Code sample

General code sample

```
<com.alipay.mobile.antui.amount.AUAmountEditText
    android:id="@+id/edit_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:amountHintText="Available balance 500.00" />

<com.alipay.mobile.antui.amount.AUAmountLabelText
    android:id="@+id/label_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal" />

<com.alipay.mobile.antui.amount.AUAmountInputBox
    android:id="@+id/amount_input_1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    app:amountTitleText="Transfer amount" />

<com.alipay.mobile.antui.amount.AUAmountInputBox
    android:id="@+id/amount_input_2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    app:amountTitleText="Transfer amount"
    app:amountHintText="Available balance 500.00"
    app:footStyle="textStyle" />

AUAmountInputBox inputBox1 = (AUAmountInputBox) findViewById(R.id.amount_input_1);
inputBox1.setFootHint("Add remark");

AUAmountInputBox inputBox2 = (AUAmountInputBox) findViewById(R.id.amount_input_2);
inputBox2.setFootText("Input not allowed");
```

Code sample defining a numeric keypad

```
<?xml version="1.0" encoding="utf-8"?>
<com.alipay.mobile.antui.basic.AULinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res/com.alipay.mobile.antui"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.alipay.mobile.antui.basic.AUScrollView
        android:id="@+id/scroll"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.alipay.mobile.antui.basic.AULinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <com.alipay.mobile.antui.amount.AUAmountInputBox
                android:id="@+id/amount_input_1"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp"
                app:amountTitleText="Transfer amount" />
            </com.alipay.mobile.antui.basic.AULinearLayout>
        </com.alipay.mobile.antui.basic.AUScrollView>

        <com.alipay.mobile.antui.keyboard.AUNumberKeyboardView
            android:id="@+id/keyboard"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:visibility="gone"/>
    </com.alipay.mobile.antui.basic.AULinearLayout>
```

```
// Initialize
keyboardView = (AUNumberKeyboardView) findViewById(R.id.keyboard);
inputBox1 = (AUAmountInputBox) findViewById(R.id.amount_input_1);
ScrollView scrollView = (ScrollView) findViewById(R.id.scroll);

// Bind the keyboard.
inputBox1.getEditLayout().setKeyBoardView(keyboardView, scrollView);
```

1.4.3.2. Input box

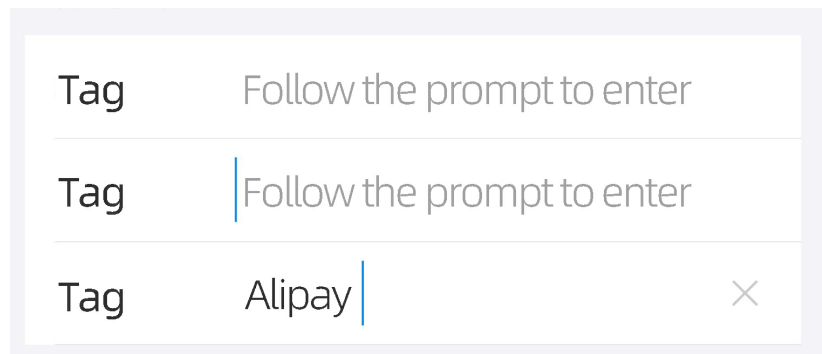
This topic describes the [AUInputBox](#), [AUImageInputBox](#), and [AUTextCodeInputBox](#) input box components provided by mPaaS. The AUImageInputBox and AUTextCodeInputBox components inherit the AUInputBox component.

AUInputBox

The AUIInputBox component contains the following:

- An AUEditText text input box
- A tag name displayed on the left of the input box
- A Delete button that is displayed when an input box gets focus and the content is not empty

Sample image



Dependency

See [Quick start](#).

API description

```
/**
 * Get the string after UBB encoding.
 */
public String getUbbStr()

/**
 * Set the emoji font size, in pixels(px).
 */
public void setEmojiSize(int emojiSize)

/**
 * Specify whether support to emoji.
 */
public void setSupportEmoji(boolean isSupport) {
    this.supportEmoji = isSupport;
}

/**
 * Set a Formatter to format the input.
 * After the setting is completed, the text that has been entered does not take effect immediately. The effects will show upon subsequent text input.
 */
public void setTextFormatter(AUFormatter formatter)

/**
 * Specify whether the input text is bold.
 *
 * @param isBold The value true indicates that the text is bold and the value false indicates that the text is normal.
 */
public void setApprerance(boolean isBold)
```

```

/**
 * Set a special listener to be called when an action is performed on the
 * text view. This will be called when the enter key is pressed, or when an
 * action supplied to the IME is selected by the user. Setting this means
 * that the normal hard key event will not insert a newline into the text
 * view, even if it is multi-line; holding down the ALT modifier will,
 * however, allow the user to insert a newline character.
 */
public void setOnEditorActionListener(OnEditorActionListener l)

/**
 * Adds a TextWatcher to the list of those whose methods are called whenever
 * this TextView's text changes.
 */
public void addTextChangedListener(TextWatcher watcher)

/**
 * The Delete button will be displayed after entering text in the input box, and setting the
 * object received by click event of the delete button here.
 */
public void setCleanButtonListener(View.OnClickListener listener)

/**
 * Set the text content of the input box.
 */
public void setText(CharSequence inputContent)

/**
 * Get the text content. If the text is formatted, the calling party needs to process the text
 * to meet the required format.
 */
public String getInputText()

/**
 * The EditText control for getting the input box.
 */
public AUEditText getInputEdit()

/**
 * Set the tag text.
 * @param title The input tag text.
 */
public void setInputName(String title)

/**
 * The control for getting the input content name (tag name).
 */
public AUTextView getInputName()

/**
 * The font size of the input content name, in pixels(px).

```

```

    */
    public void setInputNameTextSize(float textSize)

    /**
     * Set the font size of the input box, in pixels(px).
     */
    public void setInputTextSize(float textSize)

    /**
     * The text color for the input content.
     */
    public void setInputTextColor(int textColor)

    /**
     * Set the input content type.
     */
    public void setInputType(int inputType)
    /**
     * Set prompt message.
     */
    public void setHint(String hintString)

    /**
     * Set the left tag icon.
     */
    public void setInputImage(Drawable drawable)

    /**
     * Get the left tag icon.
     */
    public AUIImageView getInputImage()

    /**
     * Set the color of the prompt message.
     */
    public void setHintTextColor(int textColor)

    /**
     * Set the maximum input length of the input box.
     *
     * @param maxlength If the value is 0 or smaller, the length is not restricted.
     */
    public void setMaxLength(int maxlength)

    /**
     * The control for getting the Clear button.
     */
    public AUIIconView getClearButton()

    /**
     * Get the setting of whether to show the Clear button.
     */
    public boolean isNeedShowClearButton() {
        return isNeedShowClearButton;
    }

```

```

}

/**
 * Specify whether to show the Clear button. If the parameter is set to false, the Clear button will not be showed in any case.
 */
public void setNeedShowClearButton(boolean isNeedShowClearButton)

/**
 * Set the border style of the control, including upper, middle, lower, and independent.
 * This method is in the AULineGroupItemInterface API.
 * This method is automatically called when the control is used with LineGroupView.
 *
 * @param positionStyle Use the variables defined in AULineGroupItemInterface: TOP, CENTER, BOTTOM, NORMAL, LINE, and NONE.
 */
@Override
public void setItemPositionStyle(int positionStyle)

/**
 * Get the input content type.
 */
public int getInputType()

```

Code sample

Tag	A security keyboard will pop up here.
No more t..	Follow the prompt to enter
Transfer Amount	Follow the prompt to enter
	Transfer Amount Follow the prompt to enter
Follow the prompt to enter	

```
<com.alipay.mobile.antui.input.AUInputBox
    android:id="@+id/safeInputBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="top"
    app:inputName="Tag 1"
    app:inputType="textPassword"
    app:inputHint="This input box pops up a security keyboard"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:inputName="Tag 2"
    app:inputHint="Input as prompted"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="bottom"
    app:inputName="Transfer amount"
    app:inputHint="Input as prompted"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:inputImage="@drawable/image"
    app:inputName="Transfer amount"
    app:inputHint="Input as prompted"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:inputHint="Input as prompted"/>
```

AUImageInputBox

AUImageInputBox inherits AUInputBox and contains the following:

- A IconView on the right to show icons or Unicode
- A TextView on the right

Sample image

Name	Receiver's name	
Card Number	Receiver's debit card number	
Bank	Choose the issuing bank	>
Amount	Enter the transfer amount	Amount Limit

Dependency

See [Quick start](#).

API description

```

/**
 * Set the background of the rightmost functional button.
 * If the background of the Set button is empty, the functional button will not be showed,
 * which is consistent with the AUInputBox function.
 */
public void setLastImgDrawable(Drawable drawable)

/**
 * Set the background of the rightmost functional button.
 * @param unicode
 */
public void setLastImgUnicode(String unicode)

/**
 * Specify whether the rightmost icon is visible.
 * @param visible
 */
public void setLastImgBtnVisible(boolean visible)

/**
 * Set listening on the rightmost functional button.
 */
public void setLastImgClickListener(View.OnClickListener l)

/**
 * Set the rightmost text.
 * @param lastText
 */
public void setLastTextView(String lastText)

/**
 * Get the rightmost TextView.
 *
 * @return Get the rightmost TextView.
 */
public AUTextView getLastTextView()

/**
 * Get the rightmost IconView.
 * @return
 */
public AUIconView getLastImgBtn()

```

Code sample

Name	Receiver's name	
Card Number	Receiver's debit card number	
Bank	Choose the issuing bank	>
Amount	Enter the transfer amount	Amount Limit

```
<com.alipay.mobile.antui.input.AUImageInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:listItemType="top"
    app:inputName="Name"
    app:inputHint="Payee name"
    app:input_rightIconUnicode="@string/iconfont_phone_contact" />

<com.alipay.mobile.antui.input.AUImageInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:inputName="Card number"
    app:inputHint="Savings card number of the payee"/>

<com.alipay.mobile.antui.input.AUImageInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="bottom"
    app:inputName="Bank"
    app:inputHint="Select bank"
    app:input_rightIconDrawable="@drawable/table_arrow" />

<com.alipay.mobile.antui.input.AUImageInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:inputName="Amount"
    app:inputHint="Enter transfer amount"
    app:input_rightText="Amount limit" />
```

AUTextCodeInputBox

AUTextCodeInputBox inherits AUInputBox and contains a text button used for sending the SMS verification code on the right.

Sample image

No more than 6 digits

Dark prompt text

 Send Verification Code

Dependency

See [Quick start](#).

API description

```
/**
 * Set the callback for the Send button clicking event.
 * @param callback When a user clicks the Send button, the OnSendCallback.onSend() method will be called back.
 */
public void setOnSendCallback(OnSendCallback callback)

/**
 * Reset the current time to zero.
 */
public void currentSecond2Zero()

/**
 * Set the current time.
 */
public void setCurrentSecond(int current)

/**
 * Get the current time.
 */
public int getCurrentSecond()

/**
 * Get the Send button.
 */
public AUIButton getSendCodeButton()

/**
 * The release timer for business calling.
 */
public void releaseTimer()

/**
 * Start countdown for the button.
 */
public void scheduleTimer()

public interface SendButtonEnableChecker {
    public boolean checkIsEnabled();
}

/**
 * This method sets the method used to detect whether SendButton is available.
 * If this method detects that the button is unavailable, the button is dimmed when the upd
```

```

    ateSendButtonEnableStatus method is called.
    * Otherwise, the button is available when the updateSendButtonEnableStatus method is called according to the countdown logic.
    * The button is available only when all checks are available, or it will be dimmed.
    */
    public void setSendButtonEnableChecker(SendButtonEnableChecker checker)

    /**
     * Update the SendButton availability state based on the internal state of SendButtonEnableChecker and CheckCodeSendBox.
     * The button is available only when all checks are available, or it will be dimmed.
     */
    public void updateSendButtonEnableStatus()

    /**
     * Get SendResultCallback.
     */
    public SendResultCallback getSendResultCallback()

```

Code sample



• XML

```

<com.alipay.mobile.antui.input.AUTextCodeInputBox
    android:id="@+id/au_textcode_input"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"/>

```

• Java

```
final AUTextCodeInputBox textCodeInputBox = (AUTextCodeInputBox) findViewById(R.id.au_textc
ode_input);
textCodeInputBox.setOnSendCallback(new OnSendCallback() {
    @Override
    public void onSend(final SendResultCallback callback) {
        // The RPC request to the server for a verification code.
        boolean resendSmsRpcSuccess = true;
        if (resendSmsRpcSuccess) {
            // Start countdown when the verification code is successfully sent.
            callback.onSuccess();
            // The verification code is received.
            Toast.makeText(InputActivity.this, "Verification code received: 123456", Toast.
LENGTH_SHORT)
                .show();
            textCodeInputBox.setText("123456");
            Log.d(TAG, "Input verification code: " + textCodeInputBox.getInputText());
        } else {
            // Failed to send the verification code. Enable the Send button again.
            callback.onFail();
        }
    }
});
```

Custom properties

The following table describes custom attribute parameters of the three components.

Property	Description	Type
inputName	The input content name.	String, Reference
inputHint	The prompt content of the input box.	String, Reference
maxLength	The maximum length of the input content.	Integer
inputType	The input content type.	Enum, including textNormal, textNumber, textDecimal, and textPassword
inputImage	The image to the left of the input box.	Reference

Property	Description	Type
listItemType	The background type of an item.	Enum, including top, center, bottom, normal, line, and none
input_rightIconUnicode	The right-side icon.	String, Reference
input_rightIconDrawable	The right-side image.	Reference
input_rightText	The right-side hyperlink text.	String, Reference

1.4.3.3. Numerical keyboard

AUNumberKeyboardView provides numeric keypads in three states.

Use the component

- Use the component independently to show a view, for example, a mini program.
- Bind the component with AUAmountEditText and use them together. The binding tool is AUNumberKeyboardUtil, which is encapsulated in AUAmountEditText. For more information, see [AUAmountInputBox documentation](#).
- Bind the component with common EditText and use them together. The binding tool is AUNumberKeyboardUtil, which needs to be called independently.

Sample images

Dependency

See [Quick start](#).

API description

AUAmountEditText

```
/**
 * Set the keyboard style. The default value is STYLE_POINT.
 * @param style STYLE_POINT, STYLE_X, and STYLE_NONE.
 */
public void setStyle(int style)

/**
 * Set button listening.
 * @param listener
 */
public void setActionClickListener(OnActionClickListener listener)

/**
 * Show status listening.
 * @param windowStateChangeListener
 */
public void setWindowStateChangeListener(WindowStateChangeListener windowStateChangeListener)

/**
 * Show.
 */
public void show()

/**
 * Hide.
 */
public void hide()

/**
 * Return the show state.
 * @return
 */
public boolean isShow()
```

AUNumberKeyBoardUtil

```
/**
 * Pass in EditText and AUNumberKeyboardView.
 * @param context
 * @param editText
 * @param keyboardView
 */
public AUNumberKeyBoardUtil(Context context, EditText editText, AUNumberKeyboardView keyboardView)

/**
 * Set the scroll view.
 * @param view
 */
public void setScrollView(ScrollView view)

/**
 * Show the numeric keypad.
 */
public void showKeyboard()

/**
 * Hide the numeric keypad.
 */
public void hideKeyboard()
```

Code sample

AUAmountEditText

```
AUNumberKeyboardView auNumberKeyboardView = new AUNumberKeyboardView(this, AUNumberKeyboardView.STYLE_POINT, new AUNumberKeyboardView.OnActionClickListener() {
    @Override
    public void onNumClick(View view, CharSequence num) {

    }

    @Override
    public void onDeleteClick(View view) {

    }

    @Override
    public void onConfirmClick(View view) {

    }

    @Override
    public void onCloseClick(View view) {

    }
});
```

AUNumberKeyboardUtil

- XML

```
<com.alipay.mobile.antui.basic.AULinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.alipay.mobile.antui.basic.AUScrollView
        android:id="@+id/scroll"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.alipay.mobile.antui.basic.AULinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <EditText
                android:id="@+id/editText"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="10dp" />
        </com.alipay.mobile.antui.basic.AULinearLayout>
    </com.alipay.mobile.antui.basic.AUScrollView>

    <com.alipay.mobile.antui.keyboard.AUNumberKeyboardView
        android:id="@+id/keyboard"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:visibility="gone"/>
</com.alipay.mobile.antui.basic.AULinearLayout>
```

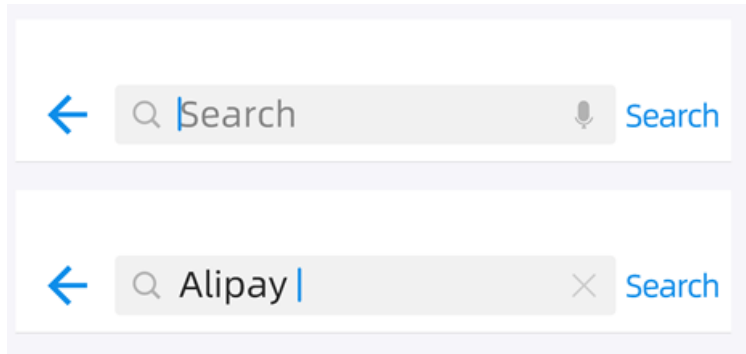
- Java

```
keyBoardUtil = new AUNumberKeyBoardUtil(context, editText, keyboardView);
keyBoardUtil.setScrollView(scrollView);
```

1.4.3.4. Search bar

AUSearchBar (previously known as APSocailSearchBar) provides a search title bar containing a Back button, a search box, and a Search button on the right.

Sample image



Dependency

See [Manage component dependency](#).

API description

```
/**
 * Set the maximum input length.
 */
public void setInputMaxLength(int length);

/**
 * Get the Back button.
 * @return
 */
public AUIconView getBackButton() ;

/**
 * Get the Delete button.
 * @return
 */
public AUIconView getClearButton();

/**
 * Get the search input box.
 * @return
 */
public AUEditText getSearchEditView();

/**
 * Get the Search button.
 * @return
 */
public AUIconView getSearchButton() ;

/**
 * Get the search layout.
 * @return
 */
public AURelativeLayout getSearchRelativeLayout() ;

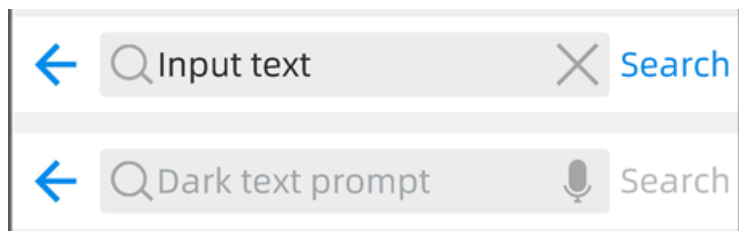
/**
 * Get the voice search button.
 * @return
 */
public AUIconView getVoiceButton();

/**
 * Add editing event listening.
 */
public void setEditChangedListener(TextWatcher watcher)
```

Custom properties

Property	Description	Type
isShowSearchBtn	Whether to show the Search button.	Boolean
isShowVoiceSearch	Whether to show the voice search button.	Boolean
searchEditText	The default text in the search box.	String, Reference
searchEditHint	The default prompt content in the search box.	String, Reference
searchButtonText	The text of the Search button.	String, Reference
inputMaxLength	The maximum length of the search box.	Integer, Reference
hintIconUnicode	The Unicode of the icon on the left side of the edit box.	String, Reference
hintIconDrawable	The resource of the icon on the left side of the edit box.	Reference
backIconUnicode	The Unicode of the Back button.	String, Reference
backIconDrawable	The resource of the Back button.	Reference
editHintColor	The color of the prompt content in the edit box.	Color, Reference
editTextColor	The color of the text in the edit box.	Color, Reference
editIconColor	The color of the icon in the edit box.	Color, Reference

Code sample



```
<com.alipay.mobile.antui.basic.AUSearchBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    app:searchEditText="Input text"
    app:isShowSearchBtn="true"
    app:isShowVoiceSearch="true"/>

<com.alipay.mobile.antui.basic.AUSearchBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    app:searchEditHint="Dark text prompt"
    app:isShowSearchBtn="true"
    app:isShowVoiceSearch="true"/>
```

1.4.3.5. Search input box

AUSearchInputBox (previously known as APSocialTagSearchBar) provides a search title bar containing a search box and a Search button on the right. To use this component, you need to set the height of View.

Sample images

Dependency

See [Quick start](#).

API description

```
/**
 * Set the maximum input length.
 */
public void setInputMaxLength();

/**
 * Get the Delete button.
 * @return
 */
public AUIconView getClearButton();

/**
 * Get the search input box.
 * @return
 */
public AUEditText getSearchEditView();

/**
 * Get the voice search button.
 * @return
 */
public AUIconView getVoiceButton();
```

Custom properties

Property	Description	Type
isShowSearchBtn	Whether to show the Search button.	Boolean
isShowVoiceSearch	Whether to show the voice search button.	Boolean
searchEditText	The default text in the search box.	String, Reference
searchEditHint	The default prompt content in the search box.	String, Reference
inputMaxLength	The maximum length of the search box.	Integer, Reference
hintIconUnicode	The Unicode of the icon on the left side of the edit box.	String, Reference

Property	Description	Type
hintIconDrawable	The resource of the icon on the left side of the edit box.	Reference
editHintColor	The color of the prompt content in the edit box.	Color, Reference
editTextColor	The color of the text in the edit box.	Color, Reference
editIconColor	The color of the icon in the edit box.	Color, Reference

Code sample

XML

```
<com.alipay.mobile.antui.basic.AUSearchInputBox
    android:layout_width="match_parent"
    android:layout_height="52dp"
    android:layout_marginTop="10dp"
    app:searchEditHint="Dimmed text prompt" />
```

```
AUSearchInputBox inputBox = new AUSearchInputBox(this);
ViewGroup.LayoutParams layoutParams = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, 300);
inputBox.setLayoutParams(layoutParams);

layout.addView(inputBox);
```

1.4.4. Item component

1.4.4.1. Auxiliary description component

AUAssistLabelView is a TextView component that displays the auxiliary text.

Sample image

Dependency

See [Quick start](#).

Code sample

```
<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="If the function is disabled, the notification will not show the sender and summary when you receive a message." />
```

```
<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:isHead="true"
android:text="Header" />
```

1.4.4.2. Bank card item component

The `AUBankCardItem` component is used to provide bank card entries such as the bank name, bank logo, and bank account number.

Sample image

Dependency

See [Quick start](#).

API description

```
/**
 * Get the bank name view.
 * @return
 */
public AUEmptyGoneTextView getBankName();

/**
 * Get the bank account view.
 * @return
 */
public AUEmptyGoneTextView getBankNumber();

/**
 * Get the bank logo view.
 * @return
 */
public AUCircleImageView getBankImage();

/**
 * Set the bank name and account number simultaneously.
 * @param bankName
 * @param bankNum
 */
public void setBankInfo(String bankName, String bankNum);
```

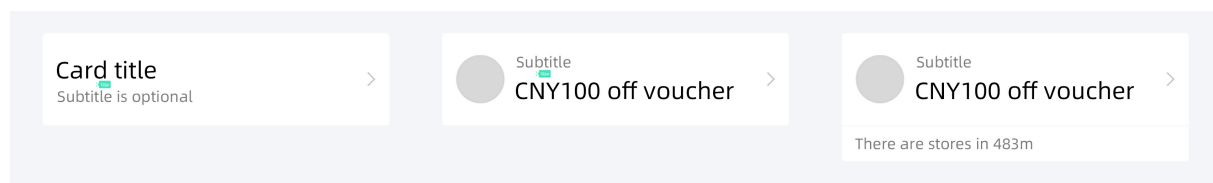
Code sample

```
AUBankCardItem cardItem = new AUBankCardItem(this);
cardItem.setBankInfo("Bank name", "Last 4 digits");
cardItem.getBankImage().setImageResource(R.drawable.image);
```

1.4.4.3. Coupons item component

The coupon entry component is used to display the coupon entry including the coupon icon, title, and auxiliary description.

Sample images



Dependency

See [Quick start](#).

Code sample

```
AUCouponsItem couponsItem1 = new AUCouponsItem(this);
couponsItem1.setCouponsInfo("Subtitle", "CNY100 off voucher", "");
couponsItem1.getCouponsImage().setImageResource(R.drawable.image);

AUCouponsItem couponsItem2 = new AUCouponsItem(this);
couponsItem2.setCouponsInfo("", "CNY100 off voucher", "Subtitle is optional");

AUCouponsItem couponsItem3 = new AUCouponsItem(this);
couponsItem3.setCouponsInfo("Subtitle", "CNY100 off voucher", "");
couponsItem3.setCouponsAssitDes("There are stores in 483m");
couponsItem3.getCouponsImage().setImageResource(R.drawable.image);
```

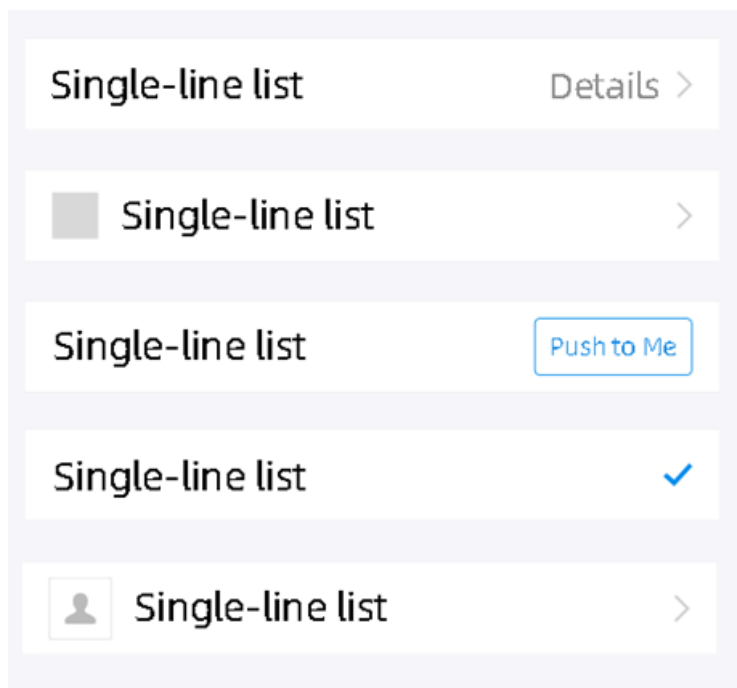
1.4.4.4. List item component

AUListItem is a list item component, including the following controls:

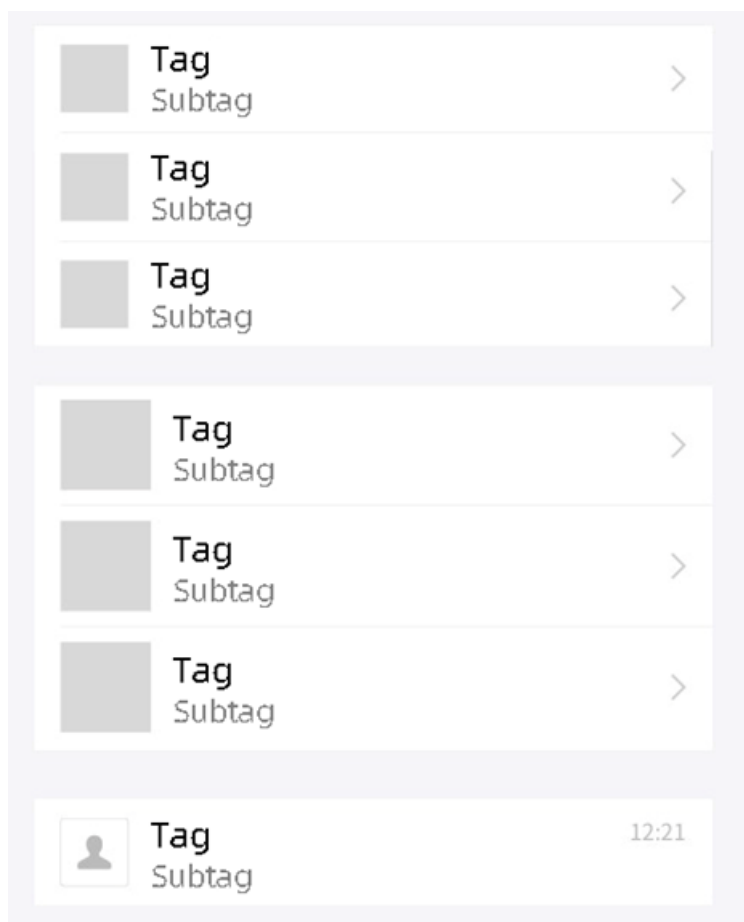
- AUSingleTitleListItem
- AUDoubleTitleListItem
- AUCheckBoxListItem
- AUSwitchListItem
- AUMultiListItem
- AUParallelTitleListItem
- AULineBreakListItem

Sample images

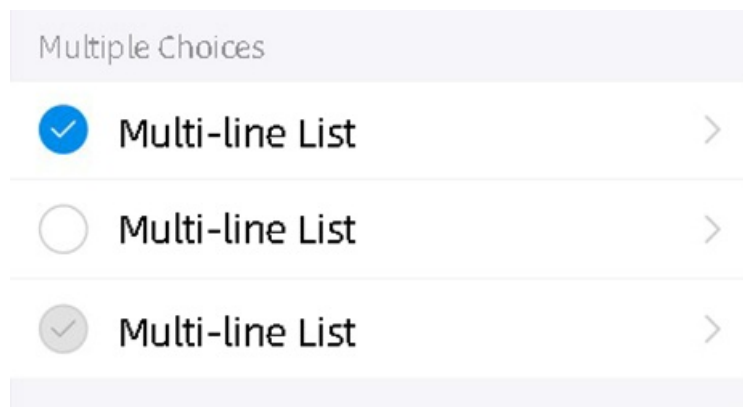
AUSingleTitleListItem



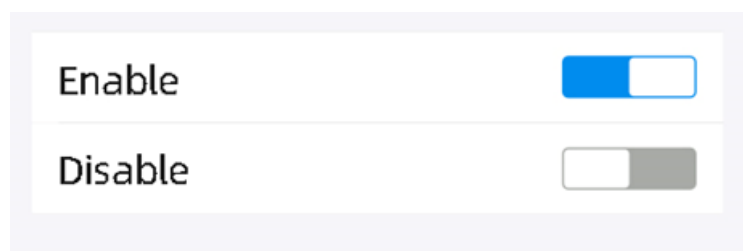
AUDoubleTitleListItem



AUCheckBoxListItem



AUSwitchListItem



Dependency

See [Quick start](#).

API description

Basic APIs

```
/**
 * Set the item type, including upper, medium, and lower.
 *
 * @param positionStyle AULineGroupItemInterface.NORMAL TOP BOTTOM CENTER LINE NONE
 */
public void setItemPositionStyle(int positionStyle)

/**
 * Specify whether the arrow on the right is visible.
 * @param isVisible
 */
public void setArrowVisibility(boolean isVisible)
```

AUParallelListItem

```
/**
 * Set the text in four positions simultaneously.
 * @param leftText
 * @param leftSubText
 * @param rightText
 * @param rightSubText
 */
public void setParallelText(String leftText, String leftSubText, String rightText, String rightSubText)

/**
 * Set the main text on the left.
 * @param leftText
 */
public void setLeftText(String leftText)

/**
 * Set the main text on the right.
 * @param rightText
 */
public void setRightText(String rightText)

/**
 * Set the auxiliary text on the left.
 * @param leftSubText
 */
public void setLeftSubText(String leftSubText)

/**
 * Set the auxiliary text on the right.
 * @param rightSubText
 */
public void setRightSubText(String rightSubText)
```

AULineBreakListItem

```
/**
 * Set the text on the left and right.
 * @param left
 * @param right
 */
public void setText(String left, String right)

/**
 * Get the left TextView.
 * @return
 */
public AUTextView getLeftText()

/**
 * Get the right TextView.
 * @return
 */
public AUTextView getRightText()
```

Public APIs

```
/**
 * Set the size of the icon image.
 */
public void setIconSize(float width, float height)

/**
 * Get the main text on the left.
 * @return
 */
public CharSequence getLeftText()

/**
 * Set the main text on the left.
 * @param text
 */
public void setLeftText(CharSequence text)

/**
 * Set the color of the main text on the left.
 * @param color
 */
public void setLeftTextColor(int color)

/**
 * Get the view of the left-side image.
 * @return
 */
public AURoundImageView getLeftRoundImageView()
public AUImageView getLeftImageView()

/**
 * Set the left-side image.
```

```

    * @param resId
    */
    public void setLeftImage(int resId)

    /**
     * Set the left-side image.
     * @param drawable
     */
    public void setLeftImage(Drawable drawable)

    /**
     * Set visibility when setLeftImage is set in the API.
     * If setLeftImage is called after this API is called, the system will reset the visibility
     *
     * @param vis View.GONE
     */
    public void setLeftImageVisibility(int vis)

    /**
     * Get text information on the left.
     * @return
     */
    public AUITextView getLeftTextView()
}

```

AUSingleTitleListItem

```

/**
 * Set the Select button on the right.
 * @param checked
 */
    public void setItemChecked (boolean checked)

    /**
     * Set text information on the right.
     * @param text
     */
    public void setRightText(CharSequence text)

    /**
     * Set the color of the text on the right.
     * @param color
     */
    public void setRightTextColor(int color)

    /**
     * Set the right-side image.
     */
    public void setRightImage(int resId)
    public void setRightImage(Bitmap bitmap)
    public void setRightImage(Drawable drawable)

```

```
/**
 * Get the view of the text on the right.
 * @return
 */
public AUTextView getRightTextView()

/**
 * Get the view of the right-side image.
 * @return
 */
public AUImageView getRightImageView()

/**
 * Set text information on the right.
 * @param text
 */
public void setRightButtonText(CharSequence text)

/**
 * Get the button.
 * @return
 */
public AUProcessButton getProcessButton()

/**
 * Set the button clicking listener.
 * @param listener
 */
public void setButtonClickListener(OnClickListener listener)

/**
 * Set the style on the right.
 * @param type AUAbsListItem.TEXT_IMAGE AUAbsListItem.BUTTON
 */
public void setRightType(int type)
```

AUCheckBoxListItem

```
/**
 * Get the check icon on the left.
 * @return
 */
public AUCheckIcon getLeftCheckIcon()

/**
 * Set the icon state.
 * @param status AUCheckIcon.STATE_CHECKED|STATE_UNCHECKED|STATE_DISABLED
 */
public void setCheckstatus(int status)

/**
 * Get the check state.
 * @return
 */
public int getIconState()
```

AUSwitchListItem

```
/**
 * Set switch status listening.
 * @param onCheckedChangeListener
 */
public void setOnSwitchListener (CompoundButton.OnCheckedChangeListener onCheckedChangeListener)

/**
 * Get the switch.
 * @return
 */
public AUSwitch getSwitch()

/**
 * Return the switch status.
 * @return Indicates whether the switch is enabled.
 */
public boolean isSwitchOn()

/**
 * Set the switch status.
 * @param status
 */
public void setSwitchStatus(boolean status)

/**
 * Set the state to enable or disable.
 * @param enabled
 */
public void setSwitchEnabled(boolean enabled)
```

AUDoubleTitleListItem

```

/**
 * Set the auxiliary text on the left.
 * @param text
 */
public void setLeftSubText(CharSequence text)

/**
 * Set the text on the right.
 * @param text
 */
public void setRightText(CharSequence text)

/**
 * Set the font and color of the text on the right.
 * @param color
 */
public void setRightTextColor(int color)

/**
 * Get the view of the text on the right.
 * @return
 */
public AUTextView getRightTextView()

/**
 * Get the view of the auxiliary text on the left.
 * @return
 */
public AUTextView getLeftSubTextView()

/**
 * Set text information on the right.
 * @param text
 */
public void setRightButtonText(CharSequence text)

/**
 * Get the button.
 * @return
 */
public AUProcessButton getProcessButton()

/**
 * Set the button clicking listener.
 * @param listener
 */
public void setButtonClickListener(OnClickListener listener)

/**
 * Set the style on the right.
 * @param type AUAbsListItem.TEXT_IMAGE AUAbsListItem.BUTTON
 */
public void setRightType(int type)

```

AUMultiListItem

```
/**
 * Add an extended view to the left.
 * @param view
 */
public void addLeftAssistantView(View view)

/**
 * Set the auxiliary text on the left.
 * @param text
 */
public void setLeftSubText(CharSequence text)

/**
 * Get the subtitle text.
 * @return
 */
public AUEmptyGoneTextView getLeftSubTextView()
```

Custom properties

Property	Description	Type
listItemType	Sets the position style.	Normal/Top/Bottom/Center/Line/None
listLeftText	The text on the left.	String, Reference
listLeftSubText	The auxiliary text on the left.	String, Reference
listLeftTextSize	The font size of the text on the left.	Dimension
listLeftSubTextSize	The font size of the auxiliary text on the left.	Dimension
listLeftTextColor	The color of the text on the left.	Color, Reference
listLeftSubTextColor	The color of the auxiliary text on the left.	Color, Reference
listLeftImage	The left-side icon.	Reference

Property	Description	Type
listLeftImageWidth	The width of the left-side image.	Dimension, Reference
listLeftImageHeight	The height of the left-side image.	Dimension, Reference
listShowArrow	Whether to show the arrow on the right.	Boolean
listArrowType	Arrow orientation	Arrow_right/Arrow_down/Arrow_up
listRightText	The text on the right.	String, Reference
listRightSubText	The auxiliary text on the right.	String, Reference
listRightType	The style on the right.	Text_image/Button
listRightImage	The right-side image.	String, Reference
listShowCheck	The checking image on the right.	Boolean

Note:

- Note: The following [code sample](#)(XML) shows the properties supported in each control.
- If you need to use the background color changing effect, add the `android:clickable="true"` property.
- The height of the control, the width and height of the left image are custom based on business requirements.

Sample code

Imported: `xmlns:app="http://schemas.android.com/apk/res/com.alipay.mobile.antui"`

AUParallelListItem

AUParallelTitleListItem	
Title 1 Content 1	Title 2 Content 2
Title 1 Content 1	Content 2
Title 1 Content 1	Title 2
Title 1 Content 1	Title 2 Content 2 >

```
<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="top"
    app:listLeftText="Title 1"
    app:listLeftSubText="Content 1"
    app:listRightText="Title 2"
    app:listRightSubText="Content 2"
    app:listShowArrow="false" />
```

```
<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftText="Title 1"
    app:listLeftSubText="Content 1"
    app:listRightSubText="Content 2"
    app:listShowArrow="false" />
```

```
<com.alipay.mobile.antui.tablelist.AUParallelTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftText="Title 1"
    app:listLeftSubText="Content 1"
    app:listRightText="Title 2"
    app:listShowArrow="false" />
```

AULineBreakListItem

AULineBreakListItem

Main Info The distance between the left part and the right part should be 30 px. >

The distance between the left part and the right part should be 30 px. Details >

Single-line text Details >

The distance between the left part and the The distance between the left part and... >

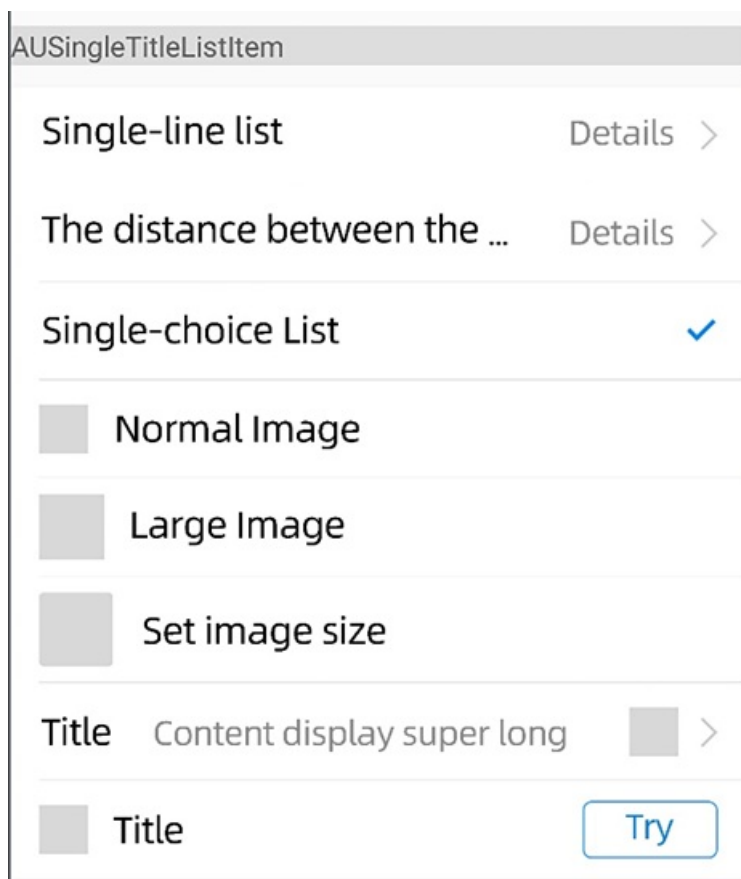
```
<com.alipay.mobile.antui.tablelist.AULineBreakListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="top"
    app:listLeftText="Main info"
    app:listRightText="The distance between the left part and the right part should be 30 px."/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:listLeftText="The distance between the left part and the right part should be 30 px."
    app:listRightText="Details"/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:listLeftText="Single-line text"
    app:listRightText="Details"/>

<com.alipay.mobile.antui.tablelist.AULineBreakListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="bottom"
    app:listLeftText="The distance between the left part and the right part should be 30 px."
    app:listRightText="The distance between the left part and the right part should be 30 px."/>
```

AUSingleTitleListItem



```
<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="top"
    app:listLeftText="Single-line list"
    app:listRightText="Details" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftText="The distance between the left part and the right part should be 30 px
."
    app:listRightText="Details" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:listLeftText="Single-choice list"
    app:listShowCheck="true" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```

        app:listItemType="center"
        app:listLeftImage="@drawable/image"
        app:listLeftText="Normal Image"
        app:listShowArrow="false" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="center"
    app:listLeftImage="@drawable/image"
    app:listLeftImageSizeType="size_large"
    app:listLeftText="Large Image"
    app:listShowArrow="false" />

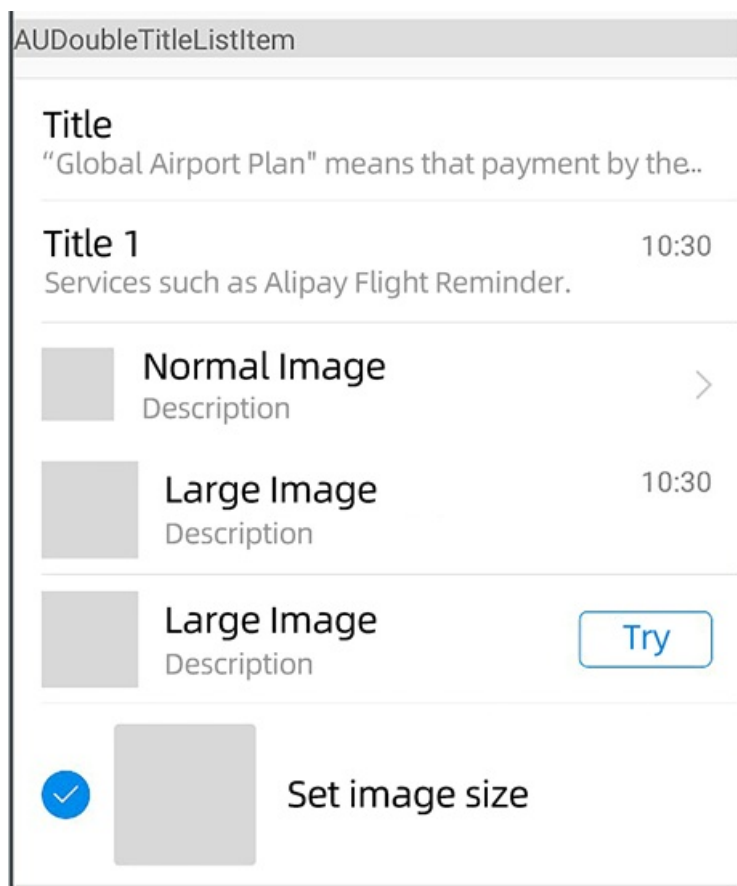
<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:hasRound="true"
    app:listItemType="center"
    app:listLeftImage="@drawable/image"
    app:listLeftImageHeight="36dp"
    app:listLeftImageWidth="36dp"
    app:listLeftText="Set image size"
    app:listShowArrow="false" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftText="Title"
    app:listRightImage="@drawable/image"
    app:listRightText="Content display extra long" />

<com.alipay.mobile.antui.tablelist.AUSingleTitleListItem
    android:id="@+id/button_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="bottom"
    app:listLeftImage="@drawable/image"
    app:listLeftText="Title"
    app:listRightText="Try"
    app:listRightType="button"/>

```

AUDoubleTitleListItem



```
<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftSubText="Services such as Alipay Flight Reminder."
    app:listLeftText="Title"
    app:listRightText="10:30"
    app:listShowArrow="false" />
```

```
<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftImage="@drawable/testapp_icon"
    app:listLeftSubText="Description text"
    app:listLeftText="Normal Image" />
```

```
<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftImage="@drawable/testapp_icon"
    app:listLeftImageSizeType="size_large"
    app:listLeftSubText="Description text"
```

```

        app:listLeftText="Large Image"
        app:listRightText="10:30"
        app:listShowArrow="false" />

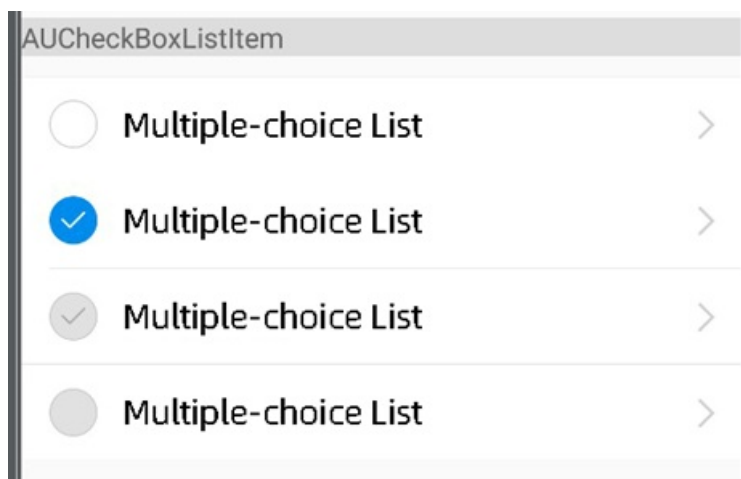
<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="bottom"
    app:listLeftImage="@drawable/testapp_icon"
    app:listLeftImageSizeType="size_multi"
    app:listLeftSubText="Global Airport Plan" means Alipay will give tourists in overseas a
    airports access to services such as Flight Reminder.
    app:listLeftText="Pics & Text list"
    app:listShowArrow="false" />

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="center"
    app:listLeftImage="@drawable/image"
    app:listLeftImageSizeType="size_large"
    app:listLeftSubText="Description"
    app:listLeftText="Large Image"
    app:listRightText="Try"
    app:listRightType="button" />

<com.alipay.mobile.antui.tablelist.AUDoubleTitleListItem
    android:id="@+id/testLitItem"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="20dp"
    android:layout_marginTop="10dp"
    android:clickable="true"
    app:listItemType="normal"
    app:listLeftImage="@drawable/testapp_icon"
    app:listLeftImageHeight="70dp"
    app:listLeftImageWidth="70dp"
    app:listLeftSubText="Click the button to set the type"
    app:listLeftText="Set image size" />

```

AUCheckBoxListItem



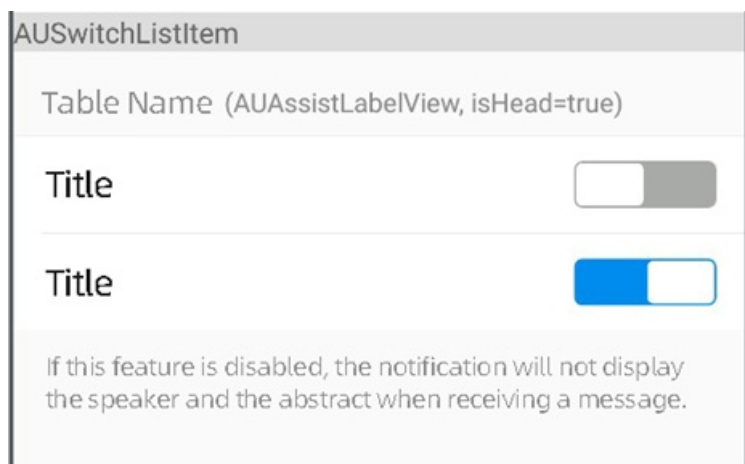
```
<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:listItemType="top"
    app:listLeftText="Multiple-choice List" />

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:checkIconState="checked"
    app:listItemType="center"
    app:listLeftText="Multiple-choice List" />

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:checkIconState="cannot_uncheck"
    app:listItemType="bottom"
    app:listLeftText="Multiple-choice List" />

<com.alipay.mobile.antui.tablelist.AUCheckBoxListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true"
    app:checkIconState="cannot_check"
    app:listItemType="bottom"
    app:listLeftText="Multiple-choice List" />
```

AUSwitchListItem



```
<com.alipay.mobile.antui.tablelist.AUSwitchListItem
    android:layout_width="match_parent"
    android:layout_height="48dp"
    app:listItemType="top"
    app:listLeftText="Title" />
```

```
<com.alipay.mobile.antui.tablelist.AUSwitchListItem
    android:id="@+id/disable_switch_list_item"
    android:layout_width="match_parent"
    android:layout_height="48dp"
    app:listItemType="bottom"
    app:listLeftText="Title" />
```

1.4.5. Result page components

1.4.5.1. Progress page

AUFlowResultView displays a result page with progress. FlowResult is used to represent a node. Each node can be set with different types and auxiliary texts.

Sample image

Dependency

See [Quick start](#).

API description

```
/**
 * Clear all FlowStepView.
 */
public void clearFlows() {
    removeAllViews();
}

/**
 * Set the FlowResult list and generate corresponding FlowStepView
 *
 * @param flowResultList
 */
public void setFlows(List<FlowResult> flowResultList) {
```

FlowResult API

```
/**
 * Construct FlowResult.
 *
 * @param resultStatus The node status. The value is ResultConstant.RESULT_STATUS_ENUM_XX.
 * @param statusIcon The status icon, which indicates ResultStatusIcon enumeration.
 * @param mainInfoText Main text.
 * @param subTitles The sub text list.
 */
public FlowResult(int resultStatus, ResultStatusIcon statusIcon, String mainInfoText,
    List<String> subTitles);

/**
 * Construct FlowResult.
 *
 * @param resultStatus The node status. The value is ResultConstant.RESULT_STATUS_ENUM_XX.
 * @param statusIconId The status icon res id.
 * @param mainInfoText The Main text.
 * @param subTitles The sub text list.
 */
public FlowResult(int resultStatus, int statusIconId, String mainInfoText,
    List<String> subTitles);
```

Sample code

```
AUFlowResultView flowResultView = (AUFlowResultView) findViewById(R.id.flow_result_view);
List<FlowResult> flows = new ArrayList<FlowResult>();
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_OK, ResultStatusIcon.OK,
    "Payment succeeded", Arrays.asList("Auxiliary text", "Auxiliary text")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_OK, ResultStatusIcon.PENDING,
    "Label text", Arrays.asList("Auxiliary text", "Auxiliary text")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_NORMAL, ResultStatusIcon.PENDING,
    "Label text", Arrays.asList("Auxiliary text", "Auxiliary text")));
flowResultView.setFlows(flows);
```

1.4.5.2. Net error page

AUNetErrorView (formerly APFlowTipView) provides a blank page indicating that there is a network exception.

Sample image

Dependency

See [Quick start](#).

API description

```
/**
 * Set the simple mode.
 * @param isSimple
 */
public void setIsSimpleType(boolean isSimple);

/**
 * Set the network exception mode.
 * @param type
 */
public void resetFlowTipType(int type);

/**
 * Set the button properties.
 *
 * @param text
 * @param clickListener
 */
public void setAction(String text, OnClickListener clickListener) ;

/**
 * Set the Cancel button.
 */
public void setNoAction();

/**
 * Set prompt message.
 *
 * @param text
 */
public void setTips(String text) ;

/**
 * Set the auxiliary prompt message.
 * @param text
 */
public void setSubTips(String text) ;

/**
 * Obtain the Operation button.
 * @return
 */
public AUIButton getActionButton();

/**
 * Obtain image view.
 * @return
 */
public AUIImageView getImageView() ;
```

Custom properties

Property	Description
netErrorType	Network exception state. Available values: <code>signalError</code> , <code>empty</code> , <code>warning</code> , <code>overflow</code> .
isSimpleMode	Whether the simple mode is used. Boolean type.

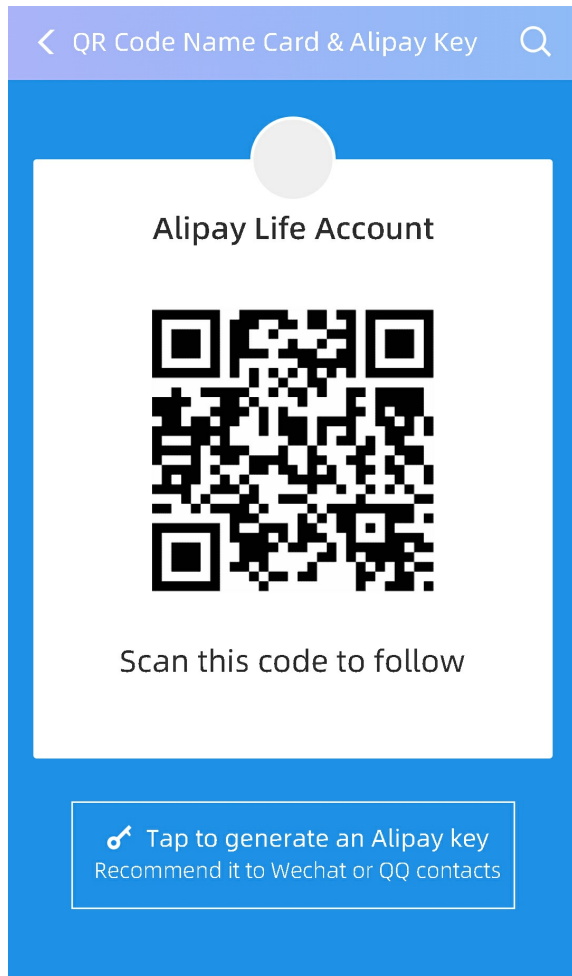
Sample code

```
<com.alipay.mobile.antui.basic.AUNetErrorView
    android:id="@+id/net_error"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:netErrorType="signalError"/>
```

1.4.5.3. QR code page

AUQRCodeView generates a QR code of current page.

Sample image



Dependency

See [Quick start](#).

API description

```
/**
 * Set the avatar name.
 * @param name
 */
public void setAvatarName(CharSequence name)

/**
 * Set QR code information.
 * @param title
 * @param description
 */
public void setCodeInfo(CharSequence title, CharSequence description)

/**
 * Set the QR code title.
 * @param title
 */
public void setCodeTitle(CharSequence title)
```

```

/**
 * Set QR code description.
 * @param description
 */
public void setCodeDescription(CharSequence description)

/**
 * Set the button information with a Zhi Token icon.
 * @param title
 * @param content
 */
public void setButtonInfo(CharSequence title, CharSequence content)

/**
 * Set button information.
 * @param title
 * @param content
 * @param isToken Whether a Zhi Token icon is provided.
 */
public void setButtonInfo(CharSequence title, CharSequence content, boolean isToken)

/**
 * Set the button title.
 * @param title
 */
public void setButtonTitle(CharSequence title)

/**
 * Whether the title includes an icon.
 * @param isToken
 */
public void setButtonToken(boolean isToken)

/**
 * Whether the button is visible.
 * @param isVisible
 */
public void setButtonVisibility(boolean isVisible)

/**
 * Set button content.
 * @param content
 */
public void setButtonContent(CharSequence content)

/**
 * Obtain vatar imageView.
 * @return
 */
public UIImageView getAvatarImage()

/**
 * Obtain avatar name View.
 * @return
 */

```

```

*/
public AUTextView getAvatarName()

/**
 * Obtain QR code imageView.
 * @return
 */
public AUImageView getCodeImage()

/**
 * Obtain the QR code title.
 * @return
 */
public AUTextView getCodeTitle()

/**
 * Obtain the QR code description.
 * @return
 */
public AUEmptyGoneTextView getCodeDescription()

/**
 * Obtain the button.
 * @return
 */
public AULinearLayout getButton()

/**
 * Obtain the button title.
 * @return
 */
public AUTextView getButtonTitle()

/**
 * Obtain button content.
 * @return
 */
public AUEmptyGoneTextView getButtonContent()

```

Code sample

```

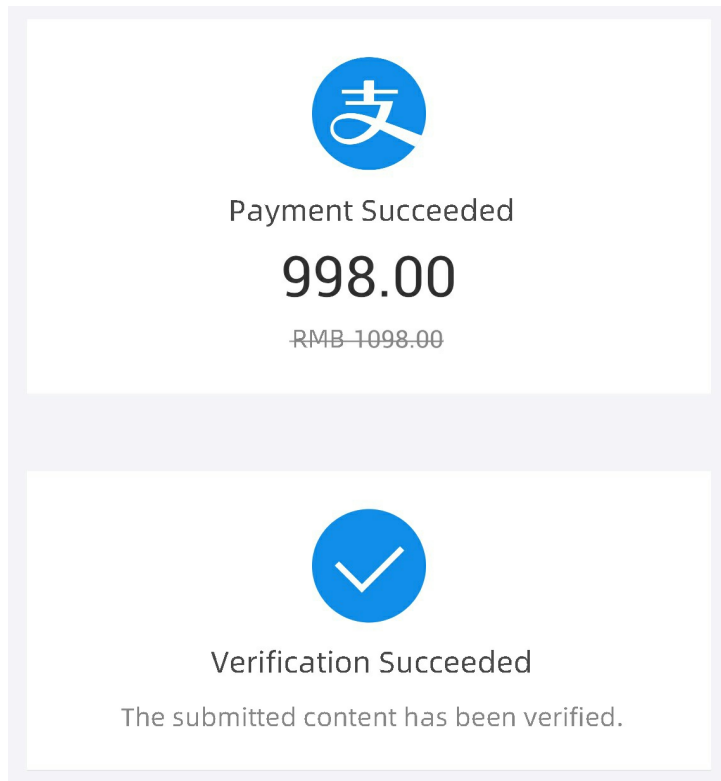
AUQRCodeView codeView = new AUQRCodeView(this);
codeView.setAvaratarName("Lifestyle name");
codeView.setCodeInfo("Scan the QR code with Alipay to join the Social Circle","The QR code will become invalid on November 05, 2017");
codeView.setButtonInfo("Click to generate a Zhi Token","Recommend Lifestyle to WeChat and QQ friends");
codeView.getCodeImage().setImageResource(R.drawable.qr_default);

```

1.4.5.4. Result page

AUResultView provides a result page containing icons and three-level text.

Sample images



Dependency

See [Quick start](#).

API description

```

/**
 * Set the icon.
 *
 * @param iconRes The icon resource ID.
 */
public void setIcon(@DrawableRes int iconRes);

/**
 * Set main title text.
 *
 * @param text The text content.
 */
public void setMainTitleText(CharSequence text);

/**
 * Set sub title text.
 *
 * @param text The text content.
 */
public void setSubTitleText(CharSequence text);

/**
 * Set auxiliary title text.
 *
 * @param text The text content.
 */
public void setThirdTitleText(CharSequence text);

/**
 * Set auxiliary title text with a strikethrough.
 *
 * @param text The text content.
 * @param strikeThrough Whether to display the strikethrough.
 */
public void setThirdTitleText(CharSequence text, boolean strikeThrough);

```

Custom properties

Property	Description	Type
icon	The icon.	Reference
mainTitleText	First-level text.	String, Reference
subTitleText	Second-level text.	String, Reference

Property	Description	Type
thirdTitleText	Third-level text.	String, Reference

Code sample

XML

```
<com.alipay.mobile.antui.status.AUResultView
    android:id="@+id/result_view2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:icon="@drawable/icon_result_alipay"
    app:mainTitleText="Payment succeeded"
    app:subTitleText="998.00"
    app:thirdTitleText="CNY 1098.00"/>
```

1.4.6. Loading component

AULoadingView provides loading pages containing the progress pattern, loading progress, and loading text.

Dependency

See [Quick start](#).

Sample images

API description

AULoadingView

```
/**
 * The constructor.
 * @param context The page context including the antu dependency.
 */
public AULoadingView(Context context)
/**
 * Set the progress.
 * @param curentProgress The progress.
 */
public void setCurrentProgress(int curentProgress)
```

AUPullLoadingView

```
/**
 * The constructor.
 * @param context The page context including the antu dependency.
 */
public AUPullLoadingView(Context context)

    /**
     * Set the progress pattern.
     * @param drawable
     */
    public void setProgressDrawable(Drawable drawable)

    /**
     * Set the bounce pattern.
     * @param mIndicatorUpDrawable
     */
    public void setIndicatorUpDrawable

    /**
     * Set the loading text.
     * @param loadingText
     */
    public void setLoadingText(String loadingText)

    /**
     * Set the dragging text.
     * @param indicatorText
     */
    public void setIndicatorText(String indicatorText)
```

AUDragLoadingView

```
/**
 * The constructor.
 * @param context The page context including the antu dependency.
 */
public AUDragLoadingView(Context context)

    /**
     * Set the loading text.
     * @param text
     */
    public void setLoadingText(CharSequence text)
```

Code sample

AULoadingView

```
private AULoadingView mAULoadingView mAULoadingView = (AULoadingView) findViewById(R.id.loadingView);

private Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        mAULoadingView.setCurrentProgress(mCurrentProgress);
    }
};

protected void onResume() {
    super.onResume();
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (mCurrentProgress < 100) {
                try {
                    Thread.currentThread().sleep(500);
                    mCurrentProgress++;
                    mHandler.sendEmptyMessage(0);
                } catch (Exception e) {
                    Log.e("EmptyPageLoadingActivity", e.getMessage());
                }
            }
        }
    }).start();
}
```

AUPullLoadingView

```
@Override
public AUPullLoadingView getOverview() {

    mAUPullLoadingView2 = (AUPullLoadingView) LayoutInflater.from(getBaseContext())
        .inflate(R.layout.au_framework_pullrefresh_overview, null);
    return mAUPullLoadingView2;
}
```

AUDragLoadingView

```
mAUDragLoadingView = (AUDragLoadingView) findViewById(R.id.dragLoadingView);
findViewById(R.id.modifyLoadingText).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mAUDragLoadingView.setLoadingText("Text after modification...") ;
    }
});
```

1.4.7. Navigation component

1.4.7.1. Carousel component

AUBannerView is used for achieving image carousel.

Sample image

The AUITitleBar control in white is provided by default.

Dependency

See [Quick start](#).

Sample code

```
BannerView bannerView = new BannerView(this, 1000);
layout.addView(bannerView);

List<BannerView.BannerItem> items = new ArrayList<BannerView.BannerItem>();
items.add(new BannerView.BannerItem());
items.add(new BannerView.BannerItem());
items.add(new BannerView.BannerItem());
final List<String> list = new ArrayList<String>();
String color1 = "#111111";
String color2 = "#666666";
String color3 = "#eeeeee";
list.add(color1);
list.add(color2);
list.add(color3);

BannerView.BaseBannerPagerAdapter adapter = new BannerView.BaseBannerPagerAdapter(
bannerView, items) {
    @Override
    public View getView(ViewGroup container, int position) {
        TextView tv = new TextView(CarouselActivity.this);
        tv.setBackgroundColor(Color.parseColor(list.get(position)));
        container.addView(tv);
        return tv;
    }
};

bannerView.setAdapter(adapter);
```

1.4.7.2. List component

AUPinnedSectionListView provides ListView by group. Titles of groups are fixed during sliding.

Note

If you use this control, you need to distinguish data model types. Otherwise, it is an ordinary ListView.

Sample image

Dependency

See [Quick start](#).

Sample code

```
public class PinnedSectionActivity extends Activity{

    private AUPullRefreshView pullRefreshView;
    AUPullLoadingView mAUPullLoadingView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.pinned_layout);
        pullRefreshView = (AUPullRefreshView) findViewById(R.id.pull_refresh);
        final AUPinnedSectionListView pinnedSectionListView = (AUPinnedSectionListView) findViewById(R.id.list_view);

        TextView tv = new TextView(this);
        tv.setText("nihao");
        pinnedSectionListView.addHeaderView(tv);
        pullRefreshView.setRefreshListener(new AUPullRefreshView.RefreshListener() {

            @Override
            public void onRefresh() {

                pullRefreshView.autoRefresh();

                pullRefreshView.postDelayed(new Runnable() {

                    @Override
                    public void run() {
                        pullRefreshView.refreshFinished();
                    }

                }, 1000);
            }

            @Override
            public AUPullLoadingView getOverView() {

                mAUPullLoadingView = (AUPullLoadingView) LayoutInflater.from(getBaseContext())
                    .inflate(com.alipay.mobile.antui.R.layout.au_framework_pullrefresh_overview, null);

                Date date = new Date(1466577757265L);
                SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
                String dateString = formatter.format(date);
                mAUPullLoadingView.setIndicatorText(dateString);
                mAUPullLoadingView.setLoadingText(dateString);
            }
        });
    }
}
```

```

        return mAUPullLoadingView;
    }

    @Override
    public boolean canRefresh() {
        return true;
    }
});

final SimpleAdapter adapter = new SimpleAdapter(this, android.R.layout.simple_list_item_1, android.R.id.text1);

pinnedSectionListView.setAdapter(adapter);

pinnedSectionListView.onFinishLoading(true);
pinnedSectionListView.setOnLoadMoreListener(new AUPinnedSectionListView.OnLoadMoreListener() {
    @Override
    public void onLoadMoreItems() {

        pinnedSectionListView.postDelayed(new Runnable() {
            @Override
            public void run() {
                pinnedSectionListView.onFinishLoading(false);
            }
        }, 3000);
    }
});
}

static class SimpleAdapter extends ArrayAdapter<Item> implements AUPinnedSectionListView.PinnedSectionListAdapter {

    public SimpleAdapter(Context context, int resource, int textViewResourceId) {
        super(context, resource, textViewResourceId);
        generateDataset('A', 'Z', false);
    }

    public void generateDataset(char from, char to, boolean clear) {

        if (clear) clear();

        final int sectionsNumber = to - from + 1;
        prepareSections(sectionsNumber);

        int sectionPosition = 0, listPosition = 0;
        for (char i=0; i<sectionsNumber; i++) {
            Item section = new Item(Item.SECTION, String.valueOf((char)('A' + i)));
            section.sectionPosition = sectionPosition;
            section.listPosition = listPosition++;
            onSectionAdded(section, sectionPosition);
        }
    }
}

```

```

        onSectionAdded(section, sectionPosition);
        add(section);

        final int itemsNumber = (int) Math.abs((Math.cos(2f*Math.PI/3f * sectionsNumber / (i+1f)) * 25f));
        for (int j=0;j<itemsNumber;j++) {
            Item item = new Item(Item.ITEM, section.text.toUpperCase(Locale.ENGLISH) + " - " + j);

            item.sectionPosition = sectionPosition;
            item.listPosition = listPosition++;
            add(item);
        }

        sectionPosition++;
    }
}

protected void prepareSections(int sectionsNumber) { }
protected void onSectionAdded(Item section, int sectionPosition) { }

@Override public View getView(int position, View convertView, ViewGroup parent) {
    TextView view = (TextView) super.getView(position, convertView, parent);
    view.setTextColor(Color.DKGRAY);
    view.setTag("" + position);
    Item item = getItem(position);
    if (item.type == Item.SECTION) {
        //view.setOnClickListener(PinnedSectionListActivity.this);
        view.setBackgroundColor(Color.parseColor("#ff0000"));
    }
    return view;
}

@Override public int getViewTypeCount() {
    return 2;
}

@Override public int getItemViewType(int position) {
    return getItem(position).type;
}

@Override
public boolean isItemViewTypePinned(int viewType) {
    return viewType == Item.SECTION;
}

}

static class Item {

    public static final int ITEM = 0;
    public static final int SECTION = 1;

    public final int type;
    public final String text;

```

```
public int sectionPosition;
public int listPosition;

public Item(int type, String text) {
    this.type = type;
    this.text = text;
}

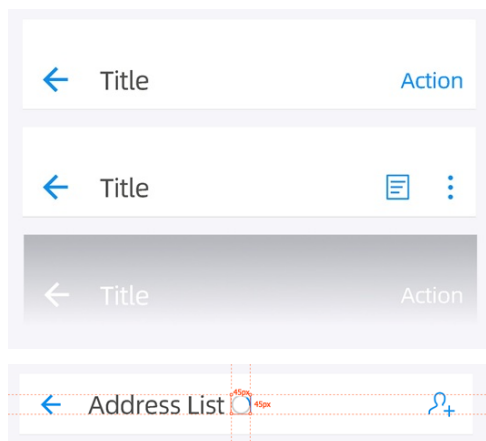
@Override public String toString() {
    return text;
}
}
```

1.4.7.3. Title bar component

AUTitleBar provides a title bar with a Back button, title text, a progress bar, a left button (text and icon), and a right button (text and icon).

Sample image

The AUTitleBar control with white background is provided by default.



Dependency

See [Quick start](#).

API description

```
/**
 * Set the button drawable.
 * @param iconView
 * @param resId
 */
public void setBtnImage(AUIconView iconView, int resId);

/**
 * Set the button size and color.
 * @param iconView
```

```

    * @param size
    * @param color
    */
    public void setIconFont(AUIconView iconView, int size, int color);

    /**
     * Get the Back button.
     * @return
     */
    public AUIconView getBackButton() ;

    /**
     * Get the left button.
     * @return
     */
    public AURelativeLayout getLeftButton() ;

    /**
     * Get the right button.
     * @return
     */
    public AURelativeLayout getRightButton() ;

    /**
     * Get the loading icon.
     * @return
     */
    public AUProgressBar getProgressBar() ;

    /**
     * Get the title text view.
     * @return
     */
    public AUTextView getTitleText() ;

    /**
     * Get the title container.
     * @return
     */
    public AURelativeLayout getTitleContainer() ;

    /**
     * Get the title bar area.
     * @return
     */
    public AURelativeLayout getTitleBarRelative() ;

    @Override
    public void setBackgroundDrawable(Drawable backgroundDrawable) ;

    /**
     * Set the progress circular.
     * @param progressDrawable

```

```

    * @param progressBarDrawable
    */
    public void setProgressBarDrawable(Drawable progressBarDrawable) ;

    /**
     * Set the text and style of the title. If you need to use the default text and style,
     set text, textSize, and textColor to null or 0.
     * @param text
     * @param textSize
     * @param textColor
     */
    public void setTitleText(String text, int textSize, int textColor) ;

    /**
     * Set the title text.
     * @param text
     */
    public void setTitleText(String text) ;

    /**
     * Set the value of resource, size, and color of the Back button. Keep the default value
     if the value is null or 0.
     * @param drawable
     * @param size
     * @param color
     */
    public void setBackBtnInfo(Object drawable, int size, int color);

    /**
     * Set the resource of the Back button.
     * @param drawable
     */
    public void setBackBtnInfo(Object drawable);

    /**
     * Set the resource, size, and color of the left button. Keep the default value if the
     value is null or 0.
     * @param drawable
     * @param size
     * @param color
     * @param isText
     */
    public void setLeftBtnInfo(Object drawable, int size, int color, boolean isText);

    /**
     * Set the resource of the left button.
     * @param drawable
     */
    public void setLeftButtonIcon(Drawable drawable);

    public void setLeftButtonIcon(String unicode);

    public void setLeftButtonText(String text);

    /**

```

```

    * Set the color and size of the left button.
    * @param size
    * @param color
    * @param isText
    */
    public void setLeftButtonFont(int size, int color, boolean isText);

    /**
     * Set the resource, size, and color of the right button. Keep the default value if the
     value is null or 0.
     * @param drawable
     * @param size
     * @param color
     * @param isText
     */
    public void setRightBtnInfo(Object drawable, int size, int color, boolean isText) ;

    /**
     * Set the resource of the right button.
     * @param drawable
     */
    public void setRightButtonIcon(Drawable drawable);

    public void setRightButtonIcon(String unicode);

    public void setRightButtonText(String text);

    /**
     * Set the color and size of the right button.
     * @param size
     * @param color
     * @param isText
     */
    public void setRightButtonFont(int size, int color, boolean isText);

    /**
     * Make the progress bar start rotation.
     */
    public void startProgressBar();

    /**
     * Make the progress bar stop and disappear.
     */
    public void stopProgressBar() ;

    /**
     * Default processing of the gradual change during sliding. Use the default value for totalHeight.
     * @param currentHeight The current height.
     */
    public void handleScrollChange(int currentHeight);

    /**

```

```

    * Default processing of the gradual change during sliding.
    *
    * @param totalHeight The total height.
    * @param currentHeight The current height.
    */
    public void handleScrollChange(int totalHeight, int currentHeight) ;

    /**
     * Set the pattern color (white) with transparent background.
     */
    public void setColorWhiteStyle();

    /**
     * Set the pattern color with white background.
     */
    public void setColorOriginalStyle();

    /**
     * Make the Back button not displayed.
     */
    public void setBackButtonGone();

    /**
     * Add a search box (input is not allowed) used only for visual adjustment.
     * @param search
     */
    public void setTitle2Search(String search);

    /**
     * Convert the search box into a title.
     */
    public void setSearch2Title();

    /**
     * The font color in the search box is set to black and the background is set to white.
     */
    public void setSearchColorOriginalStyle();

    /**
     * The font color in the search box is set to white and the background is set to transparent.
     */
    public void setSearchColorTransStyle();

    /**
     * Add a red dot to the left icon.
     * @param flagView
     */
    public void attachFlagToLeftBtn(AUWidgetMsgFlag flagView);

    /**
     * Add a red dot to the right icon
     * @param flagView

```

```

    */
    public void attachFlagToRightBtn(AUWidgetMsgFlag flagView) ;

    /**
     * Add a red dot to the targetView.
     * @param targetView
     * @param flagView
     */
    public void attachFlagView(AURelativeLayout container, View targetView, AUWidgetMsgFlag
    flagView;

```

Custom properties

Property	Description	Type
backgroundDrawable	The whole background of the title bar.	Reference
backIconColor	The color of the back arrow.	Color, Reference
titleText	Title text.	String, Reference
titleTextSize	The font size of the title.	Dimension, Reference
titleTextColor	The font color of the title.	Color, Reference
leftIconResid	ID of the .png or .jpg image corresponding to the left icon.	Reference
leftIconUnicode	The Unicode character of the left icon.	String, Reference
leftIconColor	The color of the left icon.	Color, Reference
leftIconSize	The size of the left icon.	Dimension, Reference
leftText	Content of the left text.	String, Reference
leftTextColor	The color of the left text.	Color, Reference
leftTextSize	The size of the left text.	Dimension, Reference

Property	Description	Type
rightIconResid	ID of the .png or .jpg image corresponding to the right icon.	Reference
rightIconUnicode	The Unicode character of the right icon.	String, Reference
rightIconColor	The color of the right icon.	Color, Reference
rightIconSize	The size of the right icon.	Dimension, Reference
rightText	Content of the right text.	String, Reference
rightTextColor	The color of the right text.	Color, Reference
rightTextSize	The size of the right text.	Dimension, Reference

Sample code

In the following sample code, the reference path of `alui` is:

```
xmlns:alui="http://schemas.android.com/apk/res/com.alipay.mobile.antui" .
```

Basic usage

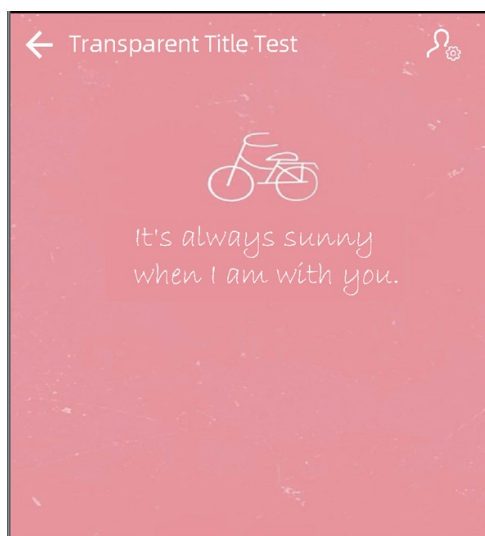
```
<com.alipay.mobile.antui.basic.AUTitleBar
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    alui:alui_titleText="Title"
    alui:alui_titleTextSize="@dimen/AU_TEXTSIZE2"
    alui:alui_titleTextColor="#f64219"
    alui:leftIconUnicode="@string/iconfont_user_setting"
    alui:rightText="Test2"/>
```



Transparent scrolling

```
<com.alipay.mobile.antui.basic.AUTitleBar
    android:id="@+id/title_bar"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    aui:rightIconUnicode="@string/iconfont_user_setting"
    aui:aui_titleText="Transparent Title Test" />
```

```
titleBar.handleScrollChange(testImg.getMeasuredHeight(), 0);
testScroll.setScrollViewListener(new AUIScrollViewListener() {
    @Override
    public void onScrollChanged(ScrollView scrollView, int x, int y, int oldx, int
oldy) {
        titleBar.handleScrollChange(y);
    }
});
```



Title with a red dot

```
<com.alipay.mobile.antui.basic.AUTitleBar
    android:id="@+id/progress_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    aui:aui_titleText="Refreshable Title"
    aui:leftIconUnicode="@string/iconfont_scan"
    aui:rightIconUnicode="@string/iconfont_more"/>
```

```
AUTitleBar progressBar = (AUTitleBar) findViewById(R.id.progress_title);

progressBar.startProgressBar();

WidgetMsgFlag j = new WidgetMsgFlag(this);
j.showMsgFlag();
progressBar.attachFlagToLeftBtn(j);

WidgetMsgFlag i = new WidgetMsgFlag(this);
i.showMsgFlag(12);
progressBar.attachFlagToRightBtn(i);
```



1.4.8. Other component

1.4.8.1. Index component

AUBladeView works with ListView which is sorted alphabetically. On the letter index on the left or right side of the page, when you click or slide to the corresponding letter, the event in the corresponding letter position is triggered. The default index ranges from letter A to letter Z. One or two customized single characters can be added to the top of the index.

Sample image

The two characters above A are customized, as shown in the following figure. The default character ranges from A to Z.



Dependency

See [Quick start](#).

API description

```
/**
 * Set the letter selection listener.
 */
public void setOnItemClickListener(OnItemClickListener listener)

public interface OnItemClickListener {

    /**
     * Set the letter selection listener.
     * @param clickChar The letter clicked or selected.
     */
    void onItemClick(String clickChar);

    /**
     * The finger raising event. You do not need to pay attention to this method if there is no special requirement.
     */
    void onClickUp();
}
```

Custom properties

Property	Description	Type
top1Text	The first customized text character.	Reference
top2Text	The second customized text character.	Reference
showSelectPop	whether to show the floating layer popped-up in the middle during sliding or clicking.	Boolean

Code sample

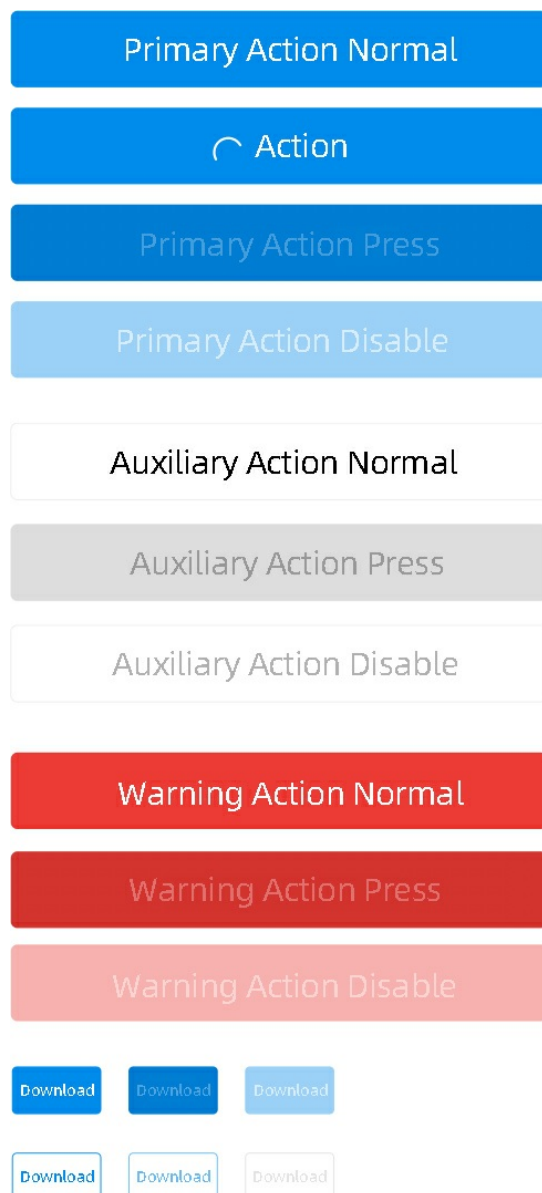
```
<com.alipay.mobile.antui.basic.AUBladeView
    android:layout_width="24dp"
    android:layout_height="wrap_content"
    app:top1Text="⊙"
    app:top2Text="Row"/>
```

As shown in [Sample image](#), top1Text and top2Text can be omitted by default.

1.4.8.2. Button component

AUButton provides buttons of different patterns.

Sample image



Dependency

See [Quick start](#).

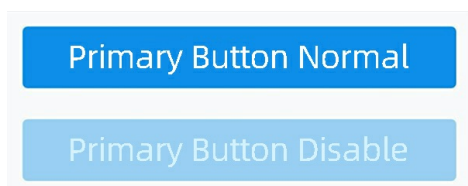
Style API

Property	Description
mainButtonStyle	The main button on a page.
subButtonStyle	The secondary button on a page.

Property	Description
warnButtonStyle	The warning button.
assMainButtonStyle	The auxiliary main button.
assButtonStyle	The auxiliary secondary button.
listButtonStyle	The list button.

Code sample

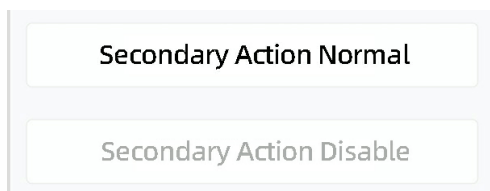
- Main button on a page



```
<com.alipay.mobile.antui.basic.AUButton
  style="@com.alipay.mobile.antui:style/mainButtonStyle"
  android:layout_margin="12dp"
  android:clickable="true"
  android:text="The main button on a page is normal." />

<com.alipay.mobile.antui.basic.AUButton
  style="@com.alipay.mobile.antui:style/mainButtonStyle"
  android:layout_margin="12dp"
  android:enabled="false"
  android:text="The main button on a page is disabled."
  app:dynamicThemeDisable="true" />
```

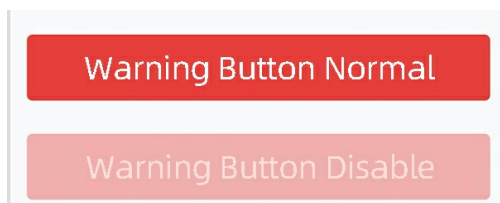
- Secondary button on a page



```
<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/subButtonStyle"
  android:layout_margin="12dp"
  android:clickable="true"
  android:text="The secondary operation on a page is normal." />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/subButtonStyle"
  android:layout_margin="12dp"
  android:enabled="false"/>
```

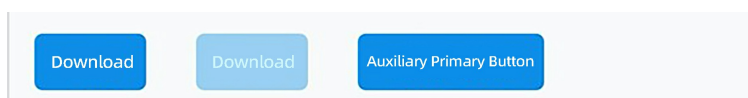
• The warning button.



```
<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/warnButtonStyle"
  android:layout_margin="12dp"
  android:clickable="true"
  android:text="The warning button is normal." />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/warnButtonStyle"
  android:layout_margin="12dp"
  android:enabled="false"
  android:text="The warning button is disabled." />
```

• Auxiliary main button



```
<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assMainButtonStyle"
  android:layout_margin="12dp"
  android:clickable="true"
  android:text="Download" />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assMainButtonStyle"
  android:layout_margin="12dp"
  android:enabled="false"
  android:text="Download" />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assMainButtonStyle"
  android:layout_margin="12dp"
  android:text="Auxiliary Primary button" />
```

• Auxiliary secondary button

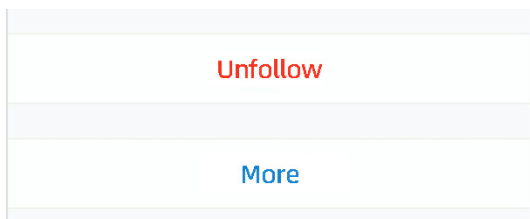


```
<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assButtonStyle"
  android:layout_margin="12dp"
  android:clickable="true"
  android:text="Download" />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assButtonStyle"
  android:layout_margin="12dp"
  android:enabled="false"
  android:text="Download" />

<com.alipay.mobile.antui.basic.AUIButton
  style="@com.alipay.mobile.antui:style/assButtonStyle"
  android:layout_margin="12dp"
  android:text="Auxiliary button" />
```

• List button



```
<com.alipay.mobile.antui.basic.AUIButton
    style="@com.alipay.mobile.antui:style/listButtonStyle"
    android:layout_marginTop="12dp"
    android:layout_marginBottom="12dp"
    android:clickable="true"
    android:text="Unfollow" />

<com.alipay.mobile.antui.basic.AUIButton
    style="@com.alipay.mobile.antui:style/listButtonStyle"
    android:layout_marginTop="12dp"
    android:layout_marginBottom="12dp"
    android:clickable="true"
    android:textColor="@com.alipay.mobile.antui:color/AU_COLOR_LINK"
    android:text="More services" />
```

1.4.8.3. Operation bar component

AUCardOptionView is a combined view providing functions such as giving likes, commenting, and rewarding. It is inherited from AULinearLayout and supporting the access of the XML layout.

Sample image

Dependency

See [Quick start](#).

API description

```
/**
 * Set the information for the entire view.
 * @param itemArrayList
 * @param textVisible
 */
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, boolean textVisible)

/**
 * Set the information for the entire view.
 * @param itemArrayList
 * @param textType = CardOptionView.TEXT_NOT_CHANGE Text is constantly displayed without switching to digits.
 */
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, String textType)

/**
 * Set the information for the entire view.
 * @param itemArrayList
 */
public void setViewInfo(ArrayList<CardOptionItem> itemArrayList)

/**
 * Set the information for the entire view.
 * @param itemArrayList
 * @param height
```

```

        * @param textVisible
        */
        public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height, boolean textVisible)

        /**
         * Set the information for the entire view.
         * @param itemArrayList
         * @param height
         */
        public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height)

        /**
         * Increase the sub view count.
         * @param childView
         */
        public void unitIncrease(View childView)

        /**
         * Decrease the sub view count.
         * @param childView
         */
        public void unitDecrease(View childView)

        /**
         * Obtain the count.
         * @param position
         * @return
         */
        public int getCount(int position)

        /**
         * Return type view.
         * @param type
         * @return
         */
        public View getChildView(String type)

        /**
         * Set listening.
         * @param cardOptionListner
         */
        public void setCardOptionListner(CardOptionClickListener cardOptionListner) {
            this.mListner = cardOptionListner;
        }
    }

```

Custom properties

It is a common ViewGroup without new custom properties.

Code sample

```
AUCardOptionView.CardOptionItem optionItem1 = new AUCardOptionView.CardOptionItem();
optionItem1.type = AUCardOptionView.TYPE_PRAISE;
optionItem1.hasClicked = false;

AUCardOptionView.CardOptionItem optionItem2 = new AUCardOptionView.CardOptionItem();
optionItem2.type = AUCardOptionView.TYPE_REWARD;
optionItem2.hasClicked = false;

AUCardOptionView.CardOptionItem optionItem3 = new AUCardOptionView.CardOptionItem();
optionItem3.type = AUCardOptionView.TYPE_COMMENT;
optionItem3.hasClicked = false;

ArrayList<AUCardOptionView.CardOptionItem> optionItems = new ArrayList<AUCardOptionView
.CardOptionItem>();
optionItems.add(optionItem1);
optionItems.add(optionItem2);
optionItems.add(optionItem3);
mAUCardOptionView.setViewInfo(optionItems, AUCardOptionView.TEXT_NOT_CHANGE);
mAUCardOptionView.setCardOptionListner(new AUCardOptionView.CardOptionClickListener() {
    @Override
    public void onCardOptionClick(View v, AUCardOptionView.CardOptionItem optionItem, int position) {
        mAUCardOptionView.unitIncrease(v);
    }
});
```

1.4.8.4. Check icon component

AUCheckIcon is used to implement the IconView of a select box.

Sample image

Dependency

See [Quick start](#).

API description

```
/** Selected.*/
public static final int STATE_CHECKED = 0x01;
/** Deselected.*/
public static final int STATE_UNCHECKED = 0x02;
/** Deselection unavailable.*/
public static final int STATE_CANNOT_UNCHECKED = 0x03;
/** Selection unavailable.*/
public static final int STATE_CANNOT_CHECKED = 0x04;

/**
 * Set the checkIcon state.
 * @param state
 */
public void setIconState(int state);

/**
 * Get the checkIcon state.
 * @return
 */
public int getIconState() ;
```

1.4.8.5. Icon component

AUIconView is an iconfont vector graphic control which implements the functions of TextView and ImageView simultaneously.

The iconfont control (can be used as TextView) actually defines special Unicode characters to map a type of images and font through the TTF font file of TextView. That is, the function of iconfont is equivalent to the effect of loading a font that maps multiple images, in which each image has a Unicode character.

Each iconfont set is actually a TTF font file. Therefore, multiple TTF font files can be loaded. Each TTF font file has a name. The default TTF font file name of the AntUI is auiconfont.

Sample image

Dependency

For how to add dependency, follow the relevant instructions in [Quick Start](#).

Icon resources

Resource ID	Corresponding name
com.alipay.mobile.antui.R.string.iconfont_more	More
com.alipay.mobile.antui.R.string.iconfont_cancel	Cancel
com.alipay.mobile.antui.R.string.iconfont_voice	Voice

Resource ID	Corresponding name
com.alipay.mobile.antui.R.string.iconfont_collect_money	Collect money
com.alipay.mobile.antui.R.string.iconfont_back	Back
com.alipay.mobile.antui.R.string.iconfont_user_setting	User settings
com.alipay.mobile.antui.R.string.iconfont_user	User
com.alipay.mobile.antui.R.string.iconfont_add	Add
com.alipay.mobile.antui.R.string.iconfont_praise	Like
com.alipay.mobile.antui.R.string.iconfont_map	Map
com.alipay.mobile.antui.R.string.iconfont_checked	Select
com.alipay.mobile.antui.R.string.iconfont_notice	Announcement
com.alipay.mobile.antui.R.string.iconfont_add_user	Add User
com.alipay.mobile.antui.R.string.iconfont_comment	Comment
com.alipay.mobile.antui.R.string.iconfont_selected	Select
com.alipay.mobile.antui.R.string.iconfont_bill	Bill
com.alipay.mobile.antui.R.string.iconfont_pulldown	Pull down
com.alipay.mobile.antui.R.string.iconfont_scan	Scan
com.alipay.mobile.antui.R.string.iconfont_list	List

Resource ID	Corresponding name
com.alipay.mobile.antui.R.string.iconfont_delete	Delete
com.alipay.mobile.antui.R.string.iconfont_share	Share
com.alipay.mobile.antui.R.string.iconfont_search	Search
com.alipay.mobile.antui.R.string.iconfont_complain	Complain
com.alipay.mobile.antui.R.string.iconfont_qrcode	QR code
com.alipay.mobile.antui.R.string.iconfont_unchecked	Deselect
com.alipay.mobile.antui.R.string.iconfont_right_arrow	Right arrow
com.alipay.mobile.antui.R.string.iconfont_help	Help
com.alipay.mobile.antui.R.string.iconfont_group_chat	Group chat
com.alipay.mobile.antui.R.string.iconfont_contacts	Contact
com.alipay.mobile.antui.R.string.iconfont_setting	Settings
com.alipay.mobile.antui.R.string.iconfont_phone_book	Address book
com.alipay.mobile.antui.R.string.iconfont_phone_contact	Contacts

API description

```
/**
 * Set the image resource ID.
 * @param resId
 * @return
 */
@Override
public AUIIconView setImageResource(int resId) {
```

```

public AUIconView setImageResource(int resId) {
    if (resId == 0) {
        return this;
    }
    clearView();
    initImageView();
    imageView.setImageResource(resId);
    this.addView(imageView);
    return this;
}

/**
 * Set the image resource drawable.
 * @param drawable
 * @return
 */
@Override
public IconfontInterface setImageDrawable(Drawable drawable)

/**
 * Set the iconfont color.
 * @param color
 * @return
 */
public AUIconView setIconfontColor(int color)

/**
 * Set the ColorStateList for the iconfont color.
 * @param color
 * @return
 */
public AUIconView setIconfontColorStates(ColorStateList color)

/**
 * Set the view size, in the unit of px.
 *
 * @param size
 */
public AUIconView setIconfontSize(float size)

/**
 * Set the iconfont resource or text of the view.
 * @param text
 * @return
 */
@Override
public AUIconView setIconfontUnicode(String text)

```

Sample code

- Set the icon information:

```
AUIconView iconView = (AUIconView) convertView.findViewById(R.id.icon_view);
iconView.setIconfontUnicode(iconUnicode);

//For example
//iconView.setIconfontUnicode(getResources().getString(com.alipay.mobile.antui.R.string.iconfont_phone_contact));
```

- Set the icon color:

```
<com.alipay.mobile.antui.iconfont.AUIconView
    android:id="@+id/icon_view"
    android:layout_width="@dimen/size"
    android:layout_height="@dimen/size"
    app:iconfontColor="@com.alipay.mobile.antui:color/AU_COLOR_APP_GREEN"
    app:iconfontUnicode="@com.alipay.mobile.antui:string/iconfont_back"/>

//or:
iconView.setIconfontColor(color)
iconView.setIconfontColorStates(colorStateList)
```

1.4.8.6. Refresh component

AURefreshListView is a ListView that supports pull-down refresh and pull-up-to-load.

Dependency

See [Quick start](#).

API description

```
/**
 * Listen to the pull-down refresh state.
 *
 * @param onPullRefreshListener
 */
public void setOnPullRefreshListener(OnPullRefreshListener onPullRefreshListener)

/**
 * Listen to the state of loading more.
 *
 * @param onLoadMoreListener
 */
public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener)

/**
 * Enable pull-down refresh by using the code.
 */
public void startRefresh()

/**
 * End the pull-down refresh.
 */
public void finishRefresh()

/**
 * Update the state of loading more on the bottom.
 *
 * @param isShowLoad
 * @param hasMore
 */
public void updateLoadMore(boolean isShowLoad, boolean hasMore)
```

Code sample

```
<com.alipay.mobile.antui.load.AURefreshListView
    android:id="@+id/refresh_list_view"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" />
```

```
listView.setOnPullRefreshListener(new OnPullRefreshListener() {
    @Override
    public void onRefresh() {
        listView.finishRefresh();
        listView.updateLoadMore(true, true);
    }

    @Override
    public void onRefreshFinished() {

    }

});
listView.setOnLoadMoreListener(new OnLoadMoreListener() {
    @Override
    public void onLoadMore() {
        for (int i = 0; i < 3; i++) {
            Map<String, Object> map = new HashMap<String, Object>();
            map.put("PIC", "Pull down to load more lists");
            map.put("TITLE", "Pull up to load more");
            contents.add(map);
        }
        adapter.notifyDataSetChanged();
        if(contents.size() > 13) {
            listView.updateLoadMore(true, false);
        } else {
            listView.updateLoadMore(true, true);
        }
    }

    @Override
    public void onLoadingFinished() {

    }

});
```

1.4.8.7. Switch tab component

AUSegment is used to replace APSwitchTab, with related codes reconstructed and the original APIs reserved for smooth scrolling.

If the SDK version is later than 10.0.20, this component supports switching by scrolling a tab. The left and right spacing of each Tab is 14dp:

- Scrolling switching is supported when the width of all tabs exceeds the initial width.
- When the width of all tabs is less than the initial width, the option indicating whether to equally divide the APIs is provided. By default, equal division is selected.

Sample image

Dependency

See [Quick start](#).

API description

```

/**
 * Reset the tab view.
 */
public void resetTabView(String[] tabNameArray)

/**
 * Adjust the position of the selected line at the bottom. Generally, this method is called in the onPageScrolled callback of ViewPager.
 *
 * @param position      The start position.
 * @param positionOffset The offset of the start position (in percentage).
 */
public void adjustLinePosition(int position, float positionOffset)

/**
 * Select a tab but do not adjust the position of the bottom line, which is suitable for tab switching using ViewPager.
 * Implement the selected line at the bottom by calling adjustLinePosition in the onPageScrolled callback of ViewPager.
 *
 * @param position      The position.
 */
public void selectTab(int position)

/**
 * Select a tab and adjust the position of the bottom line.<br>
 * Used in non-ViewPager tab switching scenarios. The duration of the cutscene before each tab switching interval is 250 ms
 *
 * @param position      The target selected tab.
 */
public void selectTabAndAdjustLine(int position)

/**
 * Select a tab and adjust the position of the bottom line.<br>
 * Used in non-ViewPager tab switching scenarios. The duration of the cutscene before each tab switching interval is customized.<br>
 * If the next animation is enabled before the previous animation finished, the previous animation is immediately terminated. The next animation starts after the final position of the previous animation is located.
 *
 * @param position      The target position.
 * @param during         The duration of the cutscene before each tab switching interval.
 */
public void selectTabAndAdjustLine(int position, int during)

/**
 * Set the tab switching listener.
 *
 * @param tabSwitchListener
 */
public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

```

```
/**
 * Add a red dot to the specified position.
 * @param view Red dot view.
 * @param position
 */
public void addTextRightView(View view, int position)

/**
 * Add a red dot to the specified position.
 * @param view Red dot view.
 * @param params The relative position of the red dot.
 * @param position
 */
public void addTextRightView(View view, RelativeLayout.LayoutParams params, int position)
```

Description of the tab scrolling switching API

To use the scrolling function, set the custom property parameter `scroll` to true, for example, adding `app:scroll="true"` to the layout file. The scrolling tab supports only the following four APIs. Other APIs are invalid to the scrollable tab.

```
/**
 * Set the tab switching listener.
 * @param tabSwitchListener
 */
public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

/**
 * Set the data source.
 * @param list
 */
public void init(List<ItemCategory> list)

/**
 * Set the selected tab.
 * @param position
 */
public void setCurrentSelTab(int position)

/**
 * Each tab has a fixed left-right spacing of 14dp. An option indicating whether to equal
ly dividing the APIs is provided when the width of all tabs is less than the initial width.
 * By default, equal division is selected. If you set this parameter to false, equal divi
sion is not allowed.
 * @param divideAutoSize
 */
public void setDivideAutoSize(boolean divideAutoSize)
```

Custom properties

The `scroll` property is added to the SDK of a version of 10.0.20 or later.

Property	Purpose	Type
tabCount	The tab quantity.	Integer
tab1Text	Tab1 text.	String, Reference
tab2Text	Tab2 text.	String, Reference
tab3Text	Tab3 text.	String, Reference
tab4Text	Tab4 text.	String, Reference
tabTextArray	The tab text array.	String, Reference
uniformlySpaced	Whether self-adaptive.	Boolean
tabTextColor	The text color.	Reference, Color
tabTextSize	The text size.	Dimension
bottomLineColor	The color of the bottom line.	Color, Reference
scroll	Whether to support scrolling.	Boolean

Code sample

XML

```
<com.alipay.mobile.antui.segement.AUSegment
    android:id="@+id/switchtab_three"
    android:layout_width="fill_parent"
    android:layout_height="50dp"
    android:layout_marginTop="10dp"
    app:tab1Text="Left text"
    app:tab2Text="Middle text"
    app:tab3Text="Right text"
    app:tabCount="3"/>
```

1.4.8.8. TabBar item component

AUTabBarItem is used to provide TabBar items in the mPaaS framework.

Sample image

Dependency

For how to add dependency, follow the relevant instructions in [Quick Start](#).

Custom properties

Property	Description	Type
topIconSid	The icon.	Reference
topIconSize	The icon size.	Dimension
textColor	The text color.	Color, Reference

Code sample

Here is the sample of using the component in XML.

```
<com.alipay.mobile.antui.bar.AUTabBarItem
    android:id="@+id/tab_2"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:text="Koubei"
    android:textSize="@dimen/AU_ICONSIZE2"
    app:topIconSize="@dimen/AU_ICONSIZE2"
    app:topIconSid="@drawable/tab_bar_alipay"
    app:textColor="@color/tabbar_text_color1"/>
```

1.5. Native based - iOS component library

1.5.1. Quick start

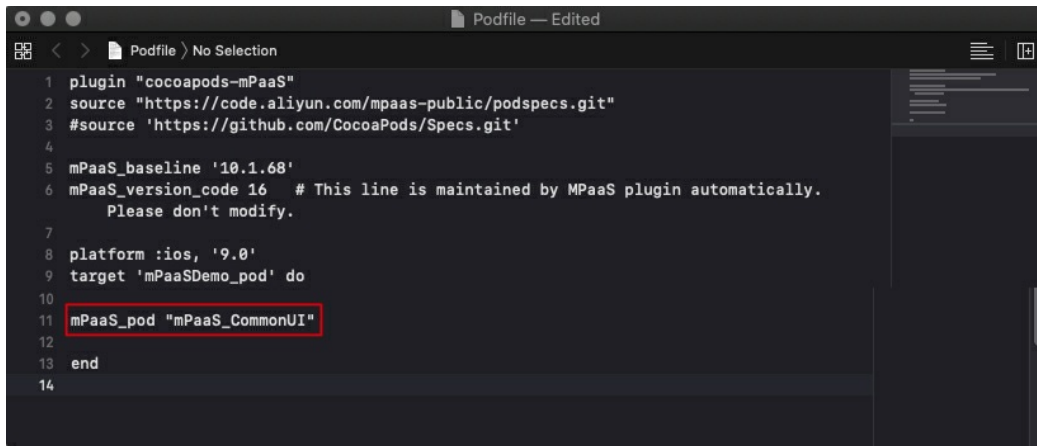
mPaaS provides a unified component library to meet the needs of users for different native controls. To enable clients to access the unified component library, you must add the SDK of the unified component library, and then call the SDK API method in the code to add controls.

Prerequisite

The project is connected to mPaaS. For more information, refer to: [Access based on native framework and using Cocoapods](#).

Add the SDK

1. In the Podfile file, enter `mPaaS_pod "mPaaS_CommonUI"` to add the dependency on the Common UI component.



2. Run `pod install` to connect the component to the mPaaS.

Use the SDK

Use Common UI SDK of the baseline version 10.1.60 and later. You can refer to the official demo of [Common UI](#).

1.5.2. Basic components

1.5.2.1. Activity Indicator base class

- The activity indicator base class `AUActivityIndicator` is the `UIActivityIndicatorView` version in mPaaS.
- To facilitate subsequent extension, use `AUActivityIndicator` instead of `UIActivityIndicatorView` of the system in all mPaaS apps.
- Since current `AUActivityIndicator` is completely inherited from `UIActivityIndicatorView` without additional properties and methods, this topic does not describe APIs and code examples.

1.5.2.2. Switch base class

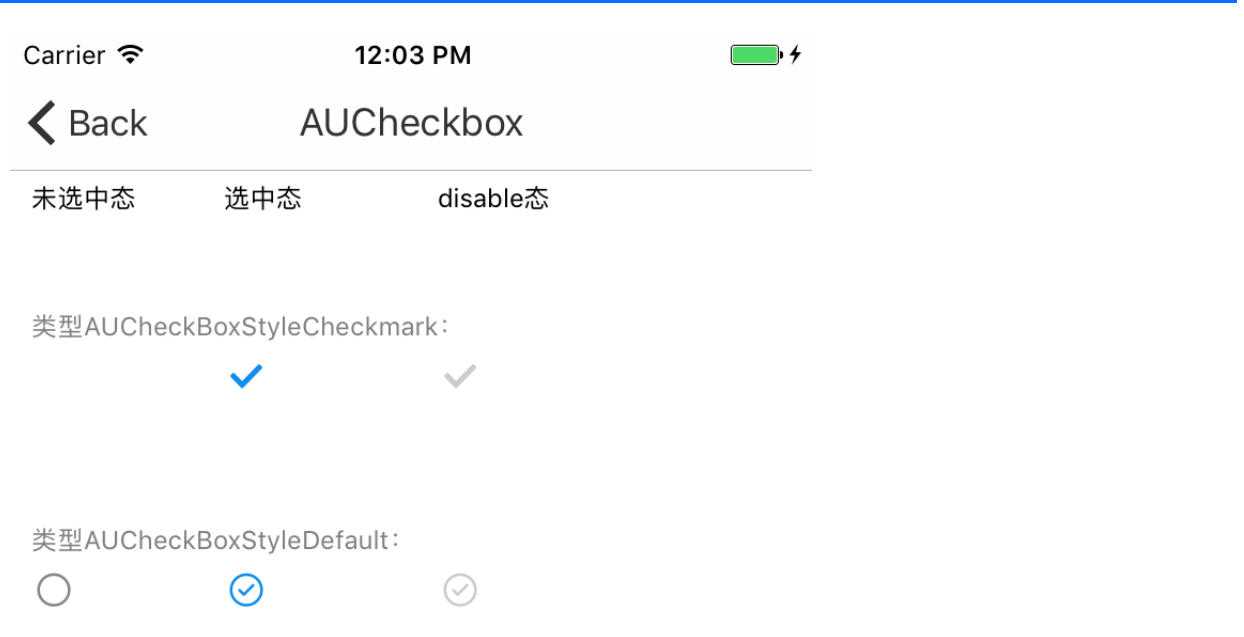
The switch base class `AUSwitch` in mPaaS is equivalent to `UISwitch`. To facilitate subsequent extension, `AUSwitch` instead of `UISwitch` must be used in all mPaaS apps.

Since the current switch base class is completely inherited from `UISwitch` without additional properties or methods, this topic does not provide APIs and code examples.

1.5.2.3. Check box control

- `AUCheckBox` is a radio button control.
- `AUCheckBox` is migrated from the `APCheckBox` of `APCommonUI`. Use latest `AUCheckBox`.

Sample image



API description

```
/**
The check box type.

- AUCheckBoxStyleDefault: The default style, similar to the check box of the web.
- AUCheckBoxStyleCheckmark: The checkmark style of tableview.
*/
typedef NS_ENUM(NSInteger, AUCheckBoxStyle) {
AUCheckBoxStyleDefault,
AUCheckBoxStyleCheckmark
};

/**
The radio button control.
*/
@interface AUCheckBox : UIControl

/**
Initialize the AUCheckBox method according to the type.

@param style The type of the check box.

@return AUCheckBox
*/
- (instancetype)initWithStyle:(AUCheckBoxStyle)style;

/**
Whether to select a property.
*/
@property(nonatomic, assign, getter = isChecked) BOOL checked;

/**
Whether to disable a property.
*/
@property(nonatomic, assign, getter = isDisabled) BOOL disabled;

/**
The check box type (read-only, set only upon initialization).
*/
@property (nonatomic, assign, readonly) AUCheckBoxStyle style;

@end
```

Code sample

```
AUCheckBox *checkbox = [[AUCheckBox alloc] initWithStyle:AUCheckBoxStyleDefault];
checkbox.checked = YES;
checkbox.disabled = NO;
checkbox.origin = CGPointMake(100, 250);
[checkbox addTarget:self action:@selector(checkboxValueChanged:) forControlEvents:UIControlEventValueChanged];
[self.view addSubview:checkbox];

- (void)checkboxValueChanged:(id)sender
{
    AUCheckBox *checkbox = (AUCheckBox *)sender;
    NSLog(@"%@", checkbox);
}
```

1.5.2.4. Image base class

- The image base class AUImage is the version of UIImage in mPaaS.
- To facilitate subsequent extension, use AUImage instead of system UIImage in all mPaaS apps.
- Since the current image base class is completely inherited from the UIImage without additional properties and methods, this topic does not describe APIs and code examples.

1.5.2.5. Label base class

The label base class AULabel in mPaaS is equivalent to UILabel. To facilitate subsequent extension, AULabel instead of UILabel must be used in all mPaaS apps.

Since the current label base class is completely inherited from UILabel with no additional properties or methods, this topic does not provide APIs and code examples.

🔍 Note

For complex label settings, `TTTAttributedLabel` (already imported to Ant UI) can be used.

1.5.2.6. Footer base class

The footer control mainly contains the text link component, the copyright component, and the combination of the text link and copyright.

Sample image



AUTextLinkView - text link

```
@protocol AUTextLinkDelegate <NSObject>

@optional
/* Callback upon text link tapping.
 * textLinkView      The text link.
 * Index             Click a subscript, which starts from 0 and corresponds to the params subscript.
 * Button            The button to click.
 */
- (void)textLinkView:(AUTextLinkView *)textLinkView didClickOnIndex:(NSInteger)index atButton:(UIButton *)button;

@end

//
@interface AUTextLinkView : UIView

@property (nonatomic, strong) UIView *containerView;    // The container.
@property (nonatomic, weak) id <AUTextLinkDelegate> delegate;

// titles: The text description array.
- (instancetype)initWithFrame:(CGRect)frame params:(NSArray *)params;

@end
```

AUCopyrightView - copyright

```
@interface AUCopyrightView : UIView

@property (nonatomic, strong) UILabel *copyrightLabel;

//
- (instancetype)initWithFrame:(CGRect)frame string:(NSString *)string;

@end
```

AUPageAnkletView - combination of the text link and copyright

```
@interface AUPageAnkletModel : NSObject

@property (nonatomic, strong) NSMutableArray *textLinkInfos;
@property (nonatomic, strong) NSString *copyrightInfo;

@end

typedef void(^paramsBlock) (AUPageAnkletModel *model);

@interface AUPageAnkletView : UIView

@property (nonatomic, strong) AULinkView *textLinkView;           // The text link.
@property (nonatomic, strong) AUCopyrightView *copyrightInfoView; // Copyright text.

//
- (instancetype)initWithFrame:(CGRect)frame params:(paramsBlock)params;

@end
```

Code sample

- Text link:

```
AULinkView *textLinkBtns = [[AULinkView alloc] initWithFrame:CGRectMake(0, CGRectGetMaxY(copyrightView1.frame)+40, self.view.width, 50) params:@{@"Bottom link", @"Bottom link", @"Bottom link"}];
textLinkBtns.centerX = self.view.centerX;
[self.view addSubview:textLinkBtns];
```

- Copyright:

```
AUCopyrightView *copyrightView1 = [[AUCopyrightView alloc] initWithFrame:CGRectMake(0, 80, self.view.width, 40) string:@"© 2004-2017 Alipay.com. All rights reserved."];
copyrightView1.centerX = self.view.centerX;
[self.view addSubview:copyrightView1];
```

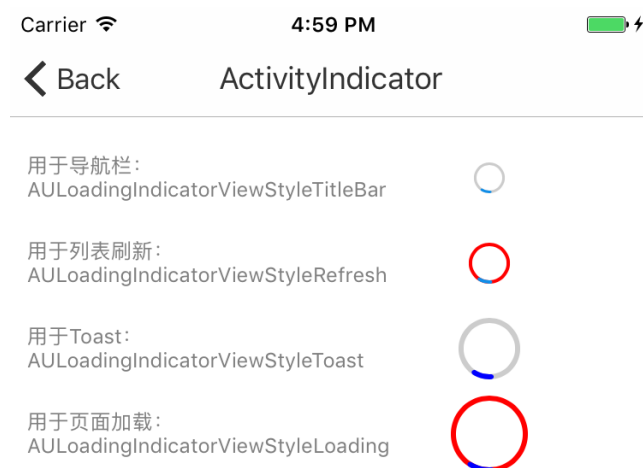
- Combination of text link and copyright:

```
AUPageAnkletView *ankletView = [[AUPageAnkletView alloc] initWithFrame:CGRectMake(0, CGRectGetMaxY(textLinkBtns.frame)+40, self.view.width, 100) params:^(AUPageAnkletModel *model) {
    model.textLinkInfos = [[NSMutableArray alloc] initWithArray:@[@"Bottom link", @"Bottom link", @"Bottom link"]];
    model.copyrightInfo = @"© 2004-2017 Alipay.com. All rights reserved.";
}];
ankletView.centerX = self.view.centerX;
[self.view addSubview:ankletView];
```

1.5.2.7. mPaaS customized loading control

- AULoadingIndicatorView is a custom loading control of mPaaS.
- The mPaaS-customized loading control is migrated from the APLoadingIndicatorView of APCommonUI. Use latest AULoadingIndicatorView.

Sample image



API description

```
typedef enum{
    AULoadingIndicatorViewStyleTitleBar, //Navigation bar loading box, diameter: 36 px, ring
    width: 3 px.
    AULoadingIndicatorViewStyleRefresh, //List loading box, diameter: 48px, ring width: 4px
    AULoadingIndicatorViewStyleToast, //Toast loading box, diameter: 72px, ring width: 6px
    AULoadingIndicatorViewStyleLoading, //Page loading box, diameter: 90px, ring width: 6px
}AULoadingIndicatorViewStyle;
```

```
/**
 * The mPaaS-customized loading control.
 */
```

```
@interface AULoadingIndicatorView : UIView
```

```
@property (nonatomic, assign) BOOL hidesWhenStopped; //Whether to hide when stopped.
@property (nonatomic, strong) UIColor *trackColor; //The color of the ring.
```

```

@property (nonatomic, strong) UIColor *progressColor;          //The color of the loading indicator.
@property (nonatomic, assign) float progressWidth;            //Set the width of the ring when customizing the ring size. Default value: 2.
@property (nonatomic, assign) CGFloat progress;              //The ratio of the arch length of the loading indicator to the perimeter of the ring. Default value: 0.1

/**
 * The circular loading box.
 * Note: If the default style is not used, define the size of the circle and use initWithFrame to initialize the circle. The default width of the ring is 2, which can be adjusted by setting progressWidth.
 */
- (instancetype)initWithLoadingIndicatorStyle:(AULoadingIndicatorViewStyle)style;

/**
 Start an animation.
 */
- (void)startAnimating;

/**
 End an animation.
 */
- (void)stopAnimating;

/**
 Whether an animation is being executed.

 @return YES: An animation is being executed. NO: No animation is being executed.
 */
- (BOOL)isAnimating;

@end

```

Code sample

```

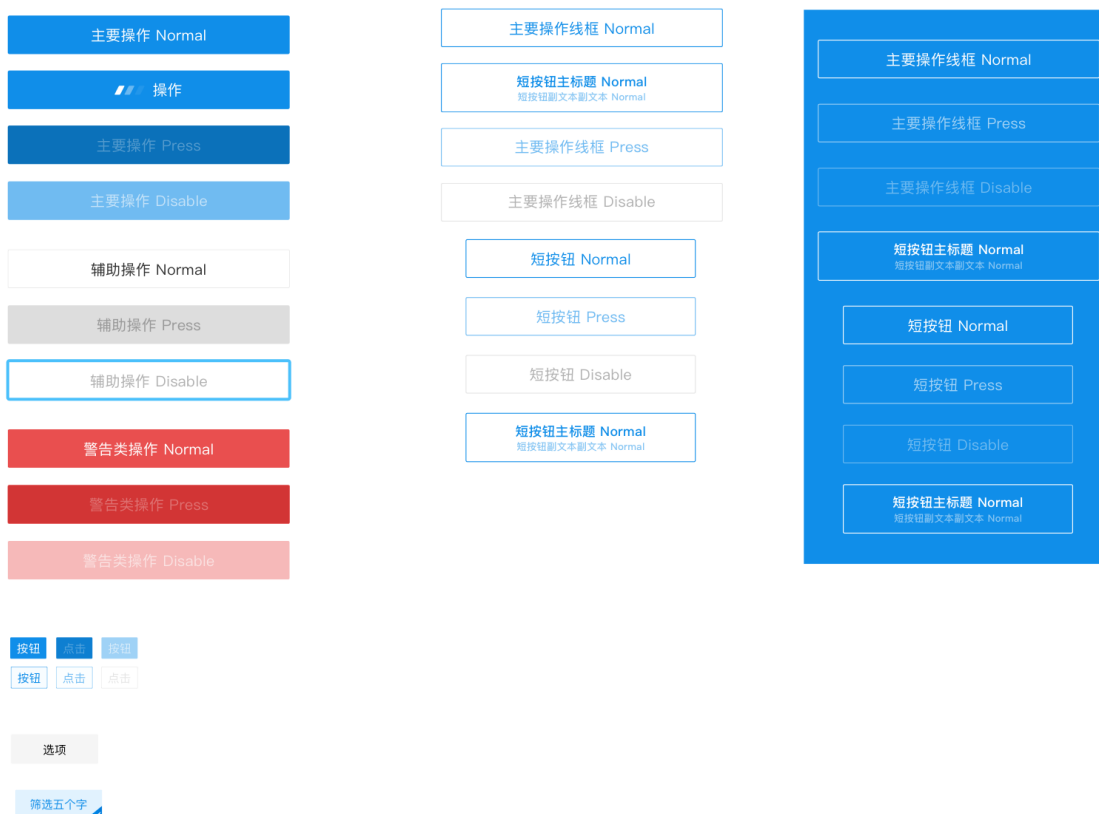
AULoadingIndicatorView *view = [[AULoadingIndicatorView alloc] initWithLoadingIndicatorStyle:AULoadingIndicatorViewStyleLoading];
view.hidesWhenStopped = YES;
view.center = CGPointMake(280, 250);
view.trackColor = [UIColor redColor];
view.progressColor = [UIColor blueColor];
[view startAnimating];
[self.view addSubview:view];

```

1.5.2.8. Button base class

AUButton follows the new UED requirements, currently contains two styles, and cannot be fully interconnected with APButton in APCommonUI. These two styles do not include the operation button of the warning type in the sample image.

Sample images



Dependency

The dependency of AUButton is as follows:

```
import <UIKit/UIKit.h>
```

API description

```
/**
    Initialization method
    @param style The style.
    @return The created initialization object.
    */
+ (instancetype)buttonWithStyle:(AUIButtonStyle)style;

/**
    * An auxiliary method of the initialization, used for creating and initializing a button object.
    *
    * @param buttonType    The button type. It must be one of the values defined in AUIButtonStyle.
    * @param title         Button title.
    * @param target        The object responding to the button tap event.
    * @param action        The function responding to the button tap event.
    *
    * @return The button object newly created and initialized.
    *
    * The initialization object of this method. A frame needs to be set.
    */
+ (instancetype)buttonWithStyle:(AUIButtonStyle)style title:(NSString *)title target:(id)target action:(SEL)action;

/**
    Display the loading icon animation and text (the loading icon is on the left and the text is on the right). When there is no text, the loading icon is centered.

    @param loadingTitle    The text to be displayed along with the loading icon. If this parameter is set to nil or an empty string, text is not displayed and the loading icon is centered.
    @param currentVC       The current VC that is used to remove the mask after the loading is complete.
    */
- (void)startLoadingWithTitle:(NSString *)loadingTitle currentViewController:(UIViewController *)currentVC;

/**
    Stop the rotation of the loading icon.
    */
- (void)stopLoading;
```

Custom properties

Property	Description
AUIButtonStyleNone	The default style.

Property	Description
AUButtonStyle1	Blue background, white text, borderless, and large button.
AUButtonStyle2	White background, black text, light gray border, and large button.
AUButtonStyle3	Transparent background, blue text, blue border, and small button.
AUButtonStyle4	White background, with upper and lower separation lines by default, and red text; applicable to page bottom operation scenarios (unfollow); default height: 44 units; width: same as screen width.
AUButtonStyle5	White background, with upper and lower separation lines by default, and ant blue text; applicable to page bottom operation scenarios (more services); default height: 44 units, and width: same as screen width
AUButtonStyle6	Red background, white text, for operations of the warning type, and large button.
AUButtonStyle7	White background, black text, light gray border, and small button.
AUButtonStyle8	Blue background, white text, borderless, and small button.

Code sample

```
AUIButton *button = [UIButton buttonWithType:AUIButtonStyle2 title:@"AUIButtonStyle2" target:self action:@selector(onButtonClicked)];
button.frame = CGRectMake(XX, XX, XX, XX);

UIButton *buttonDisable = [UIButton buttonWithType:AUIButtonStyle1];
buttonDisable.enabled = NO;
[buttonDisable setTitle:@"Style1disable" forState:UIControlStateNormal];
buttonDisable.frame = CGRectMake(XX, XX, XX, XX);

//Set the loading icon on the button.
[button startLoadingWithTitle:@"Loading" currentViewController:self];

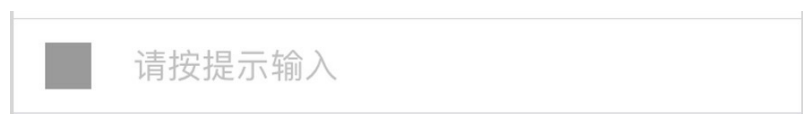
//Stop the rotation of the loading icon on the button.
[button stopLoading];
```

1.5.3. Input components

1.5.3.1. Image input box

AUImageInputBox is an input box with an icon on the left and is inherited from AUInputBox.

Sample image



API description

```
/**
The input box with an icon on the left side.
*/
@interface AUImageInputBox : AUInputBox

/**
Left-side icon view (read-only).
*/
@property (nonatomic, strong, readonly) UIImageView *iconView;

/**
Set the left-side icon.

@param image The icon image.
*/
- (void)setIconImage:(UIImage *)image;
```

Code sample

```
AUIImageInputBox *imageInputBox = [AUIImageInputBox inputBoxWithOriginY:startY inputBoxType:AUIImageInputBoxTypeNone];
imageInputBox.textField.placeholder = @"Enter as promoted";
[imageInputBox setIconImage:image];
[self.view addSubview:imageInputBox];
```

1.5.3.2. Paragraph input box

AUParagraphInputBox is a multi-line input box control. The maximum number of characters allowed in specific business can be specified for it.

Sample images

段落输入框

请输入你想表达的内容

0/1240

段落输入框

请输入你想表达的内容

API description

```
// The multi-line text input box.

@interface AUParagraphInputBox : UIView

@property (nonatomic, strong) UITextView *textView;           // Input box
@property (nonatomic, assign) NSInteger maxInputLen;         // Set the character limit (as required)

// Initialization.
- (instancetype)initWithFrame:(CGRect)frame placeholder:(NSString *)placeholder;

// Set the placeholder text.
- (void)setPlaceholder:(NSString *)placeholder;

@end
```

Code sample

```
_paragraphInputBox = [[AUParagraphInputBox alloc] init];
_paragraphInputBox.frame = CGRectMake(0, startY, AUCCommonUIGetScreenWidth(), 10);
_paragraphInputBox.maxInputLen = 1240;
_paragraphInputBox.textView.delegate = self;
[_paragraphInputBox setPlaceholder:@"Please enter text here"];
[self.view addSubview:_paragraphInputBox];
```

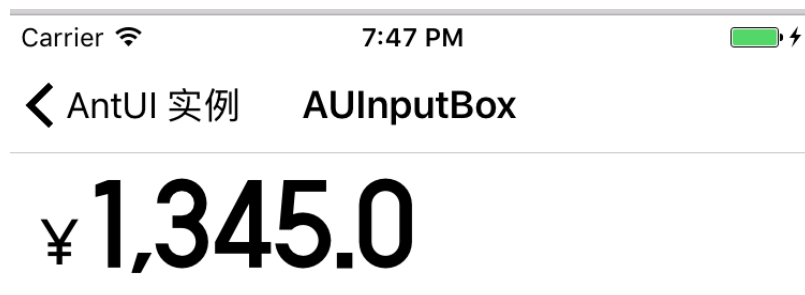
1.5.3.3. Simplified amount input box

The simplified amount input box `AUAmountEditTextField` can be used together with the amount display component `AUAmountLabelText`.

AUAmountEditTextField

Currently, the verification and preprocessing logic of input content is not contained, but can be realized by setting delegate in the business.

Sample image



API description

```

NS_ASSUME_NONNULL_BEGIN

@interface AUAmountEditTextField : UITextField

@end

/**
 The simplified amount input component with the "¥" symbol and an underscore.
 The font size of the input content is scaled with the content length.
 */
@interface AUAmountEditText : UIView

/**
 The amount input box, you can modify properties or set the delegate as needed.
 When a clear event occurs, [amountTextField sendActionsForControlEvents:UICtrlEventE
 ditingChanged] is called.
 */
@property(nonatomic, strong) AUAmountEditTextField *amountTextField;

/**
 It is open to AUAmountLabelText for adjusting the font size when the length of inputText c
 hanges.
 Business parties do not use.

 @param textLength inputText length.
 @return UIFont
 */
+ (UIFont *)resetFontSize:(NSUInteger) textLength;

@end

NS_ASSUME_NONNULL_END

// amountTextField Initialization settings.
_amountTextField.textColor = RGB(0x000000);
_amountTextField.backgroundColor = [UIColor clearColor];
_amountTextField.font = [UIFont fontWithName:kAmountNumberFontName size:45.0];
_amountTextField.contentVerticalAlignment= UIControlContentVerticalAlignmentCenter;
_amountTextField.inputView = [AUNumKeyboards sharedKeyboardWithMode:AUNumKeyboardModeCommon
];
_amountTextField.rightViewMode = UITextFieldViewModeWhileEditing;
_amountTextField.rightView = self.rightView;//Use rightView to implement clearButton.

```

Sample code

```

field = [[AUAmountEditText alloc] init]; //Set the width to the screen width and the height
to 70.
field.amountTextField.delegate = self;
[view addSubview:field];

```

AUAmountLabelText

AUAmountLabelText is an amount display component used in conjunction with AUAmountEditTextField.

Sample image



API description

```
NS_ASSUME_NONNULL_BEGIN

/**
 The amount display component used in conjunction with the AUAmountEditTextField.
 */
@interface AUAmountLabelText : UIView

@property (nonatomic, copy) NSString *amountText;//Amount, without the "¥" symbol, for example, "80.01".

@end

NS_ASSUME_NONNULL_END
```

Sample code

```
label = [[AUAmountLabelText alloc] init]; //Set the width to the screen width and the height to 64.
label.amountText = @"1,345.0";
[view addSubview:label];
```

1.5.3.4. Amount input box

AUAmountInputBox, composed of AUAmountInputField and AUAmountInputFieldFooterView, is an amount input box with the combination function.

AUAmountInputBox

- Currently, it allows you to set titles (plain text) and add footers (plain text/input box).
- Input content verification and preprocessing logic is not contained, but can be realized by setting delegate in the business.

Sample image

转账金额

¥1345.80



添加备注(50字以内)

API description

NS_ASSUME_NONNULL_BEGIN

/**

The amount input box with the combination function.

Currently, it allows you to set titles (plain text) and add footers (plain text/input box). Input content verification and preprocessing logic is not contained, but can be realized by setting delegate in the business.

*/

@interface AUAmountInputBox : UIView

/**

AUAmountInputBox initialization method

@param views @[AUAmountInputField,AUAmountInputFieldFooterView]

@return AUAmountInputBox

*/

+ (AUAmountInputBox *)amountInputBoxWithViews:(NSArray *) views;

@end

NS_ASSUME_NONNULL_END

Sample code

```
AUAmountInputField *inputField = [AUAmountInputField amountInputWithTitle:@"Transfer amount
"];
AUAmountInputFieldFooterView *footerView = [AUAmountInputFieldFooterView footerWithInput:@"
Add notes (within 50 words)"];
AUAmountInputBox *inputBox = [AUAmountInputBox amountInputBoxWithViews:[NSArray arrayWithOb
jects:inputField,footerView,nil]];
inputField.textField.delegate = self;
footerView.inputTextField.delegate = self;
[_scrollView addSubview:inputBox];
```

AUAmountInputField

It is extended based on the AUAmountEditText combination. Currently, it supports title setting.

Sample image

转账金额

¥1234.50



API description

NS_ASSUME_NONNULL_BEGIN

/**

It is extended based on the AUAmountEditText combination. Currently, it supports title setting.

*/

@interface AUAmountInputField : UIView

- (AUAmountEditText *)textField;

+ (AUAmountInputField *)amountInputWithTitle:(NSString *) title;

@end

NS_ASSUME_NONNULL_END

Code sample

See [Code sample](#) of AUAmountInputBox.

AUAmountInputFieldFooterView

Description

AUAmountInputFieldFooterView is footerView of AUAmountInputBox, and currently supports both “plain text” and “input box” .

Sample image

添加备注(50字以内)

Dependency

The dependency of AUAmountInputFieldFooterView is as follows:

```
pod 'AntUI'
```

API description

```

NS_ASSUME_NONNULL_BEGIN

@interface AUAmountInputFieldFooterView : UIView

@property (nonatomic, strong) UITextField * inputTextField;
@property (nonatomic, strong) UILabel * descTextLabel;

+ (AUAmountInputFieldFooterView *)footerWithInput:(nullable NSString *)placeholder;
+ (AUAmountInputFieldFooterView *)footerWithDesc:(nullable NSString *)text;

@end

NS_ASSUME_NONNULL_END

```

Sample code

See [Code sample](#) of AUAmountInputBox.

1.5.3.5. Normal input box

AUInputBox is a single-line input box that supports the arrangement of a title on the left side and an image button on the right side.

Sample images



API description

```

typedef NS_ENUM(NSInteger, AUInputBoxType)
{
    AUInputBoxTypeMobileNumber,    // Mobile phone number
    AUInputBoxTypeCreditCard,      // Credit card
    AUInputBoxTypeBankCard,        // Debit card
    AUInputBoxTypeAmount,          // Amount
    AUInputBoxTypeIDNumber,        // ID card
    AUInputBoxTypeNotEmpty,        // Not empty
    AUInputBoxTypeAlipayAccount,   // mPaaS app account
    AUInputBoxTypeNone             // No authentication
};

typedef enum AUInputBoxStyle
{
    AUInputBoxStyleNone,           // No background image.
    AUInputBoxStyleiOS6,           // Rounded background image.
    AUInputBoxStyleiOS7            // Non-rounded background image.
} AUInputBoxStyle;

```

```

/**
The single-line input box with title text and a button image.
*/
@interface AUInputBox : UIView

#pragma mark - AUInputBox property.

// The text input box.
@property(strong, nonatomic) UITextField *textField;
@property(strong, nonatomic) NSString *textFieldText;
@property(strong, nonatomic) NSString *textFieldFormat;
@property(assign, nonatomic) CGFloat horizontalMargin;
@property(assign, nonatomic) CGFloat textFieldHorizontalMargin;

// The button.
@property(strong, nonatomic) UIButton *iconButton;
@property(assign, nonatomic) BOOL hidesButtonWhileNotEmpty;
@property(assign, nonatomic) BOOL hidesButton;

// The label displayed on the left part of the input box.
@property(n nonatomic, readonly) UILabel *titleLabel;
@property(n nonatomic, assign) CGFloat titleLabelWidth;

The style, authenticator, background image, and input box type.
@property(assign, nonatomic) AUInputBoxStyle style;
@property(readonly, nonatomic) UIImageView *backgroundImage;
@property(assign, nonatomic) AUInputBoxType inputBoxType;

#pragma mark - The AUInputBox static method.
/**
* Create an input box component.
* @param originY The Y coordinator of the input box.
* @param type The type of the text input box.
* @return The input box component.
*/
+ (instancetype)inputboxWithOriginY:(CGFloat)originY inputboxType:(AUInputBoxType)type;

/**
* Create an input box component with an icon button.
* @param originY The Y coordinator of the input box.
* @param icon The icon on the button, 44x44.
* @param type The type of the text input box.
* @return The input box component with a button.
*/
+ (instancetype)inputboxWithOriginY:(CGFloat)originY buttonIcon:(UIImage *)icon inputboxType:(AUInputBoxType)type;

/**
* @return The control height. The default value is 44. The value is 47 for iPhone6 plus.
*/
+ (float)heightOfControl;

```

```

- (CGFloat)heightOfControl;

#pragma mark - The AUInputBox instance method.

- (instancetype)initWithFrame:(CGRect)frame inputboxType:(AUInputBoxType)type;

- (void)buildIconButton:(UIImage *)icon;

/**
 * Add a space to the text in the specified format.
 * @param text    The text content.
 * @return        The text to which a space has been added.
 */
- (NSString *)formatText:(NSString *)text;

/**
 * Add an icon by using this method for inputBox without any icon specified during the initialization.
 * @param icon    The icon on the button.
 */
- (void)setRightButtonIcon:(UIImage *)icon;

/**
 * Check the input validity.
 */
- (BOOL)checkInputValidity;

/**
 * Filter text. Only digits are allowed. The maximum length is specified.
 * The parameter is the delegate parameter and the maximum length is specified by maxLength.
 */
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withMaxLength:(int)maxLength;

/**
 * Specify the maximum length.
 * @maxLength Maximum length, excluding format spaces.
 */
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withFormatStringMaxLength:(int)maxLength;

```

Code sample

- Single-line input box

```

AUInputBox *inputBox = [AUInputBox inputBoxWithOriginY:startY inputboxType:AUInputBoxTypeNone];
inputBox.titleLabel.text = @"Label";
inputBox.textField.placeholder = @"Please enter text as prompted";
[self.view addSubview:inputBox];

```

- Icon

```
AUInputBox *iconInputBox = [AUInputBox initWithOriginY:startY buttonIcon:image inputboxType:AUInputBoxTypeNone];
iconInputBox.titleLabel.text = @"Label";
iconInputBox.textField.placeholder = @"Please enter text as prompted";
[self.view addSubview:iconInputBox];
```

1.5.3.6. Search input box

AUSearchTitleView is a search bar control. It is similar to a search bar but can only be tapped. It supports the following styles:

- `AUSearchTitleStyleDefault = 0` : search box with black text, which is applicable when a light color background is used.

Example: search box displayed on the navigation pane on an mPaaS app page.

- `AUSearchTitleStyleMiddleAlign` : search box with centered black text, which is applicable when a light color background is used.

Example: search box on the contact page.

- `AUSearchTitleStyleContent` : search box with white text, which is applicable when a deep color background is used

Example: search box displayed on the navigation pane on the mPaaS app homepage.

Sample image



Dependency

The dependency of AUSearchTitleView is as follows:

```
AntUI (iOS)
1.0.0.161108003457
APCommonUI (iOS)
1.2.0.161108102201
```

API description

```
typedef NS_ENUM(NSUInteger, AUSearchTitleStyle) {
    AUSearchTitleStyleDefault = 0, // Search box with black text, which is applicabl
```

```

    AUISearchTitleStyleDefault = 0, // Search box with black text, which is applicable
    e when a light color background is used.
    AUISearchTitleStyleMiddleAlign, // Search box with centered black text, which is
    applicable when a light color background is used.
    AUISearchTitleStyleContent, // Search box with white text, which is applicable
    e when a deep color background is used.
};

@class AUISearchTitleView;

@protocol AUISearchTitleViewDelegate <NSObject>

@optional

// The search bar control.
- (void)didPressedTitleView:(AUISearchTitleView *)titleLabel;

// The voice icon of the search bar control.
- (void)didPressedVoiceButton:(AUISearchTitleView *)titleLabel;

@end

/**
The search bar control. (The width is the same as that of the screen by default.)
*/
@interface AUISearchTitleView : UIView

@property(nonatomic, assign)AUISearchTitleStyle style; // The background style of the
search box. The light color background is used by default

@property(nonatomic, strong) NSString *placeholder; // The search box placeholder
, which is "Search" by default.
@property(nonatomic, strong) UIColor *placeholderColor; // The text color of the search
box placeholder.

@property (nonatomic, weak) id<AUISearchTitleViewDelegate> delegate;

@property(nonatomic, strong) UIImage *searchIconImage; // The search icon.
@property(nonatomic, strong) UIColor *normalBackgroundColor; // The background color of the
search box.
@property(nonatomic, assign) BOOL isShowVoiceIcon; // Whether to display the Voice
icon. Default value: No.

/**
* The padding to the left and right of the outer-layer transparent view. The default value
is 9. To configure the space between the view of the instance to be initialized and another
view for a business, consider the padding as well to prevent a visual error.
* Note: If the instance to be initialized is defined as the titleLabel of a navigationItem,
the system specifies the spacing between the titleLabel and the views on its left and right
in an adaptive manner. To meet the visual requirements, the system sets the padding between
the search box and the outer-layer view.
*

```

```
* If there are any special requirements, change the padding.
*
*/
@property(nonatomic,assign) CGFloat marginBetweenItem;

/**
 * The method for getting the instance.
 *
 * @param style The search box style.
 *
 * @return      Return the instance.
 */
- (id)initWithSearchStyle:(AUSearchTitleStyle)style;

/**
 * This method calls the global search page by default. To define the event of tapping the
 * search box for a business, override this method in the subclass.
 */
- (void)onClicked;

@end
```

Code sample

- Used in a navigation bar

```
AUSearchTitleView *titleLabel = [[AUSearchTitleView alloc] initWithSearchStyle:AUSearchTitleStyleDefault];
titleLabel.placeholder = @"The search bar style";
titleLabel.placeholderColor = [UIColor blackColor];
titleLabel.normalBackgroundColor = [UIColor orangeColor];
titleLabel.isShowVoiceIcon = YES;
titleLabel.delegate = self;
self.navigationItem.titleLabel = titleLabel;
```

- Used in a common view

```
titleLabel = [[AUSearchTitleView alloc] initWithSearchStyle:AUSearchTitleStyleMiddleAlign];
titleLabel.placeholder = @"The AUSearchTitleStyleMiddleAlign style";
titleLabel.isShowVoiceIcon = YES;
titleLabel.delegate = self;
[self.view addSubview:titleLabel];
```

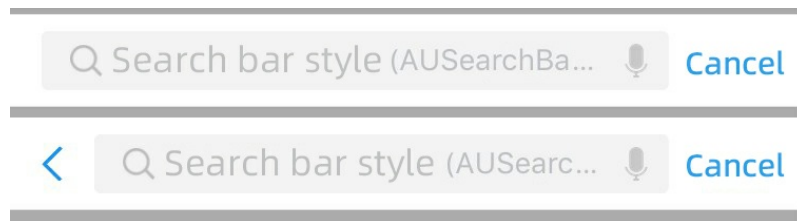
1.5.3.7. Search bar component

The dependency of AUISearchBar is as follows:

- AUISearchBar is a search bar control of mPaaS.
- It supports the following styles:

- `AUSearchBarStyleNormal` : search bar with a Cancel button, for example, search bar on the homepage for global search
- `AUSearchBarStyleDetail` : search bar with a Cancel button and a back icon, for example, search bar on a level-2 page for global search

Sample images



Dependency

The dependency of `AUSearchBar` is as follows:

```
AntUI (iOS)
1.0.0.161108003457
APCommonUI (iOS)
1.2.0.161108102201
```

API description

```
@class AUSearchBar;

@protocol AUSearchBarDelegate <NSObject>

@optional

#pragma mark - Proxy methods corresponding to UITextField.
//
- (BOOL)searchBarTextShouldBeginEditing:(AUSearchBar *)searchBar;
//
- (BOOL)searchBarTextShouldEndEditing:(AUSearchBar *)searchBar;
// Called when text starts editing.
- (void)searchBarTextDidBeginEditing:(AUSearchBar *)searchBar;
// Called when text ends editing.
- (void)searchBarTextDidEndEditing:(AUSearchBar *)searchBar;
// Called when text changes (including clear).
- (void)searchBar:(AUSearchBar *)searchBar textDidChange:(NSString *)searchText;
// Called before text changes.
- (BOOL)searchBar:(AUSearchBar *)searchBar shouldChangeTextInRange:(NSRange)range replacementText:(NSString *)text;

- (BOOL)searchBarShouldClear:(AUSearchBar *)searchBar;

#pragma mark - Other proxy methods.

// Called when the search icon is clicked.
- (void)searchBarSearchButtonClicked:(AUSearchBar *)searchBar;
```

```
// Called when the Cancel button is clicked.
- (void)searchBarCancelButtonClicked:(AUSearchBar *) searchBar;

// Called when the back icon is clicked (valid for the AUSearchBarStyleDetail style).
- (void)searchBarBackButtonClicked:(AUSearchBar *)searchBar;

// Called when the voice icon is clicked (valid when shouldShowVoiceButton is set to YES).
- (void)searchBarOpenVoiceAssister:(AUSearchBar *)searchBar;

@end

typedef NS_ENUM(NSUInteger, AUSearchBarStyle) {
    AUSearchBarStyleNormal = 0, //normal.
    AUSearchBarStyleDetail, //has back Button
};

/**
The search bar control. (By default, the width is the same as that of the screen, and the height is 44.)
*/
@interface AUSearchBar : UIView

@property (nonatomic, strong) NSString *text; // The search box text.
@property (nonatomic, assign) BOOL isSupportHanziMode; // Whether to support search while input. Default value: YES.
@property (nonatomic, assign) AUSearchBarStyle style; // The style of the search box.
@property (nonatomic, assign) BOOL shouldShowVoiceButton; // Whether to display the Voice button. Default value: NO.
@property (nonatomic, strong, readonly) UITextField *searchTextField; // The search box.
@property (nonatomic, weak) id<AUSearchBarDelegate> delegate;

/**
The initialization method.

@param style The search bar style.

@return Return an AUSearchBar instance.
*/
- (instancetype)initWithStyle:(AUSearchBarStyle)style;

@end
```

Code sample

- Add to the navigation bar

```
AUSearchBar *searchBar = [[AUSearchBar alloc] initWithStyle:AUSearchBarStyleNormal];
searchBar.searchTextField.placeholder = @"The search bar style (AUSearchBarStyleNormal)";

searchBar.delegate = self;
searchBar.isSupportHanziMode = YES;
searchBar.shouldShowVoiceButton = YES;
self.navigationItem.titleView = searchBar;
self.navigationItem.leftBarButtonItem = nil; // Add no button to the left of the search bar.
self.navigationItem.rightBarButtonItem = nil; // Add no button to the right of the search bar.
self.navigationItem.hidesBackButton = YES; // Hide the back icon.
```

- Add to a normal view

```
searchBar = [[AUSearchBar alloc] initWithStyle:AUSearchBarStyleDetail];
searchBar.searchTextField.placeholder = @"The search bar style (AUSearchBarStyleDetail)";

searchBar.delegate = self;
searchBar.isSupportHanziMode = YES;
searchBar.shouldShowVoiceButton = YES;
[self.view addSubview:searchBar];
```

1.5.3.8. Verification code input box

AUTextCodeInputBox is a verification code input control.

Sample images



API description

```
/**
The SMS verification code input box with a countdown timer.
*/
@interface AUPasswordCodeInputBox : AUPasswordCodeBox

/**
The wait time before an SMS message is sent.
*/
@property (nonatomic, assign) NSTimeInterval interval;

/**
* Create an SMS verification code input box.
* @param frame The position and size in the parent class.
* @param interval The wait time before an SMS message is sent.
* @return The input box for the text message verification code.
*/
- (AUPasswordCodeInputBox *)initWithFrame:(CGRect)frame interval:(NSTimeInterval)interval;

/**
* Create an SMS verification code input box.
* @param originY The Y-coordinate of the component.
* @param interval The wait time before an SMS message is sent.
* @return The input box for the text message verification code.
*/
- (AUPasswordCodeInputBox *)initWithOriginY:(CGFloat)originY interval:(NSTimeInterval)interval;

/**
* Set a block to be executed when countdown ends.
* @param block The block to be executed.
*/
- (void)setCountdownDidCompleteBlock:(void (^)(void))block;
```

Code sample

```
AUPasswordCodeInputBox *smsInputBox = [[AUPasswordCodeInputBox alloc] initWithOriginY:startY interval:60];
[smsInputBox.actionButton addTarget:self action:@selector(onSmsButtonClicked:) forControlEvents:UIControlEventTouchUpInside]; // The callback for processing the event that the button
on the right is tapped.
[self.view addSubview:smsInputBox];
```

1.5.4. Item component

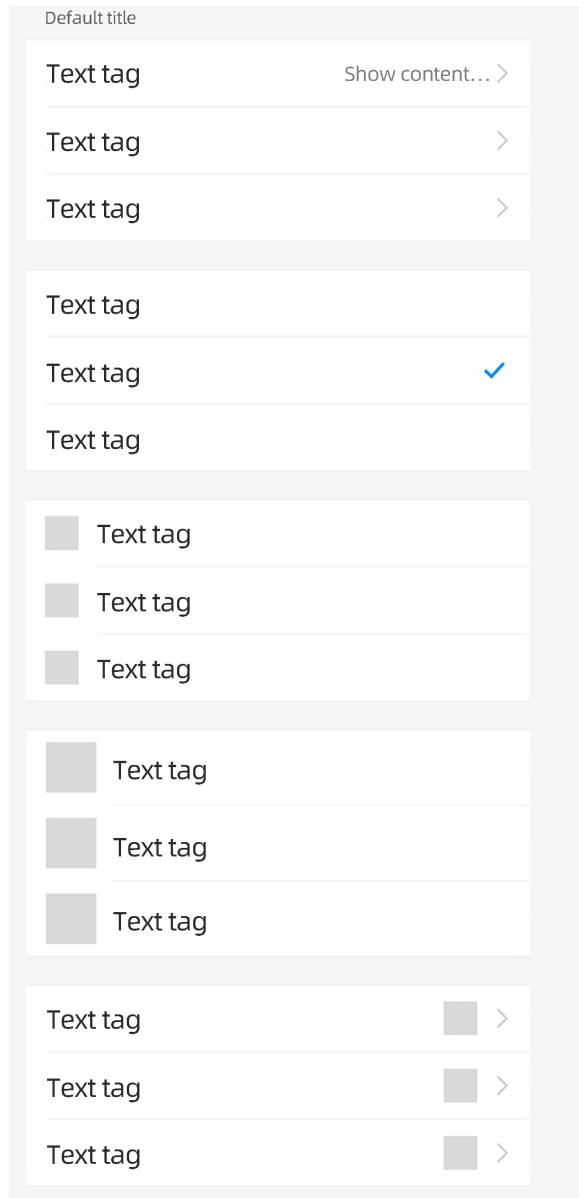
AUListItem is a series of controls designed based on the new UED requirements. It cannot be used interchangeably with the APTableView control in the original APTCommonUI because most UED styles are different.

AUListItem contains four ListItems. The following table lists the elements supported by them respectively.

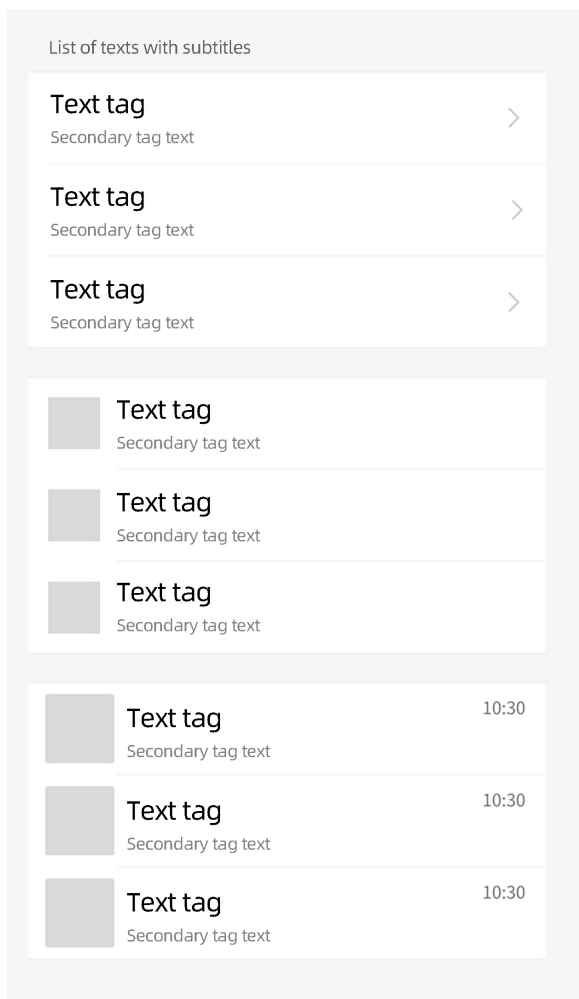
AUListItem	Title	Subtitle	Left icon	Right icon	Left icon in a customized size	Show a checkmark when selected	Rightmost assistant arrow
AUSingleTitleListItem	YES☺	YES	YES☺	YES	YES☺	YES	YES☺
AUDoubleTitleListItem	YES☺	YES	YES☺	-	YES☺	-	YES☺
AUCheckboxListItem	YES☺	-	-☺	-	-☺	-	YES☺
AUSwitchListItem	YES☺	-	-☺	-	-☺	-	-☺

Sample images

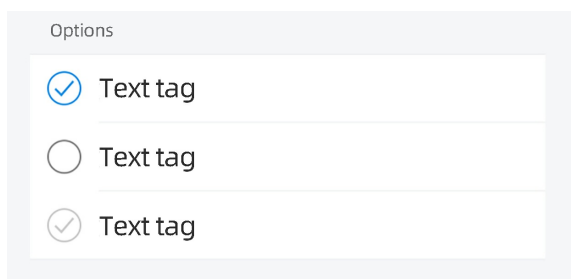
- AUSingleTitleListItem



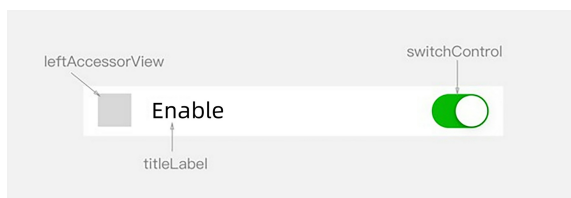
- AUNumberTitleListItem



- AUCheckBoxListItem



- AUISwitchListItem



Dependency

The dependency of AUListItem is as follows:

```
import <UIKit/UIKit.h>
```

API description

Common APIs

Model layer

Multiple delegates are set to standardize parameter transfer by external clients. For elements that are not supported, external clients cannot transfer the corresponding parameters. For example, `AUDoubleTitleListItem` does not support the right icon. Therefore, `AUDoubleTitleListItemModelDelegate` does not contain the rightImage parameter.

```
AUListItemProtocols.h
/**
Data items that can be set and accessed in AUSingleTitleListItem.
*/
@protocol AUSingleTitleListItemModelDelegate <NSObject>

@property (nonatomic, copy)    NSString    *subtitle;           // The subtitle.
@property (nonatomic, strong)  UIImage    *leftImage;          // The left-side image.
@property (nonatomic, strong)  UIImage    *rightImage;          // The image before the text on the right side.
@property (nonatomic, strong)  UIImage    *rightAssistImage;    // The image after the text on the right side.
@property (nonatomic, assign)  CGSize     leftImageSize;        // You can set the size of the left-side image. Default size: 22.
@property (nonatomic, assign)  CGSize     rightAssistImageSize; // You can set the size of the image after the text on the right side. Default size: 22.

@end

/**
Data items that can be set and accessed in AUDoubleTitleListItem.
*/
@protocol AUDoubleTitleListItemModelDelegate <NSObject>

@property (nonatomic, copy)    NSString    *subtitle;           // The subtitle.
@property (nonatomic, strong)  UIImage    *leftImage;          // The left-side image.
@property (nonatomic, assign)  CGSize     leftImageSize;        // You can set the size of the left-side image. The default size is used if you do not set this parameter.
@property (nonatomic, copy)    NSString    *timeString;         // The time displayed on the right side.
@property (nonatomic, copy)    NSString    *rightAssistString;  // The auxiliary information on the right side, which is centered by default.
@property (nonatomic, assign)  NSInteger  subtitleLines;        // The number of auxiliary subtitle lines, which must be specified by the client.
// @property (nonatomic, assign)  BOOL     showAccessory;        // Whether to display the auxiliary icon.

@end

/**
Data items that can be set and accessed in AUCheckBoxListItem.
```

```

*/
@protocol AUCheckBoxListItemModelDelegate <NSObject>
//@property (nonatomic, assign) BOOL showAccessory; // Whether to display the
auxiliary icon.

@end

/**
Data items that can be set and accessed in AUMultiListItemDelegate.

*/
@protocol AUMultiListItemDelegate <NSObject>

@property (nonatomic, copy) NSString *subtitle; // The subtitle.
@property (nonatomic, strong) UIImage *leftImage; // The left-side image.
//@property (nonatomic, assign) CGSize leftImageSize; // The size of the left-side ima
ge.
@property (nonatomic, assign) BOOL showAccessory; // Whether to display the auxili
ary icon.
@property (nonatomic, assign) NSInteger subtitleLines; // Set the number of subtitle li
nes.

@end

/**
Data items that can be set and accessed in AUMultiListBottomAssistDelegate.

*/
@protocol AUMultiListBottomAssistDelegate <NSObject>

@property (nonatomic, strong) NSString *originalText; // The source of the text.
@property (nonatomic, strong) NSString *timeDesc; // The time description.
@property (nonatomic, strong) NSString *othersDesc; // Other description.

@end

/**
Data items that can be set and accessed in AUParallelTitleListItem.

*/
@protocol AUParallelTitleListItemModelDelegate <NSObject>
@property (nonatomic, copy) NSString *subtitle; // Title 2
@property (nonatomic, copy) NSString *describe; // Description 1
@property (nonatomic, copy) NSString *subDescribe; // Description 2

@end

/**
Data items that can be set and accessed in AULineBreakListItem.

*/

```

```
@protocol AULineBreakListItemModelDelegate <NSObject>
@property (nonatomic, copy) NSString *subtitle; // The subtitle.
@end

/**
Data items that can be set and accessed in AUCouponsItemDelegate.
*/
@protocol AUCouponsItemDelegate <NSObject>

@property (nonatomic, copy) NSString *subtitle; // The subtitle.
@property (nonatomic, strong) UIImage *leftImage; // The left-side image.
@property (nonatomic, strong) UIImage *leftImageUrl; // The URL of the left-side image.
@property (nonatomic, strong) NSString *assistDesc; // The text assisted description.
@property (nonatomic, assign) NSInteger totalWidth; // Set the width of the card.

@end

/**
The rich text protocol of TTTAttributeLabelDelegate.
*/
@protocol TTTAttributeLabelDelegate <NSObject>

@property (nonatomic, copy) NSString *attributeText; // The rich-text content.
@property (nonatomic, copy) NSString *linkText; // The rich-text link text.
@property (nonatomic, copy) NSString *linkURL; // The URL of the rich-text content.

@end

AUListItemModel.h
import "AUListItemProtocols.h"
@interface AUListItemModel : NSObject
@property (nonatomic, copy) NSString *title; // The title.
@property (nonatomic, assign) UIEdgeInsets separatorLineInset; // You can set the margin between the left-side and right-side separation lines and the cell.
@end
```

View layer

```
AUBaseListItem.h:

@interface AUBaseListItem : UITableViewCell
// The following data items are open so that external clients can set extra properties, such as the title color.
```

```

@property(nonatomic,strong) UILabel *titleLabel;
@property(nonatomic,strong) UIView *separatorLine;
/**
The initialization function.
@param reuseIdentifier The reuse identifier.
@param block          The block imported externally. Generally, title and leftimage are set
in this block externally.
@return              Return a self instance.
*/
- (instancetype)initWithReuseIdentifier:(NSString *)reuseIdentifier model:(void (^)(AUListItemModel*model))block;
/**
Return the cell height.
@return              Return the cell height.
*/
+ (CGFloat)cellHeight ;
@end

#ifdef AUBaseListItem_protected
// This identifier is open only to the subclass. Before importing AUBaseListItem in the subclass, set AUBaseListItem_protected to 1.
@interface AUBaseListItem ()
@property (nonatomic,strong) AUListItemModel* baseModel;
@end

#endif
/**
Generally, a client simply needs to call the initWithReuseIdentifier:model: method in the AUBaseListItem subclass to meet the requirement.
Here, independent methods oriented to parameters such as title are provided.
All parameters, except title, are implemented in the subclass and isolated from each other.
*/
@interface AUBaseListItem (Extensions)
/**
Set the title.
@param title The title string.
*/
- (void)setTitle:(NSString* )title;

/**
The method for getting the title.
@return Return the title string.
*/
- (NSString*)title ;

/**
Set the spacing between the separation line and the left or right side of a cell.
@param separatorLineInset UIEdgeInsets parameter
*/
- (void)setSeparatorLineInset:(UIEdgeInsets)separatorLineInset;

/**
Get the inset of the separation line.
@return Return the inset of the separation line.

```

```
*/
- (UIEdgeInsets)separatorLineInset;
```

AUSingleTitleListItem

```
typedef NS_ENUM(NSUInteger, AUSingleTitleListItemStyle) {
    AUSingleTitleListItemStyleDefault, // Height: 92; icon: 58.
    AUSingleTitleListItemStyleValue1, // Height: 110; icon: 72.
};

@interface AUSingleTitleListItem : AUBaseListItem

@property(nonatomic, strong) UILabel *subtitleLabel;
@property(nonatomic, strong) UIImageView *leftImageView;
@property(nonatomic, strong) UIImageView *rightImageView;
@property(nonatomic, strong) UIImageView *rightAssistImageView;

/**Important
The initialization function.

@param reuseIdentifier The reuse identifier.
@param block The block imported externally. Generally, title and leftimage are set in this block externally.

@return Return a self instance.
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier model:(void (^)(AUListItemModel<AUSingleTitleListItemModelDelegate>*model))block __deprecated_msg("Do not use this method because it will be discarded.");

/**
Set all data required for showing a cell.

@param block The block to be transferred.
*/
- (void)setModelBlock:(void (^)(AUListItemModel<AUSingleTitleListItemModelDelegate>*model))block;

/**
The initialization function.

@param reuseIdentifier The reuse identifier.
@param style The custom style. For more information, see AUSingleTitleListItemStyle.
@return Return a self instance.
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier customStyle:(AUSingleTitleListItemStyle)style;

+++
```

```

/**
Return a height based on the style.

@param style
@return Return a custom style. For more information, see AUSingleTitleListItemStyle.
*/
+ (CGFloat)cellHeightForStyle:(AUSingleTitleListItemStyle)style;

@end

```

AUDoubleTitleListItem

```

typedef NS_ENUM(NSInteger, AUDoubleTitleListItemStyle) {
    AUDoubleTitleListItemStyleDefault, // Has a left icon; height: 120 px; icon: 76.
    AUDoubleTitleListItemStyleValue1, // Has no left icon; height: 120 px.
    AUDoubleTitleListItemStyleValue2, // Has a left icon; height: 144 px; icon: 88.
};

@interface AUDoubleTitleListItem : AUBaseListItem<AUDoubleTitleListItemModelDelegate, TTTAttributeLabelDelegate>

@property(nonatomic, strong) UILabel *subtitleLabel;
@property(nonatomic, strong) UIImageView *leftImageView;
@property(nonatomic, strong) UILabel *timeLabel;
@property(nonatomic, strong) UILabel *rightAssistLabel;

/**
Set all data required for showing a cell.

@param block The block to be transferred.
*/
- (void)setModelBlock:(void (^)(AUListItemModel<AUDoubleTitleListItemModelDelegate, TTTAttributeLabelDelegate>*model))block;

/**
The initialization function.

@param reuseIdentifier The reuse identifier.
@param style The custom style. For more information, see AUDoubleTitleListItemStyle.
@return Return a self instance.
*/
- (instancetype)initWithReuseIdentifier:(NSString*)reuseIdentifier customStyle:(AUDoubleTitleListItemStyle)style;

/**
Return a height according to the style.

@param style The custom style. For more information, see AUDoubleTitleListItemStyle.
@return Return the cell height.
*/
+ (CGFloat)cellHeightForStyle:(AUDoubleTitleListItemStyle)style;

```

```

/**
    Return a dynamic height according to the style.

    @param style The custom style. For more information, see AUDoubleTitleListItemStyle.
    @param block The data model. For more information, see AUDoubleTitleListItemModelDelegate.
    Note: 1. The method must transfer a exact value of model.accessoryType.
           2. If line feeds are required, use subtitleLines to specify the number of lines.
    @return Return the cell height.
 */
+ (CGFloat)cellHeightForStyle:(AUDoubleTitleListItemStyle)style
    modelBlock:(void (^)(AUListItemModel<AUDoubleTitleListItemModelDelegate,
TTTAttributeLabelDelagate>*model))block;

@end

```

AUCheckBoxListItem

```

@protocol AUCheckBoxListItemDelegate <NSObject>

/**
    The callback to be triggered when the check box status changes.

    @param item The check box instance.
 */
- (void)checkboxValueDidChange:(AUCheckBox *)item; // Take the tag of the cell as the tag of the item.

@end

@interface AUCheckBoxListItem : AUNBaseListItem<AUCheckBoxListItemModelDelegate>

@property(nonatomic, assign, getter = isChecked) BOOL checked; // Set the check box to the selected state.
@property(nonatomic, assign, getter = isDisableCheck) BOOL disableCheck; // Specify whether to disable the check box.
@property(nonatomic, weak) id <AUCheckBoxListItemDelegate> delegate;

@end

```

AUSwitchListItem

```

@interface AUSwitchListItem : AUNBaseListItem

@property (nonatomic, strong) UISwitch *switchControl; // The switch control in a cell.

// Specify whether to show or hide the loading icon.
- (void)showLoadingIndicator:(BOOL)show;

@end

```

Custom properties

Property	Purpose	Type
title	The title.	NSString
titleLabel	The title label.	UILabel
subtitle	The subtitle.	NSString
subtitleLabel	The subtitle label.	UILabel
leftImage	The left-side icon.	UIImage
leftImageView	The view of the left-side icon.	UIImageView
rightImage	The right-side icon.	UIImage
rightImageView	The view of the right-side icon.	UIImageView
leftimageSize	The size of the left-side icon.	CGSize
timeString	The time string on the right.	NSString
timeLabel	The time label on the right.	UILabel
showMarkWhenSelected	Whether to show a checkmark for a selected cell.	BOOL
showAccessory	Whether to show an assistant icon.	BOOL
checked	Whether AUCheckBoxListItem is selected.	BOOL
disableCheck	Whether AUCheckBoxListItem is disabled.	BOOL

 Note

Note: The following code sample shows the properties supported in each control.

Code sample

AUSingleTitleListItem

- The recommended usage is as follows:

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithReuseIdentifier:identifierSingle1 model:^(AUListItemModel<AUSingleTitleListItemModelDelegate> *model) {  
    model.title = @"Title";  
    model.subtitle = @"Subtitle";  
    model.showAccessory = YES;  
    model.XXX = XXXX;  
    // The supported properties are  
    contained in AUListItemModel and AUSingleTitleListItemDelegate. For more information, see their API descriptions.  
    }];
```

- You can also set the provided properties separately. The property names are the same as those in model in the preceding recommended usage.

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testsingle"];  
cell.title = @"Subtitle";
```

- Each element on the control can be set.

```
AUSingleTitleListItem*cell = [[AUSingleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testsingle"];  
cell.titleLabel.backgroundColor = [UIColor redColor];
```

AUDoubleTitleListItem

- The recommended usage is as follows:

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithReuseIdentifier:identifierDouble3 model:^(AUListItemModel<AUDoubleTitleListItemModelDelegate> *model) {
    model.title = @"Right icon not supported";
    model.leftImage = [UIImage imageNamed:@"AntUI.bundle/illustration_ap_expectation_limit.png"];
    model.leftImageSize = CGSizeMake(100, 100);
    model.showAccessory = YES;

    // The supported properties are contained in AUListItemModel and AUSingleTitleListItemDelegate. For more information, see their API descriptions.
}];
```

- You can also set provided properties separately.

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testdouble"];
cell.leftImage = [UIImage imageNamed:@"AntUI.bundle/illustration_ap_expectation_limit.png"];
```

- Each element on the control can be set.

```
AUDoubleTitleListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testdouble"];
cell.leftImageView.image = [UIImage imageNamed:@"AntUI.bundle/illustration_ap_expectation_limit.png"];
```

AUCheckBoxListItem

- The recommended usage is as follows:

```
AUCheckBoxListItem* cell = [[AUCheckBoxListItem alloc] initWithReuseIdentifier:identifierCheckBox model:^(AUListItemModel<AUCheckBoxListItemModelDelegate> *model) {
    model.title = @"Selected by default";
    model.showAccessory = NO;
    // Only the preceding two properties can be set.
}];
cell.disableCheck = YES; // Set the check button to the disabled state.
```

- You can also set provided properties separately.

```
AUCheckBoxListItem*cell = [[AUCheckBoxListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testcheck"];
cell.showAccessory = YES;
```

- Each element on the control can be set.

```
AUCheckBoxListItem*cell = [[AUDoubleTitleListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"testcheck"];
cell.titleLabel.text = @"Selected by default";
```

AUSwitchListItem

```
AUSwitchListItem *switchCell = [[AUSwitchListItem alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"switchCell"];
AUListItemModel *model = _datas[indexPath.row];
switchCell.titleLabel.text = model.title;
switchCell.leftAccessorView = [[UIImageView alloc] initWithImage:[UIImage imageNamed:@"certify.png"]];
switchCell.leftAccessorType = AUListItemLeftAccessorTypeIcon;
switchCell.switchControl.on = NO;
UISwitch *switchView = (UISwitch *)switchCell.accessoryView;
[switchView addTarget:self action:@selector(switchValueChanged:) forControlEvents:UIControlEventValueChanged];
return switchCell;
```

1.5.5. Pop-up window component

1.5.5.1. Action sheet

AUActionSheet is migrated from APOActionSheet. The style is slightly adjusted, supporting the common layout with the deletion button and the common sheet layout.

Sample images

- Common layout with the delete button:

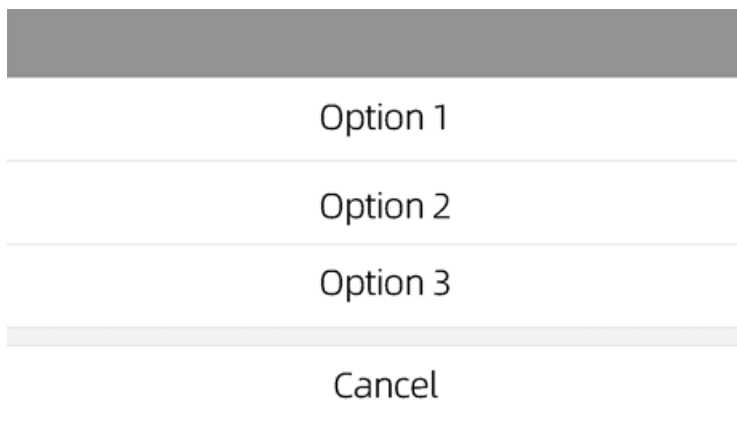


Here displays at most 2 rows of annotation. Provide clear information to make users understood.

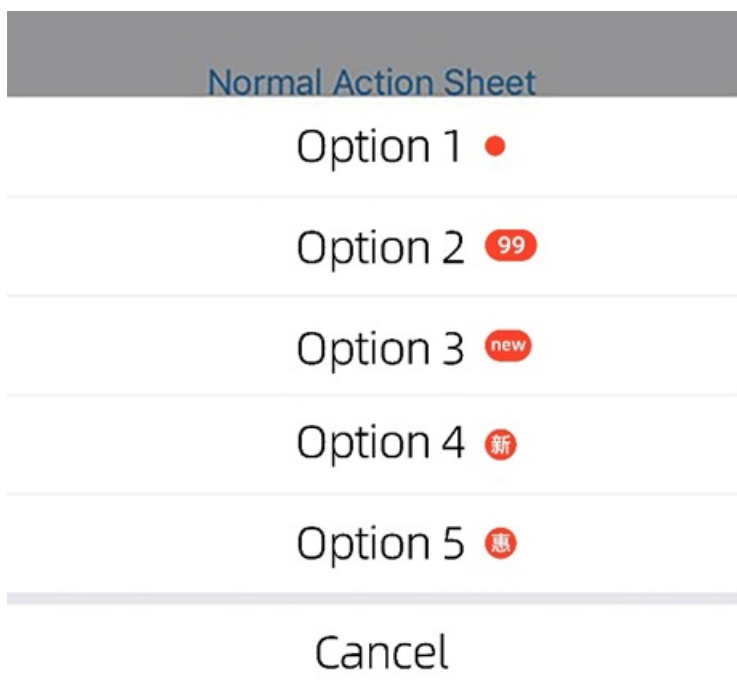
Confirm deletion

Cancel

- Tab layout:



• Badges:



Dependency

The dependency of `AUActionSheet` is as follows:

```
AntUI (iOS)
1.0.0.161108003457
APCommonUI (iOS)
1.2.0.161108102201
```

API description

```
typedef NS_ENUM(NSUInteger, AUActionSheetButtonType) {
    AUActionSheetButtonTypeDefault = 0,          // The default type.
    AUActionSheetButtonTypeDisabled,             // The button cannot be tapped.
    AUActionSheetButtonTypeDestructive,          // The red destructive button.
    AUActionSheetButtonTypeCustom                // The customized type.
};
```

```

/**
AUActionSheet The API is migrated from AActionSheet, and the style is adjusted.
*/
@interface AUActionSheet: UIView<UIAppearanceContainer>

/// The button height is 42 by default.
@property (nonatomic) CGFloat buttonHeight UI_APPEARANCE_SELECTOR;
/// The height of the Cancel button.
@property (nonatomic) CGFloat cancelButtonHeight UI_APPEARANCE_SELECTOR;
/// The color of the separation line, which is AU_COLOR_LINE by default.
@property (strong, nonatomic) UIColor *separatorColor UI_APPEARANCE_SELECTOR;
/// The background color of a tapped button.
@property (strong, nonatomic) UIColor *selectedBackgroundColor UI_APPEARANCE_SELECTOR;
// The attributes of UI components.
@property (copy, nonatomic) NSDictionary *titleTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *buttonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *disabledButtonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *destructiveButtonTextAttributes UI_APPEARANCE_SELECTOR;
@property (copy, nonatomic) NSDictionary *cancelButtonTextAttributes UI_APPEARANCE_SELECTOR;

/// The duration for showing or hiding an animation, which is 0.5s by default.
@property (nonatomic) NSTimeInterval animationDuration UI_APPEARANCE_SELECTOR;
/// The title.
@property (nonatomic, copy) NSString *title;
/// Whether the item is visible
@property (nonatomic, readonly, getter=isVisible) BOOL visible;
/// The header view of a custom button.
@property (strong, nonatomic) UIView *headerView;
/// The keyWindow before the ActionSheet instance is displayed.
@property (weak, nonatomic, readonly) UIWindow *previousKeyWindow;
/// The protocol delegate.
@property (nonatomic, weak) id<UIActionSheetDelegate> delegate;
/// The title of the Cancel button.
@property (copy, nonatomic) NSString *cancelButtonTitle;
/// The number of buttons.
@property (nonatomic, readonly) NSInteger numberOfButtons;
/// The index of the Cancel button, which is -1 by default.
@property (nonatomic) NSInteger cancelButtonIndex;
/// The index of a destructive red button, which is -1 by default and can be ignored if only one button exists.<UActionSheetButtonTypeDestructive, // The red destructive button.>
@property (nonatomic) NSInteger destructiveButtonIndex;
/**
The AUActionSheet initialization method

@param title The title information.
@param delegate The delegate object.
@param cancelButtonTitle The title of the Cancel button.
@param destructiveButtonTitle The title of a destructive button.
@param otherButtonTitles The list of other button title parameters.
@return The AUActionSheet instance.
*/

```

```

/
- (instancetype)initWithTitle:(NSString *)title delegate:(id<UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle destructiveButtonTitle:(NSString *)destructiveButtonTitle
otherButtonTitles:(NSString *)otherButtonTitles, ... NS_REQUIRES_NIL_TERMINATION;

/**
The AUIActionSheet initialization method.

@param title           The title information.
@param delegate        The delegate object.
@param cancelButtonTitle The title of the Cancel button.
@param destructiveButtonTitle The title of a destructive button.
@param items           The customOption data list (with custom title colors and badges).
@return                The AUIActionSheet instance.
*/
- (instancetype)initWithTitle:(NSString *)title
delegate:(id<UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
items:(NSArray<AUIActionSheetItem *> *)items;

/**
Add a button of a default type.

@param title The button title.
@return      The button index, starting from 0.
*/
- (NSInteger)addButtonWithTitle:(NSString *)title;

/**
Add a button.

@param title The button title.
@param type  The button type.
@return      The button index, starting from 0.
*/
- (NSInteger)addButtonWithTitle:(NSString *)title type:(AUIActionSheetButtonType)type;

/**
Obtain the button title based on the index.

@param buttonIndex The button index.
@return            The button title.
*/
- (NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex;

/**
Set a button in a position.

@param item The button type after information encapsulation.
@param index The index of the button to be replaced. The index is less than the number of existing buttons.
*/

```

```
- (void)setButton:(AUActionSheetItem *)item atIndex:(NSInteger)index;

/** The ActionSheet display method. */
- (void)show;

/**
Manually call the hiding method.

@param animate Whether a hiding animation is available.
*/
- (void)closeWithAnimate:(BOOL)animate;

/**
Manually simulate the hiding method based on button tapping (a protocol method related to button tapping is called back).

@param buttonIndex The button index.
@param animated Whether a hiding animation is available.
*/
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated;

/**
* Dynamically add an item.
* Note: Call this method after the actionSheet is shown on the screen. To add a button before the action sheet is shown, use addButtonWithTitle.
*
* @param item The custom item.
* @param index The position where the item is added.
*/
- (void)addButton:(AUActionSheetItem *)item atIndex:(NSInteger)index;

// Set the background mode. If the value is YES or @(YES), all displayed action sheets are hidden. The default value is NO.
+ (void)setIsBackGroundMode:(BOOL)isBackGroundMode;
+ (void)weakSetIsBackGroundMode:(id)isBackGroundMode;

- (void)showFromToolbar:(UIToolbar *)view;
- (void)showFromTabBar:(UITabBar *)view;
- (void)showFromBarButtonItem:(UIBarButtonItem *)item animated:(BOOL)animated NS_AVAILABLE_IOS(3_2);
- (void)showFromRect:(CGRect)rect inView:(UIView *)view animated:(BOOL)animated NS_AVAILABLE_IOS(3_2);
- (void)showInView:(UIView *)view;

@end

/** The ActionSheet button class after encapsulation. */
@interface AUActionSheetItem: NSObject
/// The button title.
@property (copy, nonatomic) NSString *title;
/// The button type.
@property (nonatomic) AUActionSheetButtonType type;
/// The color of the button title. When you set this value, manually change the button type
```

```
to AUActionSheetButtonTypeCustom.
@property (strong, nonatomic) UIColor *titleColor;

/**
 * Set the style for displaying badges.
 *
 *      badgeValue: @"."      A red dot is displayed.
 *                  @"new"    "new" is displayed.
 *                  @"digit"  A digit is displayed. If the digit is larger than 99, the mo
 *                  (...) image is displayed.
 *                  @"a Chinese character for "xin""  "xin" is displayed.
 *                  @"a Chinese character for "hui""  "hui" is displayed.
 *                  nil       Clear the displayed content.
 */
@property (nonatomic, copy) NSString *badgeValue;

@end
```

Code sample

- Common layout with the delete button:

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:@"Provide one or two lin
es of comments for information classification."

                                delegate:self
                                cancelButtonTitle:@"Cancel"
                                destructiveButtonTitle:@"Delete"
                                otherButtonTitles:nil];

[actionSheet show];
```

- Tab layout:

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:nil

                                delegate:self
                                cancelButtonTitle:@"Cancel"
                                destructiveButtonTitle:nil
                                otherButtonTitles:@"Option 1",@"Option

2",@"Option 3", nil];
[actionSheet show];
```

- Add a badge to an option:

```
AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:nil
                                delegate:self
                                cancelButtonTitle:@"Cancel"
                                destructiveButtonTitle:nil
                                otherButtonTitles:@"Option 1",@"Option
2",@"Option 3", nil];
AUActionSheetItem *item = [[AUActionSheetItem alloc] init];
item.title = @"Option 3";
item.type = AUActionSheetButtonTypeCustom;
item.badgeValue = @"new";
item.titleColor = [UIColor redColor];
[actionSheet setButton:item atIndex:2];

[actionSheet show];
```

1.5.5.2. Date picker component

AUDatePicker is a date selection control.

Sample images



下午9:20



< AntUI instance AUPicker

Create class method

Create member method

Date Picker 1

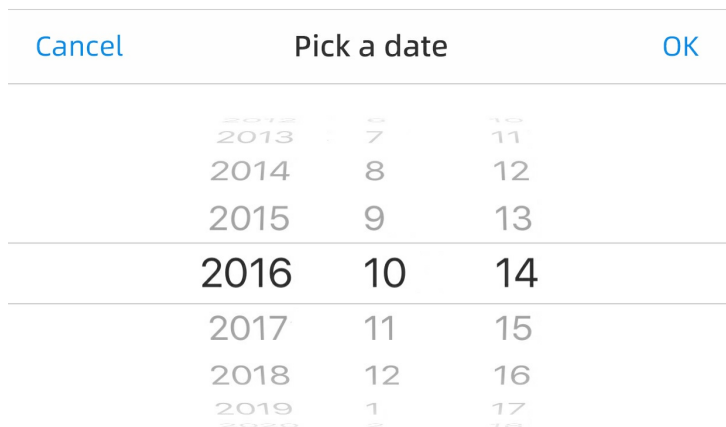
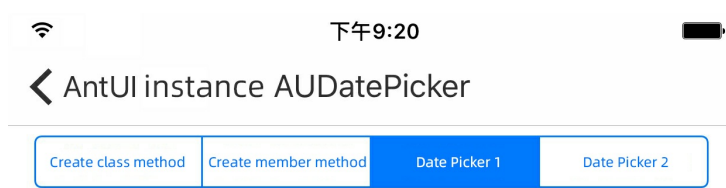
Date Picker 2

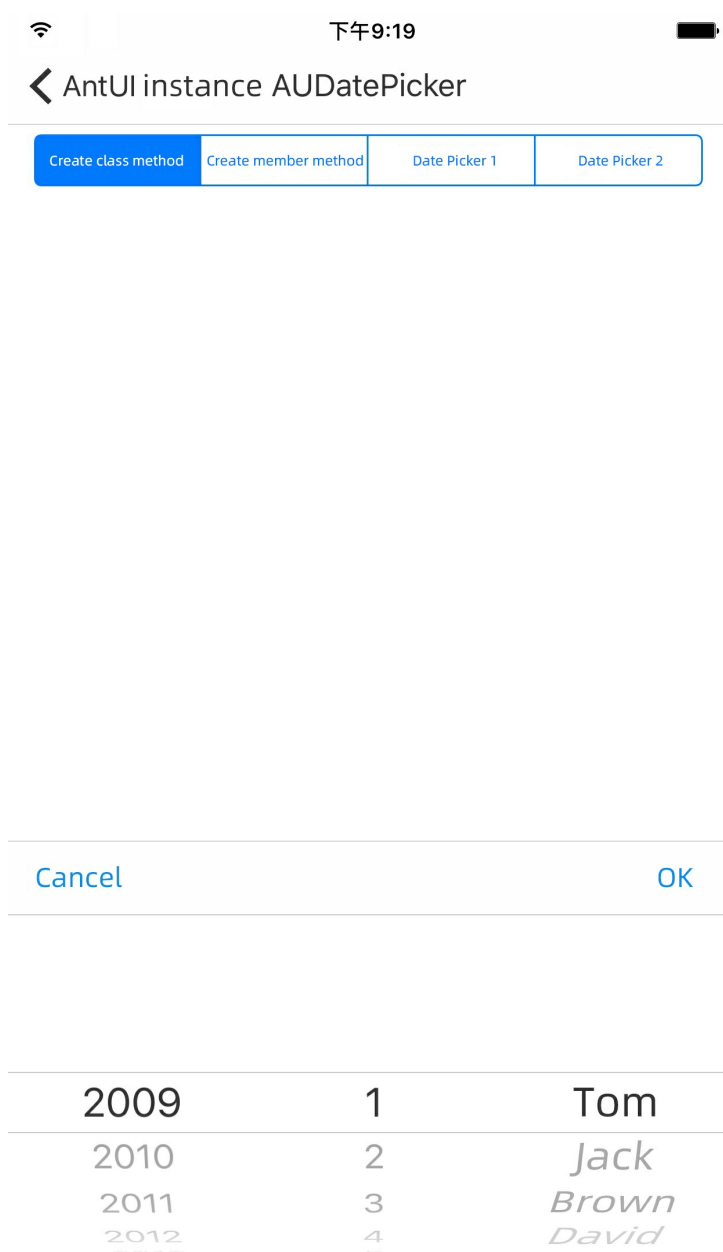
Cancel

Pick a date

OK

2016	11	17
2016	12	18
2017	1	19
2018	2	20
2019	3	21
2020	4	22
2021	5	23
2022	6	24





API description

- AUDatePicker.h

```
//
// ALPPicketView.h
// TestCell
//

#import <UIKit/UIKit.h>

@class AUDatePicker;

@protocol AUDatePickerDelegate <UIPickerViewDataSource, UIPickerViewDelegate>

/*
 * Callback is performed when Cancel is clicked.
 */

```

```

    */
    - (void)cancelPickerView:(AUDatePicker *)pickerView;

    /*
     * Callback is performed when Completed is clicked. The selected items can be returned through pickerView/Users/zhuwei/ios-phone-antui/ANTUI/Sources/Views/pickerView/AUDatePicker.h selectedRowInComponent.
     */
    - (void)selectedPickerView:(AUDatePicker *)pickerView;

@end
/*!
@class      AUDatePicker
@abstract   UIView
@discussion The frame-encapsulated picker with the Cancel and Completed buttons.
*/

@interface AUDatePicker : UIView

@property(nonatomic, strong) UIPickerView *pickerView;      // The general transaction picker.
@property(nonatomic, strong) UIDatePicker *datePickerView; // The time picker.

@property(nonatomic, assign) BOOL isDatePicker;             // Whether the current picker is the time picker. The default value is NO.

@property(nonatomic, weak) id<AUDatePickerDelegate> delegate;

/*
 * Create components.
 *
 * @param title The title. Its value can be nil.
 * @return      The created component, not shown by default. The show method needs to be called to show it.
 */
+ (AUDatePicker *)pickerViewWithTitle:(NSString *)title;

/*
 * Initialize objects.
 *
 * @param frame The display position.
 * @param title Show the title. Set the value to nil if you do not want to show the title.
 * @return      Do not show the object by default. call the show method if showing the object.
 */
- (id)initWithFrame:(CGRect)frame withTitle:(NSString *)title;

/*
 * Show
 */
- (void)show;

```

```

/*
 * Hide
 */
- (void)hide;

/**
 * Reload data.
 */
- (void)reload;

/**
 * When isDatePicker is YES, select the time using datePickerView.
 *
 * @param minDate    The earliest time.
 * @param maxDate    The latest time.
 */
- (void) setTimeDateminDate:(NSDate *)minDate MaxDate:(NSDate *)maxDate;

/**
 * When isDatePicker is YES, set the current time for datePickerView.
 *
 * @param currentDate    Set the current date.
 */
- (void) setCurrentDate:(NSDate *) currentDate;

/**
 * When isDatePicker is YES, set the time selected in the time picker.
 *
 * @param date          The selected date.
 * @param animated      Whether an animation is available.
 */
- (void)setAUIDatePickerDate:(NSDate *)date animated:(BOOL)animated; // if animated is YES,
animate the wheels of time to display the new date

@end

```

Sample code

```

//
// APPickerViewViewController.m
// UIDemo
//

#import "APPickerViewViewController.h"
#import "AUIDatePicker.h"

@interface APPickerViewViewController ()<AUIDatePickerDelegate,UIPickerViewDelegate,UIPicker
ViewDataSource>
@property(nonatomic,strong)AUIDatePicker* apPickerView;
@property(nonatomic,strong)AUIDatePicker* apPickerView2;

```

```

@property(nonatomic, strong, readonly) UIPickerView* apPickerView1;
@property(nonatomic, strong) UIPickerView* apPickerView3;
@property(nonatomic, strong) UIPickerView* apPickerView4;

@property(nonatomic, strong) UILabel* textLabel;
@property(nonatomic, strong) NSArray* yearArray;
@property(nonatomic, strong) NSArray* monthArray;
@property(nonatomic, strong) NSArray* nameArray;
@end

@implementation APPickerViewViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
        self.yearArray = @[@"2009",@"2010",@"2011",@"2012",@"2013",@"2014",@"2015",@"2016"];
    ;
        self.monthArray = @[@"1",@"2",@"3",@"4",@"5",@"6",@"7",@"8",@"9",@"10",@"11",@"12"];
    ;
        self.nameArray = @[@"Tom",@"Jack",@"Brown",@"David"];
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    [self.view setBackgroundColor:[UIColor whiteColor]];

    NSArray* items = @[@"Class Method",@"Member Method",@"Time Picker 1",@"Time Picker 2"];
    UISegmentedControl* segmentControl = [[UISegmentedControl alloc] initWithItems:items];
    [segmentControl addTarget:self action:@selector(onClick:) forControlEvents:UIControlEventEve
ntValueChanged];
    segmentControl.selectedSegmentIndex = 0;
    [segmentControl setFrame:CGRectMake(15, 70, AUCCommonUIGetScreenWidth() - 30, 30)];
    [self.view addSubview:segmentControl];

    // The label is used to display the item selected by pickerView.
    self.textLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 110, 220, 50)];
    self.textLabel.frame = CGRectOffset(self.textLabel.frame, (AUCCommonUIGetScreenWidth()-s
elf.textLabel.frame.size.width)/2, 0);
    self.textLabel.layer.cornerRadius = 12.f;
    self.textLabel.lineBreakMode = NSLineBreakByWordWrapping;
    self.textLabel.numberOfLines = 0;
    self.textLabel.textAlignment = NSTextAlignmentCenter;
    [self.view addSubview:self.textLabel];

    // pickerView created by the class method.
    self.apPickerView = [AUIDatePicker pickerViewWithTitle:nil];
    self.apPickerView.delegate = self;
    self.apPickerView.tag = 1000;
    [self.view addSubview:self.apPickerView];
}

```

```

        [self.apPickerView show];

        // pickerView created by the member method.
        _apPickerView2 = [[AUDatePicker alloc] initWithFrame:CGRectMake(0, 200, 200, 200) withTitle:nil];
        _apPickerView2.delegate = self;
        _apPickerView2.tag = 1001;
        [self.view addSubview:_apPickerView2];

        // Time picker 1.
        self.apPickerView3 = [AUDatePicker pickerViewWithTitle:@"Select time"];
        self.apPickerView3.tag = 1002;
        self.apPickerView3.isDatePicker = YES;
        NSDate * currentntDate = [NSDate date];
        NSDate * minxDate = [NSDate dateWithTimeInterval:-(3600*24*3000) sinceDate:currentntDate];
        NSDate * maxDate = [NSDate dateWithTimeInterval:3600*24*3000 sinceDate:currentntDate];
        [self.apPickerView3 setTimeDate:minDate:minxDate MaxDate:maxDate];
        [self.apPickerView3 setCurrentDate:currentntDate];
        [self.view addSubview:self.apPickerView3];

        // Time picker 2.
        self.apPickerView4 = [AUDatePicker pickerViewWithTitle:@"Select time"];
        self.apPickerView4.tag = 1003;
        self.apPickerView4.isDatePicker = YES;
        [self.apPickerView4 setTimeDate:minDate:minxDate MaxDate:maxDate];
        [self.apPickerView4 setCurrentDate:currentntDate];
        NSDate * selectDate = [NSDate dateWithTimeInterval:3600*24*888 sinceDate:currentntDate];
        [self.apPickerView4 setAUDatePickerDate:selectDate animated:NO];
        [self.view addSubview:self.apPickerView4];

        // self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButtonTitle target:self];
    }

    - (void)didReceiveMemoryWarning
    {
        [super didReceiveMemoryWarning];
        // Dispose of any resources that can be recreated.
    }

#pragma mark - Button onClick
    - (void)onBarButtonClick:(id)sender
    {
    }

    - (void)onClick:(id)sender
    {
        [self.apPickerView hide];
        [self.apPickerView2 hide];
        [self.apPickerView3 hide];
        [self.apPickerView4 hide];
        UISegmentedControl* segmentControl = (UISegmentedControl*)sender;
    }

```

```

switch (segmentControl.selectedSegmentIndex) {
    case 0:
        [self.apPickerView show];
        break;
    case 1:

        [self.apPickerView2 show];
        break;
    case 2:

        [self.apPickerView3 show];
        break;
    case 3:

        [self.apPickerView4 show];
        break;

    default:
        break;
}
}

#pragma APPickerDelegate delegate
- (void)cancelPickerView:(AUDatePicker *)pickerView
{
    switch (pickerView.tag) {
        case 1000:
            [self.apPickerView hide];
            break;
        case 1001:
            [self.apPickerView2 hide];
            break;
        case 1002:
            [self.apPickerView3 hide];
            break;
        case 1003:
            [self.apPickerView4 hide];
            break;

        default:
            break;
    }
    [self.textLabel setText:@"The callback when the Cancel button is clicked."];
}

- (void)selectedPickerView:(AUDatePicker *)pickerView
{
    NSInteger index = [pickerView.pickerView selectedRowInComponent:0];
    NSString *result = [self.yearArray objectAtIndex:index];

    index = [pickerView.pickerView selectedRowInComponent:1];

```

```

        result = [result stringByAppendingString:[NSString stringWithFormat:@"% " %@", [self.monthArray objectAtIndex:index]]];

        index = [pickerView.pickerView selectedRowInComponent:2];
        result = [result stringByAppendingString:[NSString stringWithFormat:@"% " %@", [self.nameArray objectAtIndex:index]]];

        [self.textLabel setText:result];
    }

#pragma UIPickerView delegate
- (NSString *)pickerView:(UIPickerView *)pickerView titleForRow:(NSInteger)row forComponent:(NSInteger)component
{
    if (component == 0) {
        return [self.yearArray objectAtIndex:row];
    } else if (component == 1){
        return [self.monthArray objectAtIndex:row];
    } else {
        return [self.nameArray objectAtIndex:row];
    }
}

- (NSInteger)numberOfComponentsInPickerView:(UIPickerView *)pickerView
{
    return 3;
}

- (NSInteger)pickerView:(UIPickerView *)pickerView numberOfRowsInComponent:(NSInteger)component
{
    if (component == 0) {
        return [self.yearArray count];
    } else if (component == 1){
        return [self.monthArray count];
    } else {
        return [self.nameArray count];
    }
}

@end

```

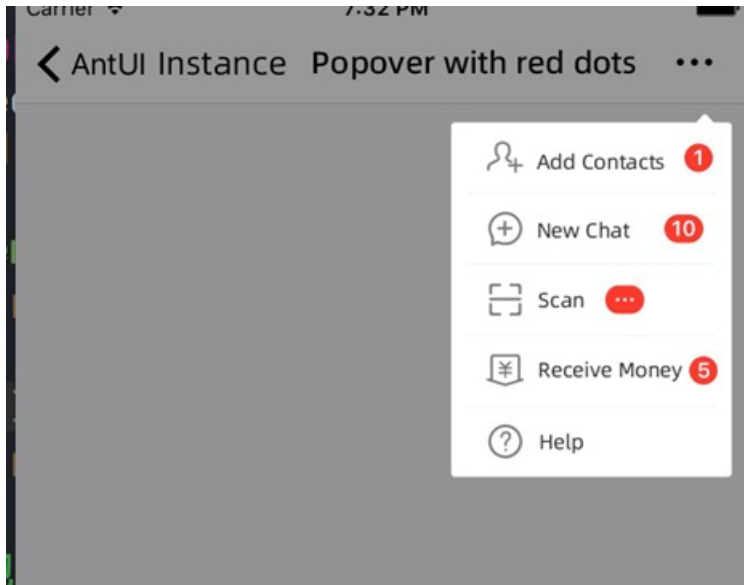
1.5.5.3. Menu component

The floating layer menu provides a menu containing icons and an option list.

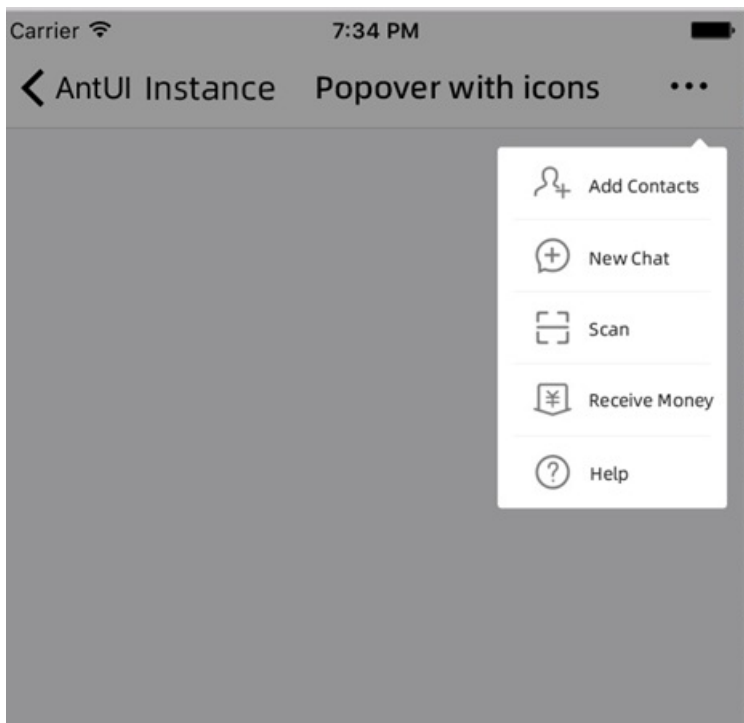
When using it, you need to change the APNavPopview and APNavItemView in the original Ant UI framework to AUFloatMenu and AUNavItemView.

Sample images

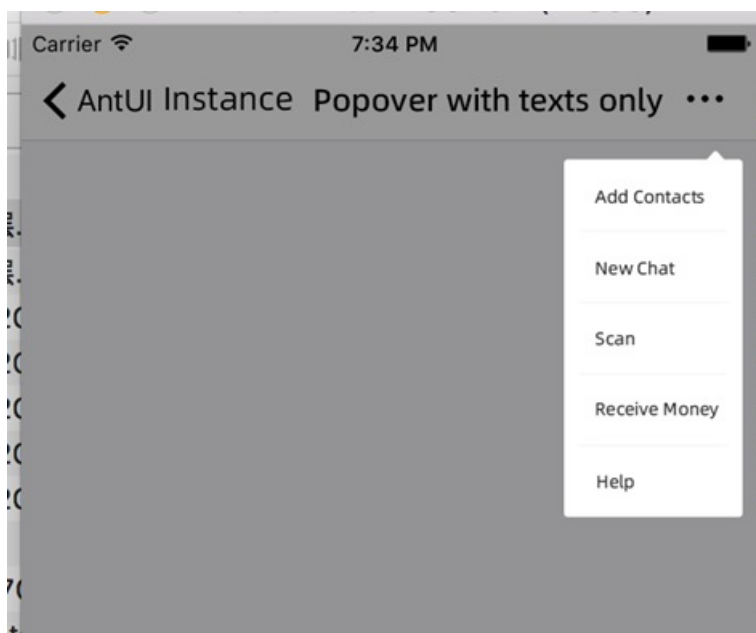
- Popover with red dots



- Popover with icons



- Popover with texts only



API description

- AUFloatMenu.h

```
//
//  AUFloatMenu.h
//  AntUI
//

#import <UIKit/UIKit.h>
/* The notice of popview hiding. */
static NSString * const APExtUIPopoverDismissedNotification = @"APExtUIPopoverDismissedNotification";

@class AUNavItemView;

/*!
 @class      AUFloatMenu
 @abstract   UIView
 @discussion floatViewMenu The floating layer.
 */

@interface AUFloatMenu : UIView<UIGestureRecognizerDelegate>
@property(n nonatomic, assign) CGFloat marginToRight; // The right margin of a white popview, which is 10 by default.

/**
 * Create a floating menu view.
 *
 * @param position The position where the floating menu is displayed on the screen.
 * @param items The array of displayed content, which is generally an AUNavItemView object.
 *
 * @return The floating menu view.
 */
+ (AUFloatMenu *)showAtPostion:(CGPoint)position items:(NSArray<AUNavItemView *> *)items;
```

```
/**
 * Create a floating menu view.
 *
 * @param position The position where the floating menu is displayed on the screen.
 * @param originY The y-coordinate value of the floating menu on the screen.
 * @param items The array of displayed content, which is generally an AUNavItemView object.
 *
 * @return The floating menu view.
 */
+ (AUFloatMenu *)showAtPostion:(CGPoint)position startOriginY:(CGFloat)originY items:(NSArray<AUNavItemView *> *)items;

/**
 * The interface method for hiding a floating menu.
 */
- (void)dismiss;

/**
 * After the menu is open, RPC is executed to load dynamically delivered menu items. After
 * RPC is completed, the update() method is called to remove the original view and add a new view.
 */
- (void)updateWithItems:(NSArray<AUNavItemView *> *)items;

@end
```

• AUNavItemView.h

```
//
//  AUNavItemView.h
//  AntUI
//

#import <UIKit/UIKit.h>

typedef NS_ENUM(NSInteger, AUCurrentTabType) {
    AUCurrentTabTypeHome = 0,
    AUCurrentTabTypeKouBei,
    AUCurrentTabTypeFriend,
    AUCurrentTabTypeWealth
};

/*!
 @class          AUNavItemView
 @abstract       UIView
 @ Discussion floatMenu The view of each column at the floating layer.
 */
@interface AUNavItemView : UIView
/**
 * title
 */
@property (nonatomic, strong) NSString *itemTitle;

@property (nonatomic, strong, readonly) UIFont *titleFont;
```

```

/**
 * The normal state.
 */
@property(nonatomic, strong) UIImage *normalStateIconImage;

/**
 * IconFont Name: If iconFont needs to be set, the AUNavItemView.h but not AUFloatMenu.h API is invoked.
 */
@property(nonatomic, strong) NSString *normalStateIconFontName;

/**
 * If widgetId is set, badgeNumber does not need to be set.
 */
@property(nonatomic, strong) NSString *badgeNumber;

/**
 * widgetId
 */
@property(nonatomic, copy) NSString *widgetId;

/**
 * The required prompt text for VoiceOver. The default value is the value of itemTitle. If itemTitle is not set, you need to manually set this attribute to support VoiceOver.
 */
@property(nonatomic, strong) NSString *voiceOverText;

@property(nonatomic, assign) BOOL isNavigationItem;

@property(nonatomic, assign, readonly) CGFloat touchEventMargin;

@property(nonatomic, assign) AUCurrentTabType currentTabType;

@property(nonatomic, assign, readonly) CGFloat marginBetweenIconTitle;

@property(nonatomic, assign, readonly) CGFloat marginBetweenLeftIcon;

@property(nonatomic, assign, readonly) CGFloat badgeViewWidth;

/**
 * This method needs to be rewritten for a sub-class, and then the clicking event is processed.
 */
- (void)onClicked;

/**
 * Return the size of the icon view.

 @return size
 */
- (CGSize) iconViewSize;

@end

```

Sample code

- Example of a popover with red dots:

```
//
//  APNavPopoverViewController.m
//  UIDemo
//

#import "APNavPopoverViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopoverViewController ()

@end

@implementation APNavPopoverViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = RGB(0xF5F5F9);

    self.navigationItem.title = @"Popover with red dots";
    UIBarButtonItem *rightItem = [[UIBarButtonItem alloc] initWithImage:[UIImage imageNamed:@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self action:@selector(onClick:)];
    self.navigationItem.rightBarButtonItem = rightItem;
    [[AUNavigationManager sharedInstance] registerAUObject:[AntUIShellObject alloc] init];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
    NSMutableArray *array = [[NSMutableArray alloc] initWithCapacity:4];

    NSArray *items = @[@"Add Contacts",@"New Chat",@"Scan",@"Receive Money",@"Help"];
    int i = 0;
    for (NSString *typeName in items) {
        AUNavItemView *item = [[AUNavItemView alloc] initWithFrame:CGRectMake(20, 0, 0, 40)];

        item.itemTitle = typeName;
        item.isNavigationItem = NO;
        // iconfont is supported.
        // item.nomarlStateIconFontName = kICONFONT_USER_ADD;
        if (i == 0 ) {
```

```

        item.badgeNumber = @"1";
        UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
        item.nomarlStateIconImage = image;
    } else if(i == 1) {
        item.badgeNumber = @"10";
        UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
        item.nomarlStateIconImage = image;
    } else if(i == 2) {
        item.badgeNumber = @"100";
        UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
        item.nomarlStateIconImage = image;
    } else if(i == 3) {
        item.badgeNumber = @"5";
        UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
        item.nomarlStateIconImage = image;
    } else if(i == 4) {
        UIImage *image = [UIImage imageNamed:@"ap_help.png"];
        item.nomarlStateIconImage = image;
    }
    i++;

    [array addObject:item];
}

[AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id)sender
{
}

@end

```

- Example of a popover with icons:

```

//
//  APNavPopoverViewController.m
//  UIDemo
//

#import "APNavPopoverNoneRedViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopoverNoneRedViewController ()

@end

@implementation APNavPopoverNoneRedViewController

```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = RGB(0xF5F5F9);

    self.navigationItem.title = @"Popover with icons";
    UIBarButtonItem *rightItem = [[UIBarButtonItem alloc] initWithImage:[UIImage imageNamed:
@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self action:@
selector(onClick:)];
    self.navigationItem.rightBarButtonItem = rightItem;
    [[AUNavigationManager sharedInstance] registerAUObject:[AntUIShellObject alloc] init]];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
    NSMutableArray *array = [[NSMutableArray alloc] initWithCapacity:4];

    NSArray *items = @[@"Add Contacts",@"New Chat",@"Scan",@"Receive Money",@"Help"];
    int i = 0;
    for (NSString *typeName in items) {
        AUNavItemView *item = [[AUNavItemView alloc] initWithFrame:CGRectMake(20, 0, 0, 40)]
;
        item.itemTitle = typeName;
        item.isNavigationItem = NO;
        // iconfont is supported.
        // item.nomarlStateIconFontName = kICONFONT_USER_ADD;
        if (i == 0 ) {
            // item.badgeNumber = @"1";
            UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
            item.nomarlStateIconImage = image;
        } else if(i == 1) {
            // item.badgeNumber = @"10";
            UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
            item.nomarlStateIconImage = image;
        } else if(i == 2) {
            // item.badgeNumber = @"100";
            UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
            item.nomarlStateIconImage = image;
        } else if(i == 3) {
            // item.badgeNumber = @"5";
            UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
            item.nomarlStateIconImage = image;
        } else if(i == 4) {
            UIImage *image = [UIImage imageNamed:@"ap_help.png"];
            item.nomarlStateIconImage = image;
        }
        i++;

        [array addObject:item];
    }
}
```

```
[AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id)sender
{
}

@end
```

- Example of a popover with texts only:

```
//
//  APNavPopoverViewController.m
//  UIDemo
//

#import "APNavPopoverOnlyViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopoverOnlyViewController ()

@end

@implementation APNavPopoverOnlyViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = RGB(0xF5F5F9);

    self.navigationItem.title = @"Popover with texts only";
    UIBarButtonItem *rightItem = [[UIBarButtonItem alloc] initWithImage:[UIImage imageNamed:@"APCommonUI_ForDemo.bundle/more.png"] style:UIBarButtonItemStylePlain target:self action:@selector(onClick:)];
    self.navigationItem.rightBarButtonItem = rightItem;
    [[AURegisterManager sharedInstance] registerAUObject:[AntUIShellObject alloc] init]];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)onClick:(id)sender
{
    NSMutableArray *array = [[NSMutableArray alloc] initWithCapacity:4];

    NSArray *items = @[@"Add Contacts",@"New Chat",@"Scan",@"Receive Money",@"Help"];
    int i = 0;
```

```

    for (NSString *typeName in items) {
        AUNavItemView *item = [[AUNavItemView alloc] initWithFrame:CGRectMake(0, 0, 0, 40)];
        item.itemTitle = typeName;
        item.isNavigationItem = NO;
        // iconfont is supported.
        // item.nomarlStateIconFontName = kICONFONT_USER_ADD;
        // if (i == 0 ) {
        ////
            item.badgeNumber = @"1";
        //      UIImage *image = [UIImage imageNamed:@"ap_add_friend.png"];
        //      item.nomarlStateIconImage = image;
        //    } else if(i == 1) {
        ////
            item.badgeNumber = @"10";
        //      UIImage *image = [UIImage imageNamed:@"ap_group_talk.png"];
        //      item.nomarlStateIconImage = image;
        //    } else if(i == 2) {
        ////
            item.badgeNumber = @"100";
        //      UIImage *image = [UIImage imageNamed:@"ap_scan.png"];
        //      item.nomarlStateIconImage = image;
        //    } else if(i == 3) {
        ////
            item.badgeNumber = @"5";
        //      UIImage *image = [UIImage imageNamed:@"ap_qrcode.png"];
        //      item.nomarlStateIconImage = image;
        //    } else if(i == 4) {
        //      UIImage *image = [UIImage imageNamed:@"ap_help.png"];
        //      item.nomarlStateIconImage = image;
        //    }
        i++;

        [array addObject:item];
    }

    [AUFloatMenu showAtPostion:CGPointMake(0, 0) startOrignY:70 items:array];
}

- (void)onBarButtonClick:(id) sender
{
}

@end

```

1.5.5.4. Recording status layer

AURecordFloatTip is a floating layer that displays the recording status. It provides users with more direct voice recording experience.

Sample image



API description

```
@interface AURecordFloatTip : UIView

@property (nonatomic, strong) UILabel *messageLabel; // The prompt for voice recording. Default value: Recording...

// Show the floating layer.
- (void)showRecordingInView:(UIView *)view;

// Hide the floating layer.
- (void)dismissRecordView;

@end
```

Code sample

```
AURecordFloatTip *_tipView = [[AURecordFloatTip alloc] init];
[_tipView showRecordingInView:self.view];
```

1.5.5.5. Image dialog

AUImageDialog is a dialog box with images. The dialog box has rounded corners, and the style can be customized, as shown in the following sample images. The window level of AUImageDialog is:

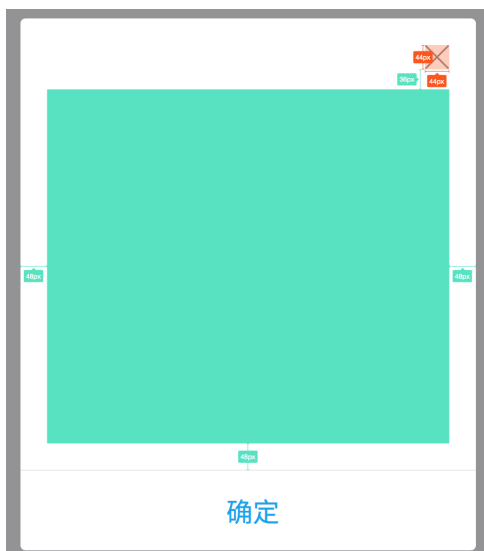
```
self.windowLevel = UIWindowLevelAlert - 1 .
```

Sample images

- With common buttons



- Custom style



- With a large image



API description

```
// The index corresponding to button clicking.
typedef NS_ENUM(NSInteger, AUIImageDialogButtonIndex) {
    AUIImageDialogButtonIndex_Close = -2,
    AUIImageDialogButtonIndex_Link = -1,
    AUIImageDialogButtonIndex_Action = 0
};

/**
    The image dialog box is a special-style dialog box meeting the UED requirements, the shape of the images is round.
    Two modes are available:
        Common image mode: The Add button is a common button.
        Action button mode: An action button and a link button can be added. A cross icon (X) is displayed in the upper right corner for users to exit.
    The buttons in one mode cannot be added in the other mode. Otherwise, the adding cannot pass the assert check.
 */
@interface AUIImageDialog : AUIDialogBaseView

/**
    The method of dialog box initialization without the button title.

    @param image    The image.
    @param title    The title.
    @param message  The message details.
    @param delegate The AUIDialogDelegate-compliant protocol object.
    @return         The AUIImageDialog instance.
 */
- (instancetype)initWithImage:(UIImage *)image
                        title:(NSString *)title
                    message:(NSString *)message
                delegate:(id<AUIDialogDelegate>)delegate;

/**
```

```

The method of dialog box initialization with the button title.

@param image      The image.
@param title      The title.
@param message    The message details.
@param delegate   The AUDialogDelegate-compliant protocol object.
@param buttonTitle The list of button title parameters.
@return          The AUIImageDialog instance.
*/
- (instancetype)initWithImage:(UIImage *)image
                        title:(NSString *)title
                        message:(NSString *)message
                        delegate:(id<AUDialogDelegate>)delegate
                        buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_N
IL_TERMINATION;

/**
The initialization method with a blue action button.

@param image      The image.
@param title      The title.
@param message    The message details.
@param delegate   The AUDialogDelegate-compliant protocol object.
@param actionTitle The title of the action button.
@return          The AUIImageDialog instance.
*/
- (instancetype)initWithImage:(UIImage *)image
                        title:(NSString *)title
                        message:(NSString *)message
                        delegate:(id<AUDialogDelegate>)delegate
                        actionButtonTitle:(NSString *)actionTitle;

/**
The initialization method with a blue action button and a link button.

@param image      The image.
@param title      The title.
@param message    The message details.
@param delegate   The AUDialogDelegate-compliant protocol object.
@param linkText    The link text.
@param actionTitle The title of the action button.
@return          The AUIImageDialog instance.
*/
- (instancetype)initWithImage:(UIImage *)image
                        title:(NSString *)title
                        message:(NSString *)message
                        delegate:(id<AUDialogDelegate>)delegate
                        linkText:(NSString *)linkText
                        actionButtonTitle:(NSString *)actionTitle;

- (instancetype)init NS_UNAVAILABLE;

- (instancetype)initWithCustomView:(UIView *)customView; // The custom view, with the X
button in the upper right corner by default.

```

```

/**
    The dialog box display method.
    */
    - (void)show;

/**
    Set the text color to gray. Default value: NO.
    */
    - (void)setGrayMessage:(BOOL)grayMessage;

/**
    Set the text alignment mode.

    @param alignment    The alignment mode.
    */
    - (void)setMessageAlignment:(NSTextAlignment)alignment;

/**
    Set the custom image size. The width cannot exceed the dialog box's maximum width of 270. Default value: 135 x 135.
    */
    - (void)configImageAreaSize:(CGSize)imageSize;

/**
    Add a common button and its callback method (The common button cannot contain an action or a link).

    @param buttonTitle    The common button title.
    @param actionBlock    The callback of the button.
    */
    - (void)addButton:(NSString *)buttonTitle actionBlock:(AUIDialogActionBlock)actionBlock;

/**
    Add an action button and its callback method.

    @param actionTitle    The title of the action button.
    @param actionBlock    The callback of the action button.
    */
    - (void)addActionButton:(NSString *)actionTitle actionBlock:(AUIDialogActionBlock)actionBlock;

/**
    Add a link button and its callback method.

    @param linkText        The link text.
    @param actionBlock    The callback of the link button.
    */
    - (void)addLinkButton:(NSString *)linkText actionBlock:(AUIDialogActionBlock)actionBlock;

/**
    Hide the close button in the upper right corner.
    */

```

```
- (void)setCloseButtonHidden:(BOOL) hidden;
```

API for a large image style AUIImageDialog

```
/**
 * The image dialog box has a special UED-required style.
 * Style: The large icon style. In this style, the image has a fixed height of 312 px, and
 * the close button is in the upper right corner of the image.
 * The icon font of the close button. Default value: white.
 */

@interface AUIImageDialog (largeImageStyle)

/**
 * The method of dialog box initialization without the button title.
 *
 * @param image      The image.
 * @param title      The title.
 * @param message    The message details.
 * @param delegate   The AUIImageDialog-compliant protocol object.
 * @return           The AUIImageDialog instance.
 */
- (instancetype)initWithLargeImage:(UIImage *)image
                           title:(NSString *)title
                           message:(NSString *)message
                           delegate:(id<AUIImageDialogDelegate>)delegate;

/**
 * Set the color of the close button in the upper right corner. Default value: white.
 */
- (void)resetCloseIconColor:(UIColor *)color;

@end
```

Sample code

- With common buttons

```
UIImage *image = [UIImage imageNamed:@"panghu.jpg"];
AUIImageDialog *dialog = [[AUIImageDialog alloc] initWithImage:image title:@"Goda Takeshi"
message:@"Strict match. The one not in the delegate is not a standard control. The one
not in the delegate but has been used in many places should be delivered in the candidate
control set. A delegate, such as the title delegate, may not be implemented as a single c
ontrol." delegate:self];
[dialog addButton:@"Cancel" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];
```

- Custom style

```
UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];
AUImageDialog *dialog = [[AUImageDialog alloc] initWithCustomView:customView];
[dialog addButton:@"Cancel" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];
```

- With a large image

```
UIImage *image = [UIImage imageWithColor:[UIColor colorWithRGB:0xD8D8D8] size:CGSizeMake(
100, 100)];
AUImageDialog *dialog = [[AUImageDialog alloc] initWithLargeImage:image title:@"Title i
n a line" message:@"Illustrate the status and prompt a solution. No more than two lines."
delegate:self];
[dialog addButton:@"Cancel" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog resetCloseIconColor:[UIColor redColor]];
[dialog show];
```

1.5.5.6. Input dialog

AUInputDialog specifies the style of a pop-up window with an input box. The Window level of the pop-up window follows the logic `self.windowLevel = UIWindowLevelAlert - 1`.

Sample images



API description

```
@interface AUInputDialog : AUIDialogBaseView

/// The input box.
```

```

// The input box.
@property (nonatomic, strong, readonly) UITextField *textField;

/**
Specify whether this instance is displayed. This applies when a pointer points at this instance.
If another dialog box overrides this one, the attribute value is fixed as YES.
*/
@property (nonatomic, assign, readonly) BOOL isDisplay;

/**
* The title.
*/
@property (nonatomic, strong) NSString *title;

/**
* The text message.
*/
@property (nonatomic, strong) NSString *message;

/**
The method of dialog box initialization without the button title.

@param title    The title.
@param message  The message details.
@return         The AUInputDialog instance.
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message;

/**
The AUInputDialog instance initialization method.

@param title        The title.
@param message      The message details.
@param placeholder  The placeholder in the text box.
@param delegate     The delegate object.
@param buttonTitle  The button title.
@return            The AUInputDialog instance.
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message
placeholder:(NSString *)placeholder
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

- (instancetype)initWithCustomView:(UIView *)customView; // The custom view.

/// The disabled initialization method.
- (instancetype)init NS_UNAVAILABLE;

/**
The dialog box display method.
*/

```

```

- (void)show;

/**
The method of closing the dialog box. If will/didDismissWithButtonIndex is monitored, the i
ndex called back is 0 by default.
*/
- (void)dismiss;

/**
Hide all dialog views in the dialog window.
*/
+ (void)dismissAll;

/**
Set the color of the text to gray. Default value: YES.
*/
- (void)setGrayMessage:(BOOL)grayMessage;

/**
Set the text alignment mode.

@param alignment The alignment mode.
*/
- (void)setMessageAlignment:(NSTextAlignment)alignment;

/**
Add a button and its callback method.

@param buttonTitle The button title.
@param actionBlock The callback of the button tapping action.
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

```

Code sample

• Common style

```

AUInputDialog *dialog = [[AUInputDialog alloc] initWithTitle:@"Title" message:@"The message
may contain the sound icon and button of a notification alarm. This message can be sent to
" placeholder:@"To friends." delegate:self buttonTitles:@"Cancel", @"Main action", nil];
[dialog show];

```

• Custom style

```

UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];
AUInputDialog *dialog = [[AUInputDialog alloc] initWithCustomView:customView];
[dialog addButton:@"Cancel" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];

```

1.5.5.7. Toast component

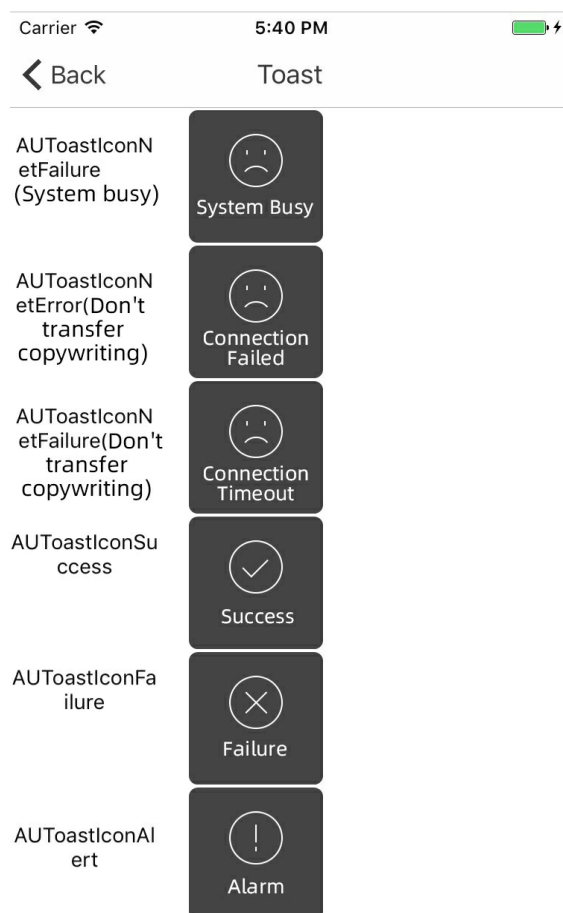
AUToast defines Toast controls for mPaaS. AUToast is developed from APToast of APCommonUI. Use AUToast instead of APToast.

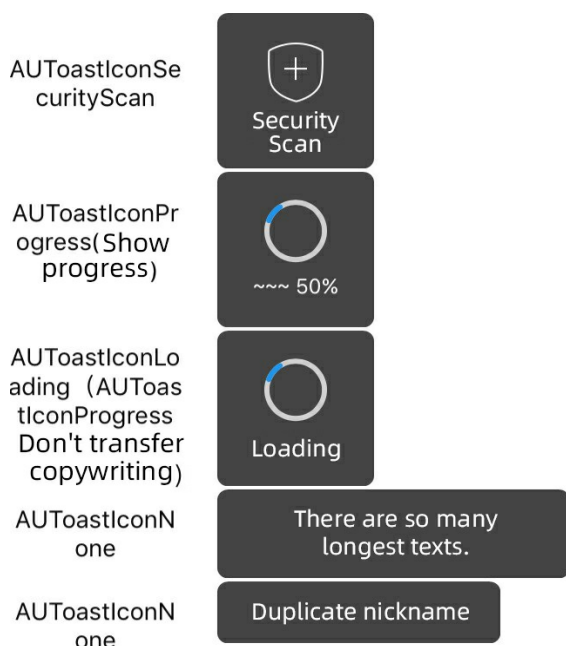
This component contains two types of Toast controls:

- Common Toast
- Modal Toast

The modal Toast has a transparent background layer, but the common Toast does not. Users cannot click the area covered by the background layer.

Sample images





API description

```
// The declaration of the log output function, which is set by the external system.
typedef void(*AUTOastLogFunc)(NSString *tag, NSString *format, ...);
extern AUTOastLogFunc g_ToastExternLogFunc; // The global variables of the log output function are set by external system.
#define AUTOastLog(fmt, ...) {if(g_ToastExternLogFunc)g_ToastExternLogFunc(@"@AUTOast",fmt, ##__VA_ARGS__);}
```

```
#define AUTOast_Default_Duration 2.0 // AUTOast Default display duration.
#define AUTOast_Strong_Duration 1.5 // AUTOast Display duration of the strong prompt.
#define AUTOast_Weak_Duration 1.0 // AUTOast Display duration of the weak prompt.
```

```
/**
 * Add a new toast icon to the end of the existing icons instead of in the middle. Otherwise, an error will occurs in business use.
 */
typedef enum{
    AUTOastIconNone = 0, // No icon.
    AUTOastIconSuccess, // The success icon.
    AUTOastIconFailure, // The failure icon.
    AUTOastIconLoading, // The loading icon.
    AUTOastIconNetFailure, // Network failure.
    AUTOastIconSecurityScan, // Security scanning.
    AUTOastIconNetError, // The network error causing connection failure.
    AUTOastIconProgress, // The loading icon indicating the loading progress.
    AUTOastIconAlert, // The alarm icon.
} AUTOastIcon;
```

```
/**
 * The Toast control.
 */
@interface AUTOast : UIView
```

```

@property (nonatomic, assign) CGFloat xOffset; // Set the offset to the central point of the
parent view in the x-axis.
@property (nonatomic, assign) CGFloat yOffset; // Set the offset to the central point of the
parent view in the y-axis.

/*
 * The modal display prompt displayed in the key window. The system does not respond to user
operations.
 * Call the dismissToast method to hide the Toast.
 *
 * @param text The displayed text. Default value: loading.
 * @param logTag The log ID.
 *
 * @return The displayed Toast object.
 */
+ (AUToast *)presentToastWithText:(NSString *)text
logTag:(NSString*)logTag;

/**
 * Show the Toast. To hide the Toast, call the dismissToast method.
 *
 * @param superview The parent view.
 * @param text      Displayed text.
 * @param logTag    The log tag.
 *
 * @return The displayed Toast object.
 */
+ (AUToast *)presentToastWithin:(UIView *)superview
text:(NSString *)text
logTag:(NSString*)logTag;

/**
 * Show the Toast. To hide the Toast, call the dismissToast method.
 *
 * @param superview The parent view.
 * @param icon      The icon type.
 * @param text      Displayed text.
 * @param logTag    The log tag.
 *
 * @return The displayed Toast object.
 */
+ (AUToast *)presentToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
logTag:(NSString*)logTag;

/**
 * Show the Toast.
 *
 * @param superview The parent view.
 * @param icon      The icon type.
 * @param text      Displayed text.
 * @param duration  The display duration.
 * @param logTag    The log tag.

```

```

*
* @return The displayed Toast object.
*/
+ (AUTOast *)presentToastWithin:(UIView *)superview
withIcon:(AUTOastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag;

/**
* Show the Toast.
*
* @param superview      In which view of Toast is displayed.
* @param icon           The icon type.
* @param text           Displayed text.
* @param duration       Display duration.
* @param logTag         The log tag.
* @param completion     Callback after Toast automatically disappears.
*
* @return The displayed Toast object.
*/
+ (AUTOast *)presentToastWithin:(UIView *)superview
withIcon:(AUTOastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag
completion:(void (^)(void))completion;

/**
* Show the Toast.
*
* @param superview      In which the view of Toast is displayed.
* @param icon           The icon type.
* @param text           Displayed text.
* @param duration       Display duration.
* @param delay          Display delay duration.
* @param logTag         The log tag.
* @param completion     Callback after Toast automatically disappears.
*
* @return The displayed Toast object.
*/
+ (AUTOast *)presentToastWithin:(UIView *)superview
withIcon:(AUTOastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
delay:(NSTimeInterval)delay
logTag:(NSString*)logTag
completion:(void (^)(void))completion;

```

```

/*
 * Show the modal Toast. To hide the Toast, call the dismissToast method.
 * Different from the common Toast, the modal Toast has a transparent background layer, which prevents users from clicking the screen.
 *
 * @param superview The parent view.
 * @param text      Displayed text.
 * @param logTag    The log tag.
 *
 * @return The displayed Toast object.
 */
+ (AUIToast *)presentModalToastWithin:(UIView *)superview
text:(NSString *)text
logTag:(NSString*)logTag;

/**
 * Show the modal Toast.
 * Different from the common Toast, the modal Toast has a transparent background layer, which prevents users from clicking the screen.
 *
 * @param superview In which view of Toast is displayed.
 * @param icon      The icon type.
 * @param text      Displayed text.
 * @param duration  Display duration.
 * @param logTag    The log tag.
 * @param completion Callback after Toast automatically disappears.
 *
 * @return The displayed Toast object.
 */
+ (AUIToast *)presentModalToastWithin:(UIView *)superview
withIcon:(AUIToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag
completion:(void (^)(void))completion;

/**
 * Show the modal Toast.
 * Different from the common Toast, the modal Toast has a transparent background layer, which prevents users from clicking the screen.
 *
 * @param superview In which the view of Toast is displayed.
 * @param icon      Icon type.
 * @param text      Displayed text.
 * @param duration  Display duration.
 * @param delay     Display delay duration.
 * @param logTag    The log tag.
 * @param completion Callback after Toast automatically disappears.
 *
 * @return The displayed Toast object.
 */
+ (AUIToast *)presentModalToastWithin:(UIView *)superview

```

```
withIcon: (AUToastIcon) icon
text: (NSString *) text
duration: (NSTimeInterval) duration
delay: (NSTimeInterval) delay
logTag: (NSString*) logTag
completion: (void (^)()) completion;

/*
 * Hide the Toast.
 */
- (void)dismissToast;

/**
 * Set the prefix text of the progress. If this parameter is not set, the default value is "
Loading data".
 * The setting is effective only when the Toast type is AUToastIconProgress. Otherwise, ignore
this parameter.
 *
 * @param prefix The text.
 */
- (void)setProgressPrefix: (NSString*) prefix;

/**
 * Show the data loading progress in percentage.
 * The setting is effective only when the Toast type is AUToastIconProgress. Otherwise, ignore
this parameter.
 *
 * @param value Currently loaded data. The value range is 0.0 to 1.0.
 */
- (void)setProgressText: (float) value;

@end
```

Sample code

```
[AUIToast presentToastWithin:self.view withIcon:AUIToastIconNetFailure text:@"System Busy" logTag:@"demo"];
[AUIToast presentToastWithin:self.view withIcon:AUIToastIconSuccess text:@"Success" logTag:@"demo"];
[AUIToast presentToastWithin:self.view withIcon:AUIToastIconFailure text:@"Failure" logTag:@"demo"];
[AUIToast presentToastWithin:self.view withIcon:AUIToastIconAlert text:@"Alarm" logTag:@"demo"];

// Loading.
[AUIToast presentToastWithin:self.view withIcon:AUIToastIconLoading text:nil logTag:@"demo"];

// Set the scenario where the progress appears.
AUIToast *toast = [AUIToast presentToastWithin:self.view withIcon:AUIToastIconProgress text:@"Loading" logTag:@"demo"];
toast.origin = point;
[toast setProgressPrefix:@"~~~"];
[toast setProgressText:0.5];

// The modal Toast.
[AUIToast presentModalToastWithin:weakSelf.view withIcon:AUIToastIconLoading text:@"Modal toast, There are so many longest texts, is too long" duration:3 logTag:@"demo" completion:NULL];
[AUIToast presentModalToastWithin:weakSelf.view withIcon:AUIToastIconLoading text:@"Modal toast, There are so many longest texts, is too long" duration:3 delay:2 logTag:@"demo" completion:NULL];
```

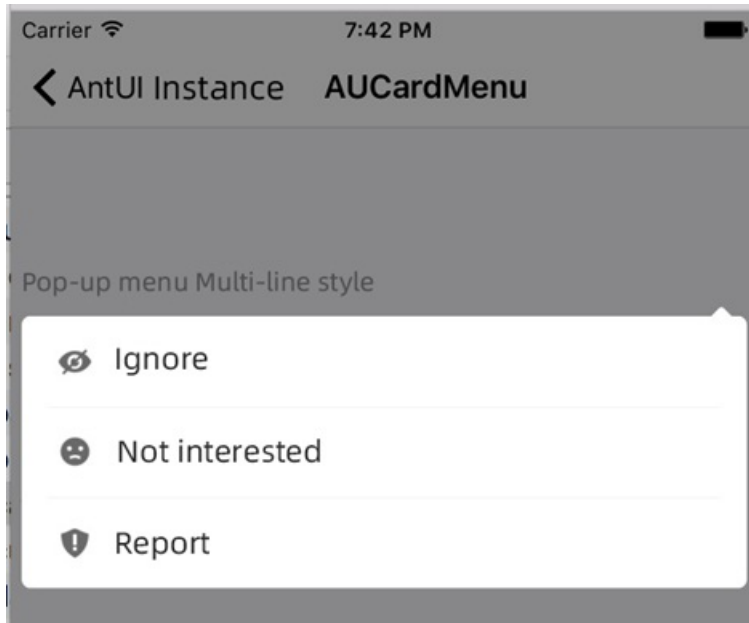
1.5.5.8. Card menu

The card menu is used to pop up a selection menu when the user clicks a card on the client page. In iOS,

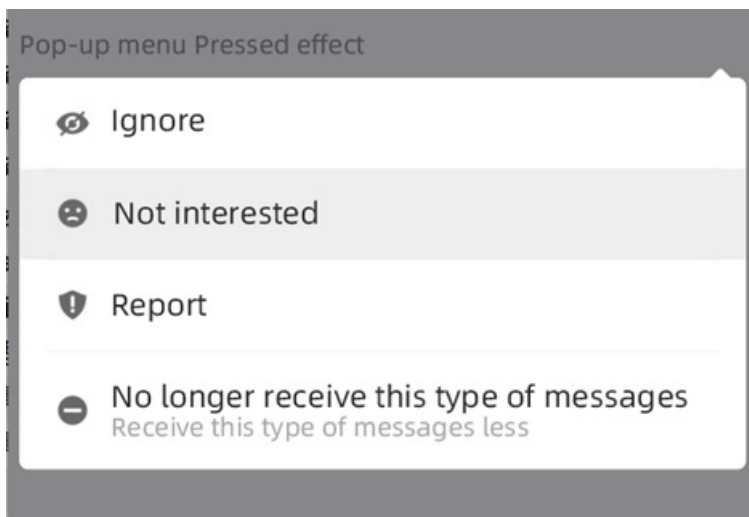
`beeviews:BEEPopMenuView` needs to be replaced with `AUCardMenu.h`.

Sample images

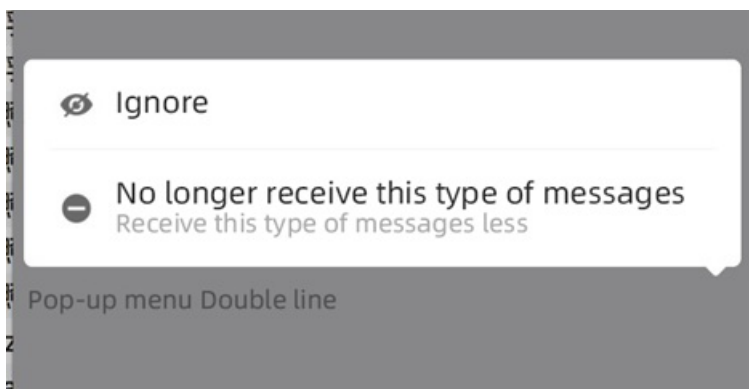
- Pop-up menu / Multi-line style



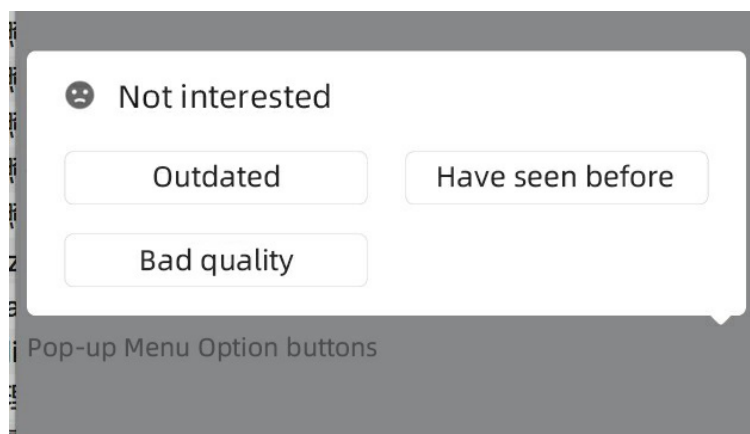
- Pop-up menu / Pressed effect



- Pop-up menu / Double line



- Pop-up menu + Option buttons



API description

- AUCardMenu.h

```
//
//  AUCardMenu.h
//  AntUI
//

@class AUMultiStyleCellView;
@class AUWindow;
/*!
    @class      AUCardMenu
    @abstract   AUWindow
    @discussion The pop-up menu with a mask for the other part of the screen and an arrow-like corner in the upper right of the menu.
 */

@interface AUCardMenu : AUWindow
{

}

/**
 * The AUMultiStyleCellDelegate protocol needs to be implemented for cellView to respond to clicking events.
 * Assign the value in your own viewcontroller. popMenuView.cellView.delegate = self
 */
@property (nonatomic, strong) AUMultiStyleCellView *cellView;

/**
 * The initialization method (highly recommended).
 *
 * @param data      The array storage object model CellDataModel.
 * @param location  The reference point of the arrow-like corner.
 * @param offset    The offset of the arrow-like corner relative to the reference point.
 *
 * @return self
 */

```

```
- (instancetype)initWithData:(NSArray *)data
    location:(CGPoint)location
    offset:(CGFloat)offset;

/**
 * Show the pop-up menu.
 *
 * @param superView superView of PopMenuView
 */
- (void)showPopMenu:(UIView *)superView;

// Hide the pop-up menu, whose calling is also recommended in the dealloc method.
- (void)hidePopMenu;

// Note: If a menu is shown with animation, the menu must be hidden with animation. If a menu is shown without animation, the menu must be hidden without animation.

// Show a menu with animation.
- (void)showPopMenu:(UIView *)superView animation:(BOOL) isAnimation;

// Hide a menu with animation.
- (void)hidePopMenuWithAnimation:(BOOL)isAnimation;

@end
```

- **AUCellDataModel.h**

```
//
//  AUCellDataModel.h
//  AntUI
//

#import <Foundation/Foundation.h>

/*!
@class      AUMultiStyleCellView
@abstract   UIView
@discussion The sub-view in a menu.
*/

@interface AUCellDataModel : NSObject

@property (nonatomic, strong) NSString *iconUrl;
@property (nonatomic, strong) NSString *titleText;
@property (nonatomic, strong) NSString *descText;
@property (nonatomic, strong) NSString *checkMarkUrl; // The check mark URL.
@property (nonatomic, strong) NSString *indicatorUrl; // The right arrow URL.
@property (nonatomic, strong) NSArray *buttonsArray; // NSArray<NSString>
@property (nonatomic, strong) NSDictionary *extendDic; // The extended field for the client
.
@property (nonatomic, assign) BOOL selectedState; // Whether the current model is selected. The default value is NO.

@end
```

- **AUMultiStyleCellView.h**

```
//
//  AUMultiStyleCellView.h
//  AntUI
//

#import <UIKit/UIKit.h>
#import "AUCellDataModel.h"

@class AUMultiStyleCellView;
@protocol AUMultiStyleCellDelegate <NSObject>

@optional
/**
 * The clicking event callback.
 *
 * @param dataModel      The data model corresponding to the clicked view.
 * @param indexPath      The index of the clicked view in CellDataModel. (If CellDataModel.buttonsArray == nil, the default value of row is - 1.)
 */
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath;
- (void)DidClickCellButton:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath;
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath cellView:(AUMultiStyleCellView *)cellView;
@end

/**
 * cellView integrating multiple styles.
 * 1. Icon + Main title
 * 2. Icon + Main title + Subtitle below the main title
 * 3. Icon + Main title + Framed button controls in multiple lines and multiple rows
 */

@interface AUMultiStyleCellView : UIView

@property (nonatomic, weak) id<AUMultiStyleCellDelegate> delegate;
@property (nonatomic, strong) NSArray *cellDataArray;

// If cellDataArray is empty, this method is equal to the initWithFrame method.
- (instancetype)initWithFrame:(CGRect)frame
    cellDataArray:(NSArray *)cellDataArray
    isUpward:(BOOL)isUpward;

// Process the status indicating whether cellView is selected.
- (void)updateSelectedState;

@end
```

Sample code

```
//
// cardMenuController.m
// AntUI
//

#import "cardMenuController.h"
#import "AUCardMenu.h"
#import "AUCellDataModel.h"
#import "AUMultiStyleCellView.h"

@interface cardMenuController ()<AUMultiStyleCellDelegate>

@property (nonatomic, strong) AUCardMenu * popMenuView;

@end

@implementation cardMenuController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    self.view.backgroundColor = RGB(0xF5F5F9);
    UIButton * button = [UIButton buttonWithTypeCustom];
    [button setFrame:CGRectMake(0, 100, self.view.width, 100)];
    [button setTitle:@"Pop-up menu/Multi-line style" forState:UIControlStateNormal];
    [button setTitleColor:RGB(0x888888) forState:UIControlStateNormal];
    [button addTarget:self
        action:@selector(handleButton:)
        forControlEvents:UIControlEventTouchUpInside];
    [button.titleLabel setTextAlignment:NSTextAlignmentLeft];
    [button.titleLabel setFont:[UIFont systemFontOfSize:14]];
    [button setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
    [button setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

    [self.view addSubview:button];

    UIButton * button2 = [UIButton buttonWithTypeCustom];
    [button2 setFrame:CGRectMake(0, 220, self.view.width, 100)];
    [button2 setTitle:@"Pop-up menu/Pressed effect" forState:UIControlStateNormal];
    [button2 addTarget:self
        action:@selector(handleButton2:)
        forControlEvents:UIControlEventTouchUpInside];
    [button2.titleLabel setTextAlignment:NSTextAlignmentLeft];
    [button2 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
    [button2 setContentMode:UIViewContentModeLeft];
    [button2 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];

    [button2 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];
    [button2.titleLabel setFont:[UIFont systemFontOfSize:14]];

    [self.view addSubview:button2];

    UIButton * button3 = [UIButton buttonWithTypeCustom];
    [button3 setFrame:CGRectMake(0, 320, self.view.width, 100)];
```

```
[button3 setTitle:@"Pop-up menu/Double line" forState:UIControlStateNormal];
[button3 addTarget:self
               action:@selector(handleButton3:)
       forControlEvents:UIControlEventTouchUpInside];
[button3.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button3 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button3 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];
[button3.titleLabel setFont:[UIFont systemFontOfSize:14]];

[button3 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];

[self.view addSubview:button3];

UIButton * button4 = [UIButton buttonWithTypeCustom];
[button4 setFrame:CGRectMake(0, 420, self.view.width, 100)];
[button4 setTitle:@"Pop-up Menu + Select button" forState:UIControlStateNormal];
[button4 addTarget:self
               action:@selector(handleButton4:)
       forControlEvents:UIControlEventTouchUpInside];
[button4.titleLabel setTextAlignment:NSTextAlignmentLeft];
[button4 setTitleEdgeInsets:UIEdgeInsetsMake(0, 5, 0, 0)];
[button4 setContentHorizontalAlignment:UIControlContentHorizontalAlignmentLeft];
[button4.titleLabel setFont:[UIFont systemFontOfSize:14]];

[button4 setTitleColor:RGB(0x888888) forState:UIControlStateNormal];

[self.view addSubview:button4];
}

- (void)handleButton4:(UIButton *)button
{
    AUCellDataModel * model = [[AUCellDataModel alloc] init];
    model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
    model.titleText = @"Not interested";
    model.buttonsArray = @[@"Outdated",@"Have seen before",@"Bad quality"];
    model.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606950000902688",@"CCard":@""};

    AUCardMenu *tmpView=[[AUCardMenu alloc] initWithData:[model] location:CGPointMake(button.n.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopupMenu:button animation:YES];
    self.popMenuView=tmpView;
}

- (void)handleButton3:(UIButton *)button
{
    AUCellDataModel * model = [[AUCellDataModel alloc] init];
    model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
    model.titleText = @"Not interested";
    model.buttonsArray = @[@"Outdated",@"Have seen before",@"Bad quality"];
    model.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606950000902688",@"CCard":@""};

    AUCardMenu *tmpView=[[AUCardMenu alloc] initWithData:[model] location:CGPointMake(button.n.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopupMenu:button animation:YES];
    self.popMenuView=tmpView;
}
```

```

        model.titleText = @"Ignore";
        //      model.buttonsArray = @[@"Hello",@"do you stutter",@"I'm not hungry",@"Hello",@"I
        'm fine"];
        model.extendDic = @{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069
        50000902688",@"CCard":@""};

        AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
        model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
        model4.titleText = @"No longer receive this type of messages";
        model4.descText = @"Receive this type of messages less";
        model4.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606
        950000902688",@"CCard":@""};
        AUCardMenu *tmpView=[[AUCardMenu alloc] initWithData:@[model,model4] location:CGPointMak
        e(button.width - 20, button.centerY) offset:13];
        tmpView.cellView.delegate=self;
        [tmpView showPopupMenu:button animation:YES];
        self.popMenuView=tmpView;
    }

- (void)handleButton2:(UIButton *)button
{
    AUCellDataModel * model = [[AUCellDataModel alloc] init];
    model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
    model.titleText = @"Ignore";
    //      model.buttonsArray = @[@"Hello",@"do you stutter",@"I'm not hungry",@"Hello",@"I
    'm fine"];
    model.extendDic = @{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069
    50000902688",@"CCard":@""};
    AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
    model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
    model2.titleText = @"Not interested";
    model2.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606
    950000902688",@"CCard":@""};
    model2.highlightState = YES;
    AUCellDataModel * model3 = [[AUCellDataModel alloc] init];
    model3.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_inform.png";
    model3.titleText = @"Report";
    model3.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606
    950000902688",@"CCard":@""};
    AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
    model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
    model4.titleText = @"No longer receive this type of messages";
    model4.descText = @"Receive this type of messages less";
    model4.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606
    950000902688",@"CCard":@""};
    AUCardMenu *tmpView=[[AUCardMenu alloc] initWithData:@[model,model2,model3,model4] locat
    ion:CGPointMake(button.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopupMenu:button animation:YES];
    self.popMenuView=tmpView;
}

- (void)handleButton:(UIButton *)button
{

```

```

        AUCellDataModel * model = [[AUCellDataModel alloc] init];
        model.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_ignore.png";
        model.titleText = @"Ignore";
        //      model.buttonsArray = @[@"Hello",@"do you stutter",@"I'm not hungry",@"Hello",@"I'm fine"];
        model.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606950000902688",@"CCard":@""};
        AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
        model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
        model2.titleText = @"Not interested";
        model2.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606950000902688",@"CCard":@""};
        AUCellDataModel * model3 = [[AUCellDataModel alloc] init];
        model3.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_inform.png";
        model3.titleText = @"Report";
        model3.extendDic = @{@"type":@"reject",@"cardId":@"201609261515032720200000091128291606950000902688",@"CCard":@""};
        AUCardMenu *tmpView=[[AUCardMenu alloc] initWithData:[model,model2,model3] location:CGPointMake(button.width - 20, button.centerY) offset:13];
        tmpView.cellView.delegate=self;
        [tmpView showPopupMenu:button animation:YES];
        self.popMenuView=tmpView;
    }

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)hidePopupMenu
{
    if (self.popMenuView) {
        [self.popMenuView hidePopupMenuWithAnimation:YES];
        self.popMenuView.cellView.delegate = nil;
        self.popMenuView = nil;
    }
}

#pragma mark --- AUMultiStyleCellDelegate
/**
 * The clicking event callback.
 *
 * @param dataModel      The data model corresponding to the tapped view.
 * @param indexPath      The index of the tapped view in CellDataModel. (If CellDataModel.buttonsArray == nil, the default value of row is - 1.)
 */
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath
{

```

```
[self hidePopupMenu];
}
- (void)DidClickCellButton:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath
{
    [self hidePopupMenu];
}
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexPath:(NSIndexPath *)indexPath cellView:(AUMultiStyleCellView *)cellView
{
    [self hidePopupMenu];
}

- (void)dealloc
{
    self.popMenuView = nil;
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
}
*/

@end
```

1.5.5.9. Operation result dialog

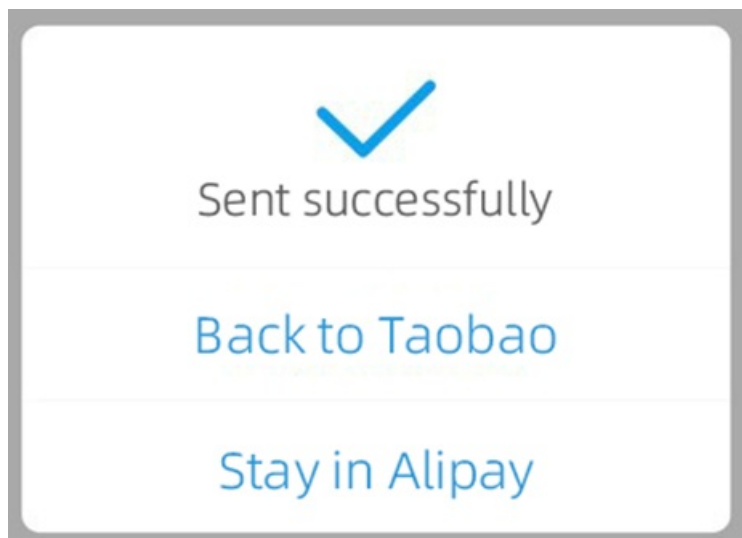
AUOperationResultDialog is a dialog box with a result image. The default size of the image is 90 x 58, in pixels. UED requires the style. The window level of AUOperationResultDialog is:

```
self.windowLevel = UIWindowLevelAlert - 1 .
```

Note

AUOperationResultDialog applies only to social sharing and checkout counter business. For the dialog box applying to other business, see [AUImageDialog](#).

Sample image



API description

```
@interface AUOperationResultDialog : AUIDialogBaseView

/**
 * Specify whether this instance is displayed. This applies when a pointer points at this instance.
 * If another dialog box overrides this one, the attribute value is fixed as YES.
 */
@property (nonatomic, assign, readonly) BOOL isDisplay;

/**
 * The dialog box description.
 */
@property (nonatomic, strong) NSString *describe;

/**
 * The method of dialog box initialization without the button title.

@param image      The image.
@param describe   The message details.
@param delegate   The AUIDialogDelegate-compliant protocol object.
@return           The AUIDialog instance.
 */
- (instancetype)initWithImage:(UIImage *)image
                        message:(NSString *)message
                        delegate:(id<AUIDialogDelegate>)delegate;

/**
 * The method of dialog box initialization with the button title.

@param image      The image.
@param describe   The message details.
@param delegate   The AUIDialogDelegate-compliant protocol object.
@param buttonTitle The list of button title parameters.
@return           The AUIDialog instance.
 */
- (instancetype)initWithImage:(UIImage *)image
```

```
(instancetype)initWithImage:(UIImage *)image
    message:(NSString *)message
    delegate:(id<AUDialogDelegate>)delegate
    buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

/**
 With a download link.

 @param imageUrl    The URL of the image.
 @param placeholder The placeholder image.
 @param describe    The message details.
 @param delegate    The AUDialogDelegate-compliant protocol object.
 @return            The AUIImageDialog instance.
 */
- (instancetype)initWithImageUrl:(NSString *)imageUrl
    placeholder:(UIImage *)placeholder
    message:(NSString *)message
    delegate:(id<AUDialogDelegate>)delegate;

/// The disabled initialization method.
- (instancetype)init NS_UNAVAILABLE;

/**
 The dialog box display method.
 */
- (void)show;

/**
 The method of closing the dialog box. If will/didDismissWithButtonIndex is monitored, the
 index called back is 0 by default.
 */
- (void)dismiss;

/**
 Hide all dialog views in the dialog window.
 */
+ (void)dismissAll;

/**
 Add a common button and its callback method. The common button cannot contain an action.

 @param buttonText The common button title.
 @param actionBlock The callback of the button.
 */
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

@end
```

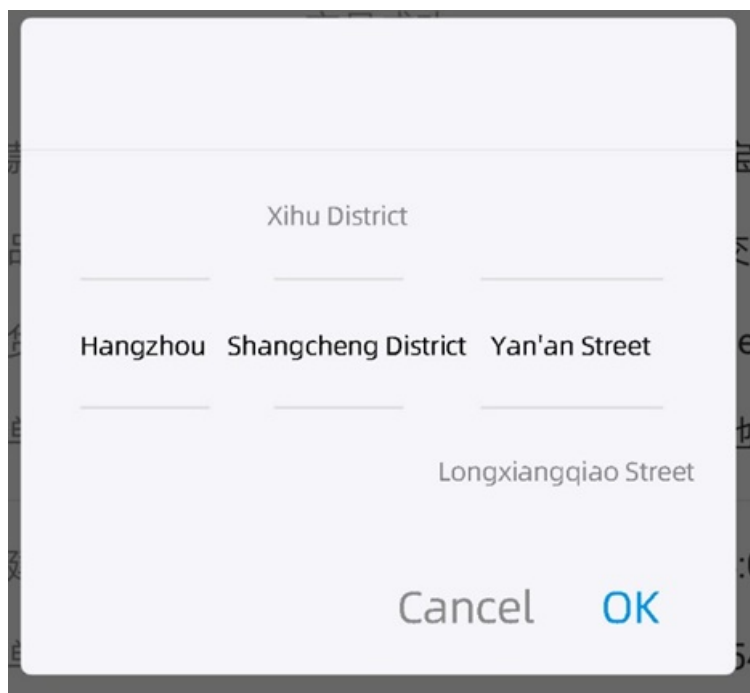
Sample code

```
UIImage *image = [UIImage imageNamed:@"panghu.jpg"];
AUOperationResultDialog *dialog = [[AUOperationResultDialog alloc] initWithImage:image
message:@"Sent successfully" delegate:self];
[dialog addButton:@"Back to Taobao" actionBlock:nil];
[dialog addButton:@"Stay in Alipay" actionBlock:nil];
[dialog show];
```

1.5.5.10. Cascade picker

AUCascadePicker is a multi-level cascade picker control that supports up to three levels.

Sample image



API description

```
// Set the selected item for the picker.
@interface AUCascadePickerSelectedItem : NSObject

@property (nonatomic, strong) NSString *selectedLeftTitle;    // The title selected for the
sublist on the left.
@property (nonatomic, strong) NSString *selectedMiddleTitle; // The title selected for the
sublist in the middle.
@property (nonatomic, strong) NSString *selectedRightTitle;  // The title selected for the
sublist on the right.

@end

@interface AUCascadePickerRowItemModel : NSObject

@property (nonatomic, strong) NSString *rowTitle;
@property (nonatomic, strong) NSArray<AUCascadePickerRowItemModel *> *rowSubList;
```

```
@end

// The data model required for the linkage effect.
@interface AUCascadePickerModel : NSObject

@property (nonatomic, strong) AUCascadePickerSelectedItem *preSelected; // The selected item transferred from the client.
@property (nonatomic, strong) AUCascadePickerSelectedItem *selectedItem; // The selected data list that is automatically recorded in the current component.
@property (nonatomic, strong) NSArray<AUCascadePickerRowItemModel *> *dataList; // The data list.
@property (nonatomic, strong) NSString *title; // The picker title.

@end

@interface AUCascadePicker : AUPickerBaseView <UIPickerViewDataSource, UIPickerViewDelegate>

@property (nonatomic, strong) AUCascadePickerModel *dataModel;
@property (nonatomic, assign) NSInteger numberOfComponents;
@property (nonatomic, weak) id <AUCascadePickerDelegate> linkageDelegate;

- (instancetype)initWithPickerModel:(AUCascadePickerModel *)model;

@end

// The callback for the Cancel and Completed buttons at the top.
@protocol AUCascadePickerDelegate <AUPickerBaseViewDelegate>
/*
 * Callback is performed when Cancel is clicked.
 */
- (void)cancelPickerView:(AUCustomDatePicker *)pickerView;

/*
 * Callback is performed when Completed is clicked. The selected items can be returned through selectedRowInComponent.
 */
- (void)selectedPickerView:(AUCustomDatePicker *)pickerView

@end
```

JSAPI description

API

antUIGetCascadePicker

Usage

```
AlipayJSBridge.call('antUIGetCascadePicker',
{
  title: 'nihao',// The cascade option title.
  selectedList:[{"name":"Hangzhou",subList:[{"name":"Shangcheng district"}]},
  list: [
    {
      name: "Hangzhou",// The entry name.
      subList: [
        {
          name: "Xihu district",
          subList: [
            {
              name: "Gucui Street"
            },
            {
              name: "Wenxin Street"
            }
          ]
        },
        {
          name: "Shangcheng district",
          subList: [
            {
              name: "Yan'an Street"
            },
            {
              name: "Longxiangqiao Street"
            }
          ]
        }
      ]
    }
  ]// The cascade sub-data list.
}
]// The cascade data list.
},
function(result){
  console.log(result);
});
```

Input parameters

Name	Type	Description	Required	Version
title	String	The cascade control title.	NO	10.1.2
selectedList	Json	The selected sub-items, in the same format as Input parameters. <pre>([{ "name": "Hangzhou", subList: [{"name": "Shangcheng District"}]})</pre>	NO	10.1.2

Name	Type	Description	Required	Version
list	Json	The selector data list.	YES	10.1.2
name	String	The entry name in <code>list</code> .	YES	10.1.2
subList	Json	The sub-list in <code>list</code> .	NO	10.1.2
fn	function	The callback function after selection is complete.	NO	10.1.2

Output parameters

Name	Type	Description	Version
success	Bool	Whether selection is complete. If selection is canceled, false is returned.	10.1.2
result	Json	The selected items, for example, <code>[{"name": "Hangzhou", subList: [{"name": "Shangcheng District"}]}</code>	10.1.2

Sample code

```

model = [[AULinkagePickerModel alloc] init];

NSMutableArray *modelList = [[NSMutableArray alloc] init];
for (int i=0; i<6; i++)
{
    AULinkagePickerRowItemModel *item = [[AULinkagePickerRowItemModel alloc] init];
    item.rowTitle = [NSString stringWithFormat:@"Level-1 %d", i];
    NSMutableArray *array = [[NSMutableArray alloc] init];
    for (int j=0; j<7; j++)
    {
        if (i == 0)
        {
            break;
        }
        AULinkagePickerRowItemModel *item1 = [[AULinkagePickerRowItemModel alloc] init];
;
        item1.rowTitle = [NSString stringWithFormat:@"Level-2 %d", j];
        NSMutableArray *array1 = [[NSMutableArray alloc] init];
        for (int k=0; k<5; k++) {
            AULinkagePickerRowItemModel *item2 = [[AULinkagePickerRowItemModel alloc] i
nit];

            item2.rowTitle = [NSString stringWithFormat:@"Level-3 %d", k];
            [array1 addObject:item2];
            if (j == 1 || j== 2) {
                break;
            }
        }
        item1.rowSubList = array1;
        [array addObject:item1];
        if (i == 3 || i== 5) {
            break;
        }
    }
    item.rowSubList = array;
    [modelList addObject:item];
}

model.dataList = modelList;

AULinkagePickerSelectedItem *item = [[AULinkagePickerSelectedItem alloc] init];
item.selectedLeftTitle = @"Level-1 0";
item.selectedMiddleTitle = @"Level-2 0";
item.selectedRightTitle = @"Level-3 0";

model.selectedItem = item;

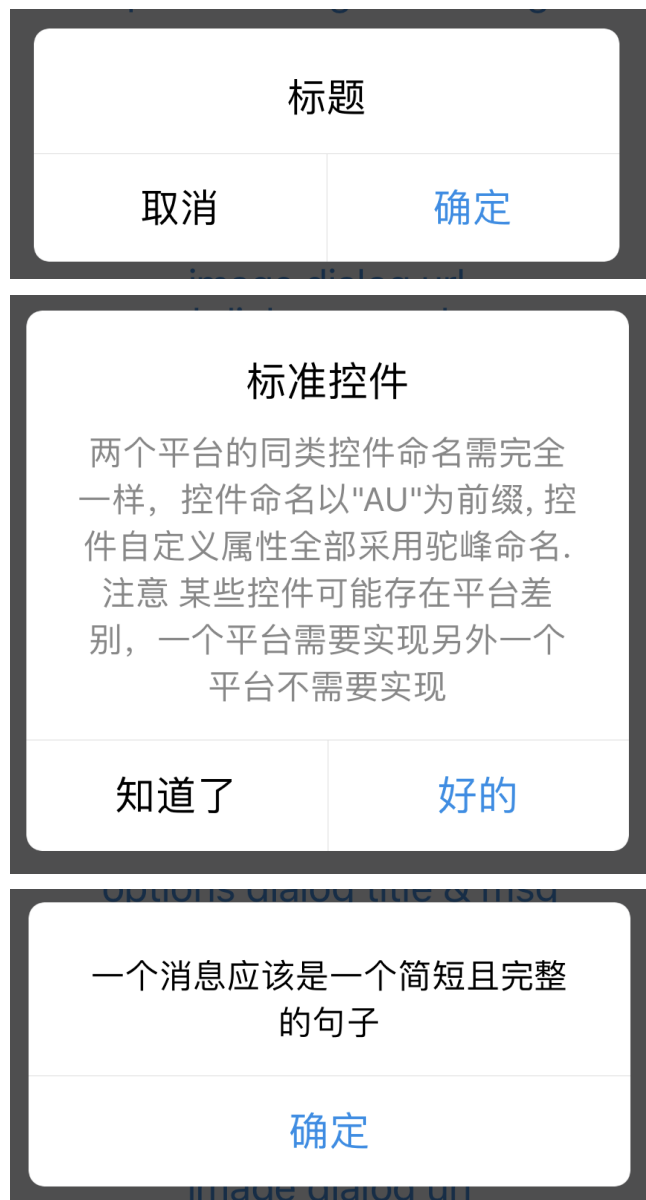
self.linkagePickerView = [[AULinkagePickerView alloc] initWithPickerModel:model];
self.linkagePickerView.linkageDelegate = self;
[self.linkagePickerView show];

```

1.5.5.11. Notification dialog

AUNoticeDialog specifies the style of a common dialog box. It references AlertView but does not contain the blur background. The Window level of the dialog box follows the logic `self.windowLevel = UIWindowLevelAlert - 1`.

Sample images





API description

```
/**
The common dialog box style. It is the same as the system style but does not contain the blur background.
*/
@interface AUNoticeDialog : AUDialogBaseView

/**
The method of dialog box initialization without the button title.

@param title    The title.
@param message  The message details.
@return         The AUNoticeDialog instance.
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message;

/**
The method of initializing the dialog box with the button title.

@param title    The title.
@param message  The message details.
@param delegate  The AUDialogDelegate-compliant protocol object.
@param buttonTitle  The button title list.
@return         The AUNoticeDialog instance.
*/
- (instancetype)initWithTitle:(NSString *)title
message:(NSString *)message
delegate:(id<AUDialogDelegate>)delegate
buttonTitles:(NSString *)buttonTitle, ... NS_REQUIRES_NIL_TERMINATION;

- (instancetype)initWithCustomView:(UIView *)customView; // The custom view.

- (instancetype)init NS_UNAVAILABLE;

/**
The dialog box display method.
*/
- (void)show;

/**
```

Add a button and its callback method.

```
@param buttonTitle The button title.
@param actionBlock The callback of the button tapping action.
*/
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUNoticeDialogActionBlock)actionBlock;

/**
The method of closing the dialog box. It is similar to the dismissWithClickedButtonIndex method of UIAlertView.
*/
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated;

/**
Set the text alignment mode.
@param alignment The alignment mode.
*/
- (void)setMessageAlignment:(NSTextAlignment)alignment;
```

Add a new dialog box

- Use the block to add a callback of the button tapping action.

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"Title" message:@"Message details"];
[dialog addButton:@"Got it" actionBlock:^(
    NSLog(@"print pressed")
)];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];
```

- Use the delegate to add a callback of the button tapping action.

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"Title" message:@"Message details" delegate:delegate buttonTitles:@"OK", nil];
[dialog show];

The delegate protocol is AUNoticeDialogDelegate, similar to UIAlertViewDelegate.
```

- Create a dialog box in a simple way.

```
NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitle(NSString *title)
NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitleAndMessage(NSString *title, NSString *message)
```

Use UIAlertView and UIAlertView

This section describes how to modify UIAlertView and UIAlertView to AUNoticeDialog.

Most APIs of AUNoticeDialog support UIAlertView and UIAlertView. In most cases, you may only need to modify the class name. The detailed operations are as follows:

- Use an API of AUNoticeDialog that supports UIAlertView to create a dialog box.

```
- (instancetype)initWithTitle:(NSString *)title
    message:(NSString *)message
    delegate:(id<AUNoticeDialogDelegate>)delegate
    cancelButtonTitle:(NSString *)cancelButtonTitle
    otherButtonTitles:(NSString *)otherButtonTitles, ... NS_REQUIRES_NIL_TERMINATION;
```

Change the class name from `[[UIAlertView alloc] initWithxxxxxx]` to `[[AUNoticeDialog alloc] initWithxxxxxx]`.

- Use the following method to create UIAlertView. **No modification is required**, because relevant modification has been added to the API.

```
NS_INLINE UIAlertView *UIAlertViewWithTitleAndMessage(NSString *title, NSString *message)
//
NS_INLINE UIAlertView *UIAlertViewWithTitle(NSString *title)
NS_INLINE UIAlertView *UIAlertViewWithMessage(NSString *message)
```

- Use the `addButtonWithTitle` API that supports UIAlertView to create a dialog box. **No modification is required**.

```
- (NSInteger)addButtonWithTitle:(NSString *)title callback:(void (^)(int index, NSString *title))callback;

/**
 * @brief Add a cancel button and its callback.
 * @param title The button title.
 * @param callback The callback.
 */
- (NSInteger)addCancelButtonWithTitle:(NSString *)title callback:(void (^)(int index, NSString *title))callback;

/**
 * @brief Add a button.
 * @param title The button title.
 */
- (NSInteger)addButtonWithTitle:(NSString *)title;

/**
 * @brief Add a cancel button.
 * @param title The button title.
 */
- (NSInteger)addCancelButtonWithTitle:(NSString *)title;

+ (void)setBackgroundMode:(BOOL)isBackMode;
```

- Use the following method of UIAlertView to create a dialog box. **No modification is required**. AUNoticeDialog has a method of the same name.

```
/**
 * The method of closing the dialog box. It is similar to the dismissWithClickedButtonIndex
 * method of APAlertView.
 */
- (void)dismissWithClickedButtonIndex:(NSInteger)buttonIndex animated:(BOOL)animated
- (nullable NSString *)buttonTitleAtIndex:(NSInteger)buttonIndex;
/**
 * Specify the number of buttons. (It is similar to numberOfButtons of APAlertView.)
 */
@property(nonatomic, readonly) NSInteger numberOfButtons;

/**
 * The index of the cancel button. (It is similar to cancelButtonIndex of APAlertView.)
 */
@property(nonatomic) NSInteger cancelButtonIndex;
```

- When you call the following APIs of APAlertView, you only need to change the method name.
 - Change the `showAlert` method to `show` .
For example, change `[alertView showAlert]` to `[alertView show]` .
 - Change the `removeAllAlertViews` method to `dismissAll` .
For example, change `[APAlertView removeAllAlertViews]` to `[AUNoticeDialog dismissAll]` .
- To use the input box feature of APAlertView or UIAlertView, replace the corresponding methods with those of AUIInputDialog. The operations are the same as those of AUNoticeDialog.

 **Note** The class file is AUIInputDialog.h.

Use UIAlertController

- Modify the creation method as follows:

```
[UIAlertController alertControllerWithTitle:title message:message preferredStyle:UIAlertControllerStyleAlert]
Modified to:
[[AUNoticeDialog alloc] initWithTitle:@"Title" message:@"Message details"]
```

- Modify the button and callback addition method as follows:

```
[UIAlertAction actionWithTitle:title style:(UIAlertActionStyle)style handler:handler]
Modified to:
[dialog addButton:@"Got it" actionBlock:^(
    NSLog(@"xxxx");
}]
```

Code sample

- Standard style

```
AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithTitle:@"Standard control" message:@"The controls of the same type must have the same name on two platforms. The prefix in a control name is \"AU\". Custom properties of a control are named in the camel-case format. Note: Some controls may be implemented in one platform but not in the other platform."];
[dialog addButton:@"Got it" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];
```

• Custom style

```
UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 240, 60)];
customView.backgroundColor = [UIColor greenColor];

AUNoticeDialog *dialog = [[AUNoticeDialog alloc] initWithCustomView:customView];
[dialog addButton:@"Cancel" actionBlock:nil];
[dialog addButton:@"OK" actionBlock:nil];
[dialog show];
```

1.5.5.12. Custom date picker

ACustomDatePicker is a custom date selection control and currently supports the following modes:

- ADatePickerModeTime : hour/minute, 24-hour clock
- ADatePickerModeDate : year/month/day
- ADatePickerModeDateAndTime : month/day/day of week/hour/minute, 24-hour clock

🔍 Note

The year is defined based on minimumDate and is 2000 (leap year) by default. Therefore, February 29 exists.

- ADatePickerYear : year
- ADatePickerYearMonth : year/month

Sample images

- ADatePickerModeTime

取消	ADatePickerModeTime	完成
	7	45
	8	46
	9	47
	10	48
	11	49
	12	50

- ADatePickerModeDate

取消	AUDatePickerModeDate	完成
	2014年	5月
	2015年	6月
	2016年	7月
	2017年	8月
	2018年	9月
	2019年	10月

- AUDatePickerModeDateAndTime

取消	AUDatePickerModeDateAndTi...	完成
	08月05日 星期六	7 55
	08月06日 星期日	8 56
	08月07日 星期一	9 57
	08月08日 星期二	10 58
	08月09日 星期三	11 59
	08月10日 星期四	12

- AUDatePickerYear

取消	AUDatePickerYear	完成
	2014年	
	2015年	
	2016年	
	2017年	
	2018年	
	2019年	

- AUDatePickerYearMonth

取消	AUDatePickerYearMonth	完成
	2014年	5月
	2015年	6月
	2016年	7月
	2017年	8月
	2018年	9月
	2019年	10月

- With a custom bottom view



API description

AUCustomDatePicker.h

```
// The custom bottom view.
@property (nonatomic, strong) UIView *bottomView;

/**
 * Create a picker, in AUPickerModeDate mode by default.
 *
 */
+ (AUPicker *)pickerWithTitle:(NSString *)title;

+ (AUPicker *)pickerWithTitle:(NSString *)title pickerMode:(AUPickerMode)mode;

/**
 * Set an available date range.
 * @param minDate The earliest time (included), which is 00:00:00 on January 1, 2000 by default.
 * @param maxDate The latest time (included), which is 23:59:59 on December 31, 2050 by default.
 */
- (void) setTimeDateMinDate:(NSDate *)minDate MaxDate:(NSDate *)maxDate;

/**
 * @param currentDate The time selected by default.
 */
- (void) setCurrentDate:(NSDate *) currentDate animated:(BOOL) animated;

/**
 * Show the date selection control.
 */
- (void) show;

/**
 * Hide the date selection control.
 */
- (void) hide;
```

Code sample

- Create

```
self.apCustomDatePickerView = [AUCustomDatePicker pickerViewWithTitle:@"AUDatePickerYearMonth" pickerMode:AUDatePickerYearMonth];

UIView *customBottomView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, AUCCommonUIGetScreenWidth(), 40)];
customBottomView.backgroundColor = RGB(0x00AAEE);
self.apCustomDatePickerView.bottomView = customBottomView;

[self.apCustomDatePickerView setCurrentDate:[NSDate date] animated:NO];
self.apCustomDatePickerView.tag = 1004;
self.apCustomDatePickerView.delegate = self;
[self.view addSubview:self.apCustomDatePickerView];
```

- Show/Hide

```
[self.apCustomDatePickerView show];
[self.apCustomDatePickerView hide];
```

- Value

```
- (void)cancelPickerView:(AUCustomDatePicker *)pickerView
{
    [self.apCustomDatePickerView hide];
}

- (void)selectedPickerView:(AUCustomDatePicker *)pickerView
{
    NSDate *selectedDate = picker.selectedDate;

    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    formatter.dateFormat = @"YYYY-MM-dd HH:mm:ss";

    [self.textLabel setText:[formatter stringFromDate:selectedDate]];

    [pickerView hide];
}
```

1.5.6. Loading components

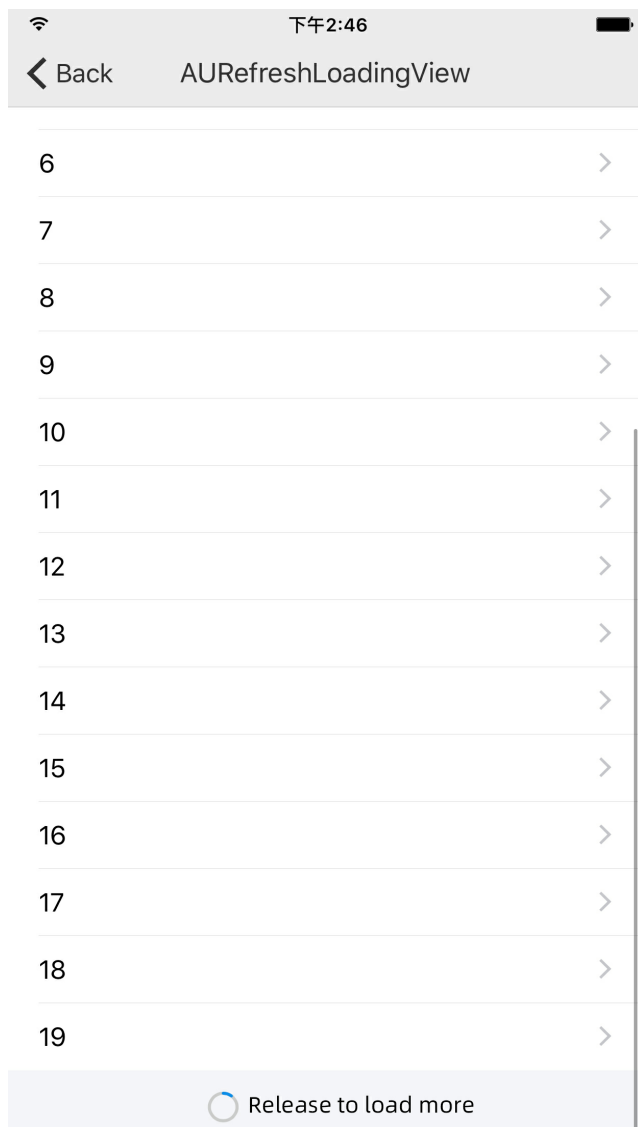
1.5.6.1. Pull-up refresh control

AUDragLoadingView and Aupullloadingview provide the loading style when the page is pulled up or down.

The following controls are not customized for businesses and must be switched to the AUPullLoadingView or AUDragLoadingView control:

CommonUI: ODRefreshControl, APCircleRefreshControl, EGOResfreshTableHeaderView, and APNextPagePullView

Sample image



API description

• AUIDragLoadingView.h

```
//
// AUIDragLoadingView.h
// AntUI
//

#import <AntUI/AntUI.h>

@interface AUIDragLoadingView : AUIPullLoadingView

@end
```

• AUIPullLoadingView.h

For more information, see [Pull loading view control](#).

Sample code

```
//
```

```
//
//  APRefreshTableViewController.m
//  UIDemo
//

#import "APRefreshTableViewController.h"
@interface APRefreshTableViewController ()
{
    BOOL _headerReloading;
    BOOL _footerReloading;
    BOOL _isHeader;
}
@property(nonatomic, strong) AUPullLoadingView *refreshHeaderView;
@property(nonatomic, strong) AUDragLoadingView *refreshFooterView;
@property(nonatomic, strong) UITableView *tableView;
@property(nonatomic, strong) NSMutableArray* listArray;

@end

@implementation APRefreshTableViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
        NSArray *array = @[@"0",
                            @"1",
                            @"2",
                            @"3",
                            @"4",
                            @"5",
                            @"6",
                            @"7",
                            @"8",
                            @"9",
                            @"10",
                            @"11",
                            @"12",
                            @"13",
                            @"14",
                            @"15",
                            @"16",
                            @"17",
                            @"18",
                            @"19"];

        self.listArray = [NSMutableArray arrayWithArray:array];
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
}
```

```

- (void)loadView {
    // Do any additional setup after loading the view.
    self.edgesForExtendedLayout = UIRectEdgeNone;
    // self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButton
    Title target:self];

    self.tableView = [[UITableView alloc] initWithFrame:self.view.bounds style:UITableViewSt
    ylePlain];
    self.tableView.dataSource = self;
    self.tableView.delegate = self;
    self.tableView.backgroundColor = [UIColor colorWithRGB:0xf5f5f9];
    self.tableView.separatorColor = [UIColor colorWithRGB:0xdddddd];
    [self.view addSubview:self.tableView];

    if (_refreshHeaderView == nil) {
        AUPullLoadingView *view = [[AUPullLoadingView alloc] initWithFrame:CGRectMake(0.0f,
        0.0f - self.tableView.bounds.size.height, self.view.frame.size.width, self.tableView.bounds
        .size.height)];
        view.delegate = self;
        [view ShowLastPullDate:YES];
        [view ShowStatusLabel:NO];

        [self.tableView addSubview:view];
        _refreshHeaderView = view;
    }
    [_refreshHeaderView refreshLastUpdatedDate];

    if (_refreshFooterView == nil) {
        AUDragLoadingView *view = [[AUDragLoadingView alloc] initWithFrame:CGRectMake(0, 0,
        self.view.bounds.size.width, 48)];
        view.delegate = self;
        [view setPullUp:@"release to load more"];
        [view setRelease:@"Relax"];
        self.tableView.tableFooterView = view;
        _refreshFooterView = view;
    }

    _isHeader = YES;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

#pragma tableview datasource
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{

```

```

        return _listArray.count;
    }

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"RefreshCell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (nil == cell)
    {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
                                         reuseIdentifier:CellIdentifier];
    }
    cell.textLabel.text = _listArray[indexPath.row];
    cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

    return cell;
}

#pragma mark -
#pragma mark Data Source Loading / Reloading Methods

- (void)reloadHeaderTableViewDataSource{

    // should be calling your tableviews data source model to reload
    // put here just for demo
    NSInteger first = [_listArray[0] integerValue] - 1;
    [_listArray insertObject:[NSString stringWithFormat:@"%li", (long)first] atIndex:0];

    _headerReloading = YES;
}

- (void)doneLoadingHeaderTableViewData{

    // model should call this when its done loading
    _headerReloading = NO;
    [_refreshHeaderView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
    [self.tableView reloadData];
}

- (void)reloadFooterTableViewDataSource{

    // should be calling your tableviews data source model to reload
    // put here just for demo
    NSInteger count = [_listArray count];
    NSInteger last = [_listArray[count-1] integerValue] + 1;
    [_listArray addObject:[NSString stringWithFormat:@"%li", (long)last]];

    _footerReloading = YES;
}

- (void)doneLoadingFooterTableViewData{

```

```
// model should call this when its done loading
_footerReloading = NO;
[_refreshFooterView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
[self.tableView reloadData];

}

#pragma mark -
#pragma mark UIScrollViewDelegate Methods

- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
    if (scrollView.contentInset.top + scrollView.contentOffset.y < 0) {
        _isHeader = YES;
    } else {
        _isHeader = NO;
    }

    if (_isHeader) {
        [_refreshHeaderView egoRefreshScrollViewDidScroll:scrollView];
    } else {
        [_refreshFooterView egoRefreshScrollViewDidScroll:scrollView];
    }
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
    if (_isHeader) {
        [_refreshHeaderView egoRefreshScrollViewDidEndDragging:scrollView];
    } else {
        [_refreshFooterView egoRefreshScrollViewDidEndDragging:scrollView];
    }
}

#pragma mark -
#pragma mark EGORefreshTableHeaderDelegate Methods

- (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*)view
{
    if (_isHeader) {
        [self reloadHeaderTableViewDataSource];
        [self performSelector:@selector(doneLoadingHeaderTableViewData) withObject:nil afterDelay:2.0];
    } else {
        [self reloadFooterTableViewDataSource];
        [self performSelector:@selector(doneLoadingFooterTableViewData) withObject:nil afterDelay:2.0];
    }
}

- (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*)view
{
    if (_isHeader) {
```

```
        return _headerReloading;
    } else {
        return _footerReloading; // should return if data source model is reloading
    }

}

- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view{

    return [NSDate date]; // should return date data source was last changed

}

@end
```

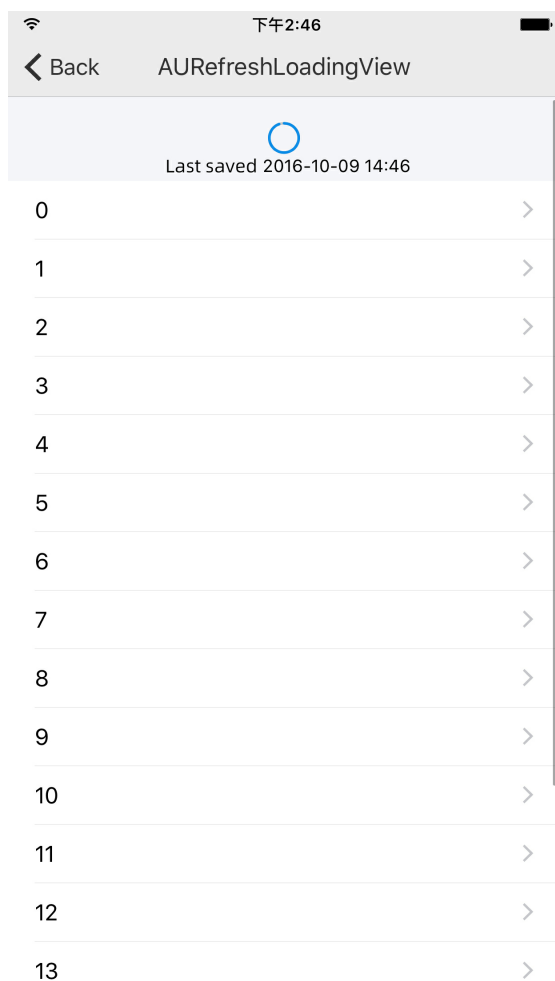
1.5.6.2. Pull-down refresh component

The AUDragLoadingView and the Aupullloadingview provide the loading style when the page is pulled up or down.

The following controls are not customized for businesses and must be switched to the AUPullLoadingView or AUDragLoadingView control.

CommonUI: ODRefreshControl, APCircleRefreshControl, EGOResfreshTableHeaderView, and APNextPagePullView.

Sample image



API description

- **AUPullLoadingView.h**

```
//
// EGORefreshTableHeaderView.h
// Demo
//

#import <UIKit/UIKit.h>
#import <QuartzCore/QuartzCore.h>

typedef enum {
    AUEGOPullingDown = 1000,
    AUEGOPullingUp
} AUEGOPullDirection;

typedef enum{
    AUEGOOPullRefreshPulling = 0,
    AUEGOOPullRefreshNormal,
    AUEGOOPullRefreshLoading,
} AUEGOPullRefreshState;

@class AULoadingIndicatorView;
@protocol AURefreshLoadingViewDelegate;
```

```

/ **
 * @class AUPullRefreshLoadingView
 * @abstract UIView
 * @discussion This class is migrated from EGOResfreshTableHeaderView. The class is used to load more views through pulling up or down.
 */

@interface AUPullRefreshLoadingView : UIView {

    __weak id _delegate;
    AUEGOPullRefreshState _state;

    UILabel *_lastUpdatedLabel;
    UILabel *_statusLabel;
    // AUIActivityIndicatorView *_activityView;
    AUEGOPullDirection _pullDirection;

    BOOL isAutoPullFlag;
}
@property(n nonatomic, strong) AUIActivityIndicatorView *_activityView;

/ **
 * Set the text for pull-up-to-load in initial state. The default value is "Pull up to load more", which is displayed upon pull-up.
 *
 * @param tip The tip text.
 *
 */
- (void)setPullUp:(NSString *)tip;

/ **
 * Set the text for pull-down refresh in initial state. The default value is "Pull down to refresh", which is not displayed upon pull-down.
 *
 * @param tip The tip text.
 *
 */
- (void)setPullDown:(NSString *)tip;

/ **
 * Set the text for the loading process displayed after release. The default value is "Loading".
 * Note: The tip text is displayed in the loading process upon pull-up but not pull-down by default.
 *
 * @param tip The tip text.
 *
 */
- (void)setLoading:(NSString *)tip;

/ **
 * The text displayed to prompt a user to release. The default value is "Release to refresh".
 *

```

```

    * @param tip The tip text.
    *
    */
    - (void)setRelease:(NSString *)tip;

/**
    * Specify whether to display the text about the last update. The text is not displayed by default.
    *
    * @param isOpen If this parameter is set to YES, the text is displayed.
    *
    */
    - (void)ShowLastPullDate:(BOOL)isOpen;

/**
    * Specify whether to display the text for the loading process.
    *
    * Default: No text is displayed upon pull-down refresh, and "Loading" is displayed upon pull-up-to-load.
    *
    * @param isShow If this parameter is set to YES, the text is displayed.
    *
    */
    - (void)ShowStatusLabel:(BOOL)isShow;

    - (void)setDateFormat:(NSDateFormatter *)dateFormatter;

    - (void)setAutoPull:(BOOL)isAutoPull;

@property(nonatomic,weak) id <AURefreshLoadingViewDelegate> delegate;

    - (void)refreshLastUpdatedDate;
    - (void)egoRefreshScrollViewDidScroll:(UIScrollView *)scrollView;
    - (void)egoRefreshScrollViewDidEndDragging:(UIScrollView *)scrollView;
    - (void)egoRefreshScrollViewDataSourceDidFinishedLoading:(UIScrollView *)scrollView;
    - (void)egoRefreshScrollViewDataSourceDidFinishedLoadingWithoutUpdate:(UIScrollView *)scrollView;

    - (void)autoUpdateScrollView:(UIScrollView *)scrollView;

#pragma Mark -- for LegacySystem not recommend
@property(nonatomic,assign) AUEGOPullRefreshState state;
@property(nonatomic,retain) NSString *statusText;
@property (nonatomic, retain) UILabel *lastUpdatedLabel;
@property (nonatomic, retain) UILabel *statusLabel;

    - (void)setCurrentDate;

@end

@protocol AURefreshLoadingViewDelegate
    - (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*) view;
    - (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*) view;
@optional

```

```
- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view;
@end
```

• AUDragLoadingView.h

For more information, see [Drag loading view control](#).

Sample code

```
//
//  ARefreshTableViewController.m
//  UIDemo
//

#import "ARefreshTableViewController.h"
@interface ARefreshTableViewController ()
{
    BOOL _headerReloading;
    BOOL _footerReloading;
    BOOL _isHeader;
}
@property(nonatomic, strong) AUPullLoadingView *refreshHeaderView;
@property(nonatomic, strong) AUDragLoadingView *refreshFooterView;
@property(nonatomic, strong) UITableView *tableView;
@property(nonatomic, strong) NSMutableArray* listArray;

@end

@implementation ARefreshTableViewController

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
        NSArray *array = @[@"0",
                           @"1",
                           @"2",
                           @"3",
                           @"4",
                           @"5",
                           @"6",
                           @"7",
                           @"8",
                           @"9",
                           @"10",
                           @"11",
                           @"12",
                           @"13",
                           @"14",
                           @"15",
                           @"16",
                           @"17",
                           @"18",
                           @"19",
                           @"20",
                           @"21",
                           @"22",
                           @"23",
                           @"24",
                           @"25",
                           @"26",
                           @"27",
                           @"28",
                           @"29",
                           @"30",
                           @"31",
                           @"32",
                           @"33",
                           @"34",
                           @"35",
                           @"36",
                           @"37",
                           @"38",
                           @"39",
                           @"40",
                           @"41",
                           @"42",
                           @"43",
                           @"44",
                           @"45",
                           @"46",
                           @"47",
                           @"48",
                           @"49",
                           @"50",
                           @"51",
                           @"52",
                           @"53",
                           @"54",
                           @"55",
                           @"56",
                           @"57",
                           @"58",
                           @"59",
                           @"60",
                           @"61",
                           @"62",
                           @"63",
                           @"64",
                           @"65",
                           @"66",
                           @"67",
                           @"68",
                           @"69",
                           @"70",
                           @"71",
                           @"72",
                           @"73",
                           @"74",
                           @"75",
                           @"76",
                           @"77",
                           @"78",
                           @"79",
                           @"80",
                           @"81",
                           @"82",
                           @"83",
                           @"84",
                           @"85",
                           @"86",
                           @"87",
                           @"88",
                           @"89",
                           @"90",
                           @"91",
                           @"92",
                           @"93",
                           @"94",
                           @"95",
                           @"96",
                           @"97",
                           @"98",
                           @"99"];
    }
```

```

        @"19"];
        self.listArray = [NSMutableArray arrayWithArray:array];
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the view.
    self.edgesForExtendedLayout = UIRectEdgeNone;
    // self.navigationItem.rightBarButtonItem = [APUtil getBarButtonWithTitle:RightBarButton
    Title target:self];

    self.tableView = [[UITableView alloc] initWithFrame:self.view.bounds style:UITableViewSt
ylePlain];
    self.tableView.dataSource = self;
    self.tableView.delegate = self;
    self.tableView.backgroundColor = [UIColor colorWithRGB:0xf5f5f9];
    self.tableView.separatorColor = [UIColor colorWithRGB:0xdddddd];
    [self.view addSubview:self.tableView];

    if (_refreshHeaderView == nil) {

        AUPullLoadingView *view = [[AUPullLoadingView alloc] initWithFrame:CGRectMake(0.0f,
0.0f - self.tableView.bounds.size.height, self.view.frame.size.width, self.tableView.bounds
.size.height)];
        view.delegate = self;
        [view ShowLastPullDate:YES];
        [view ShowStatusLabel:NO];

        [self.tableView addSubview:view];
        _refreshHeaderView = view;
    }
    [_refreshHeaderView refreshLastUpdatedDate];

    if (_refreshFooterView == nil) {
        AUDragLoadingView *view = [[AUDragLoadingView alloc] initWithFrame:CGRectMake(0, 0,
self.view.bounds.size.width, 48)];
        view.delegate = self;
        [view setPullUp:@"Pull up to load more"];
        [view setRelease:@"Relax"];
        self.tableView.tableFooterView = view;
        _refreshFooterView = view;
    }

    _isHeader = YES;
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

```

```
#pragma tableview datasource
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    return _listArray.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *CellIdentifier = @"RefreshCell";
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];
    if (nil == cell)
    {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault
                                         reuseIdentifier:CellIdentifier];
    }
    cell.textLabel.text = _listArray[indexPath.row];
    cell.accessoryType = UITableViewCellAccessoryDisclosureIndicator;

    return cell;
}

#pragma mark -
#pragma mark Data Source Loading / Reloading Methods

- (void)reloadHeaderTableViewDataSource{

    // should be calling your tableviews data source model to reload
    // put here just for demo
    NSInteger first = [_listArray[0] integerValue] - 1;
    [_listArray insertObject:[NSString stringWithFormat:@"%li", (long)first] atIndex:0];

    _headerReloading = YES;
}

- (void)doneLoadingHeaderTableViewData{

    // model should call this when its done loading
    _headerReloading = NO;
    [_refreshHeaderView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
    [self.tableView reloadData];
}

- (void)reloadFooterTableViewDataSource{

    // should be calling your tableviews data source model to reload
    // put here just for demo
    NSInteger count = [listArray count];
```

```

        NSInteger last = [_listArray[count-1] integerValue] + 1;
        [_listArray addObject:[NSString stringWithFormat:@"%li", (long)last]];

        _footerReloading = YES;
    }

- (void)doneLoadingFooterTableViewData{

    // model should call this when its done loading
    _footerReloading = NO;
    [_refreshFooterView egoRefreshScrollViewDataSourceDidFinishedLoading:self.tableView];
    [self.tableView reloadData];

}

#pragma mark -
#pragma mark UIScrollViewDelegate Methods

- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
    if (scrollView.contentInset.top + scrollView.contentOffset.y < 0) {
        _isHeader = YES;
    } else {
        _isHeader = NO;
    }

    if (_isHeader) {
        [_refreshHeaderView egoRefreshScrollViewDidScroll:scrollView];
    } else {
        [_refreshFooterView egoRefreshScrollViewDidScroll:scrollView];
    }
}

- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
    if (_isHeader) {
        [_refreshHeaderView egoRefreshScrollViewDidEndDragging:scrollView];
    } else {
        [_refreshFooterView egoRefreshScrollViewDidEndDragging:scrollView];
    }
}

#pragma mark -
#pragma mark EGORefreshTableHeaderDelegate Methods

- (void)egoRefreshTableHeaderDidTriggerRefresh:(AUPullLoadingView*)view
{
    if (_isHeader) {
        [self reloadHeaderTableViewDataSource];
        [self performSelector:@selector(doneLoadingHeaderTableViewData) withObject:nil afterDelay:2.0];
    } else {
        [self reloadFooterTableViewDataSource];
        [self performSelector:@selector(doneLoadingFooterTableViewData) withObject:nil after

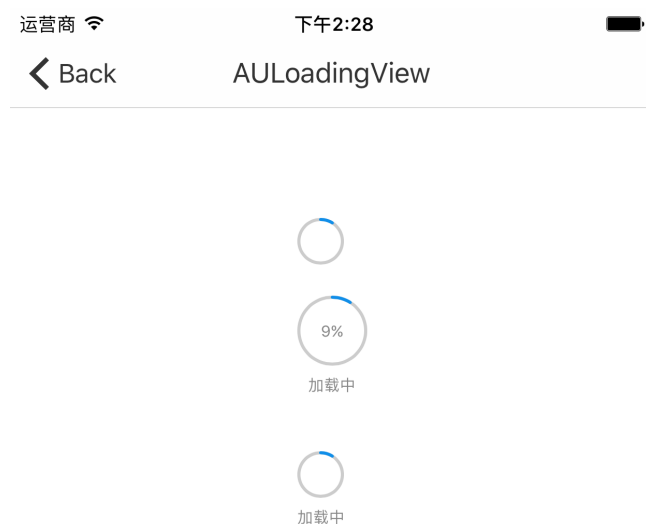
```

```
rDelay:2.0];  
    }  
  
    }  
  
- (BOOL)egoRefreshTableHeaderDataSourceIsLoading:(AUPullLoadingView*)view  
{  
    if (_isHeader) {  
        return _headerReloading;  
    } else {  
        return _footerReloading; // should return if data source model is reloading  
    }  
}  
  
- (NSDate*)egoRefreshTableHeaderDataSourceLastUpdated:(AUPullLoadingView*)view{  
    return [NSDate date]; // should return date data source was last changed  
}  
  
@end
```

1.5.6.3. Loading component

AULoadingView is a new loading control that provides a loading page with progress indicator, loading progress, and loading text.

Sample image



API description

AULoadingView.h

```
//
//  AULoadingView.h
//  AntUI
//

#import <UIKit/UIKit.h>

/**
 * Set the middle loading control to display a percentage.
 */
@interface AULoadingView : UIView

@property (nonatomic,assign) BOOL isShowProgressPer; // Specify whether to display the progress percentage. The default value is NO.
@property (nonatomic,assign) BOOL isShowLoadingText; // Specify whether to display the loading text. The default value is NO.

/**
 * Set the progress percentage.
 *
 * @param progress The percentage.
 */
- (void) setProgressPer:(CGFloat) progress;

@end
```

Sample code

```
//
//  AULoadingViewController.m
//  AntUI
//

#import "AULoadingViewController.h"
#import "AULoadingView.h"

@interface AULoadingViewController ()
@property (nonatomic,strong) AULoadingView * loadingView;
@property (nonatomic,strong) AULoadingView * loadingView2;
@property (nonatomic,strong) AULoadingView * loadingView3;

@property (nonatomic,assign) CGFloat progress;

@end

@implementation AULoadingViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.view.backgroundColor = [UIColor whiteColor];
    // Do any additional setup after loading the view.
}
```

```

self.loadingView = [[AULoadingView alloc] init];
self.loadingView.center = CGPointMake(200, 200);
self.loadingView.isShowProgressPer = YES;
self.loadingView.isShowLoadingText = YES;
[self.view addSubview:self.loadingView];

self.loadingView2 = [[AULoadingView alloc] init];
self.loadingView2.center = CGPointMake(200, 150);
// self.loadingView2.isShowProgressPer = YES;
// self.loadingView2.isShowLoadingText = YES;
[self.view addSubview:self.loadingView2];

self.loadingView3 = [[AULoadingView alloc] init];
self.loadingView3.center = CGPointMake(200, 300);
// self.loadingView3.isShowProgressPer = YES;
self.loadingView3.isShowLoadingText = YES;
[self.view addSubview:self.loadingView3];

[NSTimer scheduledTimerWithTimeInterval:0.1
                                target:self
                                selector:@selector(loadingTimer:)
                                userInfo:nil
                                repeats:YES];
}

- (void) loadingTimer:(id)timer
{
    self.progress += 0.01;
    if ((int)(self.progress *100) > 100) {
        self.progress = 0.0;
        [timer invalidate];
        return;
    }
    [self.loadingView setProgressPer:self.progress];
    [self.loadingView2 setProgressPer:self.progress];
    [self.loadingView3 setProgressPer:self.progress];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before
navigation

```

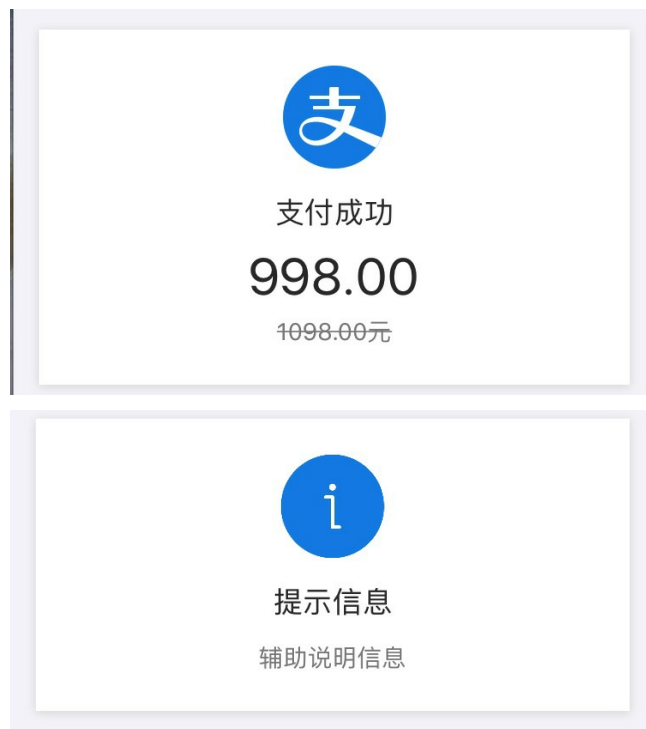
```
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {  
    // Get the new view controller using [segue destinationViewController].  
    // Pass the selected object to the new view controller.  
}  
*/  
  
@end
```

1.5.7. Result page component

1.5.7.1. Result page component

AUResultView displays status result views with an image.

Sample images



API description

```
/**
The result view that displays the status.
*/
@interface AUResultView : UIView

/**
The image at the top.
*/
@property (nonatomic, strong) UIImage *icon;

/**
The middle-sized title in black at the top of the text field.
*/
@property (nonatomic, strong) NSString *mainTitleText;

/**
The large-sized title in black in the middle.
*/
@property (nonatomic, strong) NSString *midTitleText;

/**
The message in gray at the bottom.
*/
@property (nonatomic, strong) NSString *bottomMessage;

/**
Specify whether strikethrough is added to the message at the bottom.
*/
@property (nonatomic, assign) BOOL messageThrough;

/**
The expected height of the view. You can obtain the value after initialization is completed
.
*/
@property (nonatomic, assign, readonly) CGFloat expectHeight;

/**
The ResultView instance method.

@param icon          The image.
@param mainTitleText The first title.
@param midTitleText  The large-sized title in the middle.
@param bottomMessage The message in gray at the bottom.
@param messageThrough Specify whether strikethrough is added to the message.
@return              The AUResultView instance.
*/
- (instancetype)initWithIcon:(UIImage *)icon mainTitleText:(NSString *)mainTitleText midTitleText:(NSString *)midTitleText bottomMessage:(NSString *)bottomMessage messageThrough:(BOOL)messageThrough;
```

Code sample

```
UIImage *image = AUBundleImage(@"alipay-60");
AUResultView *resultView = [[AUResultView alloc] initWithIcon:image
mainTitleText:@"Payment succeeded"
midTitleText:@"998.00"
bottomMessage:@"CNY 1098.00"
messageThrough:YES];
resultView.frame = CGRectMake(marginX, originY, AUCCommonUIGetScreenWidth()-2*marginX, resultView.expectHeight);
[self.view addSubview:resultView];
```

1.5.7.2. Exception page component

AUNetErrorView is the control that displays empty page errors, including two prompt styles:

- Simple style (default): includes five types.
- Illustrated style: includes five types.

The two styles differ in the used prompt images, as shown in the following sample images.

Sample images

- Simple style (half-screen)



- Complex style (full-screen)



API description

```
typedef NS_ENUM(NSInteger, AUNetErrorType) {

    AUNetErrorTypeLimit,           // Throttling.
    AUNetErrorTypeAlert,           // System busy(error) or warning.
    AUNetErrorTypeNetworkError,    // Poor network
    AUNetErrorTypeEmpty,           // Empty content.
    AUNetErrorTypeNotFound,        // Page not found. (The image is the same as that used in A
    UNetErrorTypeAlert.)
    AUNetErrorTypeUserLogout,      // User logout.

    AUNetErrorTypeFailure __attribute__((deprecated)) = AUNetErrorTypeNetworkError,
    AUNetErrorTypeError __attribute__((deprecated)) = AUNetErrorTypeNetworkError,        // N
    o network connection.
    AUNetErrorTypeSystemBusy __attribute__((deprecated)) = AUNetErrorTypeAlert,        // W
    arning.
    APEXceptionEnumNetworkError __attribute__((deprecated)) = AUNetErrorTypeNetworkError,
    // No network connection.
    APEXceptionEnumEmpty __attribute__((deprecated)) = AUNetErrorTypeEmpty,            // E
    mpty content.
    APEXceptionEnumAlert __attribute__((deprecated)) = AUNetErrorTypeAlert,            // W
    arning.
    APEXceptionEnumLimit __attribute__((deprecated)) = AUNetErrorTypeLimit,            // T
    hrottling.
    APEXceptionEnumNetworkFailure __attribute__((deprecated)) = AUNetErrorTypeNetworkError
    , // The network signal strength is poor
};

typedef NS_ENUM(NSInteger, AUNetErrorStyle) {

    AUNetErrorStyleMinimalist,      // The simple style.
    AUNetErrorStyleIllustration,    // The complex style.

    APEXceptionStyleIllustration __attribute__((deprecated)) = AUNetErrorStyleIllustration,
    // The complex style.
    APEXceptionStyleMinimalist __attribute__((deprecated)) = AUNetErrorStyleMinimalist
    // The simple style.
};
```

```
/**
    The control that displays empty page errors.

    Two prompt styles are supported:
        1. Simple style (default): includes three types.
        2. Illustrate style: includes seven types.

    The two styles differ in the used prompt images.
    */
@interface AUNetErrorView : UIView

    @property(nonatomic, strong, readonly) UIButton *actionButton;        // The default text
    is refresh.
    @property(nonatomic, strong, readonly) UIImageView *iconImageView;    // The icon view.
    @property(nonatomic, strong, readonly) UILabel *infoLabel;            // The label of the
```

```
primary prompt text.
    @property(nonatomic, strong, readonly) UILabel *detailLabel;           // The label of the
    detailed prompt text.

    @property(nonatomic, strong) NSString *infoTitle;                     // The primary text
    .
    @property(nonatomic, strong) NSString *detailTitle;                   // The secondary te
    xt.

    /**
     * Initialize the error view and set the error style and type.
     * (When target and action are empty, the refresh button will not be displayed.)
     *
     * @param frame      Required. The coordinates of the view.
     * @param style      Required. The complex or simple style.
     * @param type       Required. The error type.
     * @param target     The object to be processed in the refresh event.
     * @param action     The method of processing the refresh event.
     *
     * @return APEExceptionView
     */
    - (id)initWithFrame:(CGRect)frame
                      style:(AUNetErrorStyle)style
                      type:(AUNetErrorType)type
                      target:(id)target
                      action:(SEL)action;

    /**
     * Initialize the error view and display it on the specified view.
     * (When target and action are empty, the refresh button will not be displayed.)
     *
     * @param parent     Required. The superView of view.
     * @param style      Required. error style, the complex or simple style.
     * @param type       Required. The error type.
     * @param target     The object to be processed in the refresh event.
     * @param action     The method of processing the refresh event.
     *
     * @return APEExceptionView
     */
    + (id)showInView:(UIView *)parent
                  style:(AUNetErrorStyle)style
                  type:(AUNetErrorType)type
                  target:(id)target
                  action:(SEL)action;

    /**
     * Cancel the display of the error view.
     */
    - (void)dismiss;

    /**
     * The countdown, which can only be used in the case of throttling.
     * If completeBlock is nil and the business does not set a clicking response event for acti
     onButton, the countdown function does not take effect.
```

```
* If completeBlock isn't nil, directly execute completeBlock and hide actionBar when the
countdown is over.
* To call getActionBar to add a button response event, ensure that the actionBar response
event has been added.
*/
- (void)setCountdownTimeInterval:(NSInteger)startTime // The countdown start time.
        completeBlock:(void (^)(void))completeBlock; // Countdown ends.

@end
```

Sample code

```
netErrorView = [[AUNetErrorView alloc] initWithFrame:CGRectMake(0, CGRectGetMaxY(label.frame)
+ 5, self.view.width, 300) style:AUNetErrorStyleIllustration type:AUNetErrorTypeError target:self
action:@selector(pressedNetErrorView)];
netErrorView.detailTitle = @"AUNetErrorTypeError type";
[self.view addSubview:netErrorView];

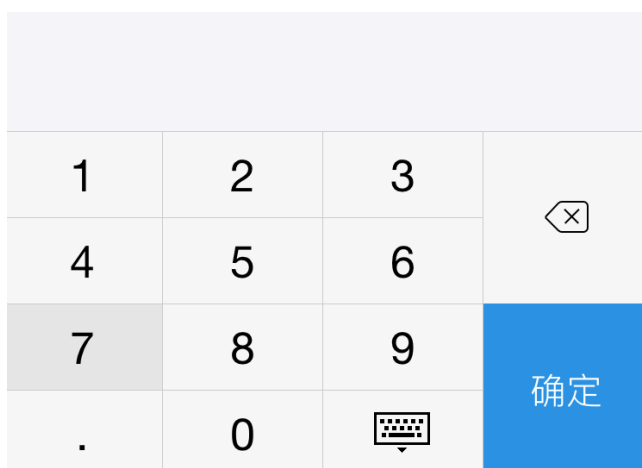
// Set the countdown.
[netErrorView setCountdownTimeInterval:10 completeBlock:^(
    NSLog(@"Countdown ends");
)];
```

1.5.8. Numeric keypad component

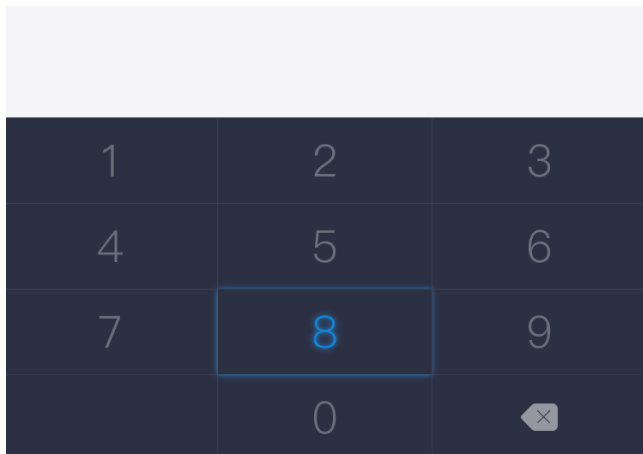
AUNumKeyboards is a custom numeric keypad component.

Sample images

- Common mode



- Chat mode



API description

```
typedef NS_ENUM(NSInteger, AUNumKeyboardMode) {
    AUNumKeyboardModeCommon, // The numeric keypad is in common mode.
    AUNumKeyboardModeChat,   // The keypad is in chat mode.
    AUNumKeyboardModeInvalid // The keypad is in invalid mode and unavailable.
};

/**
 * Define a numeric keypad.
 */
@interface AUNumKeyboards : UIView

/**
 * Create a numeric keypad component, which uses the common mode by default.
 *
 * @return Return the initialized numeric keypad component.
 */
+ (AUNumKeyboards *)sharedKeyboard;

/**
 * Create a numeric keypad component.
 *
 * @param mode The numeric keypad mode.
 *
 * @return Return the initialized numeric keypad component.
 */
+ (AUNumKeyboards *)sharedKeyboardWithMode:(AUNumKeyboardMode)mode;

/**
 * Manually set textInput. The Y-coordinate of the numeric keypad needs to be set external
 * ly.
 */
@property (nonatomic, weak) id<UITextInput> textInput;

/**
 * The ID card number.
 */
@property (nonatomic, assign) BOOL idNumber;
```

```
/**
 * Set the numeric keypad mode.
 */
@property (nonatomic, assign, readonly) AUNumKeyboardMode mode;

/**
 * Specify whether to hide the decimal point.
 */
@property (nonatomic, assign) BOOL dotHidden;

/**
 * Specify whether to hide the numeric keypad.
 */
@property (nonatomic, assign) BOOL dismissHidden;

/**
 * Specify whether the submit button is clickable.
 */
@property (nonatomic, assign) BOOL submitEnable;

/**
 * The text of the submit button.
 * Note: To ensure the visual requirement, the text supports a maximum of six characters.
 */
@property (nonatomic, strong) NSString *submitText;
```

Code sample

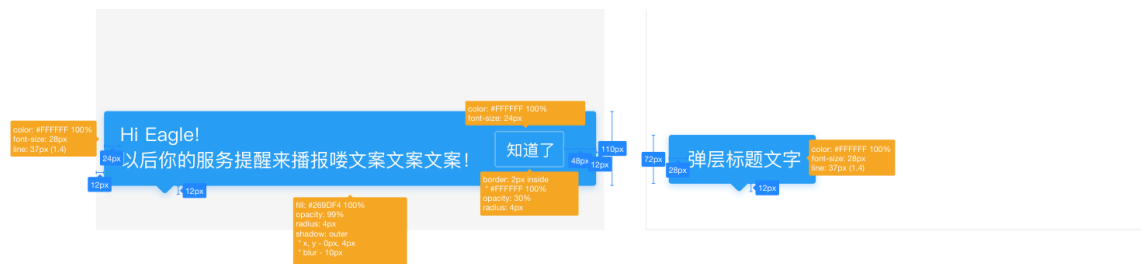
```
UITextField *numTextField = ...
numTextField.inputView = [AUNumKeyboards sharedKeyboardWithMode:AUNumKeyboardModeCommon] ;
// The chat mode parameter: AUNumKeyboardModeChat.
[self.view addSubview:numTextField];
```

1.5.9. Guidance component

1.5.9.1. Prompt component

AUPopTipView is a boot prompt component.

Sample image



API description

```
typedef NS_ENUM(NSInteger, AUPopViewIndicatorDirection) {
    AUPopViewIndicatorDirectionUp,
    AUPopViewIndicatorDirectionDown,
};

@interface AUPopTipView : AUPopDrawBoardView

AU_UNAVAILABLE_INIT

@property (nonatomic, assign) AUPopViewIndicatorDirection indicatorDirection;

- (void)dismiss:(BOOL)animated;

+ (instancetype)showFromView:(UIView *)fromView
    fromPoint:(CGPoint)fromPoint
    toView:(UIView *)toView
    animated:(BOOL)animated
    withText:(NSString *)text
    buttonTitle:(NSString *)buttonTitle;

@end
```

Sample code

```
// Display
AUPopTipView *popTipView = [AUPopTipView showFromView:button
                                fromPoint:CGPointZero
                                toView:self.view
                                animated:YES
                                withText:@"Text sample"
                                buttonTitle:@"Close"]; // When buttontitle is not
transferred, the right button is not displayed.

// Hide
[popTipView dismiss:YES];
```

1.5.9.2. Floating layer bar component

AUPopBar is a guide floating layer bar component.

Sample image



API description

```
@interface AUPopBar : UIView

AU_UNAVAILABLE_INIT

+ (instancetype)showInViewBottom:(UIView *)view
    animated:(BOOL)animated
    withText:(NSString *)text
    icon:(UIImage *)icon
    buttonTitle:(NSString *)buttonTitle
    actionBlock:(BOOL (^)(void))actionBlock;

- (void)dismiss:(BOOL)animated;

@end
```

Sample code

```
// Display
AUPopBar *popBar = [AUPopBar showInViewBottom:weakSelf.view animated:YES withText:@"Add "City services" to the homepage" icon:[UIImage imageNamed:@"ap_scan"] buttonTitle:@"Add now" actionBlock:^(
    NSLog(@"Clicked");
    return YES;
}];

// Hide
[popBar dismiss:YES];
```

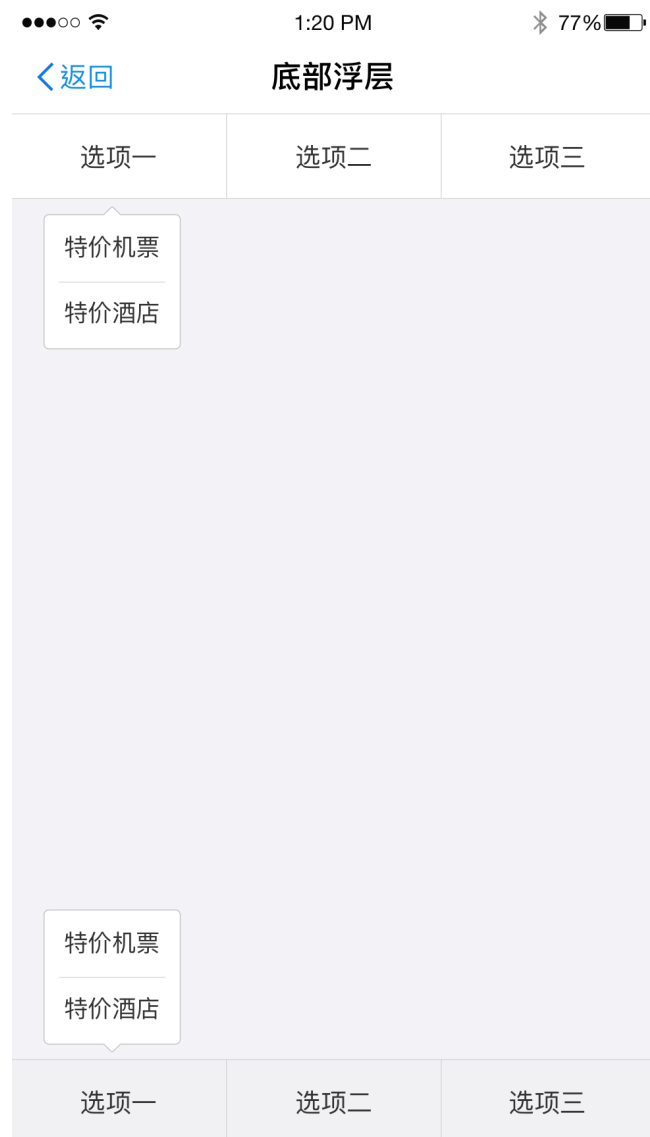
1.5.10. Pop menu component

The AUPopMenu component provides popping-up menus when the user clicks the navigation bar tabs.

- Different from AUFloatMenu, AUPopMenu has menu outlines but no bottom masks. All menus are aligned in the center. The separation lines have a fixed length and are aligned in the center.

- Basic functions: Menus in this dialog box popping out upwards or downwards, the popping positions are all defined by the business needs.

Sample image



API description

- AUPopMenu.h

```
@protocol AUPopMenuDelegate <NSObject>

@optional
- (void)DidClickPopItemView:(AUPopItemModel *)viewModel;

@end

@interface AUPopMenu : UIView

@property (nonatomic, weak) id<AUPopMenuDelegate> delegate;

/*  datas          The AUPopItemModel object list.
 *  position       The position of the direction angle.
 *  superView      The parent view.
 *  isArchViewUp   The orientation of the arrow-like corner. Default value: Down.
 */
- (instancetype)initWithDatas:(NSArray *)datas
                      position:(CGPoint)position
                      superView:(UIView *)superView
                      isArchViewUp:(BOOL)isArchViewUp;

/* Show and hide the menus with animation by default.
 *  position The start and end positions of the arrow-like corner.
 *  superView Describe which parent view the current floating layer is displayed on.
 */
- (void)showMenu;

//
- (void)hideMenu;

@end
```

• AUPopItemView.h

```
@interface AUPopItemView : AUPopItemBaseView

@property (nonatomic, strong) AUIconView *iconView; // Support the icon font image.
//@property (nonatomic, strong) UIView *badgeView // The badge is not supported currently.

- (instancetype)initWithModel:(AUPopItemModel *)model position:(CGPoint )position;

@end
```

• AUPopItemBaseView.h

```
//
@interface AUPopItemBaseView : UIControl

@property (nonatomic, strong) AULabel *titleLabel; //

@end
```

• AUPopItemModel.h

```
// The object model.
@interface AUPopItemModel : NSObject

@property (nonatomic, strong) NSString *titleString; // The main description.
@property (nonatomic, strong) id iconImage; // The left-side icon, which can be a UIImage object or URL.

@end
```

Sample code

```
_menu = [[AUPopMenu alloc] initWithDatas:array position:CGPointMake(CGRectGetMidX(button.frame),
CGRectGetMaxY(button.frame)+5) superView:self.view isArchViewUp:YES];
_menu.delegate = self;
[_menu showMenu];
```

1.5.11. Navigation components

1.5.11.1. Vertical tab

AUVerticalTabView is a vertical tab-based component.

Sample image



Dependency

The dependency of AUVerticalTabView is as follows:

AntUI

API description

```
#import <UIKit/UIKit.h>

@protocol AUVerticalTabViewDataProtocol <NSObject>

@required
- (NSString *) tabName;

@end

@class AUVerticalTabView;

typedef void (^AUVerticalTabSelectedCallback)(AUVerticalTabView *verticalTabView);

@interface AUVerticalTabView : UIView

/**
 The recommended initialization method. The layout parameters are standardized by AntDNA.
 AUVerticalTabView : width=110pt
 TabCell : width=110pt,height=55pt

@param verticalTabViewDatas Set tab data.
@param selectedCallback Set the tapping event callback.
@param height The height of AUVerticalTabView.
@param business The business identifier, such as GoldWord or BeeCityPicker.
@return AUVerticalTabView
*/
+ (AUVerticalTabView *)verticalTabViewWithDatas:(NSArray <id<AUVerticalTabViewDataProtocol>
>*) verticalTabViewDatas
    selectedCallback:(AUVerticalTabSelectedCallback)selectedCall
back
    height:(CGFloat)height
    business:(NSString *)business;

@property(nonatomic, strong) NSArray <id<AUVerticalTabViewDataProtocol>>* verticalTabViewDa
tas;
@property(nonatomic, assign) NSUInteger selectedIndex;//default 0
@property(nonatomic, copy) AUVerticalTabSelectedCallback selectedCallback;

@end
```

Code sample

```
// The external data object implementation AUVerticalTabViewDataProtocol, which returns the
required tabName.
@interface DemoVerticalTabData : NSObject <AUVerticalTabViewDataProtocol>

- (NSString *)tabName;

@end
```

```
NSArray *datas = @[ [DemoVerticalTabData new],
                    [DemoVerticalTabData new],
                    [DemoVerticalTabData new],
                    [DemoVerticalTabData new],
                    [DemoVerticalTabData new],
                    [DemoVerticalTabData new],
                    [DemoVerticalTabData new] ];

AUVerticalTabView *tabView = [AUVerticalTabView verticalTabViewWithDatas:datas
                                selectedCallback:^(AUVerticalTabView *verticalTabView ){
    NSUInteger selectedIndex = verticalTabView.selectedIndex;
    id<AUVerticalTabViewDataProtocol> selectedData = [verticalTabView.verticalTabViewDatas objectAtIndex:selectedIndex];

    height:self.view.height
    business:@"AntUI"];

[self.view addSubview:tabView];
```

1.5.11.2. Double title

AUDoubleTitleView is a view control containing a title line and a subtitle line in the navigation pane.

Sample image



API description

```

    /**
     The titleView of a navigation pane that contains two lines.
    */
@interface AUDoubleTitleView : UIView

/**
 * Create TitleViews of the title and subtitle.
 *
 * @param title          The main title.
 * @param detaileTitle The subtitle.
 *
 * @return Return the initialized APTitleView control.
 */
- (UIView *)initWithTitle:(NSString *)title detailTitle:(NSString *)detaileTitle;

/**
 * Modify the title text.
 *
 * @param title          Main title text.
 *
 */
- (void)updateTitle:(NSString *)title;

/**
 * Modify the subtitle text.
 *
 * @param detailTitle    Main title text.
 *
 */
- (void)updateDetailTitle:(NSString *)detailTitle;

/**
 Change the title font.

 @param titleFont The title font.
 */
- (void)updateTitleFont:(UIFont *)titleFont;

/**
 Change the subtitle font.
 *
 * @param detailTitleFont Main title font.
 *
 */
- (void)updateDetailTitleFont:(UIFont *)detailTitleFont;

@end

```

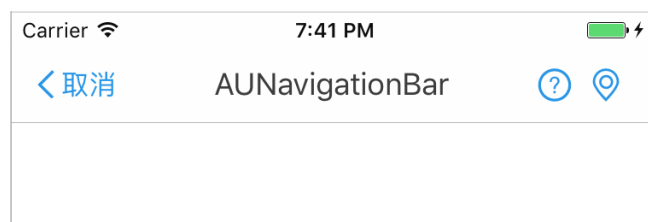
Code sample

```
self.navigationItem.titleView = [[AUDoubleTitleView alloc] initWithTitle:@"Title" detailTitle:@"Subtitle"];
```

1.5.11.3. Navigation bar

AUNavigationBar is a navigation bar control of mPaaS. It extends the UINavigationController control and provides default mPaaS navigation bar styles. To facilitate subsequent extension, use AUNavigationBar rather than UINavigationController in all mPaaS apps.

Sample image



API description

```
/**
 * The mPaaS navigation bar control that contains mPaaS navigation bar styles.
 * Initialize:
 * UINavigationController *navBar = [[UINavigationController alloc] initWithNavigationBarClass:
 *   NSStringFromClass(@"AUNavigationBar") toolbarClass:nil];
 */
@interface AUNavigationBar : UINavigationController

@end

/**
 * The UINavigationController extension that defines default UINavigationController styles.
 */
@interface UINavigationController (AUNavigationBarExtensions)

/**
 * Return the default color of the title of framework navigation bar. The default value is #000000.
 */
@property (nonatomic, assign) UIColor *navigationBarTitleDefaultColor;

/**
 * Return the color of the item of framework navigation bar. The default value is #108EE9.
 */
@property (nonatomic, assign) UIColor *navigationBarButtonItemDefaultColor;

/**
 * Return the color of the item of framework navigation bar. The default value is #ffffff.
 */
```

```

*
* @return
*/
+ (UIColor*)getNavigationBarDefaultColor;

/**
 * Get the color of the bottom line of the navigation bar. The default value is #e1e1e1.
 *
 * @return
 */
+ (UIColor*)getNavigationBarBotLineColor;

/**
 * Note:
 * 1. The base class DTViewController sets the default style of navigation bar in ViewWillAppear.
 * 2. Business personnel can call the system APIs or the following APIs to change the style of navigation bar. The style is typically set in ViewWillAppear.
 * 3. If VC is a subclass of DTViewController, it must be set in ViewWillAppear, or it will be overridden.
 * 4. Ensure that setNavigationBarDefaultStyle is used to restore the default style when ViewWillDisappear is called after the modification.
 * 5. If VC defines the homepage in the UITabBarController container, do not use setNavigationBarDefaultStyle to restore the default style, or the VC will be overridden upon tab switching.
 */

/**
 *
 * Set the background of default navigation bar. The default background color and bottom line color are #ffffff and #e1e1e1, respectively.
 *
 */
- (void)setNavigationBarDefaultStyle;

/**
 *
 * Set the default title style of the navigation bar.
 *
 */
- (void)setNavigationBarDefaultTitleTextAttributes;

/**
 *
 * Set the title color of the navigation bar in ViewWillAppear, or the title color will be overridden by the default title color in the framework.
 *
 */
- (void)setNavigationBarTitleTextAttributesWithTextColor:(UIColor *)textColor;

/**
 *
 * Set the transparency of the navigation bar.
 * Note: If this method sets the navigation bar to be completely transparent, returned anim

```

```

ation will flash white. Currently, this issue has not been resolved. Do not call this method. If the method is required, evaluate whether the impact is acceptable.
*/
- (void)setNavigationBarTranslucentStyle;

/**
 * Set the color of the navigation bar. To achieve the frosted glass effect, set translucent to Yes.
 * Note: After calling this API, call the bottom line setting API if necessary, or the bottom line color will be overridden by the default color #e1e1e1.
 *
 * @param color          The color to be displayed.
 * @param translucent    Whether to be transparent.
 *
 */
- (void)setNavigationBarStyleWithColor:(UIColor *)color translucent:(BOOL)translucent;

/**
 * There may be a separation line under the navigation bar, and the UI may fail to meet some UI requirements. Call this method to set the color of the separation line to prevent it from being recognized.
 * Note: If you have defined the background of navigation bar, for example, by calling setNavigationBarStyleWithColor or rewriting opaqueNavigationBarColor, call this API after changing the background color.
 * Otherwise, the bottom line color will be overridden by the default color #e1e1e1.
 */
- (void)setNavigationBarBottomLineColor:(UIColor*)color;

/**
 * Before calling the system methods setBarTintColor, setBackgroundImage, and setBackgroundColor to set the color of navigation bar, call this method to eliminate the default effect.
 * Otherwise, the color will be overlaid with the default color and cause a color deviation.
 */
- (void)resetNavigationBarColor;

/**
 *
 * To prevent the issue that the navigation bar flashes when a user swipes right to go back or cancel an operation, do not call this method.
 */
- (void)setNavigationBarMaskLayerWithColor:(UIColor *)color;

/**
 * Return the current background color of the navigation bar.
 *
 * @return Return the current background color of the navigation bar.
 */
- (UIColor*)getNavigationBarCurrentColor;

@end

```

Sample code

```
// Initialize UINavigationController.
UINavigationController *navBar = [[UINavigationController alloc] initWithNavigationBarClass:
:NSClassFromString(@"AUNavigationBar") toolbarClass:nil];

// Configure the navigation bar in VC.
AUBarButtonItem *cancelItem = [AUBarButtonItem backBarButtonItemWithTitle:@"Back" target:self
action:@selector(cancel)];
cancelItem.backBarButtonItemTitle = @"Cancel";
self.navigationItem.leftBarButtonItem = cancelItem;

UIImage *image1 = [AUIconView iconWithName:kICONFONT_MAP width:22 color:AU_COLOR_LINK];
UIImage *image2 = [AUIconView iconWithName:kICONFONT_HELP width:22 color:AU_COLOR_LINK];
AUBarButtonItem *rightItem1 = [[AUBarButtonItem alloc] initWithImage:image1 style:UIBarButtonItemStylePlain
target:self action:@selector(rightBarItemPressed)];
AUBarButtonItem *rightItem2 = [[AUBarButtonItem alloc] initWithImage:image2 style:UIBarButtonItemStylePlain
target:self action:@selector(rightBarItemPressed)];
self.navigationItem.rightBarButtonItem = [rightItem1 rightItem2];
```

1.5.11.4. Custom navigation bar

AUCustomNavigationBar is a navigation bar component customized by mPaaS for the transparent navigation bar scenario.

After the native navigation bar is changed from transparent to solid, users may have bad visual experience. This class is provided for avoiding this issue.

Sample image



API description

```
/**
Customize a transparent navigation bar as required.
After the native navigation pane is changed from transparent to solid, users may have bad
visual experience. This class is provided for avoiding this issue.
*/
@interface AUCustomNavigationBar : UIView

@property(nonatomic, strong) UIView *backgroundView; // Ground glass background view.

@property(nonatomic, strong) NSString *backBarButtonItem; // The back button title (no title by default).

@property(nonatomic, strong) UIColor *backBarButtonItemTitleColor; // The title color of the
```

```

@property(nonatomic, strong, class=AUCCustomNavigationBar, readwrite) UIImage *backButtonImage; // The image of the back button.

@property(nonatomic, strong) UIImage *backButtonImage; // The image of the back button.

@property(nonatomic, strong) NSString *title; // The title.
@property(nonatomic, strong) UIColor *titleColor; // The title color.
@property(nonatomic, strong) UIView *titleLabel; // Customized titleview.

@property(nonatomic, strong) NSString *rightItemTitle; // The right item title.
@property(nonatomic, strong) UIColor *rightItemTitleColor; // The color of the right item title.
@property(nonatomic, strong) UIImage *rightItemImage; // The image of the right item.

/**
 * The VoiceOver text for the item displayed on the right.
 * The item displayed on the left, which is "Back" by default.
 * The item displayed on the right, which is specified by rightItemTitle by default. If rightItemTitle is not set, manually set this property to support VoiceOver.
 */
@property(nonatomic, strong) NSString *rightItemVoiceOverText;

@property(nonatomic, strong) NSString *leftItemVoiceOverText;

/**
 * Create a specified view of transparent navigation bar.
 *
 * (1) By default, the navigation bar displays an arrow instead of "Back" on the left for users to go back. If the current page needs to set the back text that is consistent with the framework logic, override the (UIView *)customNavigationBar method in VC.
 * (2) To set the title, item to be displayed on the right, and background in frosted glass effect, call related APIs.
 *
 * @param currentVC The current VC.
 *
 * @return The view of transparent navigation bar.
 */
+ (AUCCustomNavigationBar *)navigationBarForCurrentVC:(UIViewController *)currentVC;

/**
 * Set the background view of frosted glass. The transparency is 0 by default.
 */
- (void)setNavigationBarBlurEffective;

/**
 * Create an item to be displayed on the right of the navigation bar.
 *
 * @param rightItemTitle Displayed text.
 * @param target target
 * @param action action
 *
 */
- (void)setNavigationBarRightItemWithTitle:(NSString *)rightItemTitle target:(id)target act

```

```
ion: (SEL)action;

/**
 * Create an item to be displayed on the right of the navigation bar.
 *
 * @param rightItemImage    Displayed image.
 * @param target            target
 * @param action            action
 *
 */
- (void)setNavigationBarRightItemWithImage:(UIImage *)rightItemImage target:(id)target action: (SEL)action;

/**
 * Create an item to be displayed on the left of the navigation bar.
 *
 * @param leftItemTitle     Displayed text.
 * @param target            target
 * @param action            action
 *
 */
- (void)setNavigationBarLeftItemWithTitle:(NSString *)leftItemTitle target:(id)target action: (SEL)action;

/**
 * Create an item to be displayed on the left of the navigation bar.
 *
 * @param leftItemTitle     Displayed image.
 * @param target            target
 * @param action            action
 *
 */
- (void)setNavigationBarLeftItemWithImage:(UIImage *)leftItemTitle target:(id)target action: (SEL)action;

@end
```

Sample code

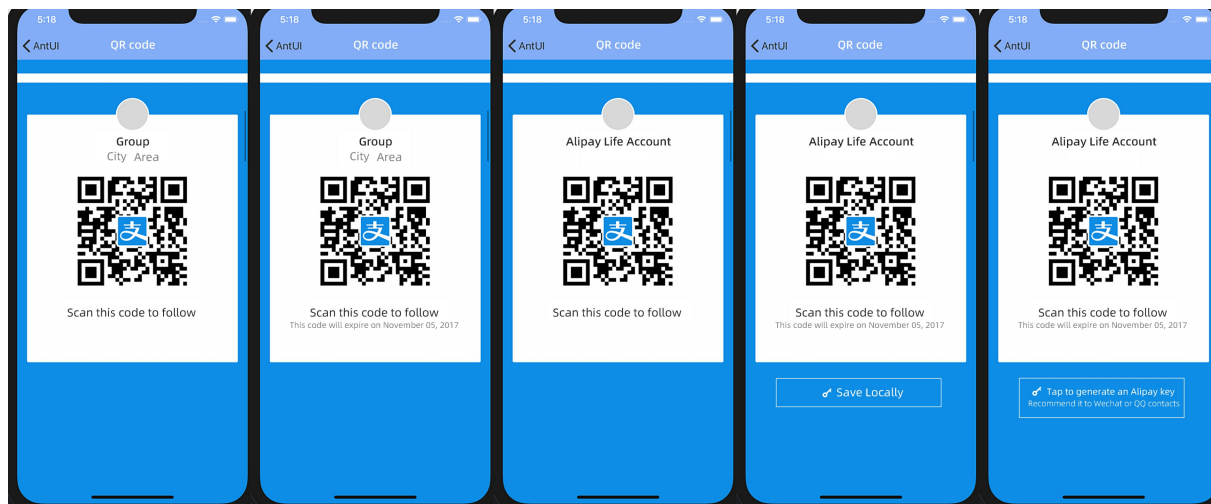
```
AUCustomNavigationBar *navBar = [AUCustomNavigationBar navigationBarForCurrentVC:self];
[navBar setNavigationBarBlurEffective]; // The frosted glass effect.
[self.view addSubview:navBar];
navBar.title = @"Title";
navBar.backButtonImage = [AUIconView iconWithName:kICONFONT_BILL width:22 color:AU_COLOR_LINK];
navBar.backButtonTitle = @"Bill";
navBar.rightBarButtonItemImage = [AUIconView iconWithName:kICONFONT_ADD width:22 color:AU_COLOR_LINK];

// When using this API on mPaaS, override the following methods of the parent class:
- (BOOL)autohideNavigationBar
{
    return YES;
}
- (UIView *)customNavigationBar
{
    return self.navBar;
}
```

1.5.12. QR code component

AUQRCodeView is the Alert view that supports multiple option buttons. The Window level of QR code component follows the logic `self.windowLevel = UIWindowLevelAlert - 1`.

Sample image



API description

```
// The data model object.
@interface QRDataModel : NSObject

@property (nonatomic, strong) id topLeftIcon; // An image, a URL, or cloudID
can be imported.
@property (nonatomic, strong) NSString *topTitle; // An image, a URL, or cloudID
can be imported.
@property (nonatomic, strong) id qrCodeIcon; // The QR code image.
@property (nonatomic, strong) NSString *bottomTitle;
@property (nonatomic, strong) NSString *bottomMessage;
@property (nonatomic, strong) id actionButtonIcon; // An image, a URL, or cloudID
can be imported.
@property (nonatomic, strong) NSString *actionButtonTitle; // The primary description of
the action button at the bottom.
@property (nonatomic, strong) NSString *actionButtonMessage; // The secondary description o
f the action button at the bottom.

@end

// The action button under the QR code.
@interface QRActionButton : UIControl

@end

// The QR code component.
@interface AUQRCodeView : UIView

@property (nonatomic, strong) UIView *maskView;
@property (nonatomic, strong) UIView *containerView; // The QR code container.
@property (nonatomic, strong) UIImageView *topLeftImageView; // The image in the upper lef
t corner.
@property (nonatomic, strong) UILabel *topTitleLabel; // Description text for the t
itle on the top.
@property (nonatomic, strong) UIImageView *qrCodeView; // The QR code image.
@property (nonatomic, strong) UILabel *bottomTitleLabel; // Main description text at t
he bottom.
@property (nonatomic, strong) UILabel *bottomMessageLabel; // Secondary description text
at the bottom.
@property (nonatomic, strong) QRActionButton *actionButton; // The action button at the b
ottom.

// The control frame used to block the initialization data model.
- (instancetype)initWithFrame:(CGRect)frame model:(void (^)(QRDataModel *model))block;

// Start loading.
- (void)startLoading;

// Stop loading.
- (void)stopLoading;

@end
```

Code sample

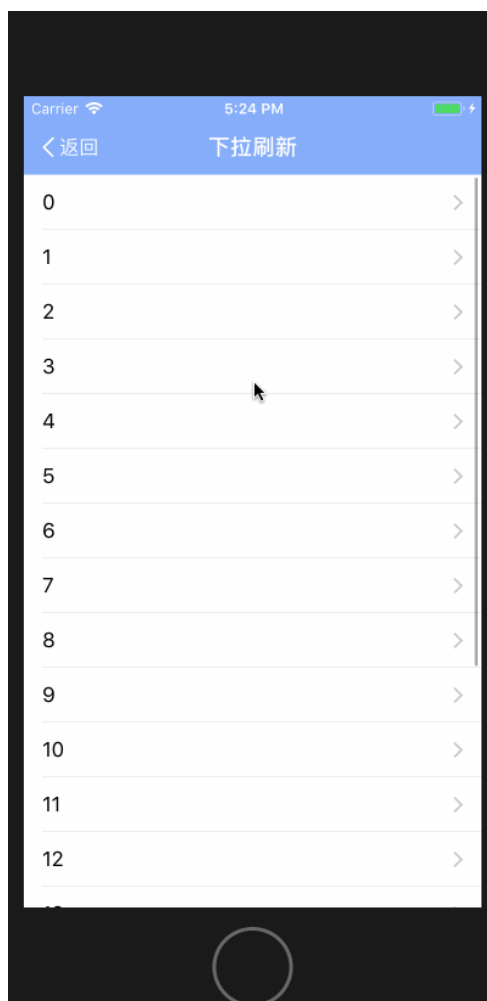
The following sample shows the code of a standard style QR code.

```
AUQRCodeView *qrCodeView = [[AUQRCodeView alloc] initWithFrame:frame model:^(QRDataModel *model) {  
    model.topLeftIcon = [UIImage imageWithColor:[UIColor colorWithRGB:0xbbbbbb] size:CGSizeMake(54, 54)];  
    model.topTitle = @"Alipay life account";  
    model.bottomTitle = @"Scan this QR code to follow";  
    model.bottomMessage = @"This QR code will expire on November 05, 2017";  
    model.actionButtonTitle = @"Save locally";  
}];  
[self.view addSubview:qrCodeView];
```

1.5.13. Refresh component

AURefreshView is a pull-down refresh view with an ant icon. At present, two colors are supported, as shown in the sample image.

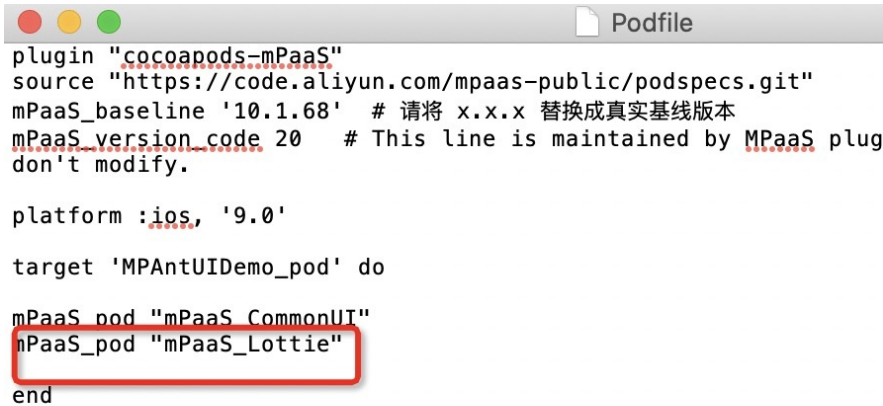
Sample image



Add components

To enable AUIRefreshView, add the **Common UI** component and the **Lottie** component first.

1. In the Podfile file, enter `mPaaS_pod "mPaaS_CommonUI"` and `mPaaS_pod "mPaaS_Lottie"` to add the dependencies on the **Common UI** component and the **Lottie** components.



```
plugin "cocoapods-mPaaS"
source "https://code.aliyun.com/mpaas-public/podspecs.git"
mPaaS_baseline '10.1.68' # 请将 x.x.x 替换成真实基线版本
mPaaS_version_code 20 # This line is maintained by MPaaS plug
don't modify.

platform :ios, '9.0'

target 'MPAntUIDemo_pod' do

  mPaaS_pod "mPaaS_CommonUI"
  mPaaS_pod "mPaaS_Lottie"
end
```

2. Run `pod install` to complete adding the components.

Interface description

```
typedef NS_ENUM(NSUInteger, AUIRefreshViewState) {
    AUIRefreshViewStateNomal = 0,           // Restore the list to the initial position.
    AUIRefreshViewStateBeginPulling = 1,    // A user starts to pull the bar down.
    AUIRefreshViewStateLoading = 2,         // Trigger RPC loading. The contentInset list is
    delivered in the default position.
    AUIRefreshViewStateFinishedLoading = 3, // RPC loaded. The contentInset list is about to
    be restored to the original position.
    AUIRefreshViewStateBeginResetting = 4,  // The contentInset list starts to restore to th
    e default inset.
};

typedef NS_ENUM(NSUInteger, AUIRefreshViewType) {
    AUIRefreshViewDefault,           // The refresh style on the page.
    AUIRefreshViewTypeFeature1      // Apply it to the title bar, such as the homepage or the f
    ortune tab with a background.
};

@protocol AUIRefreshViewDelegate;

/**
 Set the animation view of mPaaS during pull-down refresh.
 */

@interface AUIRefreshView : UIView
@property (nonatomic, readonly) AUIRefreshViewState state;
@property (nonatomic, weak) id <AUIRefreshViewDelegate> delegate;
/**
 Set the style of the Lottie component during pull-down refresh.
 */

@property (nonatomic, strong) UIView /*LOTAnimationView */ *lottieAnimationView;
/* Specify the parent view where pull-down refresh is performed. The initial default height
of pull-down refresh is the height of scrollView. By default, refreshView is to add the par
ent scrollView.
 * The default initial frame is (0, 0 - scrollView.height, scrollView.width, scrollView.hei
ght)). */
- (instancetype)initWithSuperView:(UIScrollView *)scrollView
```

```

        type: (AURefreshViewType)type
        bizType: (NSString *)bizType;
// Set a text that is displayed during pull-down refresh.
- (void)setupLabelText:(NSString *)text;
// Call the following methods in "delegate" of UIScrollView:
- (void)auRefreshScrollViewWillBeginDragging:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidScroll:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidEndDragging:(UIScrollView *)scrollView;
// To end the animation and hide the list, call the following method:
- (void)auRefreshScrollViewDidFinishedLoading:(UIScrollView *)scrollView;
// The client needs to scroll the page to the initial position and then call the automatic
pull-down refresh function. Otherwise, a scrolling error will occur.
- (void)autoPullRefreshScrollView:(UIScrollView *)scrollView;
//
- (void)pauseAnimation;
// Expand the page.
- (void)resumeAnimation;
@end
@protocol AURefreshViewDelegate <NSObject>
@optional
// This protocol is triggered when the bar is dropped down to the default position, which i
s the height of (Lottie)View.
- (void)auRefreshViewDidTriggerloading:(AURefreshView *)view;
// Complete the reset action after a pull-down refresh.
- (void)auRefreshViewDidDidFinishAnimation:(AURefreshView *)view;
@end

```

Sample code

This section provides the sample code of standard style and custom style refresh view.

Standard style

The following shows the sample code for a standard style refresh view.

```
_refreshView = [[AURefreshView alloc] initWithSuperView:self.tableView type:AURefreshViewDefault bizType:@"demo"];
[_refreshView setupLabelText:@"Refreshing"];
[self.tableView addSubview:_refreshView];
- (void)viewDidAppear:(BOOL)animated
{
    [super viewDidAppear:animated];
    [_refreshView resumeAnimation];
}
- (void)viewDidDisappear:(BOOL)animated
{
    [super viewDidDisappear:animated];
    [_refreshView pauseAnimation];
}
- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView
{
    [_refreshView auRefreshScrollViewWillBeginDragging:scrollView];
}
- (void)scrollViewDidScroll:(UIScrollView *)scrollView
{
    [_refreshView auRefreshScrollViewDidScroll:scrollView];
}
- (void)scrollViewDidEndDragging:(UIScrollView *)scrollView willDecelerate:(BOOL)decelerate
{
    [_refreshView auRefreshScrollViewDidEndDragging:scrollView];
}
```

Custom style

To customize styles, you must use the Lottie component and rewrite the `Category` of `AUThemeManager`.

The following sample code is for your reference:

```
+ (NSString *)au_defaultTheme_refresh_lottie_path{

    NSString *path = [[NSBundle mainBundle] pathForResource:@"ani" ofType:@"json"];
    return path;

}
```

1.5.14. Other components

1.5.14.1. Carousel component

AUBannerView is a carousel component.

Sample image



API description

```
typedef NS_ENUM(NSUInteger, AUBannerStyle) {
    AUBannerStyleDeepColor, // The deep color style.
    AUBannerStyleLightColor // The light color style.
};

@interface AUBannerViewConfig : NSObject

@property (nonatomic, assign) AUBannerStyle style; //
// The default style.
@property (nonatomic, strong) UIColor *pageControlNormalColor; //
// The default color.
@property (nonatomic, strong) UIColor *pageControlSelectedColor; //
// The selected color.
@property (nonatomic, assign) CGFloat pageControlMarginBottom; //
// The margin between the pagination identifier and bottom.
@property (nonatomic, assign) BOOL pageControlDotTapEnabled; //
// Specify whether the pagination identifier (dot) is clickable. The default value is NO.
@property (nonatomic, assign) UIEdgeInsets contentViewMargin; //
// The margin of the content area.
@property (nonatomic, assign) UIEdgeInsets contentViewPadding; //
// The padding of the content area. An image will pass the padding when it scrolls.
@property (nonatomic, assign) BOOL autoTurn; //
// Whether to enable automatic carousel. Default value: YES.
@property (nonatomic, assign) BOOL autoStartTurn; //
// Whether to automatically start carousel.
@property (nonatomic, assign) CGFloat duration; //
// The automatic carousel interval.
```

```

@end

@class AUBannerView;
@protocol AUBannerViewDelegate <NSObject>

@required
- (NSInteger)numberOfItemsInBannerView:(AUBannerView *)bannerView;
- (UIView *)bannerView:(AUBannerView *)bannerView itemViewAtPos:(NSInteger)pos;

@optional
- (void)bannerView:(AUBannerView *)bannerView didTapedItemAtPos:(NSInteger)pos;
- (CGFloat)bannerView:(AUBannerView *)bannerView durationOfItemAtPos:(NSInteger)pos;

@end

@interface AUBannerView : UIView

AU_UNAVAILABLE_INIT

@property (nonatomic, readonly) UIView *contentView; // The content area view.
@property (nonatomic, readonly) AUPageControl *pageControl; // The pagination identifier view.

@property (nonatomic, copy) NSString *bizType; // The business type.
@property (nonatomic, assign) NSInteger currentPage; // The current page, which starts from page 0.
@property (nonatomic, weak) id<AUBannerViewDelegate> delegate; // The data source and event delegate.

/**
 Create a banner view.

@param frame frame
@param bizType The business type, which cannot be left empty.
@param configOperation The configuration block.
@return The banner view.
*/
- (instancetype)initWithFrame:(CGRect)frame
    bizType:(NSString *)bizType
    makeConfig:(void (^)(AUBannerViewConfig *config))configOperation;

/**
 Start automatic carousel. (Call this method only when autoStartTurn is set to NO.)
*/
- (void)startTurning;

```

```

/**
 * Reload the banner. (Call this method to reload data when the data source changes.)
 */
- (void)reloadData;

@end

//#####
//##### UIImage #####
//#####

@interface AUBannerView (Image)

/**
 * Create a banner view for images.
 * Note: Ensure that the value of images is the same as that of actionURLs, or the banner view will fail to be created.
 *
 * @param frame          The frame.
 * @param bizType         The business type, which cannot be left empty.
 * @param images          The image set, which can be an array of image link strings or image objects.
 * @param placeholder     The image placeholder, or the UIImage object.
 * @param actionURLs      The link to which a user is redirected after the user taps the corresponding image. The link is a string. If an image does not support redirection, set this parameter to [NSNull null].
 * @param configOperation The banner view configuration parameter.
 * @return                The banner view of image carousel.
 */
+ (instancetype)bannerViewWithFrame:(CGRect)frame
                        bizType:(NSString *)bizType
                        images:(NSArray *)images
            placeholder:(UIImage *)placeholder
            actionURLs:(NSArray *)actionURLs
      makeConfig:(void (^)(AUBannerViewConfig *config))configOperation;

@end

//#####
//##### Extension #####
//#####

@interface AUBannerView (Extension)

/**

```

```
Update the banner view configuration.
A reloading event will be automatically triggered.

@param update The update block.
*/
- (void)updateConfigOperation:(void(^)(AUBannerViewConfig *config))update;

@end
```

Code sample

```
// The common deep color banner.
for (NSInteger i = 0; i < 1; i++) {
    CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);

    AUBannerView *bannerView = [[AUBannerView alloc] initWithFrame:rect
                                                                    bizType:@"demo"
                                                                    makeConfig:^(AUBannerViewConfig
*config)
                                                                    {
                                                                    config.duration = 1.5;
                                                                    config.contentViewMargin = UIEdgeInsetsMake(5, 5,
                                                                    10, 5);
                                                                    config.contentViewPadding = UIEdgeInsetsMake(0, 5
                                                                    0, 0, 50);

                                                                    config.style = AUBannerStyleDeepColor;
                                                                    config.autoTurn = YES;
                                                                    config.autoStartTurn = YES;
                                                                    }]];

    bannerView.delegate = self;
    bannerView.tag = 1;
    bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
    [self.view addSubview:bannerView];
}

// The common light color banner.
for (NSInteger i = 1; i < 2; i++) {
    CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);

    AUBannerView *bannerView = [[AUBannerView alloc] initWithFrame:rect
                                                                    bizType:@"demo"
                                                                    makeConfig:^(AUBannerViewConfig
*config)
                                                                    {
                                                                    config.duration = 1.5;
                                                                    config.style = AUBannerStyleLightColor;
                                                                    config.autoTurn = NO;
                                                                    config.pageControlDotTapEnabled = YES;
                                                                    }]];

    bannerView.delegate = self;
    bannerView.tag = 2;
    bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
}
```

```

        bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1],
        [self.view addSubview:bannerView];
    }

    // The banner with images only.
    for (NSInteger i = 2; i < 3; i++) {
        CGRect rect = CGRectMake(10, 10 + (height + spaceY) * i, self.view.width - 20, height);

        NSMutableArray *images = [NSMutableArray array];
        for (NSInteger j = 0; j < 5; j++) {
            UIImage *image = [UIImage imageNamed:[NSString stringWithFormat:@"%d.jpg", @(j + 1)]];
            [images addObject:image];
        }
        AUBannerView *bannerView = [AUBannerView bannerViewWithFrame:rect
                                                                    bizType:@"demo"
                                                                    images:images
                                                                    placeholder:nil
                                                                    actionURLs:nil
                                                                    makeConfig:NULL];
        bannerView.backgroundColor = [UIColor colorWithWhite:0 alpha:0.1];
        [self.view addSubview:bannerView];
    }

```

```
#pragma mark - AUBannerViewDelegate

- (NSInteger)numberOfItemsInBannerView:(AUBannerView *)bannerView
{
    return bannerView.tag == 1 ? 2 : 4;
}

- (UIView *)bannerView:(AUBannerView *)bannerView itemViewAtPos:(NSInteger)pos
{
    NSArray *array = nil;
    // The deep color.
    if (bannerView.tag == 1) {
        array = @[RGB(0x108EE9), RGB_A(0x108EE9, 0.5), [UIColor blueColor], [UIColor yellowColor]];
    }
    // The light color.
    else {
        array = @[RGB(0xFFFFFFFF), RGB_A(0xFFFFFFFF, 0.7), RGB(0xcFFFFFFF), RGB_A(0xcFFFFFFF, 0.5), RGB_A(0xcFFFFFFF, 0.9)];
    }

    UIView *view = [[UIView alloc] init];
    view.backgroundColor = array[pos];
    return view;
}

- (void)bannerView:(AUBannerView *)bannerView didTapedItemAtPos:(NSInteger)pos
{
    NSLog(@"didTapedItemAtPos %@", @(pos));
}

// - (CGFloat)bannerView:(AUBannerView *)bannerView durationOfItemAtPos:(NSInteger)pos
// {
//     return 1;
// }
```

1.5.14.2. Segment component

AUSegment provides a switching bar style that supports scrolling.

AUSegment is provided based on the latest UED requirements. It cannot be used interchangeably with APSegmentedControl in APCommonUI because APSegmentedControl encapsulates the system component UISegmentedControl but does not provide any other functions.

Sample image



Dependency

The dependency of AUSegment is as follows:

```
import "AUSegmentedControlItem.h"
```

API description

```
@protocol AUSegmentedControlDelegate <UIScrollViewDelegate>
// The AUSegment clicking event callback.
@optional

- (void)didSegmentValueChanged:(AUSegment*) segmentControl;

- (void)didSelectSegmentItemModel:(AUSegmentItemModel*) selectedItemModel;

@end

// The default segment height.
#define AUSegmentHeight AU_SPACE13

/**
 The AUSegment component.
 */
```

```

@interface AUSegment : UIScrollView

/**
 The initialization function.

@param frame The frame.
@param titles The array that contains all title strings.

@return Return the AUSegment instance.
*/
- (instancetype)initWithFrame:(CGRect)frame titles:(NSArray<NSString*> *)titles;

/**
 Disable the init method.
*/
- (instancetype)init NS_UNAVAILABLE;

/**
 Disable the initWithFrame method.
*/
- (instancetype)initWithFrame:(CGRect)frame NS_UNAVAILABLE;

/**
 AUISegmentedControlDelegate
*/
@property (nonatomic, weak) id <AUISegmentedControlDelegate> delegate;

/**/

/**
 The title array.
*/
@property (nonatomic, strong) NSMutableArray *titles;

/**
 * The title font.
*/
@property (nonatomic, copy) UIFont *titleFont;

/**
 The selected segment index.
*/
@property (nonatomic, assign) NSInteger selectedIndex;

/**
 The color of the selected item (including the text and slider).
*/
@property (nonatomic, copy) UIColor *selectedColor;

/**
 * The left and right margins of each text menu in the horizontal direction.
 * The default value is 21 px.
 * When a menu contains a red dot, the value of fixedItemWidth is invalid and the menu width

```

```

h is not fixed.
 */
@property(nonatomic, assign) NSInteger textHorizontalPadding;

/**
 * Whether to use a fixed menu width.
 * The default value is YES, for compatibility with old menu styles.
 * When the value is YES, the value of textHorizontalPadding is invalid, and all menus are
in fixed width.
 */
@property (nonatomic, assign) BOOL fixedItemWidth;

/**
 * Whether to automatically scroll the selected menu to an appropriate position (middle pos
ition preferred, and displayed to the side when the space is insufficient).
 * The default value is NO.
 */
@property (nonatomic, assign) BOOL autoScroll;

/**
 * Whether to automatically move the indicator bar below the index of the selected item aft
er clicked.
 * The default value is YES.
 */
@property (nonatomic, assign) BOOL autoChangeSelectedIndex;

/*
 * The model array.
 */
@property(nonatomic, strong) NSMutableArray<AUSegmentItemModel *> *itemModels;

/**
Multiple items can be inserted in the middle.

@param array The inserted title array.
@param indexes The inserted indexes.
 */
- (void)insertTitleArray:(NSArray<NSString*> *)array atIndexes:(NSIndexSet *)indexes;

/**
Multiple items can be added to the end.

@param array The added title array.
 */
- (void)addTitleArray:(NSArray<NSString*>*)array;

/**
 * Set automatic scrolling to the specified subscript position. Note: Items are displayed i
n scrolling mode, and the indicator bar stays in the same position.
 * The value is the same as that of selectedIndex (indicating the index of the selec
ted segment) by default.
 */
- (void)autoScrollToIndex:(NSInteger)index;

```

```

- (BOOL)segmentItemIsInVisualAear:(NSInteger)index;

@end

@interface AUSegmentItemModel : NSObject

@property(nonatomic, copy) NSString *title;
@property(nonatomic, copy) UIImage *img;
@property(nonatomic, copy) NSString *imgId;
@property(nonatomic, copy) NSString *badgeNumber;
@property(nonatomic, copy) NSString *badgeWidgetId;
@property(nonatomic, assign) BOOL badgeReserved; // Whether to reserve a red dot position for the current item. If no red dot position is reserved, the item may flicker when containing a red dot.
@property(nonatomic, strong) NSDictionary *extendInfo; // The extended field.

@end

@interface AUSegment (ItemModel)

/**
 * The initialization function for version 2.
 * @param frame The frame.
 * @param menus The item array.
 */
- (instancetype) initWithFrame:(CGRect)frame menus:(NSArray<AUSegmentItemModel *>*)menus;

/**
 Control items can be updated.

 @param items The array of items that need to be updated, mainly for adding or deleting model data or updating all existing model data.
 */
- (void)updateItems:(NSArray<AUSegmentItemModel *>*)items;

/**
 Control items can be updated.

 @param items Delete existing item data and replace it with new item data.
 */
- (void)updateItemModel:(AUSegmentItemModel *)model
                    atIndex:(NSInteger)index;

@end

// The action button, plus sign (+) by default, is displayed on the right.
@interface AUSegment (AUActionIcon)

```

```
- (void)showActionIcon:(BOOL)showIcon target:(id)target action:(SEL)action;

@end
```

Custom properties

Property	Purpose	Type
titles	The segment title array.	NSArray
selectedSegmentIndex	The selected segment index.	NSInteger
delegate	Implement AUISegmentedControlDelegate.	ID
autoScroll	Whether to automatically scroll the selected item to an appropriate position (middle position preferred, and displayed the side when the space is insufficient).	BOOL
fixedItemWidth	Whether to use a fixed menu width.	BOOL
textHorizontalPadding	The left and right margins of each text menu in the horizontal direction.'	BOOL
titleFont	The custom title font.	UIFont

Sample code

- Segment controls without red dot:

```
NSArray *testArray1 = @[@"tab1",@"tab2",@"tab3",@"tab4",@"tab5",@"tab6",@"tab7",@"tab8"];

AUISegment *segment = [[AUISegment alloc] initWithFrame:CGRectMake(0, 300, self.view.width, 44) titles:testArray1];
segment.delegate = self;
[self.view addSubview:segment];

// The callback.
- (void)didSegmentValueChanged:(AUISegment*)segmentControl {
    NSLog(@"AUISegmented switched");
}
```

- Segment controls with red dot:

```
NSMutableArray *array = [[NSMutableArray alloc] init];
for (int i=0; i<7; i++)
{
    AUSegmentItemModel *model = [[AUSegmentItemModel alloc] init];
    model.title = [NSString stringWithFormat:@"Option %d", i];
    if (i == 0)
    {
        model.badgeNumber = @".";
    }
    if (i == 1)
    {
        model.badgeNumber = @"new";
    }
    if (i == 6)
    {
        model.badgeNumber = @"6";
    }
    model.badgeReserved = YES;
    [array addObject:model];
}
AUSegment *segment2 = [[AUSegment alloc] initWithFrame:CGRectMake(0, topMargin, self.view.width, 44) menus:array];
[self.view addSubview:segment2];
[segment2 scrollToIndex:6];
segment2.backgroundColor = [UIColor whiteColor];
[segment2 showActionIcon:YES target:self action:@selector(clickActionIcon)];
```

1.5.14.3. Icon component

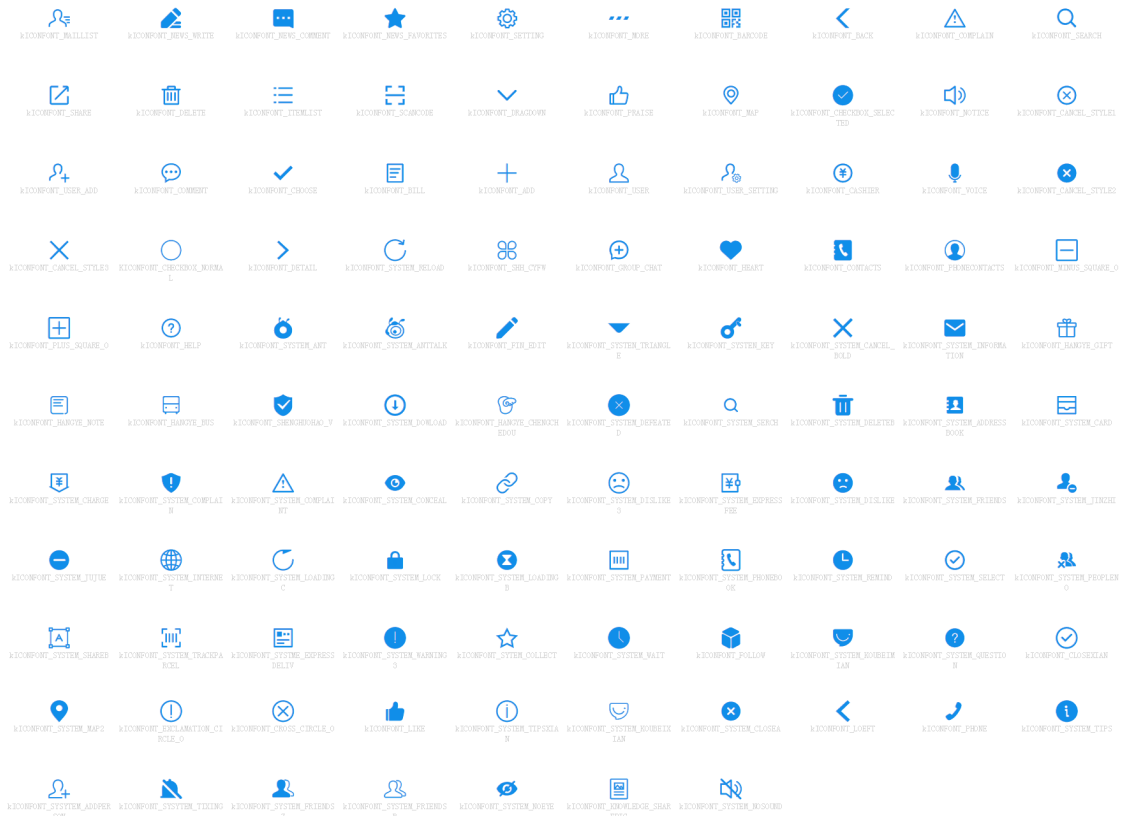
AUIconView is an iconfont vector icon control. The usage is similar to that of UIImageView. The control, which can be used as an UIImageView, is an image object drawn by using the drawRect feature of the string.

ⓘ Note

Currently, only square vector icons are supported.

- You can consider that an iconfont loads a font. The font is associated with multiple images and each image has a Unicode character. Therefore, you can set text to the corresponding Unicode character and call the drawInRect method of the string to render the iconfont.
- Each iconfont set is a .ttf font file. You can load multiple .ttf font files, each of which has a name. The default iconfont is the ttf font of AntUI, named auiconfont.

Sample image



API description

```
// The default AntUI iconfont name.
#define kICONFONT_FONTNAME                (@"auiconfont")
// The default AntUI iconfont path.
#define kICONFONT_FONTPATH                (@"APCommonUI.bundle/iconfont/auiconfont")
```

/ **

The `iconfont` control, which can be used as an `ImageView`.

Actually, the control is an image object drawn by using the drawRect feature of the string.
Note: Currently, only square vector icons are supported.

You can consider that an iconfont loads a font. The font is associated with multiple images and each image has a Unicode character.

Therefore, you can set text as the corresponding Unicode character and call the `drawInRect` method of the string to render the iconfont.

Each iconfont set is a .ttf font file. You can load multiple .ttf font files, each of which has a name. The default iconfont is the ttf font of AntUI, named auiconfont.

 $\ast /$

```
@interface AUIconView : UIImageView
```

```
@property (nonatomic, strong) UIColor *color;           // The vector diagram color (ant blue b
y default).
```

```

@property (nonatomic, strong) NSString *name; // The vector diagram name.
@property (nonatomic, strong) NSString *fontName; // The vector icon library name.

/**
The initialization method.

@param frame The view frame.
@param name The iconfont vector icon name.

@return Return an AUIconView instance.
*/
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name;

/**
The initialization method.
(If the iconfont has been loaded, it can be rendered without fontPath.)

@param frame The view frame.
@param name The iconfont image name.
@param fontName The iconfont name.

@return Return an AUIconView instance.
*/
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name fontName:(NSString *)fontName;

/**
Get the iconView size.

@return If an iconfont is used, the iconfont size is returned. If an common UIImageView is used, the image size is returned.
*/
- (CGSize)iconViewSize;

@end

@interface UIImage (AUIconFont)

/**
Register the iconfont. (This method needs to be called only once.)

@param fontName The iconfont name.
@param fontPath The iconfont path, such as @"AntUI.bundle/iconfont/auiconfont".
*/
+ (void)registerIconFont:(NSString *)fontName fontPath:(NSString *)fontPath;

/**
Get a square vector icon (with the same width and length).

@param name The image name.
@param width The image width.
@param color The image color. If the value is nil, the color is ant blue by default.

```

```

@param color The image color. If the value is nil, the color is ant blue by default.

@return Return a square vector icon.
*/
+ (UIImage *)iconWithName:(NSString *)name
width:(CGFloat)width
color:(UIColor *)color;

/**
Get a square vector icon (with the same width and length).

@param name The name.
@param fontName The vector font name.
@param width The size.
@param color The image color. If nil is imported, the color is ant blue by default.

@return Return a square vector icon.
*/
+ (UIImage *)iconWithName:(NSString *)name
fontName:(NSString *)fontName
width:(CGFloat)width
color:(UIColor *)color;

@end

```

Sample code

```

// Use AUIconView.
AUIconView *view = [[AUIconView alloc] initWithFrame:CGRectZero name:_array[indexPath.row]]
;
view.tag = 1;

view.size = CGSizeMake(30, 30);
view.origin = CGPointMake(100, 10);
view.color = RGB(0x2b91e2);
[cell.contentView addSubview:view];

// Use the image extension independently.
self.image = [UIImage iconWithName:self.name fontName:self.fontName width:width color:self.
color];

```

1.5.14.4. Index component

The AUBladeView provides alphabetical index function. clicking or sliding to the letter on the alphabetical index on the left or right side of the page to trigger the event at the corresponding region.

Sample image



API description

AUBladeView.h

```
//
//  indexBar.h
//

#import <Foundation/Foundation.h>

#define kIndexSearchTitle    @"Search"

@protocol AUBladeViewDelegate;
/*!
 @class      AUBladeView
 @abstract   UIView
 @discussion The alphabetical index view.
 */
@interface AUBladeView : UIView

- (id)init;
- (id)initWithFrame:(CGRect)frame;
- (void)clearIndex;

@property (nonatomic, weak) id<AUBladeViewDelegate> delegate;
@property (nonatomic, strong) UIColor *highlightedBackgroundColor;
@property (nonatomic, strong) UIColor *textColor;
@property (nonatomic, strong) UIFont *textFont;
@property (nonatomic, strong) NSArray * iconImageNames;
@property (nonatomic, strong) NSArray * iconTitles;
@property (nonatomic, assign) BOOL enableSearch;
@property (nonatomic, strong) NSArray * defaultIndexes;

- (void)updateIndexes;

@end

@protocol AUBladeViewDelegate<NSObject>
@optional
- (void)indexSelectionDidChange:(AUBladeView *)indexBar index:(NSInteger)index title:(NSString*)title;
@end
```

Sample code

```
//
//  bladeViewController.m
//  AntUI
//

#import "bladeViewController.h"
#import "AUBladeView.h"

@interface bladeViewController ()<AUBladeViewDelegate,UITableViewDelegate,UITableViewDataSource>
@property (nonatomic,strong) AUBladeView * bladeView;
```

```
@property (nonatomic, strong)    NSArray * sectionArr;
@property (nonatomic, strong)    NSArray * mainDataIndexChar;
@property (nonatomic, strong)    UITableView * tableView;
@end

@implementation bladeViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    self.title = @"Select a city";
    // Do any additional setup after loading the view.
    self.tableView = [[UITableView alloc] initWithFrame:CGRectMake(0, 0, self.view.frame.size.width, self.view.frame.size.height) style:UITableViewStylePlain];
    self.tableView.delegate = self;
    self.tableView.dataSource = self;
    [self.view addSubview:self.tableView];
    self.bladeView = [[AUBladeView alloc] initWithFrame:CGRectMake(self.view.frame.size.width-16.0, 80, 16.0, self.view.bounds.size.height-60)];
    self.bladeView.delegate = self;

    NSString * plistStr = [[NSBundle mainBundle] pathForResource:@"APCommonUI_ForDemo.bundle/citydict" ofType:@"plist"];
    NSDictionary * srcPlistDic = [NSDictionary dictionaryWithContentsOfFile:plistStr];
    NSMutableArray * citysList = [[NSMutableArray alloc] initWithCapacity:27];
    [citysList addObject:@{@"Popular":@[@"Shanghai",@"Hangzhou",@"Guangzhou",@"Beijing",@"Shenzhen"]}];

    NSMutableArray * indexArrList = [[NSMutableArray alloc] initWithCapacity:27];
    [indexArrList addObject:@"Popular"];
    NSArray * keyList = [srcPlistDic allKeys];
    NSArray * sortedList = [keyList sortedArrayUsingComparator:^NSComparisonResult(id _Nonnull obj1, id _Nonnull obj2) {
        return [(NSString *)obj1 compare:obj2];
    }];
    [sortedList enumerateObjectsUsingBlock:^(id _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
        if (obj) {
            NSDictionary * tmpDic = [[NSDictionary alloc] initWithObjectsAndKeys:[srcPlistDic objectForKey:obj],obj, nil];
            [citysList addObject:tmpDic];
            [indexArrList addObject:obj];
        }
    }];
    self.sectionArr = citysList;
    self.mainDataIndexChar = indexArrList;
    // self.bladeView.iconTitles = secondaryIndexsTitles;
    // self.bladeView.iconImageNames = secondaryIndexsIcons;
    self.bladeView.defaultIndexes = self.mainDataIndexChar;
    [self.bladeView updateIndexes];
    [self.view addSubview:self.bladeView];
    self.tableView.showsVerticalScrollIndicator = NO;
}
```

```

        self.tableView.showsHorizontalScrollIndicator = NO ;
        [self.view bringSubviewToFront:self.bladeView];

    }

#pragma mark -----UITableViewDelegate
// Display customization
// Variable height support

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath
{
    return 44;
} // Use the estimatedHeight methods to quickly calculate guessed values which will allow for
fast load times of the table.

- (CGFloat)tableView:(UITableView *)tableView heightForHeaderInSection:(NSInteger)section
{
    return 35;
}
#pragma mark -----UITableViewDataSource
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section
{
    NSDictionary * test = [self.sectionArr objectAtIndex:section] ;
    NSArray * valueArr = [test objectForKey:([test allKeys] firstObject)];

    return [valueArr count];
}

- (nullable NSString *)tableView:(UITableView *)tableView titleForHeaderInSection:(NSInteger)section
{
    NSDictionary * test = [self.sectionArr objectAtIndex:section] ;

    return [[test allKeys] firstObject];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:@"BladeTableViewCell"];
    if (!cell) {
        cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:@"BladeTableViewCell"];
    }
    NSInteger section = [indexPath section];
    NSInteger row = [indexPath row];
    NSDictionary * test = [self.sectionArr objectAtIndex:section] ;
    NSArray * valueArr = [test objectForKey:([test allKeys] firstObject)];
    NSString * text = [valueArr objectAtIndex:row];
    cell.textLabel.text = text;
}

```

```

        cell.textLabel.text = text ;
        return cell;
    }

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView
{
    return [self.sectionArr count];
} // Default is 1 if not implemented

- (void)dealloc
{
    self.tableView.delegate = nil;
    self.tableView.dataSource = nil;
    self.bladeView.delegate = nil;
    self.bladeView = nil;
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation before navigation
- (void)prepareForSegue:(UIStoryboardSegue *)segue sender:(id)sender {
    // Get the new view controller using [segue destinationViewController].
    // Pass the selected object to the new view controller.
}
*/

#pragma mark ---- AUBladeViewDelegate
- (void)indexSelectionDidChange:(AUBladeView *)indexBar index:(NSInteger)index title:(NSString*)title
{
    if (self.tableView){
        NSInteger ret = 0;
        if ([title isEqualToString:kIndexSearchTitle]){
            [self.tableView scrollToRowAtIndexPath:[NSIndexPath indexPathForRow:0 inSection:ret]
                                atScrollPosition:UITableViewScrollPositionBottom
                                animated:NO];
            return;
        }
        else {
            ret = [self findIndexSection:title];
            if (ret != NSNotFound) {
                [self.tableView scrollToRowAtIndexPath:[NSIndexPath indexPathForRow:0 inSection:ret]
                                atScrollPosition:UITableViewScrollPositionTop
                                animated:NO];
            }
        }
    }
}

```

```
    }  
  
    }  
  
    - (NSInteger)findIndexSection:(NSString *)title {  
        NSInteger ret = NSNotFound;  
        int beginIndex = 0;  
        /*  
        The following shows a custom section calculation rule.  
        beginIndex += self.customSectionCount;  
        if (self.secondarySectionCount > 0 ){  
            for (NSInteger i = 0 ; i < [self.secondarySectionTitles count]; i++) {  
                NSString * secondaryTitle = [self.secondarySectionTitles objectAtIndex:i];  
                if ([secondaryTitle isEqualToString:title]) {  
                    ret = beginIndex + i;  
                    return ret;  
                }  
            }  
            beginIndex += self.secondarySectionCount;  
        }  
        */  
        if ([self.mainDataIndexChar count] > 0){  
            for(NSInteger i = 0; i < [self.mainDataIndexChar count]; i++) {  
                NSString * indexChar = [self.mainDataIndexChar objectAtIndex:i];  
                if ([indexChar isEqualToString:title]) {  
                    ret = i;  
                    break;  
                }  
            }  
        }  
        if (ret != NSNotFound) {  
            ret = ret + beginIndex;  
        }  
        return ret;  
    }  
  
@end
```

1.5.14.5. Title bar segment component

AUTitleBarSegment is a segmented control used at the top of the navigation bar. AUTitleBarSegment encapsulates UISegmentedControl, simply modifies the UI style of UISegmentedControl, and provides default width and height of each segment.

Sample image



API description

```

/*
    mPaaS standard: The segment controls can be used only at the top of the navigation bar
    .
    The default color value is used, and the default height of the navigation bar is 26 px
    .
    */

#define AUTitleBarSegment_DefaultHeight          26          // The default height of ea
ch segment is 26 px.
#define AUTitleBarSegment_DefaultSegmentWidth    90          // The default width of eac
h segment is 90 px.

@interface AUTitleBarSegment : UISegmentedControl

@end

```

Sample code

```

AUTitleBarSegment *titleBatSegment = [[AUTitleBarSegment alloc] initWithItems:@[@"Label", @
"Label"]];
self.navigationItem.titleView = titleBatSegment;

```

1.5.14.6. Navigation button

AUBarButtonItem in mPaaS is equivalent to UIBarButtonItem. It contains predefined items such as the color and font. To facilitate subsequent extension, AUBarButtonItem instead of UIBarButtonItem must be used in all mPaaS apps.

Currently, AUBarButtonItem completely is completely inherited from AUISwitch without any new properties or methods.

API description

```
/**
 * /
@interface AUBarButtonItem : UIBarButtonItem

@property(nonatomic, strong) NSString *backButtonTitle; // The title of the Back button.
@property(nonatomic, strong) UIImage *backButtonImage; // The icon of the Back button.
@property(nonatomic, strong) UIColor *titleColor; // The text color of the Back button.

/**
 * Set the spacing between buttons.
 *
 * @return Return an empty button of the UIBarButtonSystemItemFlexibleSpace style.
 */
+ (AUBarButtonItem *)flexibleSpaceItem;

/**
 * Create a default Back button style.
 *
 * @param title The title to be displayed.
 * @param target The tapping target.
 * @param action The action to be executed upon tapping.
 *
 * @return UIBarButtonItem
 */
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title target:(id)target action:
(SEL)action;

/**
 * Create a default Back button style.
 *
 * @param title The title to be displayed.
 * @param count The maximum number of characters to be displayed.
 * @param target The tapping target.
 * @param action The action to be executed upon tapping.
 *
 * @return UIBarButtonItem
 */
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title maxWordsCount:(NSInteger)
count target:(id)target action:(SEL)action;

@end
```

Code sample

```
// Define a backButtonItem.
// The backButtonItem contains a back icon by default.
AUIButtonItem *cancelItem = [AUIButtonItem backButtonItemWithTitle:@"Back" target:self
    action:@selector(cancel)];
cancelItem.backButtonItemTitle = @"Cancel";
self.navigationItem.leftBarButtonItem = cancelItem;

AUIButtonItem *rightItem1 = [[AUIButtonItem alloc] initWithImage:image1 style:UIBarButtonItemStylePlain
    target:self action:@selector(rightBarItemPressed)];
```

1.5.14.7. Adaptation and dependency

As the shell of Ant UI, AntUIShell is mainly used to implement third-party protocols in Ant UI. It can be embedded to an mPaaS app and reduce external dependencies of Ant UI.

API description

AntUIShellObject.h

```
//
//  AntUIShellObject.h
//  AntUIShell
//

#import <Foundation/Foundation.h>
#import <AntUI/AntUI.h>

@interface AntUIShellObject : NSObject<AThirdPartyAdapter>

@end
```

Code sample

```
//
//  AntUIShellObject.m
//  AntUIShell
//

#import "AntUIShellObject.h"
#import <APMonitor/APMonitor.h>
#import <APMultimedia/APMultimedia.h>
#import <MPBadgeService/MPBadgeService.h>

@implementation AntUIShellObject

#pragma mark ----AThirdPartyAdapter
/*****
// The image protocol APMultimedia.
*/
API adaptation for third-party to download image.
It wraps multimedia APIs and is implemented by a third party.
```

```

*/
- (NSString *)thirdPartyGetImage:(NSString *)identifier
    business:(NSString *)business
    zoom:(CGSize)size
    originalSize:(CGSize)originSize
    progress:(void (^)(double percentage, long long partialBytes, long long totalBytes))progress
    completion:(void (^)(UIImage *image, NSError *error))complete
{
    return [[APIImageManager manager] getImage:identifier business:business zoom:size originalSize:originSize progress:progress completion:complete];
}

/*
API adaptation for third-party to download UIImageView image.
It is implemented by a third party.
*/
- (void)thirdPartyFromImageView:(UIImageView *)fromImageView
    setImageWithKey:(NSString *)key
    business:(NSString *)business
    placeholderImage:(UIImage *)placeholder
    zoom:(CGSize)zoom
    originalSize:(CGSize)originalSize
    progress:(void (^)(double percentage, long long partialBytes, long long totalBytes))progress
    completion:(void (^)(UIImage *image, NSError *error))complete
{
    if(fromImageView && [fromImageView isKindOfClass:[UIImageView class]]) {
        [fromImageView setImageWithKey:key business:business placeholderImage:placeholder zoom:zoom originalSize:originalSize progress:progress completion:complete];
    }
}

/*****
// The badge protocol MPBadgeService.
*/
Initialize the badge view.
*/
- (UIView *) thirdPartyBadgeViewWithFrame:(CGRect)frame
{
    return [[MPBadgeView alloc] initWithFrame:frame];
}

/*
Set widgetId for the badge.
*/
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
    widgetId:(NSString *) widgetId
{
    if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
        MPBadgeView * tmpBadgeView = (MPBadgeView *)badgeView;
        tmpBadgeView.widgetId = widgetId;
    }
}

```

```

}
/*
Register the badge view to MPBadgeManager.
*/
- (void) thirdPartyBadgeViewReg:(UIView *)badgeView
{
    if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
        MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
        [[MPBadgeManager sharedInstance] registerBadgeView:tmpBadgeView];
    }
}

/**
 * Update the badge style.
 * @param badgeView The badge view.
 * @param badgeValue: @"." Display a red dot.
 *                  @"new" Display "new".
 *                  @"Number" Display a number. For a number greater than 99, display the more icon (...).
 *                  @"hui" Display "hui".
 *                  @"xin" Display "xin".
 *                  nil Clear the currently displayed content.
 * @return There is no return value.
 */
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
                           updateValue:(NSString *)badgeValue
{
    if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
        MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
        [tmpBadgeView updateBadgeValue:badgeValue];
    }
}

/*
Provide business personnel with an API for monitoring badge control updates.
The type of widgetInfo is MPWidgetInfo.
*/
- (void) thirdPartyBadgeViewWith:(UIView *)badgeView
                           updateBlock:(void(^)(id widgetInfo, BOOL isShow)) updateBlock
{
    if(badgeView && [badgeView isKindOfClass:[MPBadgeView class]]) {
        MPBadgeView * tmpBadgeView =(MPBadgeView *)badgeView;
        if(updateBlock) {
            tmpBadgeView.updateBlock = updateBlock;
        }
    }
}

/*
The tracking protocol APMonitor.
*/

```

```

// The tracking protocol of actionName of the button.
- (void) thirdPartySetButtonActionLog:(UIButton *)button
    actionNameLog:(NSString *)actionName
{
    if(button && [button isKindOfClass:[UIButton class]]) {
        button.actionName = actionName;
    }
}

/*
The notification protocol AUCardMenu/AUFloatMenu.
*/

/*
AUCardMenu registers the logout notification, ensuring that AUCardMenu is destroyed upon l
ogout in a timely manner.
*/
- (NSString *) thirdPartyCardMenuDismissNotiName
{
    return @"SAAccountDidExitNotification";
}

/*
AUFloatMenu registers alerView kShareTokenAlertViewShownNotification.
*/
- (NSString *) thirdPartyFloatMenuDismissFromAlertNotiName
{
    return @"kShareTokenAlertViewShownNotification";
}

/*
AUFloatMenu registers alerView SALoginAppWillStartNotification.
*/
- (NSString *) thirdPartyFloatMenuDismissFromLoginNotiName
{
    return @"SALoginAppWillStartNotification";
}

@end

```

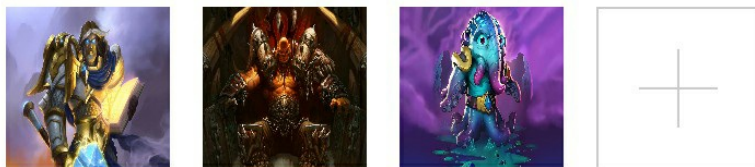
1.5.14.8. Image picker encapsulation

AUImagePickerSkeleton is an image selection component that encapsulates the vision and interaction functions. It does not support album calling, browsing, or uploading. Currently, the image selection functions are incomplete. The component that inherits functions will be added to BEEViews.

Sample image

Image
(optional, upload screenshots as an evidence)

3/4



Dependency

Currently, this component is not added to the AntUI baseline.

API description

```
@protocol AUIImagePickerDataProtocol <NSObject>

- (UIImage *)image;

@end

@interface AUIImagePickerSkeleton : UIView

- (AUIImagePickerSkeleton *)initWithTitle:(NSString *)title
    numberOfImages:(NSUInteger)numberOfImages;

@property(nonatomic, assign, readonly) NSUInteger numberOfImages;
@property(nonatomic, weak) id<AUIImagePickerDelegate> delegate;
@property(nonatomic, strong, readonly) NSArray<id<AUIImagePickerDataProtocol>> *imagePickerDatas;

- (void)updateImagePickerDatas:(NSArray <id<AUIImagePickerDataProtocol>>*) datas;

@end

@protocol AUIImagePickerDelegate <NSObject>

@required
- (void)imagePickerAddButtonClick:(AUIImagePickerSkeleton *)imagePicker;

@optional
- (void)imagePickerImageClick:(AUIImagePickerSkeleton *)imagePicker
    clickData:(id<AUIImagePickerDataProtocol>)clickData;

@end
```

Sample code

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.datas = [[NSMutableArray alloc] init];
    self.picker = [[AUIImagePickerSkeleton alloc] initWithTitle:@"Image(optional, upload screenshots as an evidence)"numberOfImages:4];
    self.picker.top = 100;
    self.picker.delegate = self;
    [self.view addSubview:self.picker];
    [self.view addSubview:self.button];
    self.view.backgroundColor = RGB(0xEBEBEB);
}

- (void)imagePickerAddButtonClick:(AUIImagePickerSkeleton *)imagePicker
{
    AUIImagePickerData *data = [AUIImagePickerData new];
    data.originalImage = [self getImageWithCount:[self.datas count]];
    [self.datas addObject:data];
    [self updatePickerAndResize];
}

- (void)imagePickerImageClick:(AUIImagePickerSkeleton *)imagePicker
    clickData:(id<AUIImagePickerDataProtocol>)clickData
{
    NSString *msg = @"";
    if ([self.datas containsObject:clickData]) {
        msg = [NSString stringWithFormat:@"Tap image %d", (int)[self.datas indexOfObject:clickData]+1];
    }else{
        msg = @"The image tapped is abnormal";
    }
    [AUIToast presentModalToastWithin:self.view
        withIcon:AUIToastIconNone
        text:msg
        duration:1
        logTag:@"demo"
        completion:NULL];
}
```