Ant Technology

Client UI Components
User Guide

Document Version: 20260303

蚂蚁集团
ANT GROUP

Legal disclaimer

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 `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

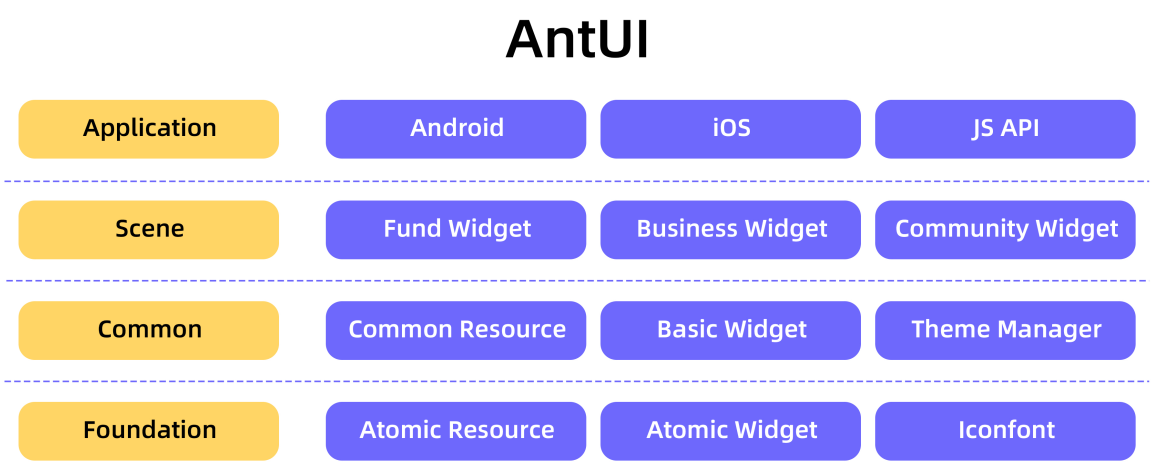
1.Client UI components

1.1. Introduction to the Native framework

The mPaaS unified component library (AntUI) translates abstract visual concepts into concrete controls based on standardized visual specifications. This ensures a consistent visual style on the client when you integrate controls.

Unified component library architecture

The AntUI architecture is built from the bottom up, similar to building blocks, to create a unified system of controls.



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

Architecture layer	Description
Foundation layer	This layer modularizes visual specifications and forms the base of the AntUI system. It includes atomic resources, atomic controls, and Iconfont icons. The foundation layer is built from the smallest units of the visual specifications.
Common layer	This is the core module of AntUI and contains the most frequently used controls. It includes common resources, basic controls, and a style manager. The common layer is built by combining elements from the foundation layer. It can be used in all common client scenarios.
Scene layer	This layer builds collections of controls with scenario-specific features, such as controls for funds, merchants, and social networking. Because mPaaS is a super app, its scale means that many services require custom handling. The scene layer builds custom controls for these scenarios on top of the common layer.

Application layer	This layer provides platform-specific handling and H5 container support. It resolves the conflict between unification and platform customization. While atomic, composite, and scene elements form the foundation of AntUI, practical applications must meet the needs of Android, iOS, and H5. The application layer provides interfaces for platform-specific customization to address these differences.

Foundation layer

The foundation layer modularizes visual specifications and forms the base of the AntUI system. It includes atomic resources, atomic controls, and Iconfont icons.

- **Atomic resources**

  Defines atomic resources used by controls, such as colors, sizes, and spacing, to ensure their uniqueness. Examples include colors such as red, yellow, and blue, and font sizes such as 12, 14, and 16.

- **Atomic controls**

  Wraps the native controls of the platform framework to build a basic library of atomic controls.

- **Iconfont icons**

  Collects icons for common scenarios and provides them in the Iconfont format as a library of usable control icons.

Common layer

The common layer is the core unified module of AntUI. It contains the most frequently used controls and includes common resources, basic controls, and a style manager.

- **Common resources**

  Redefines atomic resources according to usage scenarios, such as title color, content color, and link color.

- **Basic controls**

  This layer visually reproduces the controls defined in the visual design mockups on a one-to-one basis. It maintains consistent naming and implementation across the Android and iOS platforms to simplify client development.

- **Style manager**

  Defines styles abstractly and manages them centrally. This lets you switch a specific control between multiple skins. Style abstraction is implemented using incremental definitions. This means you only need to focus on the styles of the elements that your business requires.

Scene layer

The scene layer builds collections of controls with scenario-specific features, such as controls for funds, merchants, and social networking.

Application layer

The application layer provides capabilities such as platform-specific handling and H5 container support. It resolves the conflict between unification and platform customization.

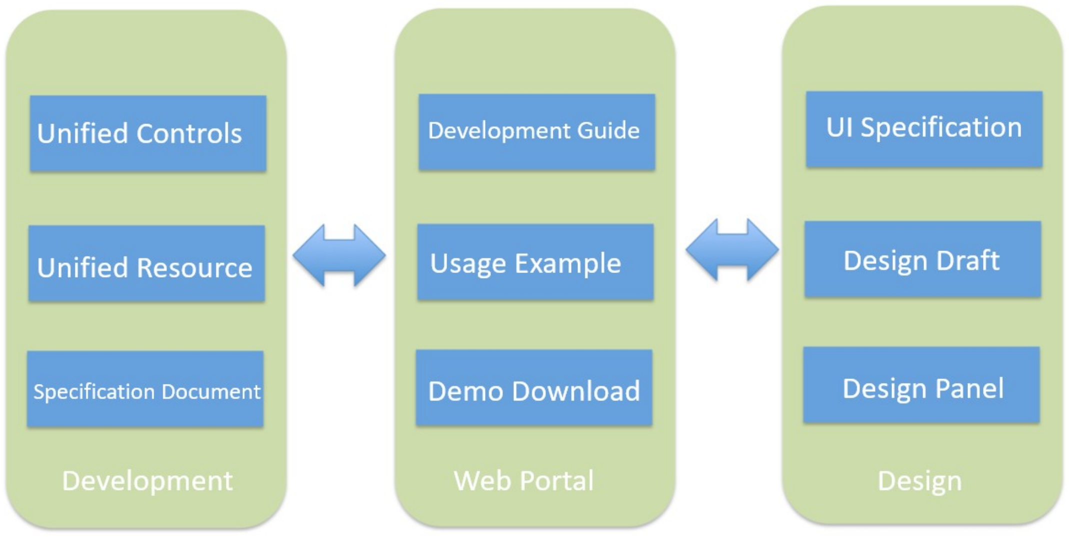
The Android and iOS platforms have different visual specifications. For example, AntUI handles actionsheets differently on each platform:

- On iOS, the actionsheet appears from the bottom.
- On Android, it is handled as a pop-up list in the center of the screen.

H5 content often requires different handling for elements such as pop-up dialog boxes and title bars. To ensure H5 content has a native look and feel, the unified component library defines a unified JavaScript Application Programming Interface (JSAPI) for the H5 container. This API makes it easy to call the appropriate platform controls. This allows H5 pages to be handled differently on Android and iOS.

Linkage

To reduce communication costs between designers and developers and to avoid redundant control development and visual design work, the unified component library (AntUI) integrates development and visual design tasks.



Designers create specifications, and developers interpret these specifications to create controls. A complete development guide simplifies implementation, forming a one-stop control system.

- Unified naming creates a shared understanding between developers and designers. For more information about naming conventions, see Component specifications and principles in this topic.
- Design boards help designers understand existing controls. They can build the basic structure of a page by simply dragging and dropping elements.
- The portal aggregates development documents and visual specifications. It also provides demo downloads so you can observe the visual effects of the controls directly.

Component specifications and principles

- **Naming style**

  Controls of the same type must have the same name on both the Android and iOS platforms. Control names are prefixed with AU. All custom control properties use camel case.

  > **Important**
  >
  > Some components may have platform differences. A component may need to be implemented on one platform but not on another.

- **Matching basic controls with visual/interaction specifications**
  - Controls not included in the specifications cannot be added to the standard controls.

    - Controls that are not in the specifications but are already used in multiple places should be placed in the candidate control collection.
    - A specification does not have to be implemented as a single control. For example, the title bar specification.
- **Usability**
    - Unlike commonui, do not create simple wrappers for system controls (such as APImageView and APTextView). When you need a system control, use the native control.
    - Names must be accurate and unambiguous.
    - Similar features should be consistent across different controls.
    - Follow user conventions.
- **Extensibility**
    - Do not use hard coding for control features. For example, support dynamic changes to the number of switchable tabs.
    - Some controls, such as dialog boxes and navigation bars, must allow their layout to be modified externally.
- **Novelty**

  You can use the latest platform features, such as Android's RecyclerView.

1.2. Native based - Android component library

1.2.1. Quick start

AntUI supports three types of access: **native AAR** and **component-based connection (Portal & Bundle)**.

Prerequisites

- For the native AAR connection type, complete the prerequisites and follow the steps in Add mPaaS to your project.
- For the component-based connection type, complete the Component-based connection flow.

Add SDK

Native AAR connection

Use **Component Management (AAR)** to install the **AntUI** component in your project. For more information, see AAR component management.

Component-based connection

In the Portal and Bundle projects, install the **AntUI** component using **Component Management**. For more information, see Manage component dependencies.

Code example

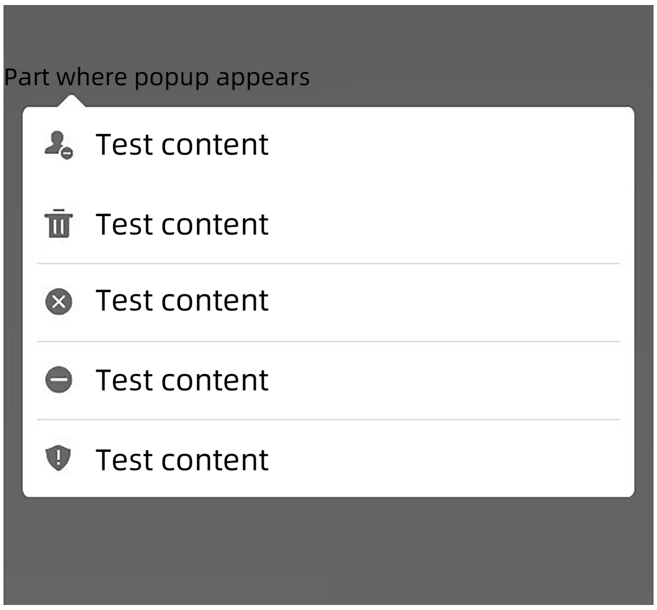
Download the Android demo and view the code in the `AntUI` project. For more information, see Get code examples.

1.2.2. Dialog component

1.2.2.1. Card menu

The AUCardMenu component is a dialog that functions as a popup window to display a selection menu when a user clicks a card on the mPaaS client homepage.

Preview



API reference

```
     /**
      * 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
marginRight)

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

        // If an image is from a network URL, you must download it.
        public void setOnLoadImageListener(OnLoadImageListener onLoadImageListener)

        public interface OnLoadImageListener {

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

Custom properties

None. XML layouts are not supported.

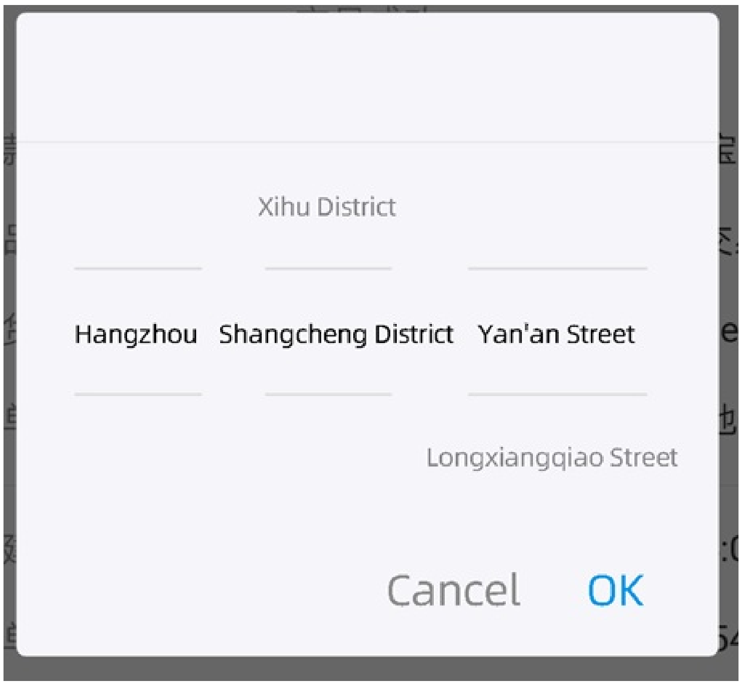
Code example

```
    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 = "Sample Text 1";
        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 = "Sample Text 2";
        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 = "Sample Text 3";
        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 = "Sample Text 4";
        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 = "Sample Text 5";
        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);
        }
```

1.2.2.2. 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

antUIGetCascadePicker

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 ([{“name”:”Hangzhou City”,subList:[{“name”:”Shangcheng District”}]}])	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+ m
odel);
                        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();
```

1.2.2.3. Date picker

AUDatePicker is a date picker control that appears as a pop-up window.

API reference

```
  /**
     * 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 AUDatePicker(Activity activity, @Mode int mode)

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

··/**
     * Selects a specific year, month, and day.
     *
     * @param year  the year
     * @param month the month
     * @param day   the day
     */
    public void setSelectedItem(int year, int month, int day)

  /**
     * Selects a specific date.
     *
     * @param yearOrMonth the year or month
     * @param monthOrDay  the month or day
     */
    public void setSelectedItem(int yearOrMonth, int monthOrDay)

  /**
     * Sets the listener for date selection.
     *
     * @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

This control has no custom properties and does not support XML layout files.

Code examples

```
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();
```

To use AUDatePicker on an embedded page:

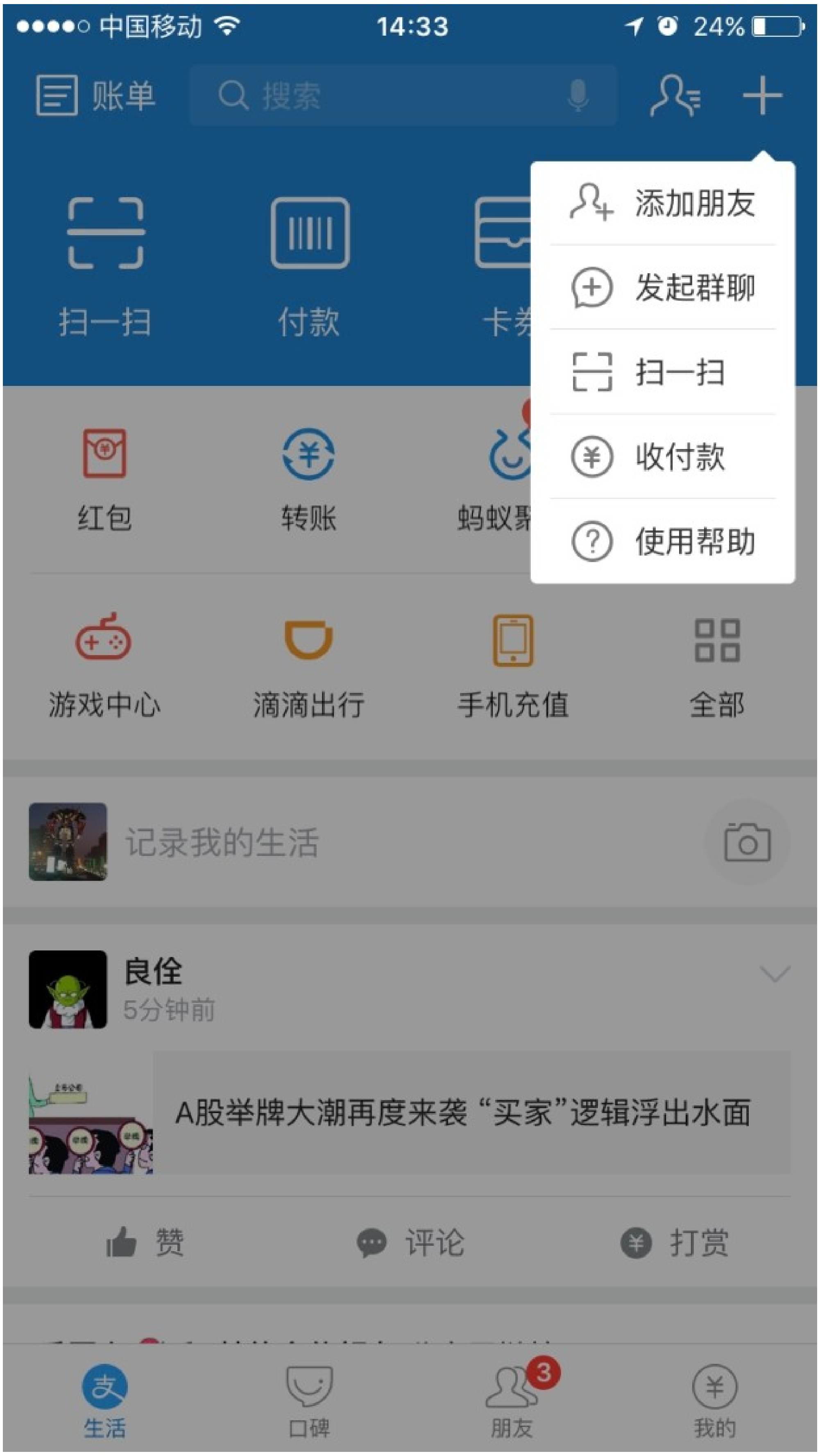
```
AUDatePicker picker = new AUDatePicker(this);
        picker.show();
        picker.dismiss();
        View view = picker.getOutterView();
        LinearLayout layout = (LinearLayout) findViewById(R.id.layout);
        layout.removeAllViews();
        if(view != null) {
            ((ViewGroup) view.getParent()).removeAllViews();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

For assistance, join our DingTalk group by searching for group number 145930007362 and contact the helpdesk.

1.2.2.4. Menu

The AUFloatMenu component provides a menu that contains icons and a list of options.

Preview



API reference

```
    /**
     * Constructor
     *
     * @param context The context of the activity that contains the antui-build depende
ncy.
     */
    public AUFloatMenu(Context context)
        /**
     * Displays the menu on the right by default.
     * @param view The view to which the menu is anchored.
     * @param popItems The model for the list to be displayed.
     */
    @Override
    public void showDrop(View view, ArrayList<MessagePopItem> popItems);

    /**
     * Displays the menu on the left.
     * @param view The view to which the menu is anchored.
     * @param popItems The model for the list to be displayed.
     */
    public void showAsDropDownLeft(View view, ArrayList<MessagePopItem> popItems);

    /**
     * Displays the menu in the center of the screen.
     * @param parent The view to which the menu is anchored.
     * @param title The title of the list to be displayed.
     * @param popItems The model for the list to be displayed.
     */
    public void showAsDropDownTitleCenter(View parent, String title,
ArrayList<MessagePopItem> popItems);

    /**
     * Adds a click event listener for the list items.
     * @param listener
     */
    public void setOnClickListener(AdapterView.OnItemClickListener listener)
```

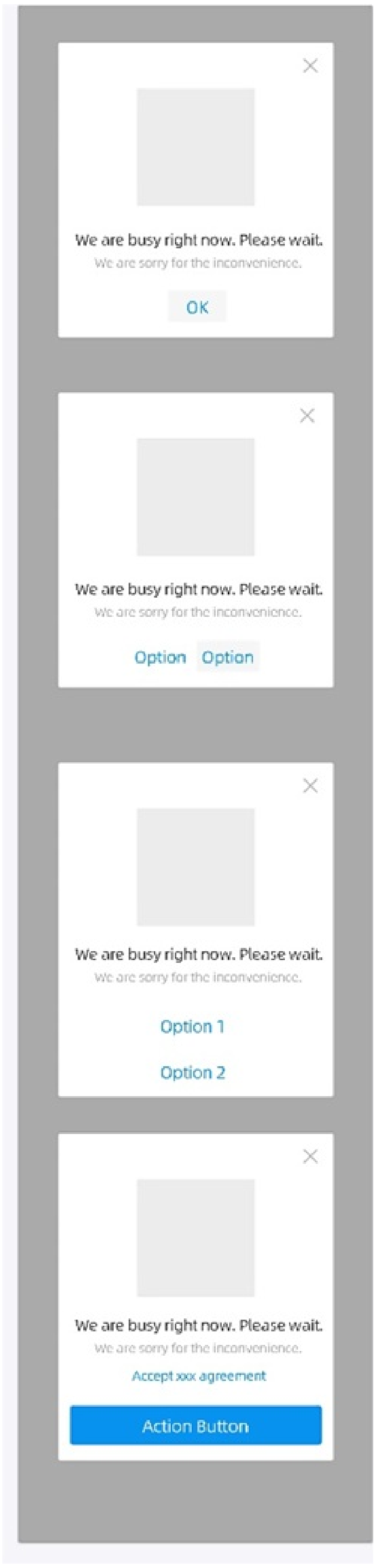
Code example

```
    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 friend";
    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 = "Receive/Pay";
    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();
        }
    });
```

1.2.2.5. Image dialog

AUImageDialog, formerly SalesPromotionLimitDialog, provides a dialog box with a title, up to three levels of text, and an ImageView. The dialog box has a confirmation button or two buttons at the bottom. You can use this component to display traffic-limiting messages.

Preview

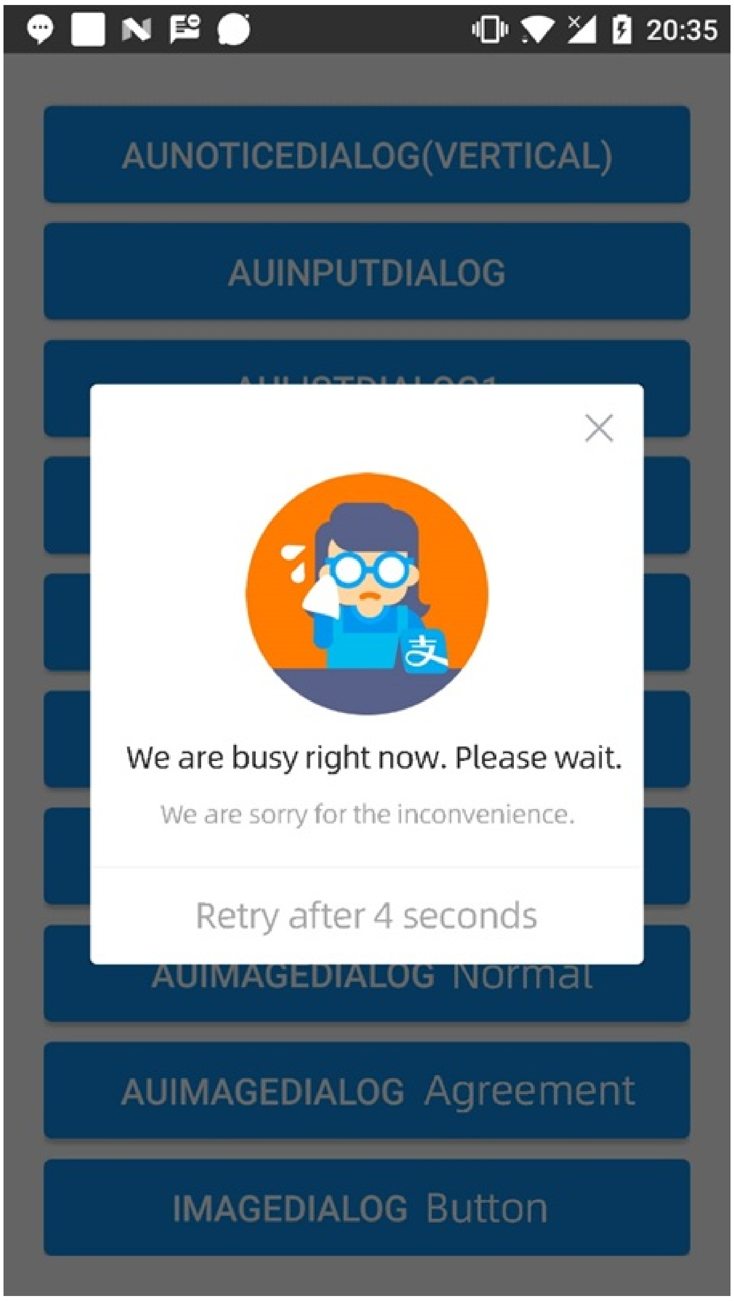

We are busy right now. Please wait.
We are sorry for the inconvenience.
OK
We are busy right now. Please wait.
We are sorry for the inconvenience.
Option
Option
We are busy right now. Please wait.
We are sorry for the inconvenience.
Option 1
Option 2
We are busy right now. Please wait.
We are sorry for the inconvenience.
Accept xxx agreement
Action Button

API reference

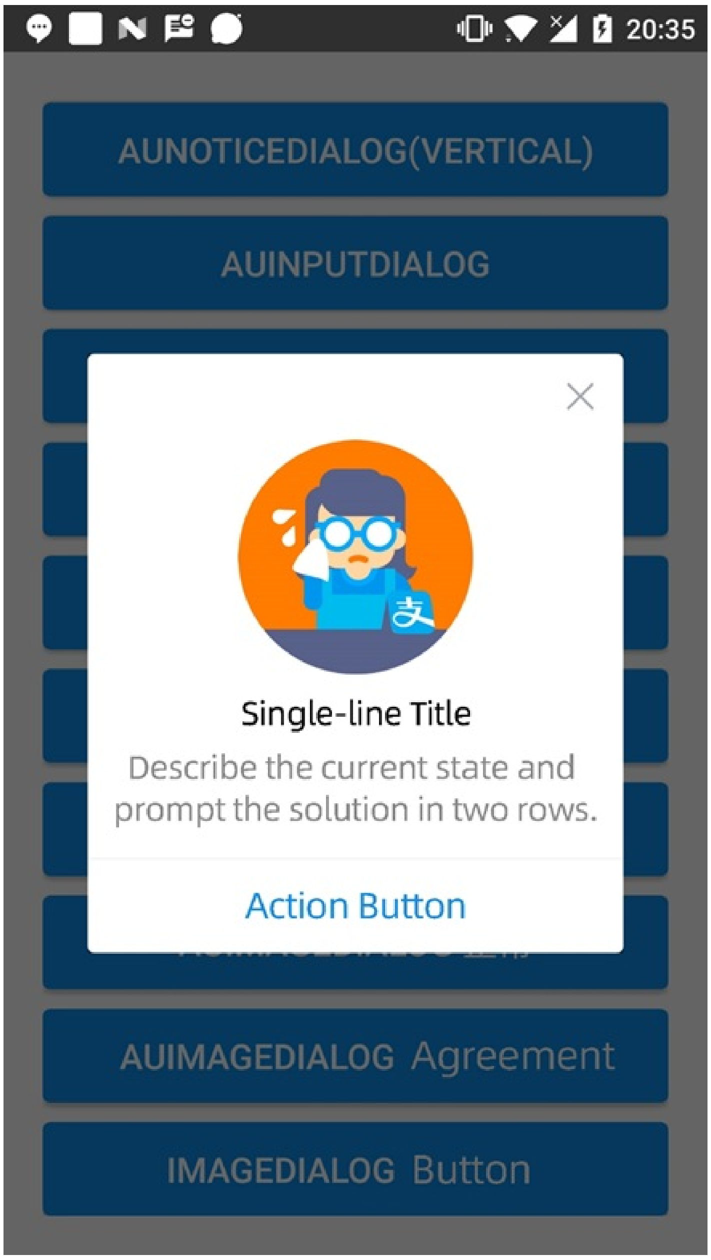
```
public interface OnItemClickListener {
    void onItemClick(int index);
}

/**
 * Gets an AUImageDialog instance.
 *
 * @param context The context object.
 * @return Returns an AUImageDialog instance.
 */
public static AUImageDialog getInstance(Context context)

/**
 * Sets a listener for the Close button.
 *
 * @param mCloseBtnClickListener
 */
public void setCloseBtnClickListener(View.OnClickListener mCloseBtnClickListener)


/**
 * Sets the primary title text.
 */
public void setTitle(CharSequence title)

/**
 * Sets the font size of the primary title text in sp.
 *
 * @param size
 */
public void setTitleTextSize(float size)

/**
 * Sets the visibility of the primary title.
 *
 * @param visibility
 */
public void setTitleTextVisibility(int visibility)
}

/**
 * Sets the visibility of the secondary title.
 *
 * @param visibility
 */
public void setSubTitleTextVisibility(int visibility)

/**
 * Sets the color of the primary title.
 *
 * @param color
 */
public void setTitleTextColor(int color)

/**
 * Sets the secondary title text.
 *
 * @param title
 */
public void setSubTitle(CharSequence title)

/**
 * Sets the font size of the secondary title text in sp.
 *
 * @param size
 */
public void setSubTitleTextSize(float size)

/**
 * Sets the color of the secondary title text.
 *
 * @param color
 */
public void setSubTitleTextColor(int color)

/**
 * Sets the tertiary title text.
 *
 * @param text
 */
public void setThirdTitleText(String text)

/**
 * Sets the color of the tertiary title.
 *
 * @param color
 */
public void setThirdTitleTextColor(int color)

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

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

/**
 * Sets the background color of the ImageView.
 *
 * @param color
 */
public void setLogoBackgroundColor(int color)

/**
```

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

/**
 * Checks whether an animation is used.
 */
public boolean isUsdAnim()

/**
 * Specifies whether to use an animation when the dialog box appears or disappears. The
default value is true.
 *
 * @param usdAnim
 */
public void setUsdAnim(boolean usdAnim)

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

/**
 * Sets the text of the confirmation button.
 *
 * @param text
 */
public void setConfirmBtnText(String text)


/**
 * Gets the confirmation button.
 */
public Button getConfirmBtn()

/**
 * Sets a click listener for the confirmation button.
 *
 * @param clickListener
 */
public void setOnConfirmBtnClickListener(View.OnClickListener clickListener)


/**
 * Shows the dialog box without an animation.
 */
public void showWithoutAnim()

/**
 * Sets a countdown timer.
 * @param seconds The countdown duration in 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)

/**
 * Gets the default color of the countdown timer.
 * @return
 */
public String getDefaultTimeColorStr()

/**
 * Dismisses the dialog box without an animation.
 */
public void dismissWithoutAnim()

@Override
public void dismiss()


public boolean isCanceledOnTouch() {
    return canceledOnTouch;
}

/**
 * Specifies whether the dialog box is automatically canceled when the image in the mid
dle is tapped.
 *
 * @param canceledOnTouch
 */
public void setCanceledOnTouch(boolean canceledOnTouch)


/**
 * Sets the list buttons.
 * @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;
}
```

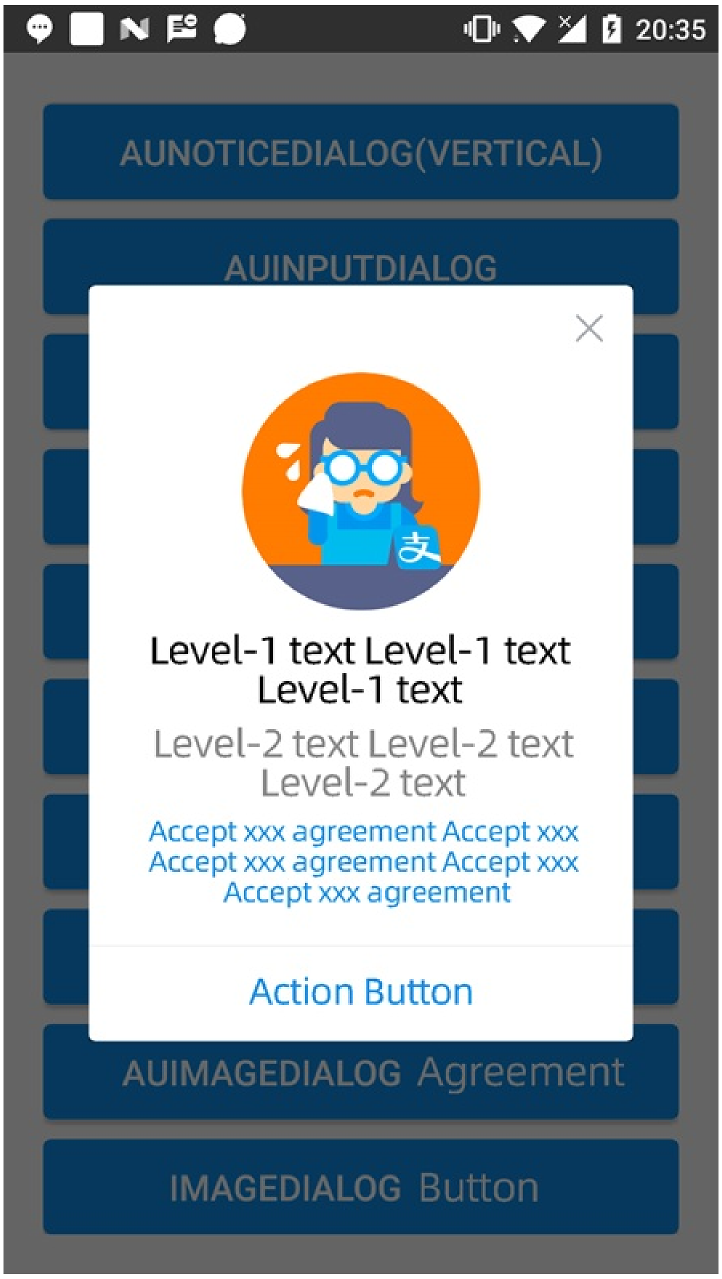
Code examples



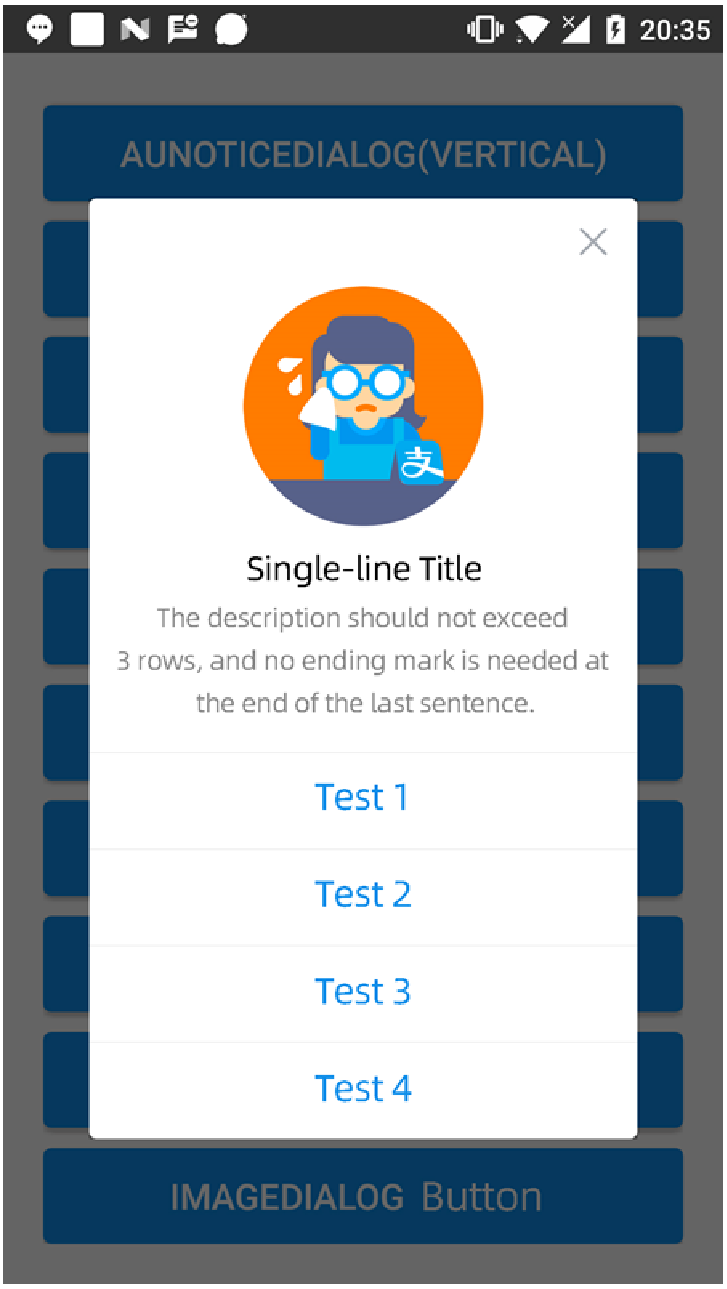
```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.showWithTimer(5, null, null);
```



```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("Single-line title");
dialog.setSubTitle("Describe the current status and suggest a solution. The text should
not exceed two lines.");
dialog.setConfirmBtnText("Action button");
dialog.showWithoutAnim();
```


20:35
AUNOTICEDIALOG(VERTICAL)
AUINPUTDIALOG
Level-1 text Level-1 text Level-1 text
Level-2 text Level-2 text Level-2 text
Accept xxx agreement Accept xxx Accept xxx agreement Accept xxx Accept xxx agreement
Action Button
AUIMAGEDIALOG Agreement
IMAGEDIALOG Button

```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setCanceledOnTouch(true);
dialog.setTitle("Primary text");
dialog.setSubTitle("Secondary text");
dialog.setThirdTitleText("Agree to the xxx protocol");
dialog.setConfirmBtnText("Action button");
dialog.showWithoutAnim();
```

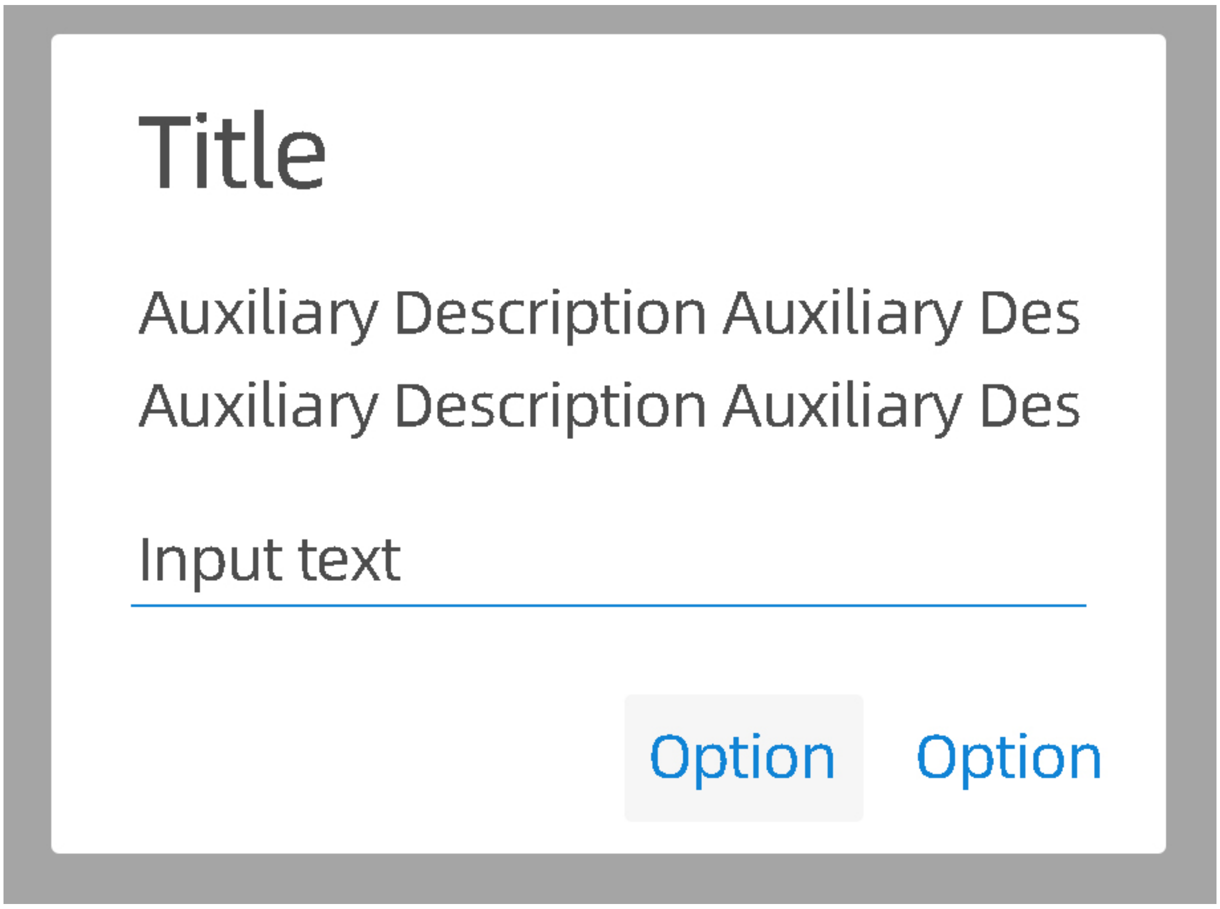

20:35
AUNOTICEDIALOG(VERTICAL)
Single-line Title
The description should not exceed 3 rows, and no ending mark is needed at the end of the last sentence.
Test 1
Test 2
Test 3
Test 4
IMAGEDIALOG Button

```
AUImageDialog dialog = AUImageDialog.getInstance(this);
dialog.setTitle("Single-line title");
dialog.setSubTitle("The description should be within three lines. Avoid ending a line w
ith a punctuation mark.");
dialog.setButtonListInfo(getData(), new AUImageDialog.OnItemClickListener() {
    @Override
    public void onItemClick(int index) {

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

1.2.2.6. Input dialog

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

Preview


Title
Auxiliary Description Auxiliary Des
Auxiliary Description Auxiliary Des
Input text
Option
Option

API reference

```
    /**
     * Constructs an AUInputDialog based on the input parameters.
     *
     * @param context The context object.
     * @param title The title.
     * @param msg The message.
     * @param positiveString The text for the confirm button.
     * @param negativeString The text for the cancel button.
     * @param isAutoCancel Specifies whether to automatically close the dialog box when
a user taps the area outside the dialog box.
     */
    public AUInputDialog(Context context, String title, String msg, String positiveStri
ng,
            String negativeString, boolean isAutoCancel)

    /**
     * Gets the cancel button.
     */
    public Button getCancelBtn();

    /**
```

```
     * Gets the confirm button.
     */
    public Button getEnsureBtn();

    /**
     * Gets the title TextView.
     */
    public TextView getTitle();

    /**
     * Gets the message TextView.
     */
    public TextView getMsg();

    /**
     * Gets the LinearLayout of the bottom buttons.
     */
    public LinearLayout getBottomLayout();

    /**
     * Gets the root RelativeLayout of the dialog box.
     */
    public RelativeLayout getDialogBg();

    /**
     * Sets the listener for the confirm button.
     */
    public void setPositiveListener(OnClickPositiveListener listener);

    /**
     * Sets the listener for the cancel button.
     */
    public void setNegativeListener(OnClickNegativeListener listener);

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

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

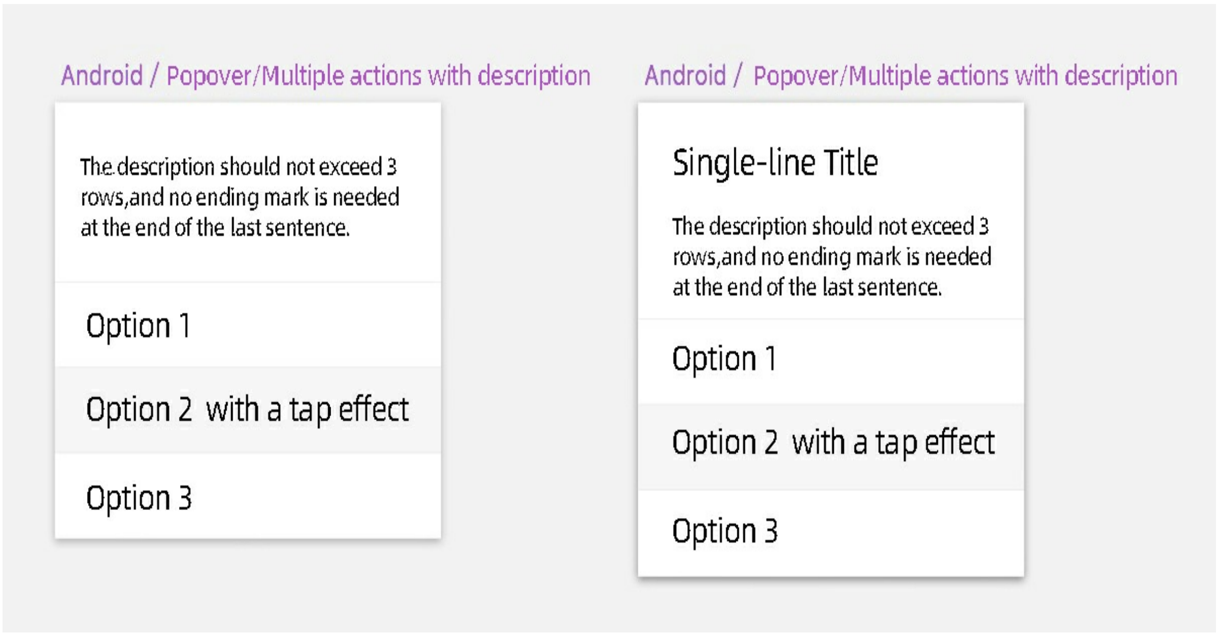
Code example

```
    AUInputDialog dialog = new AUInputDialog(this, "Title text", "Helper text",
                "Confirm", "Cancel", true);
        dialog.show();
```

1.2.2.7. List dialog

AUListDialog (formerly APListPopDialog) creates a list dialog box that can include a title, a list of options, and OK and Cancel buttons. Each option is a PopMenuItem object that contains information such as an icon, an option name, and a selection state.

Preview


Android / Popover/Multiple actions with description

The description should not exceed 3 rows,and no ending mark is needed at the end of the last sentence.

Option 1

Option 2 with a tap effect

Option 3

Android / Popover/Multiple actions with description

Single-line Title

The description should not exceed 3 rows,and no ending mark is needed at the end of the last sentence.

Option 1

Option 2 with a tap effect

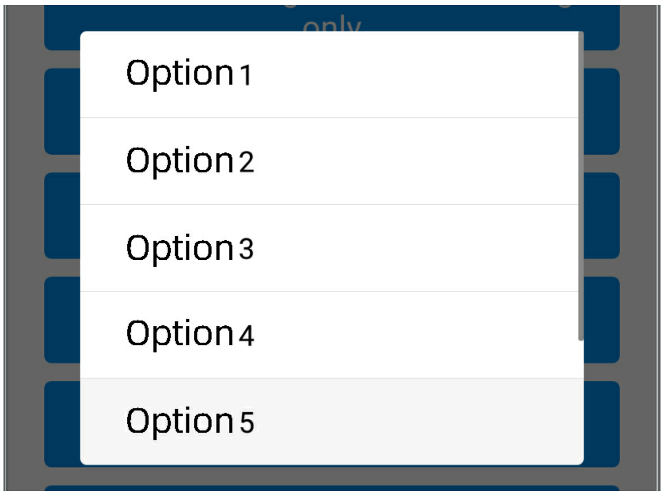
Option 3

API reference

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

/**
 * Creates an AUListDialog from the specified list data. The items contain only text an
d no images.
 *
 * @param context The context object.
 * @param list A list of strings. These are pure ItemName properties with no images.
 */
public AUListDialog(Context context, ArrayList<String> list)

/**
 * Creates an AUListDialog from the specified list data.
 *
 * @param list A list of PopMenuItem objects.
 * @param context The context object.
 */
public AUListDialog(ArrayList<PopMenuItem> list, Context context)

/**
 * Creates an AUListDialog from the specified list data.
 *
 * @param title The title.
 * @param list A list of PopMenuItem objects. Icons can be set.
 * @param context The context object.
 */
public AUListDialog(String title, ArrayList<PopMenuItem> list, Context context)

/**
 * Creates an AUListDialog from the specified list data.
 *
 * @param title The title.
 * @param message The message body.
 * @param list A list of PopMenuItem objects. Icons can be set.
 * @param context The context object.
 */
public AUListDialog(String title, String message, ArrayList<PopMenuItem> list, Context
context)

/**
 * Creates an AUListDialog from the specified list data.
 *
 * @param title The title.
 * @param list A list of PopMenuItem objects.
 * @param showSelectionState Specifies whether to display an icon for the selection sta
te of an option.
 * @param positiveString The text of the OK button.
 * @param positiveListener The listener for the OK button.
 * @param negativeString The text of the Cancel button.
 * @param negativeListener The listener for the Cancel button.
 * @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)

/**
 * Creates an AUListDialog from the specified list data.
 *
 * @param title The title.
 * @param message The message body.
 * @param list A list of PopMenuItem objects.
 * @param showSelectionState Specifies whether to display an icon for the selection sta
te of an option.
 * @param positiveString The text of the OK button.
 * @param positiveListener The listener for the OK button.
 * @param negativeString The text of the Cancel button.
 * @param negativeListener The listener for the Cancel button.
 * @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)

/**
 * Sets a listener for item click events in the list.
 * Sets a listener for item click events in the list.
 */
public void setOnItemClickListener(OnItemClickListener listener) {
    this.listener = listener;
}


/**
 * Updates the data in the list.
 *
 * @param list
 */
public void updateData(ArrayList<PopMenuItem> list)
```

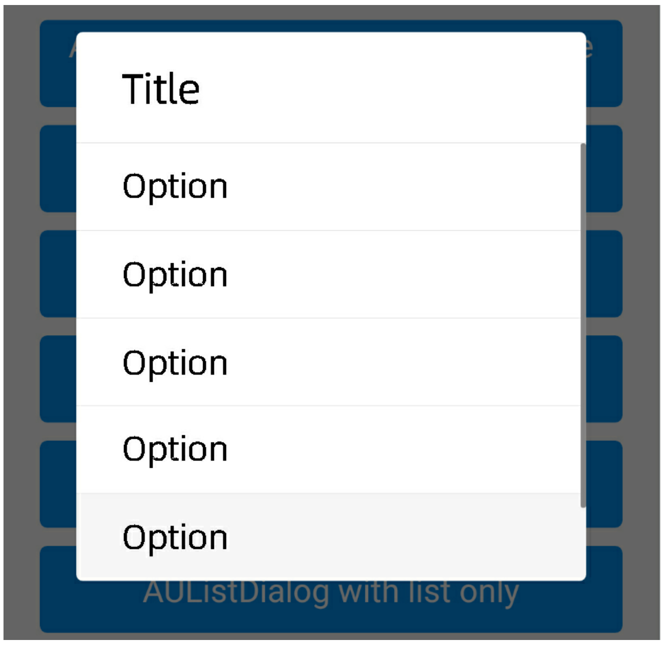
Code examples

- List-only dialog box



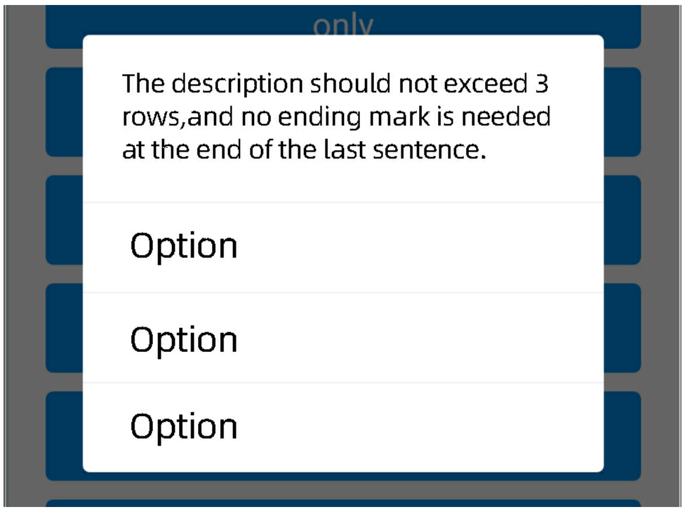
```
new AUListDialog(this, getData(7)).show();

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

- List dialog box with a title



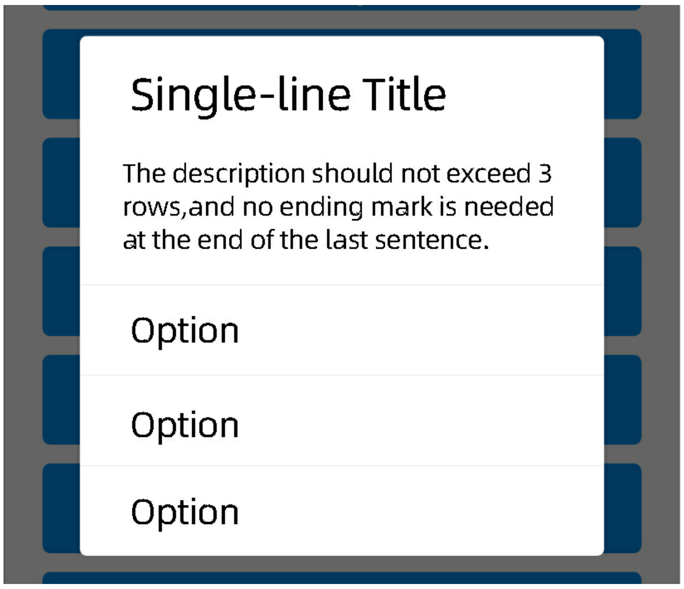
```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text",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

- List dialog box with descriptive text


only
The description should not exceed 3 rows,and no ending mark is needed at the end of the last sentence.
Option
Option
Option

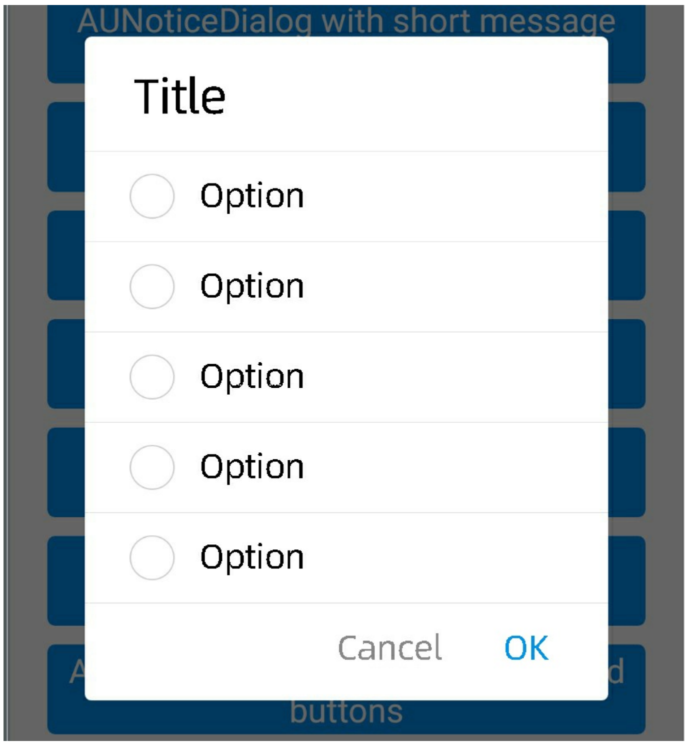
```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
new AUListDialog("", "Keep the descriptive text within three lines. Avoid ending a
line with a punctuation mark.", items, this).show();
```

- List dialog box with a title and descriptive text


Single-line Title
The description should not exceed 3 rows,and no ending mark is needed at the end of the last sentence.
Option
Option
Option

```
ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
items.add(new PopMenuItem("Option text", null));
new AUListDialog("Single-line title", "Keep the descriptive text within three lines
. Avoid ending a line with a punctuation mark.", items, this).show();
```

- List dialog box with checkable items

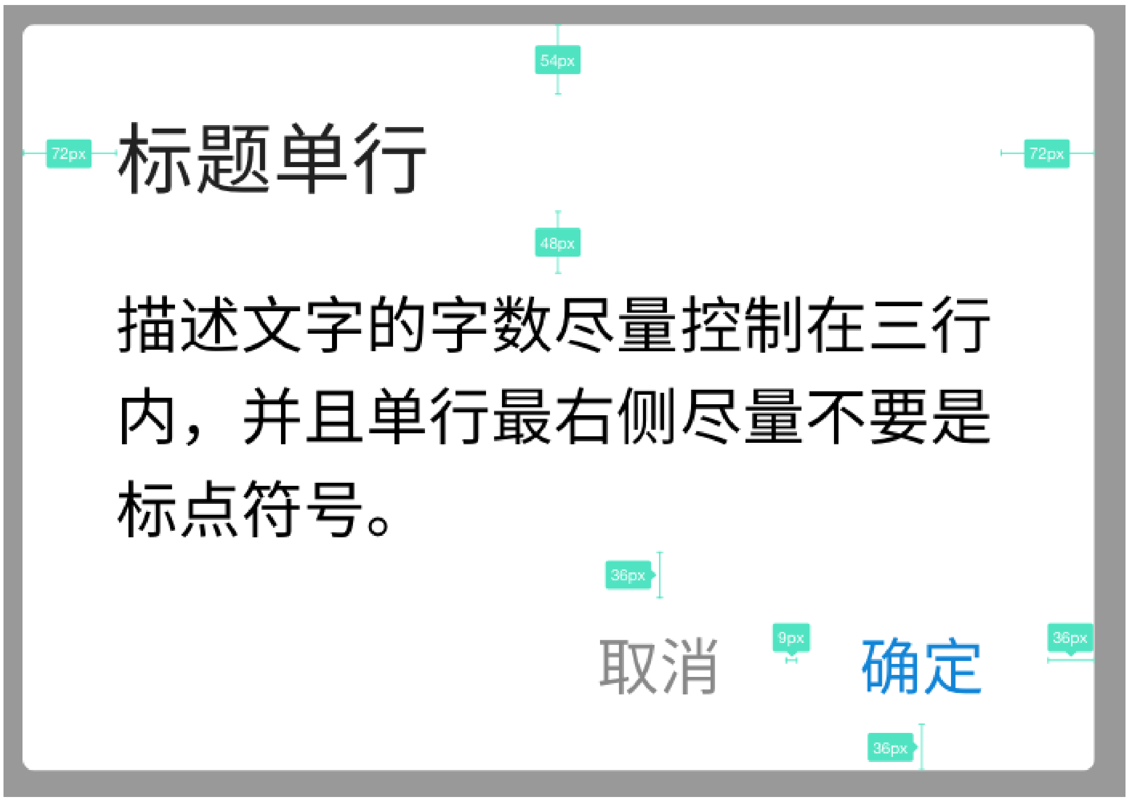


```
    ArrayList<PopMenuItem> items = new ArrayList<PopMenuItem>();
    PopMenuItem item = new PopMenuItem("Option text", null);
    item.setType(AUCheckIcon.STATE_UNCHECKED);
    items.add(item);
    items.add(new PopMenuItem("Option text", null));
    items.add(new PopMenuItem("Option text", null));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

1.2.2.8. Message dialog box

AUNoticeDialog (formerly APNoticePopDialog) provides a dialog box that contains a title, body text, and Confirm and Cancel buttons. This dialog box is used to display common business messages.

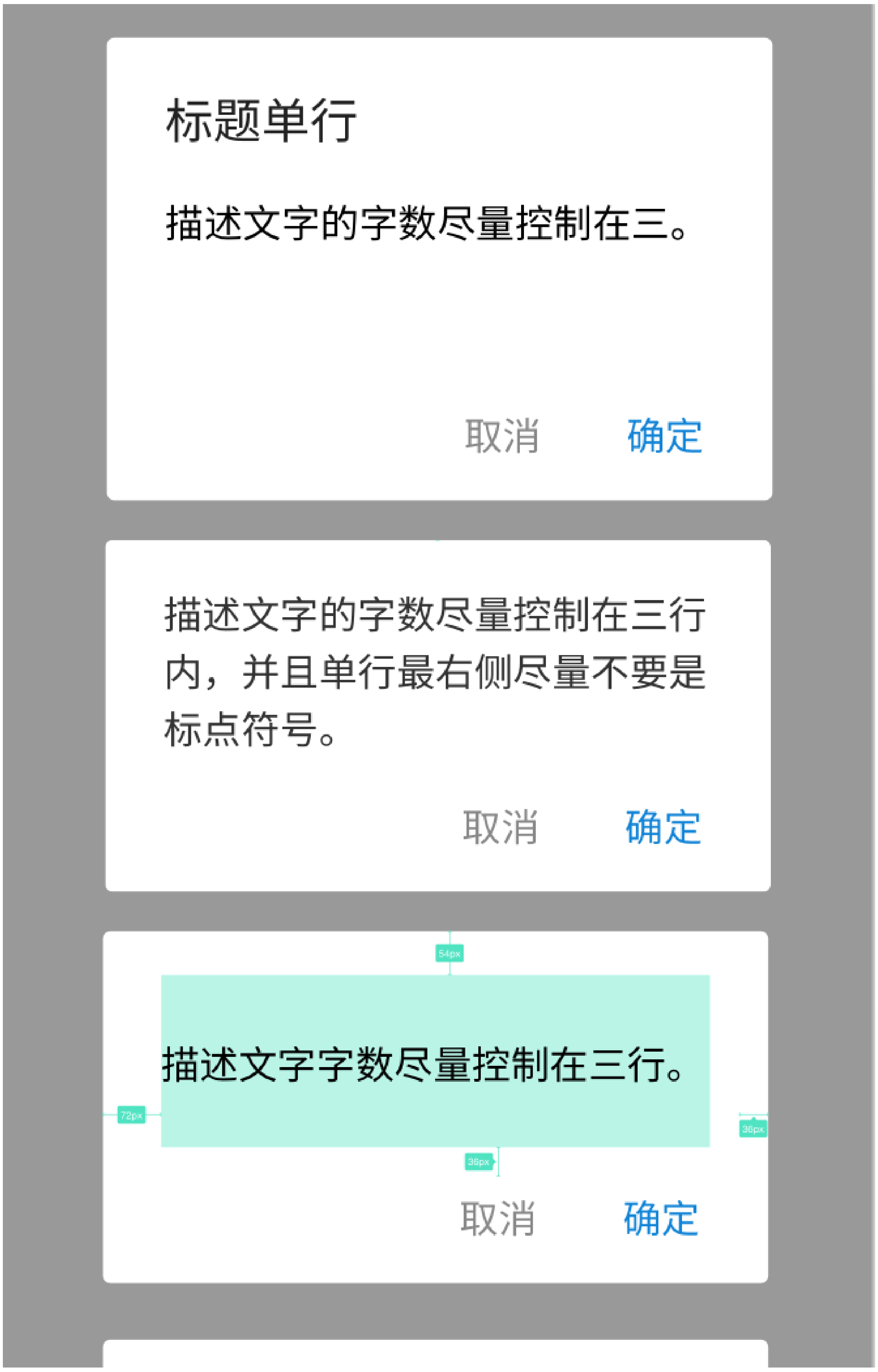
Preview


54px
72px
标题单行
72px
48px
描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。
36px
取消
9px
确定
36px
36px

```
AUNoticeDialog dialog = new AUNoticeDialog(this, "Single-line title",
    "Keep the description within three lines. Avoid ending a line with a punctuation ma
rk.",
    "Confirm", "Cancel", true);
dialog.show();
```

Basic rules

- The dialog box has a minimum height.
- If the dialog box contains only a title or only body text, the content is vertically centered.

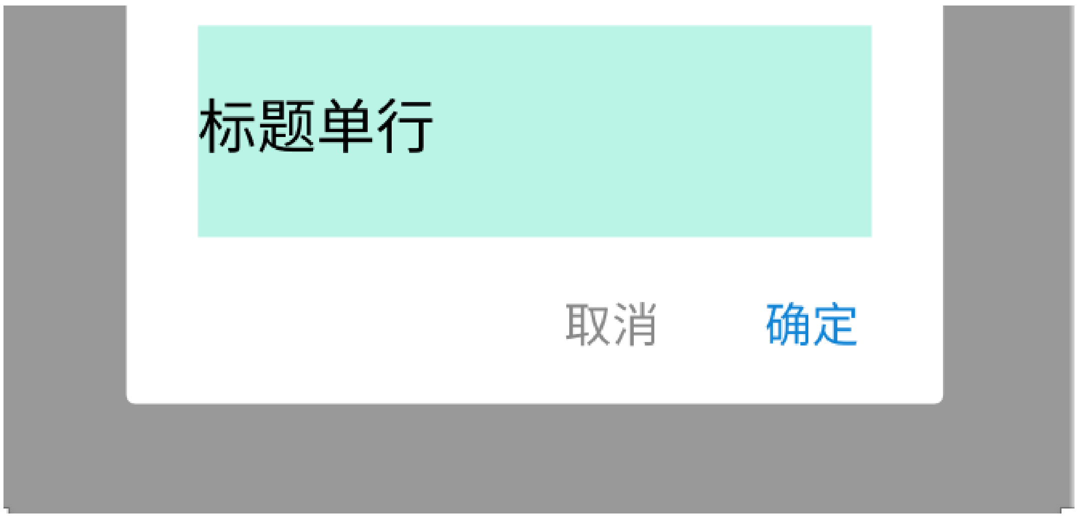


标题单行

描述文字的字数尽量控制在三。

描述文字的字数尽量控制在三行内，并且单行最右侧尽量不要是标点符号。

描述文字字数尽量控制在三行。



- Keep the text for the **Confirm** and **Cancel** buttons short. Long text may not display correctly on small-screen phones, such as the VIVO Y23L.

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) ;

/**
 * Creates an AUNoticeDialog based on the input parameters.
 *
 * @param context The context object.
 * @param title The title.
 * @param msg The message.
 * @param positiveString The text for the confirm button.
 * @param negativeString The text for the cancel button.
 * @param isAutoCancel Specifies whether to automatically close the dialog box when a u
ser taps outside of it.
 */
public AUNoticeDialog(Context context, CharSequence title, CharSequence msg, String pos
itiveString, String negativeString, boolean isAutoCancel);

/**
 * Sets the text color of the confirm button.
 *
 * @param c The color value.
 */
public void setPositiveTextColor(ColorStateList c);

/**
 * Sets the text color of the cancel button.
 *
 * @param c The color value.
 */
public void setNegativeTextColor(ColorStateList c);
```

```
/**
 * Gets the cancel button.
 */
public Button getCancelBtn();

/**
 * Gets the confirm button.
 */
public Button getEnsureBtn();

/**
 * Gets the title TextView.
 */
public TextView getTitle();

/**
 * Gets the message TextView.
 */
public TextView getMsg();

/**
 * Sets the click listener for the confirm button.
 *
 * @param listener The listener.
 */
public void setPositiveListener(OnClickPositiveListener listener);

/**
 * Sets the click listener for the cancel button.
 *
 * @param listener The listener.
 */
public void setNegativeListener(OnClickNegativeListener listener);

/**
 * Gets the root RelativeLayout of the dialog box layout.
 */
public RelativeLayout getDialogBg();

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

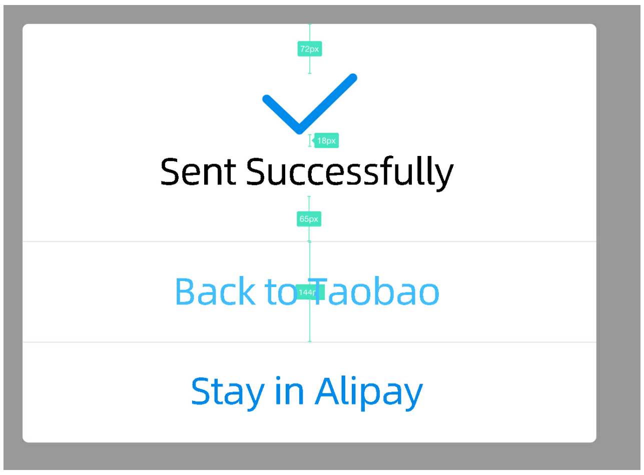
Code examples

```
// Without a title
AUNoticeDialog dialog = new AUNoticeDialog(this, "",
            "Keep the description within three lines. Avoid ending a line with a
punctuation mark.",
            "Confirm", "Cancel", true);
dialog.show();

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

1.2.2.9. 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



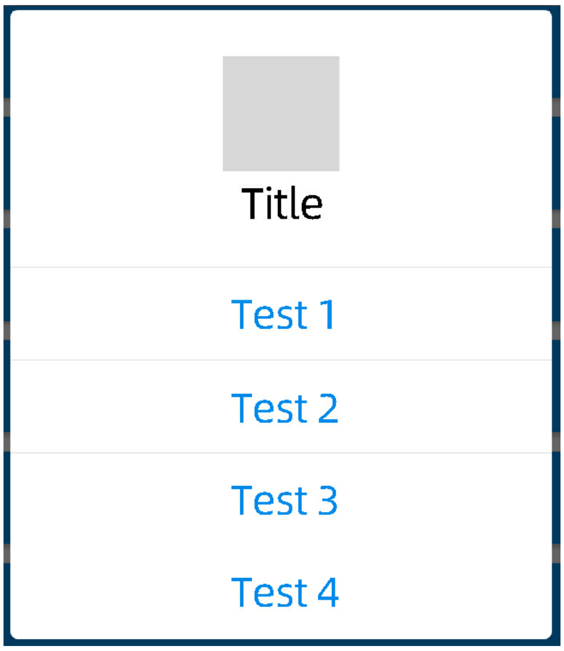
Dependency

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();
}
```

1.2.2.10. Pop-up menu

The AUPopMenu component lets you create a pop-up menu that appears when a tab in the navigation bar is clicked. It is a type of popup window.

Unlike AUFloatMenu, AUPopMenu does not have a bottom mask layer and has an outer border. All layouts within AUPopMenu are centered.

Basic features

- The business control expands or collapses.
- Supports passing a list of strings to use the default style or passing an adapter directly.

API reference

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

/**
* Constructs the adapter using a 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)

/**
* Dismisses the window.
*/
public void dismiss()


/**
* Sets a listener for item clicks.
* @param listener
*/
public void setOnItemClickListener(AdapterView.OnItemClickListener listener)
```

Custom properties

No custom properties. XML layouts are not supported.

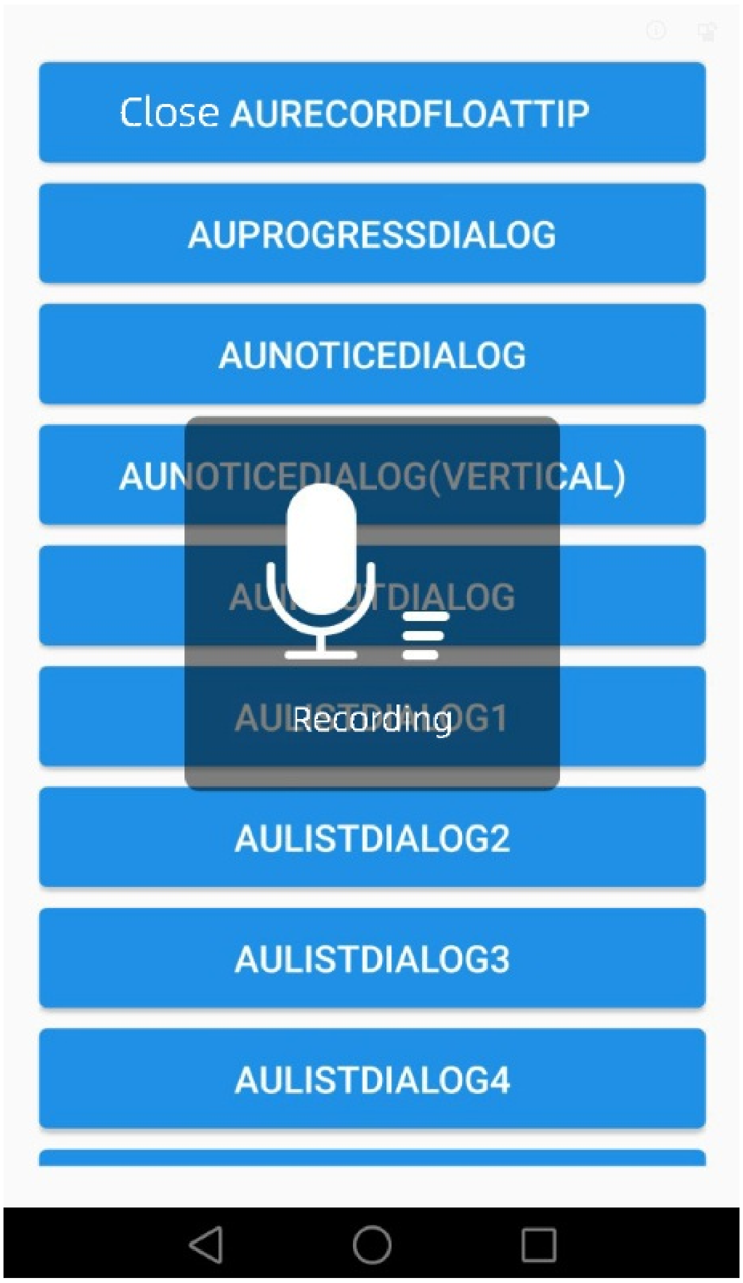
Code sample

```
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) {

    }
});
```

1.2.2.11. Recording

The AURecordFloatTip component displays a floating layer that shows the **Recording** status. This provides a more intuitive recording experience for users.

Preview



Constructors

```
    public AURecordFloatTip(Activity activity) ;

    public AURecordFloatTip(Activity activity, String tip);
```

API reference

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

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

/**
 * Gets the text view of the floating layer.
 *
 * @return
 */
public AUTextView getTipTextView();

/**
 * Gets the icon view of the floating layer.
 *
 * @return
 */
public AUImageView getIconView();
```

Code example

```
if (!isSHowAURecordFloatTip) {
    ((AUButton) view).setText("Hide AURecordFloatTip");
    if (mAURecordFloatTip == null) {
        mAURecordFloatTip = new AURecordFloatTip(this, "Recording");
    }
    mAURecordFloatTip.show();
    isSHowAURecordFloatTip = true;
} else {
    ((AUButton) view).setText("Show AURecordFloatTip");
    if (mAURecordFloatTip != null) {
        mAURecordFloatTip.dismiss();
    }
    isSHowAURecordFloatTip = false;
}
```

1.2.2.12. Toast notification

AUToast is a component for displaying toast notifications on a page. AUProgressDialog is a component for displaying progress indicators.

When you call the toast method from BaseFragmentActivity or BaseActivity, the mPaaS framework automatically customizes the toast notifications.

Activities that use BaseFragmentActivity and BaseActivity use AUToast by default.

Dependencies

For more information, see Getting started.

API reference

```
    /**
     * Instantiates a Toast.
     *
     * @param context    The context. Use the activity of the current page.
     * @param drawableId The image resource.
     * @param tipSrcId   The ID of the text message.
     * @param duration   The display duration. Toast.LONG or Toast.SHORT.
     * @return A Toast object.
     */
    public static Toast makeToast(Context context, int drawableId, int tipSrcId, int du
ration) {
        CharSequence tipSrc = context.getResources().getText(tipSrcId);
        return makeToast(context, drawableId, tipSrc, duration);
    }

    /**
     * Creates a Toast.
     *
     * @param context  The context. Use the activity of the current page.
     * @param tipSrcId The message to show.
     * @param duration The duration.
     * @return A toast object.
     */
    public static Toast makeToast(Context context, int tipSrcId, int duration) {
        CharSequence tipSrc = context.getResources().getText(tipSrcId);
        return makeToast(context, 0, tipSrc, duration);
    }

    /**
     * Makes a toast that contains an image view and a text view.
     *
     * @param context    The context. Use the activity of the current page.
     * @param drawableId The image resource ID.
     * @param tipSrc     The text to show. Can be formatted text.
     * @param duration   How long to display the message. Either Toast.LONG or Toast.SH
ORT.
     * @return A Toast object.
     */
    public static Toast makeToast(Context context, int drawableId, CharSequence tipSrc,
int duration)
```

Code example

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


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

    // Warning
        AUToast.makeToast(ToastActivity.this,
com.alipay.mobile.antui.R.drawable.toast_warn, "Warning notification",
Toast.LENGTH_SHORT).show();
    }

    // Text
        AUToast.showToastWithSuper(ToastActivity.this, 0, "The text cannot exceed 14 ch
aracters.", Toast.LENGTH_SHORT);

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

}
```

1.2.3. Input components

1.2.3.1. Amount input

The AUAmountInputBox component provides an input box for amounts and uses a special numeric font. This component includes an edit box (AUAmountEditText) and a note section (AUAmountFootView). The AUAmountFootView is available in two styles: an editable input box and a text display. You can combine these parts as needed.

The component also provides AUAmountLabelText to display text with the special numeric font.

API reference

AUAmountInputBox

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

/**
 * Gets the edit layout.
 * @return
 */
public AUAmountEditText getEditLayout()

/**
 * Gets the footView of the amount input box.
 * @return
 */
public AUAmountFootView getFootView()

/**
 * Gets the title bar of the input box.
 * @return
 */
public AUTextView getTitleView()

/**
 * Sets the style of the FootView.
 * @param style The style. Valid values: EDIT_STYLE and TEXT_STYLE.
 */
public void setFootStyle(int style)

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


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

AUAmountEditText

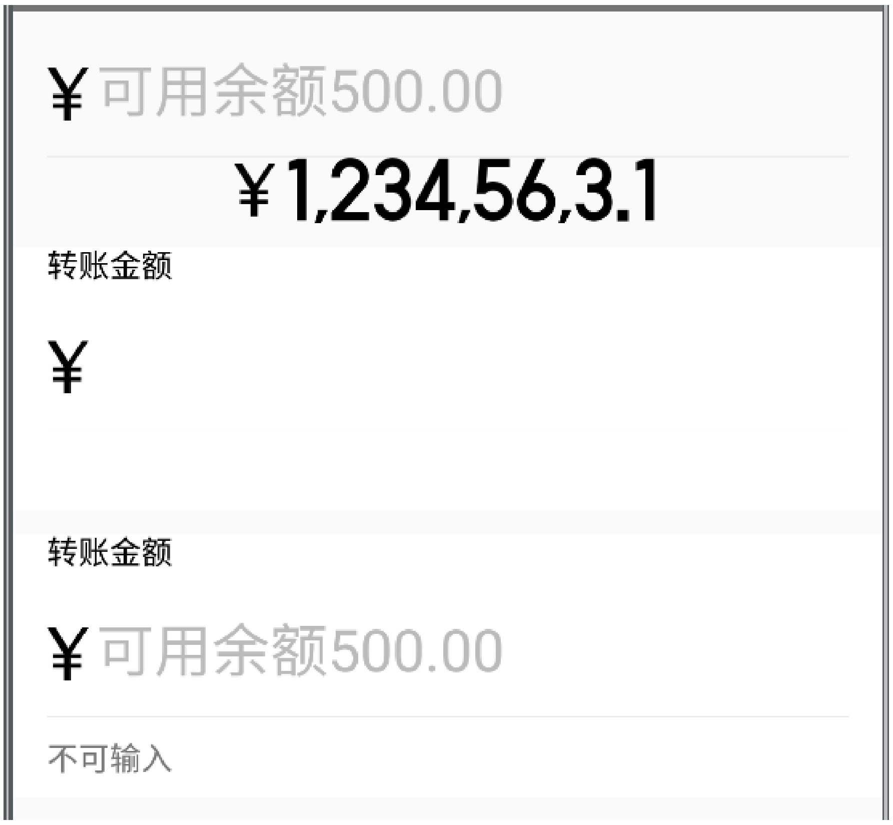
```
    /**
     * Gets the EditText.
     * @return
     */
    public AUEditText getEditText()


    /**
     * Gets the editable text from the input box.
     * @return
     */
    public Editable getEditTextEditable()

    /**
     * Shows or hides the line separator.
     * @param visible
     */
    public void setDividerVisible(boolean visible)

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

    /**
     * Specifies whether to show the delete button.
     * @param isShow
     */
    public void isShowClearIcon(boolean isShow)

    /**
     * Adds a focus change listener.
     * @param listener
     */
    public void addOnFocusChangeListeners(OnFocusChangeListener listener)

    /**
     * Attaches an external AUNumberKeyboardView and ScrollView.
     * @param keyboardView
     * @param scrollView
     */
    public void setKeyBoardView(AUNumberKeyboardView keyboardView, ScrollView
scrollView)

    /**
     * Attaches an external AUNumberKeyboardView.
     * @param keyboardView
     */
    public void setKeyBoardView(AUNumberKeyboardView keyboardView)
```

Custom attributes

Property	Description	Type
footStyle	The header view type.	editStyle, textStyle
amountTitleText	The title of the edit box.	string, reference
amountHintText	The hint for the edit box.	string, reference

Code examples

General code example



```
<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 transfer description");

AUAmountInputBox inputBox2 = (AUAmountInputBox)findViewById(R.id.amount_input_2);
inputBox2.setFootText("Not editable");
```

Code example with 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);

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

1.2.3.2. Input box

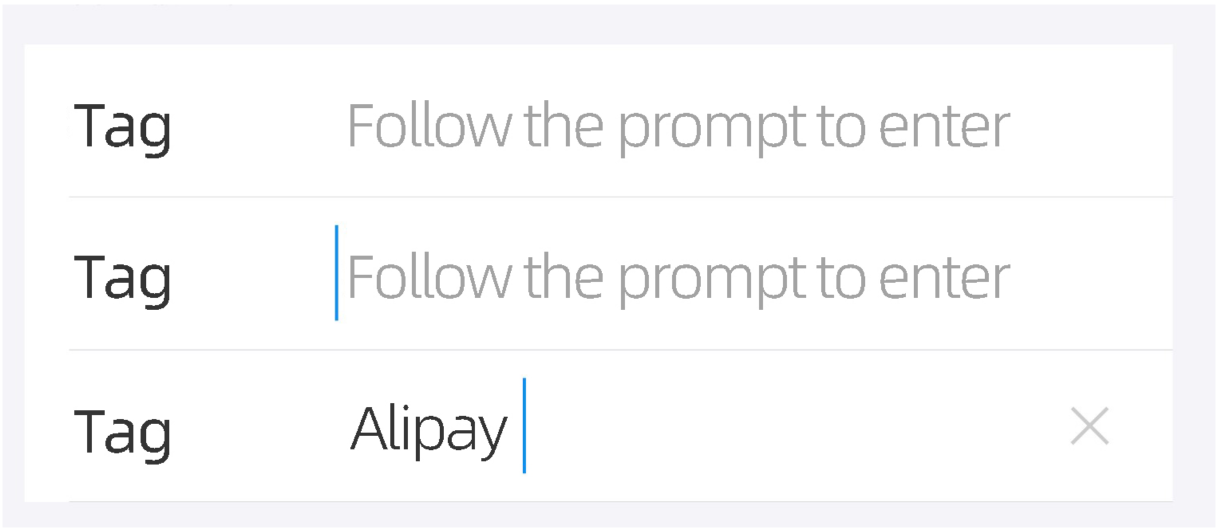
This topic describes the three input box components that mPaaS provides: AUInputBox, AUImageInputBox, and AUTextCodeInputBox. AUImageInputBox and AUTextCodeInputBox inherit from AUInputBox.

AUInputBox

The AUInputBox component contains the following items:

- An AUEditText text input box
- A tag name that appears to the left of the input box
- A delete button that appears when the input box has focus and is not empty

Preview



API reference

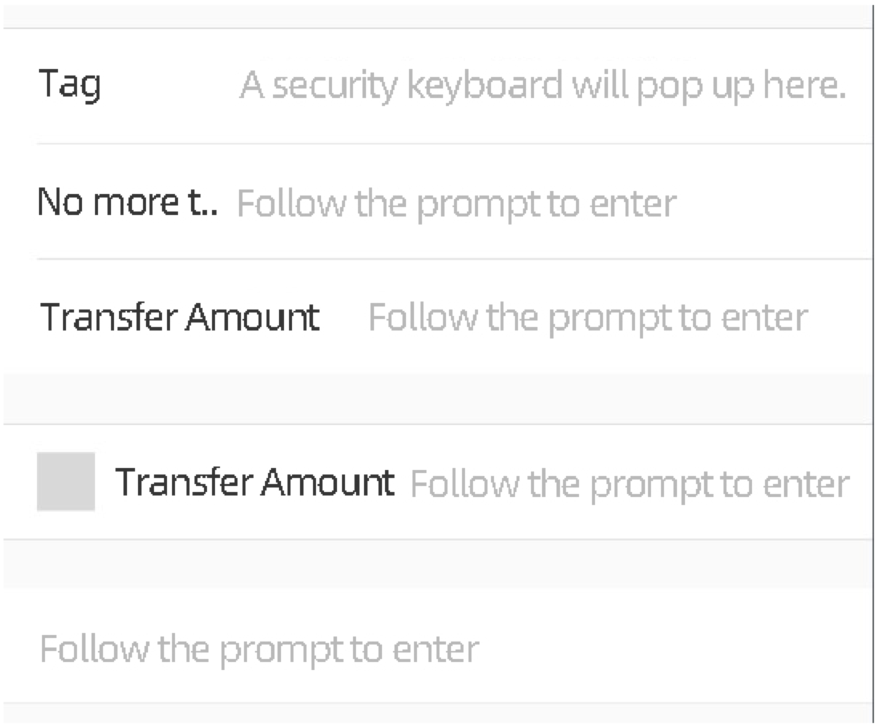
```
/**
 * Gets the UBB-encoded string.
 */
public String getUbbStr()

/***
 * Sets the emoji font size in px.
 */
public void setEmojiSize(int emojiSize)
/***
 * Sets whether to support emojis.
 */
public void setSupportEmoji(boolean isSupport) {
    this.supportEmoji = isSupport;
}

/**
 * Sets a Formatter to format the input.
 * This setting does not take effect on existing text until new text is entered.
 */
public void setTextFormatter(AUFormatter formatter)

/**
 * Sets whether to display the input text in bold.
 *
 * @param isBold Set to true for bold, or false for 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)


/**
 * After text is entered in the input box, a delete button appears. This method sets th
e listener object for the delete button's click event.
 */
public void setCleanButtonListener(View.OnClickListener listener)

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

/**
 * Gets the text content. If the text is formatted, the caller must process it into the
required format.
 */
public String getInputedText()

/**
 * Gets the EditText control of the input box.
 */
public AUEditText getInputEdit()

/**
 * Sets the tag text.
 * @param title The tag text to enter.
 */
public void setInputName(String title)


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

/**
 * Sets the font size of the input content name in px.
 */
public void setInputNameTextSize(float textSize)
```

```
/**
 * Sets the font size of the input box in px.
 */
public void setInputTextSize(float textSize)

/**
 * Sets the text color of the input content.
 */
public void setInputTextColor(int textColor)

/**
 * Sets the content type of the input.
 */
public void setInputType(int inputType)
/**
 * Sets the hint message.
 */
public void setHint(String hintString)

/**
 * Sets the icon for the left tag.
 */
public void setInputImage(Drawable drawable)

/**
 * Gets the icon for the left tag.
 */
public AUImageView getInputImage()

/**
 * Sets the color of the hint message.
 */
public void setHintTextColor(int textColor)

/**
 * Sets the maximum input length for the input box.
 *
 * @param maxlength If the parameter is <=0, no length limit is applied.
 */
public void setMaxLength(int maxlength)

/**
 * Gets the clear button control.
 */
public AUIconView getClearButton()

/**
 * Gets whether the clear button needs to be displayed.
 */
public boolean isNeedShowClearButton() {
    return isNeedShowClearButton;
}

/**
```

```
 * Sets whether to display the clear button. If set to false, the clear button is never displayed.
 */
public void setNeedShowClearButton(boolean isNeedShowClearButton)

/**
 * Sets the border style of the control, including top, center, bottom, and normal.
 * This method is from the AULineGroupItemInterface interface.
 * When the control is used with LineGroupView, this method is automatically called.
 *
 * @param positionStyle Use the variables defined in AULineGroupItemInterface: AULineGroupItemInterface.TOP, CENTER, BOTTOM, NORMAL, LINE, or NONE.
 */
@Override
public void setItemPositionStyle(int positionStyle)

/**
 * Gets 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

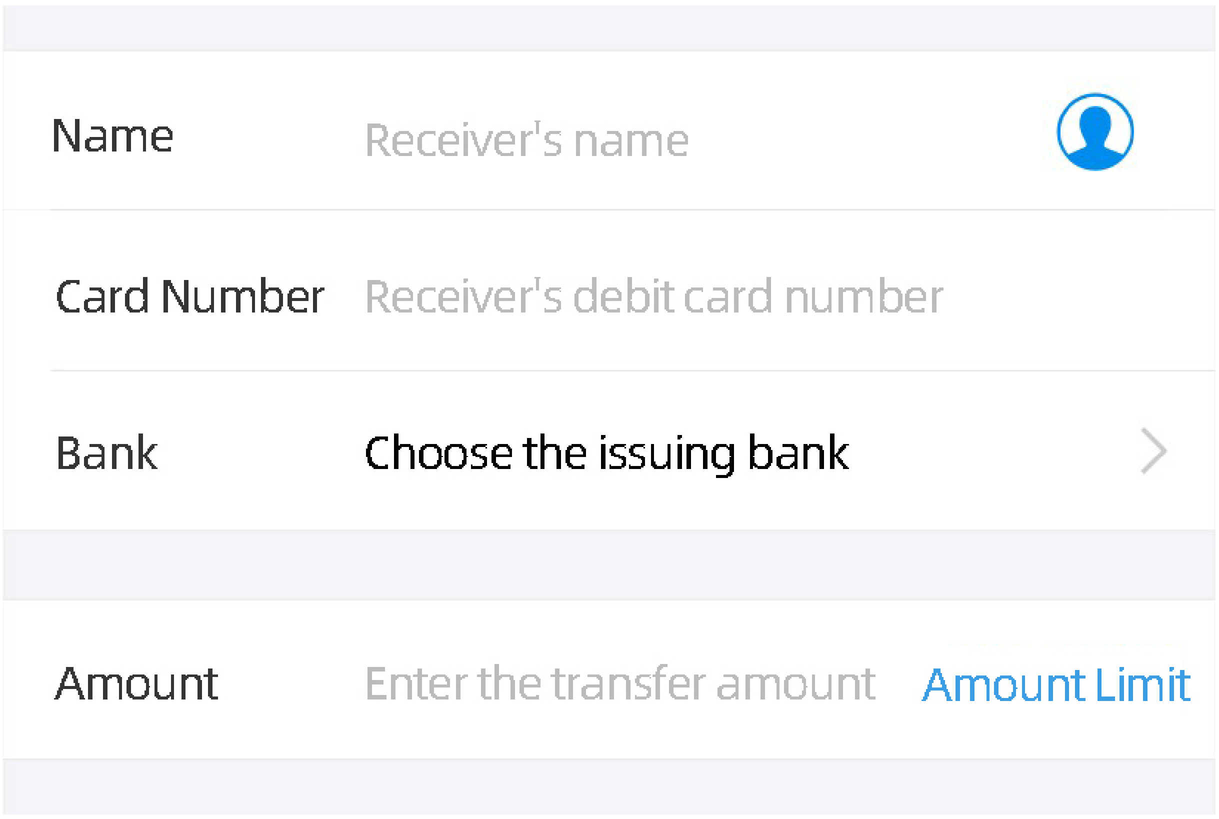
```
<com.alipay.mobile.antui.input.AUInputBox
    android:id="@+id/safeInputBox"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="top"
    app:inputName="tag 1"
    app:inputType="textPassword"
    app:inputHint="This input box opens the secure keyboard"/>

<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="Enter text as prompted"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="bottom"
    app:inputName="Transfer-in amount"
    app:inputHint="Enter text 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-in amount"
    app:inputHint="Enter text as prompted"/>

<com.alipay.mobile.antui.input.AUInputBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:inputHint="Enter text as prompted"/>
```

AUImageInputBox

AUImageInputBox inherits from AUInputBox and contains:

- An IconView on the right that can display an icon or a Unicode character
- A TextView on the right

Preview


Name | Receiver's name
Card Number | Receiver's debit card number
Bank | Choose the issuing bank
Amount | Enter the transfer amount | Amount Limit

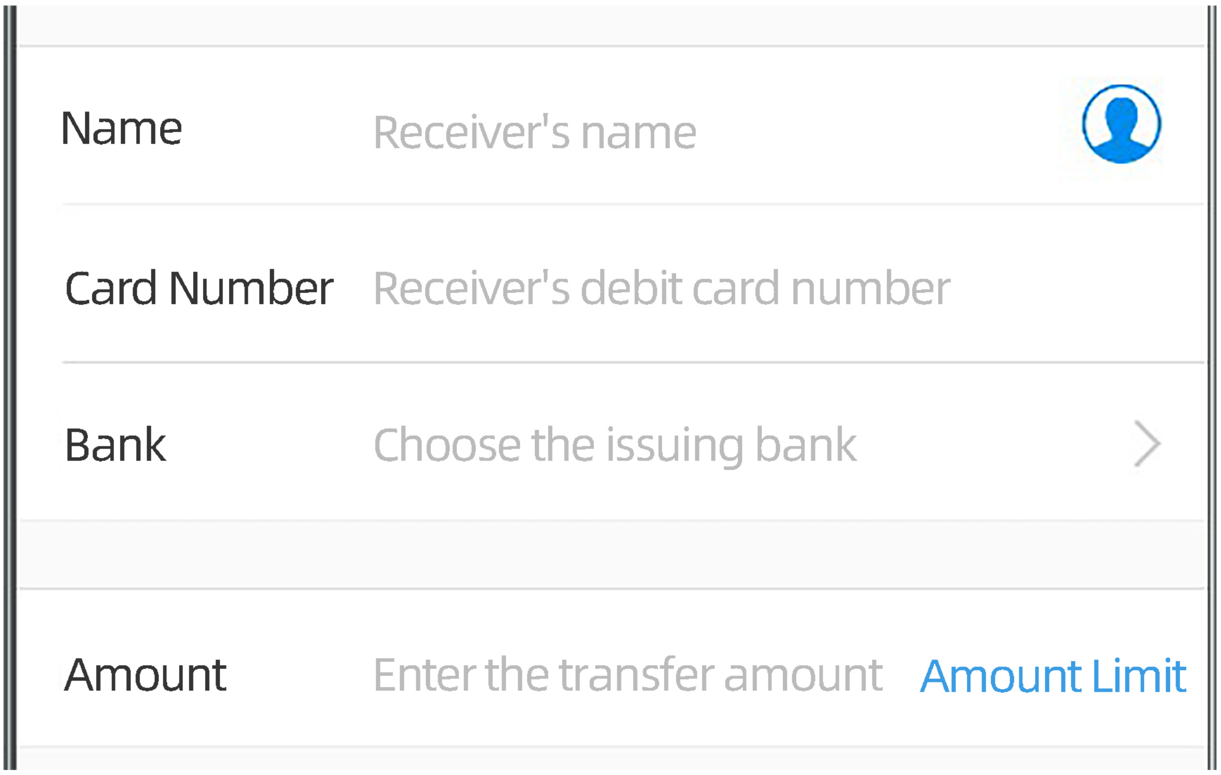
Dependencies

For more information, see Quick start.

API reference

```
/**
 * Sets the background of the rightmost function button.
 * If the button background is set to empty, the function button is not displayed, whic
h is consistent with the behavior of AUInputBox.
 */
public void setLastImgDrawable(Drawable drawable)

/**
 * Sets the rightmost function icon using a Unicode string.
 * @param unicode
 */
public void setLastImgUnicode(String unicode)

/**
 * Sets the visibility of the rightmost icon.
 * @param visible
 */
public void setLastImgBtnVisible(boolean visible)

/**
 * Sets the listener for the rightmost function button.
 */
public void setLastImgClickListener(View.OnClickListener l)

/**
 * Sets the text on the far right.
 * @param lastText
 */
public void setLastTextView(String lastText)

/**
 * Gets the rightmost TextView.
 *
 * @return The rightmost TextView.
 */
public AUTextView getLastTextView()

/**
 * Gets the rightmost icon View.
 * @return
 */
public AUIconView getLastImgBtn()
```

Code sample



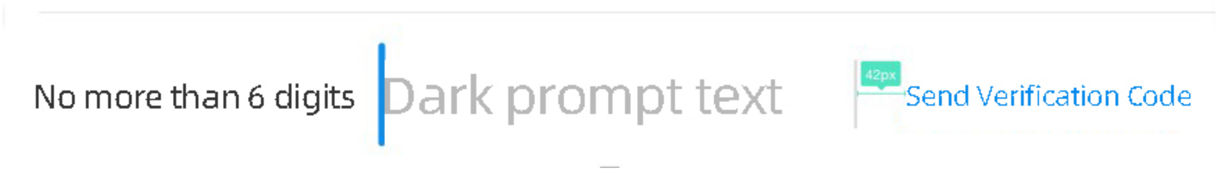
Name	Receiver's name	
Card Number	Receiver's debit card number	
Bank	Choose the issuing bank	
Amount	Enter the transfer amount	Amount Limit

```
<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="Payee debit card number" />

<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 a 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="Limit description" />
```

AUTextCodeInputBox

AUTextCodeInputBox inherits from AUInputBox and includes a text button on the right to send a text message verification code.

Preview



Dependencies

For more information, see Quick start.

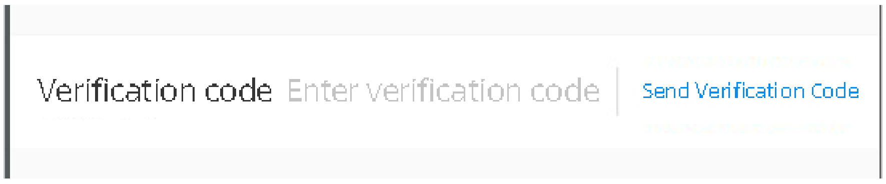
API reference

```
/**
 * Sets the callback for the send button's click event.
 * @param callback When a user clicks the send button, the OnSendCallback.onSend() meth
od is called.
```

```
 */
public void setOnSendCallback(OnSendCallback callback)

/**
 * Resets the current time to 0.
 */
public void currentSecond2Zero()

/**
 * Sets the current time.
 */
public void setCurrentSecond(int current)

/**
 * Gets the current time.
 */
public int getCurrentSecond()

/**
 * Gets the send button.
 */
public AUButton getSendCodeButton()

/**
 * Allows the service to call this method to release the timer.
 */
public void releaseTimer()


/**
 * Starts the button's countdown timer.
 */
public void scheduleTimer()

public interface SendButtonEnableChecker {
    public boolean checkIsEnabled();
}

/**
 * This method sets a checker to determine if the SendButton is enabled.
 * If this checker finds the button is disabled, the button is grayed out when updateSe
ndButtonEnableStatus is called.
 * Otherwise, the button's enabled state is set according to the countdown logic when u
pdateSendButtonEnableStatus is called.
 * In short, the button is enabled only if all checks pass. Otherwise, it is grayed out
.
 */
public void setSendButtonEnableChecker(SendButtonEnableChecker checker)

/**
 * Updates the enabled state of the SendButton based on SendButtonEnableChecker and the
internal state of CheckCodeSendBox.
 * The button is enabled only if all checks pass. Otherwise, it is grayed out.
 */
```

```
public void updateSendButtonEnableStatus()

/**
 * Gets the 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_textcode_input);
textCodeInputBox.setOnSendCallback(new OnSendCallback() {
    @Override
    public void onSend(final SendResultCallback callback) {
        // Request a verification code from the server via RPC.
        boolean resendSmsRpcSuccess = true;
        if (resendSmsRpcSuccess) {
            // Verification code sent successfully. Start the countdown.
            callback.onSuccess();
            // Verification code received...
            Toast.makeText(InputActivity.this, "Verification code received: 123456",
Toast.LENGTH_SHORT)
                    .show();
            textCodeInputBox.setText("123456");
            Log.d(TAG, "The entered verification code is: " +
textCodeInputBox.getInputedText());
        } else {
            // Failed to send the verification code. Re-enable the send button.
            callback.onFail();
        }
    }
});
```

Custom attributes

The following table lists the custom attribute parameters for the three components.

Attribute name	Description	Type

inputName	Enter the content name.	string, reference
inputHint	The hint text in the input box.	string, reference
maxLength	The maximum length of the input content.	integer
inputType	The type of the input content.	enum. Valid values: textNormal, textNumber, textDecimal, and textPassword.
inputImage	The image to the left of the input box.	reference
listItemType	The background type of the item.	enum. Valid values: top, center, bottom, normal, line, and none.
input_rightIconUnicode	The icon on the right.	string, reference
input_rightIconDrawable	The image on the right.	reference
input_rightText	The hyperlink text on the right.	string, reference

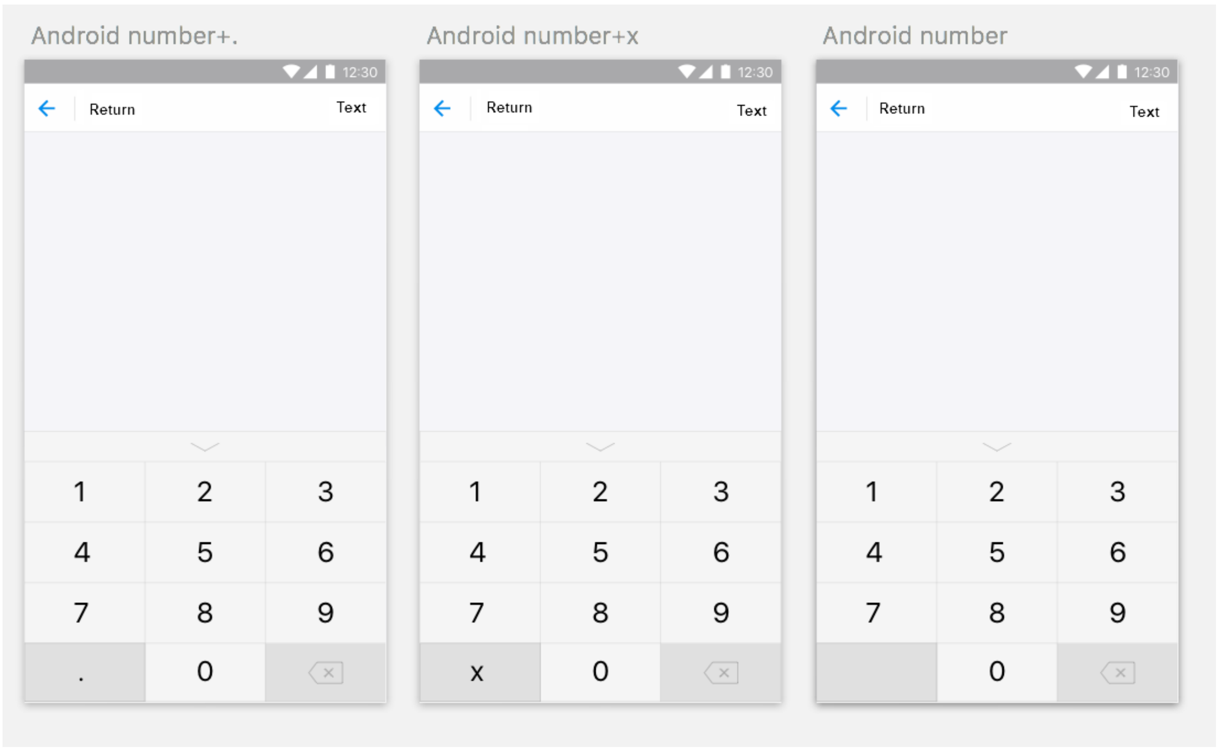
1.2.3.3. Numeric keyboard

AUNumberKeyboardView provides a numeric keyboard with three states.

Usage instructions

- You can display it as a separate view, such as in a miniapp.
- You can attach it to AUAmountEditText using AUNumberKeyBoardUtil. This utility is already encapsulated in AUAmountEditText. For more information, see the AUAmountInputBox document.
- You can attach it to a standard EditText using AUNumberKeyBoardUtil. You must invoke this utility.

Preview



API reference

AUAmountEditText

```
    /**
     * Sets the style of the keyboard. The default value is STYLE_POINT.
     * @param style STYLE_POINT, STYLE_X, or STYLE_NONE
     */
    public void setStyle(int style)

    /**
     * Sets the button listener.
     * @param listener
     */
    public void setActionClickListener(OnActionClickListener listener)

    /**
     * Sets the display state listener.
     * @param windowStateChangeListener
     */
    public void setWindowStateChangeListener(WindowStateChangeListener
windowStateChangeListener)

    /**
     * Shows the keyboard.
     */
    public void show()

    /**
     * Hides the keyboard.
     */
    public void hide()

    /**
     * Returns the display state.
     * @return
     */
    public boolean isShow()
```

AUNumberKeyBoardUtil

```
    /**
     * Passes the EditText and AUNumberKeyboardView.
     * @param context
     * @param editText
     * @param keyboardView
     */
    public AUNumberKeyBoardUtil(Context context, EditText editText,
AUNumberKeyboardView keyboardView)

    /**
     * Sets the scroll view.
     * @param view
     */
    public void setScrollView(ScrollView view)

    /**
     * Shows the numeric keypad.
     */
    public void showKeyboard()

    /**
     * Hides the numeric keypad.
     */
    public void hideKeyboard()
```

Code examples

AUAmountEditText

```
AUNumberKeyboardView  auNumberKeyboardView = new AUNumberKeyboardView(this, AUNumberKey
boardView.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>
```

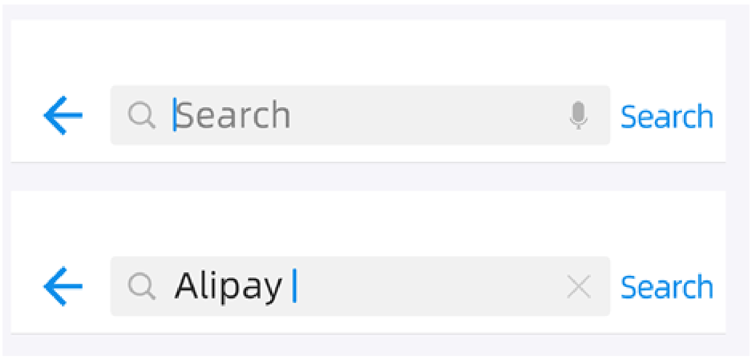
- **Java:**

```
keyBoardUtil = new AUNumberKeyBoardUtil(context, editText, keyboardView);
keyBoardUtil.setScrollView(scrollView);
```

1.2.3.4. Search bar

AUSearchBar, formerly APSocialSearchBar, provides a search title bar that includes a back button, a search box, and a search button on the right.

Preview



API reference

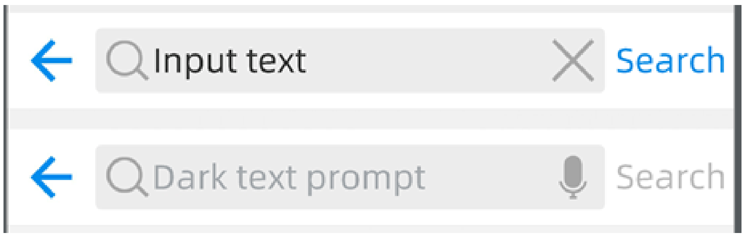
```
    /**
      * Sets the maximum input length.
      */
     public void setInputMaxLength(int length);

     /**
      * Gets the back button.
      * @return
      */
     public AUIconView getBackButton() ;

     /**
      * Gets the delete button.
      * @return
      */
     public AUIconView getClearButton();

     /**
      * Gets the search input box.
      * @return
      */
     public AUEditText getSearchEditView();

     /**
      * Gets the search button.
      * @return
      */
     public AUIconView getSearchButton() ;

     /**
      * Gets the search layout.
      * @return
      */
     public AURelativeLayout getSearchRelativeLayout() ;

     /**
      * Gets the voice search button.
      * @return
      */
     public AUIconView getVoiceButton();

          /**
      * Adds a listener for edit events.
      */
         public void setEditChangedListener(TextWatcher watcher)
```

Custom properties

Property	Description	Type

isShowSearchBtn	Specifies whether to show the search button.	boolean
isShowVoiceSearch	Specifies whether to show the voice search button.	boolean
searchEditText	The default text in the search box.	string, reference
searchEditHint	The default hint text in the search box.	string, reference
searchButtonText	The text of the search button.	string, reference
inputMaxLength	The maximum length of the input in the search box.	integer, reference
hintIconUnicode	The Unicode of the icon on the left of the edit box.	string, reference
hintIconDrawable	The resource for the icon on the left of the edit box.	reference
backIconUnicode	The Unicode of the back button.	string, reference
backIconDrawable	The resource for the back button.	reference
editHintColor	The color of the hint text 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 example



```
<com.alipay.mobile.antui.basic.AUSearchBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        aui:searchEditText="Input text"
        aui:isShowSearchBtn="true"
        aui:isShowVoiceSearch="true"/>

    <com.alipay.mobile.antui.basic.AUSearchBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        aui:searchEditHint="Hint text"
        aui:isShowSearchBtn="true"
        aui:isShowVoiceSearch="true"/>
```

1.2.3.5. Search input box

The AUSearchInputBox component (formerly APSocialTagSearchBar) is a search title bar that includes a search box and a search button on the right. When you use this component, you must set the height of the View.

API reference

```
    /**
      * Sets the maximum input length.
      */
     public void setInputMaxLength();

     /**
      * Gets the delete button.
      * @return
      */
     public AUIconView getClearButton();

     /**
      * Gets the search input box.
      * @return
      */
     public AUEditText getSearchEditView();

     /**
      * Gets the voice search button.
      * @return
      */
     public AUIconView getVoiceButton();
```

Custom properties

Property	Description	Type

isShowSearchBtn	Specifies whether to display the search button.	boolean
isShowVoiceSearch	Specifies whether to display the voice search button.	boolean
searchEditText	The default text in the search box.	string, reference
searchEditHint	The default hint text in the search box.	string, reference
inputMaxLength	The maximum input length for the search box.	integer, reference
hintIconUnicode	The Unicode for the icon on the left of the edit box.	string, reference
hintIconDrawable	The resource for the icon on the left of the edit box.	reference
editHintColor	The color of the hint text 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 examples

XML example:

```
<com.alipay.mobile.antui.basic.AUSearchInputBox
        android:layout_width="match_parent"
        android:layout_height="52dp"
        android:layout_marginTop="10dp"
        app:searchEditHint="Hint text" />
```

```
AUSearchInputBox inputBox = new AUSearchInputBox(this);
ViewGroup.LayoutParams layoutParams = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,300);
inputBox.setLayoutParams(layoutParams);

layout.addView(inputBox);
```

1.2.4. Item component

1.2.4.1. Assist label widget

AUAssistLabelView is a TextView component that displays assistive text.

Code examples

```
<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:text="If disabled, notifications for friend messages will not show the sender a
nd content summary." />
```

```
<com.alipay.mobile.antui.basic.AUAssistLabelView
android:layout_width="match_parent"
android:layout_height="wrap_content"
app:isHead="true"
android:text="Table header" />
```

1.2.4.2. Bank card item component

The AUBankCardItem component is a bank card item that displays a bank name, a bank logo, and a bank account number.

API reference

```
/**
 * Gets the bank name view.
 * @return
 */
public AUEmptyGoneTextView getBankName();

/**
 * Gets the bank account number view.
 * @return
 */
public AUEmptyGoneTextView getBankNumber();

/**
 * Gets the bank logo view.
 * @return
 */
public AUCircleImageView getBankImage();

/**
 * Sets the bank name and account number.
 * @param bankName
 * @param bankNum
 */
public void setBankInfo(String bankName, String bankNum) ;
```

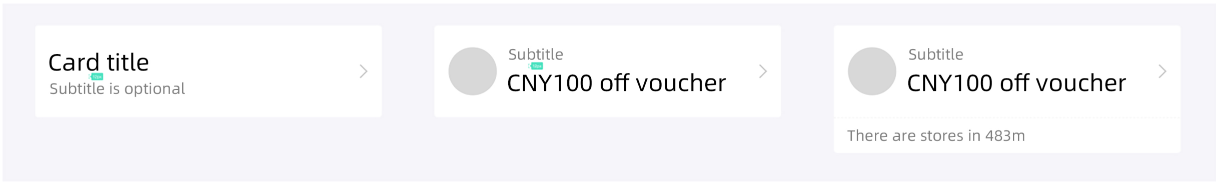
Code example

```
AUBankCardItem cardItem = new AUBankCardItem(this);
cardItem.setBankInfo("Bank Name","Last 4 digits");
cardItem.getBankImage().setImageResource(R.drawable.image);
```

1.2.4.3. Coupon item component

The coupon item component displays a coupon with an icon, a title, and a description.

Preview



Code example

```
AUCouponsItem couponsItem1 = new AUCouponsItem(this);
couponsItem1.setCouponsInfo("Subtitle","100 CNY voucher","");
couponsItem1.getCouponsImage().setImageResource(R.drawable.image);

AUCouponsItem couponsItem2 = new AUCouponsItem(this);
couponsItem2.setCouponsInfo("","100 CNY voucher","Subtitle (optional)");

AUCouponsItem couponsItem3 = new AUCouponsItem(this);
couponsItem3.setCouponsInfo("Subtitle","100 CNY voucher","");
couponsItem3.setCouponsAssitDes("Outlet at 483 m");
couponsItem3.getCouponsImage().setImageResource(R.drawable.image);
```

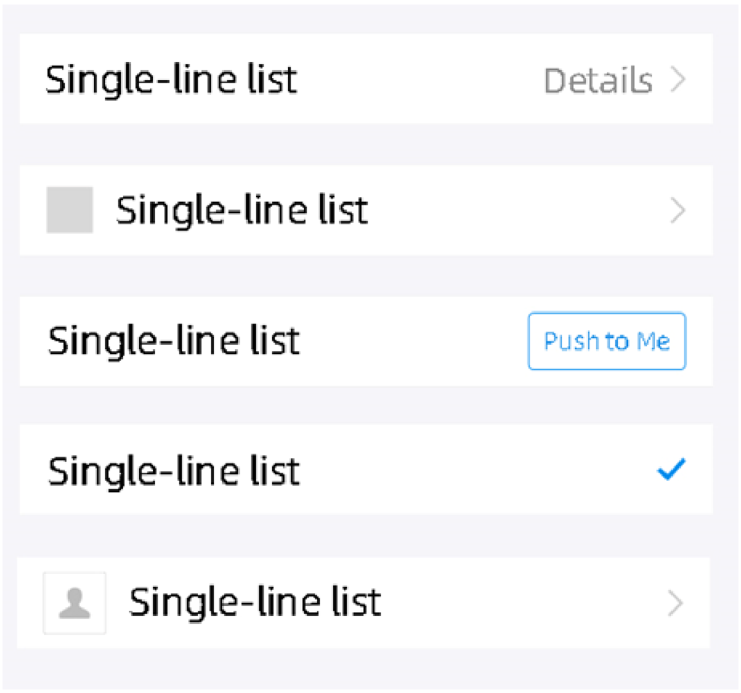
1.2.4.4. List item component

AUListItem is a list item component, including the following controls:

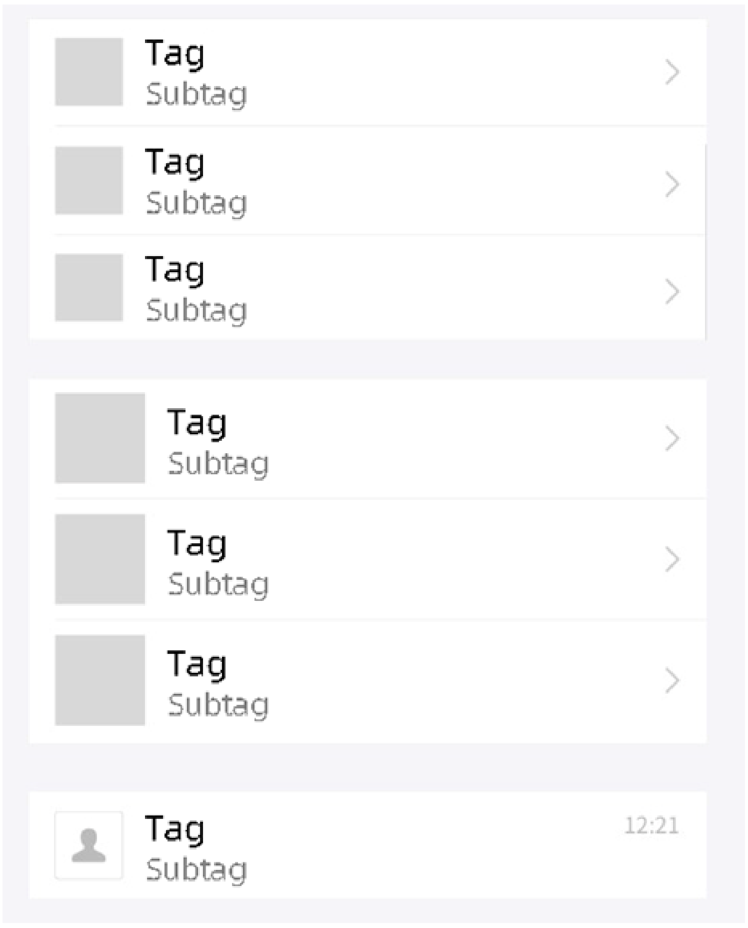
- AUSingleTitleListItem
- AUDoubleTitleListItem
- AUCheckBoxListItem
- AUSwitchListItem
- AUMultiListItem
- AUParallelTitleListItem
- AULineBreakListItem

Sample images

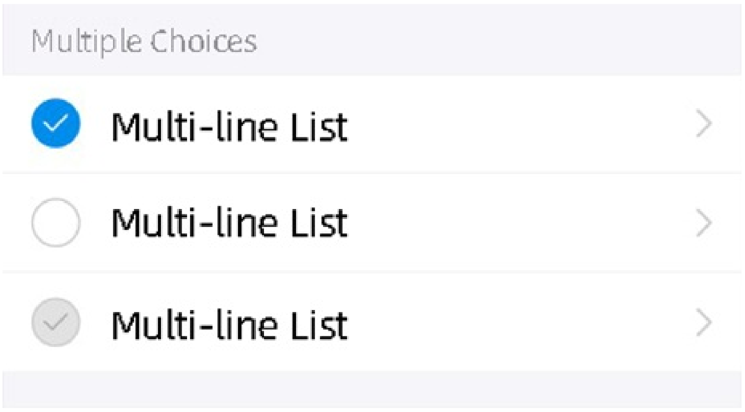
AUSingleTitleListItem



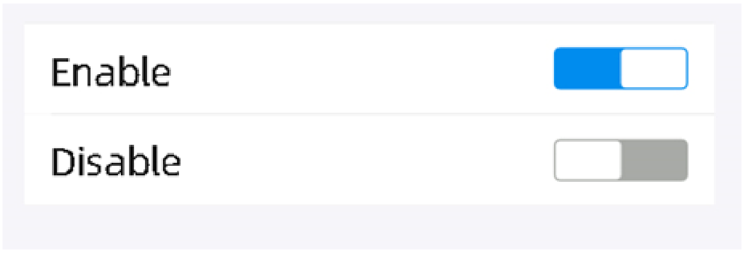
AUDoubleTitleListItem



AUCheckBoxListItem



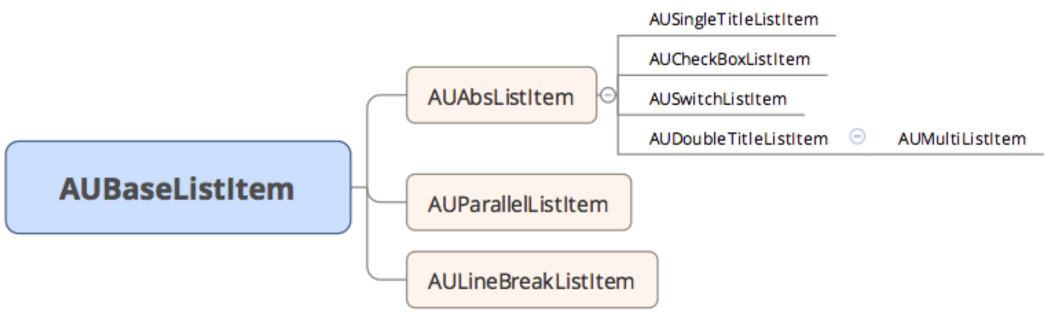
AUSwitchListItem



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, Stri
ng 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 visibi
lity.
 *
 * @param vis View.GONE
 */
public void setLeftImageVisibility(int vis)

/**
 * Get text information on the left.
 * @return
 */
public AUTextView 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 attributes

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

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	Direction of arrow.	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

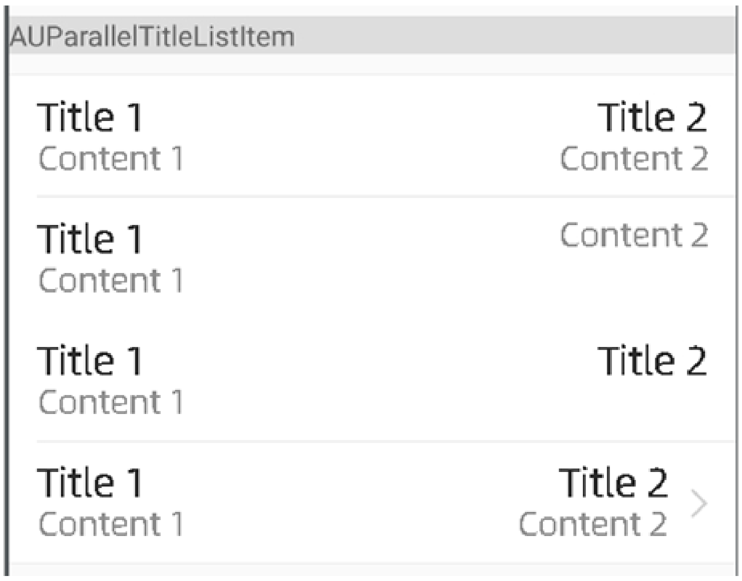
- The following code sample (XML) shows the attributes supported in each control.
- If you want to change the color of the background by clicking it, please add the attribute: `android:clickable="true"` .
- The height of the control, the width and height of the left image are custom based on business requirements.

Sample code

Introduce XML namespace. When the SDK is accessed in **Native AAR mode**, define `xmlns:app="http://schemas.android.com/apk/res-auto"` to make all the elements with the same `app` prefix are associated with the same namespace. **The following takes** `app` **as an example**.

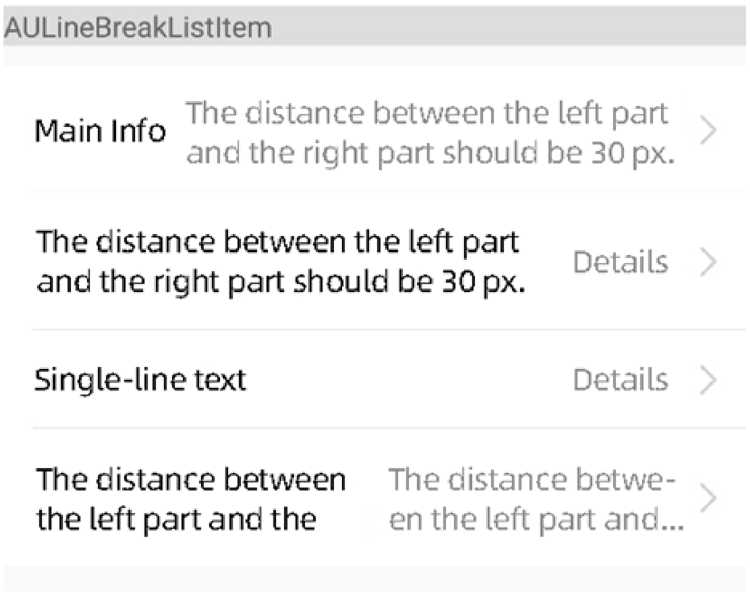
Introduce XML namespace. When the SDK is accessed in **component mode**, define `xmlns:aui="http://schemas.android.com/apk/res/com.alipay.mobile.antui"` to make all the elements with the same `aui` prefix are associated with the same namespace.

AUParallelListItem



```
<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.

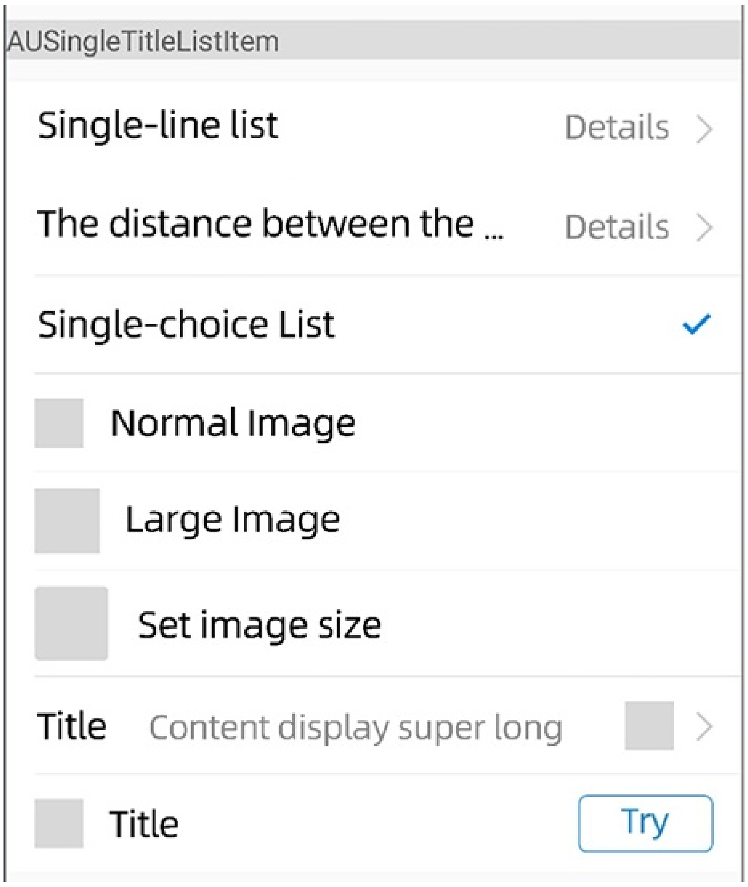
The distance between the left part and the right part should be 30 px. Details

Single-line text Details

The distance between the left part and the The distance betwe-en the left part and...

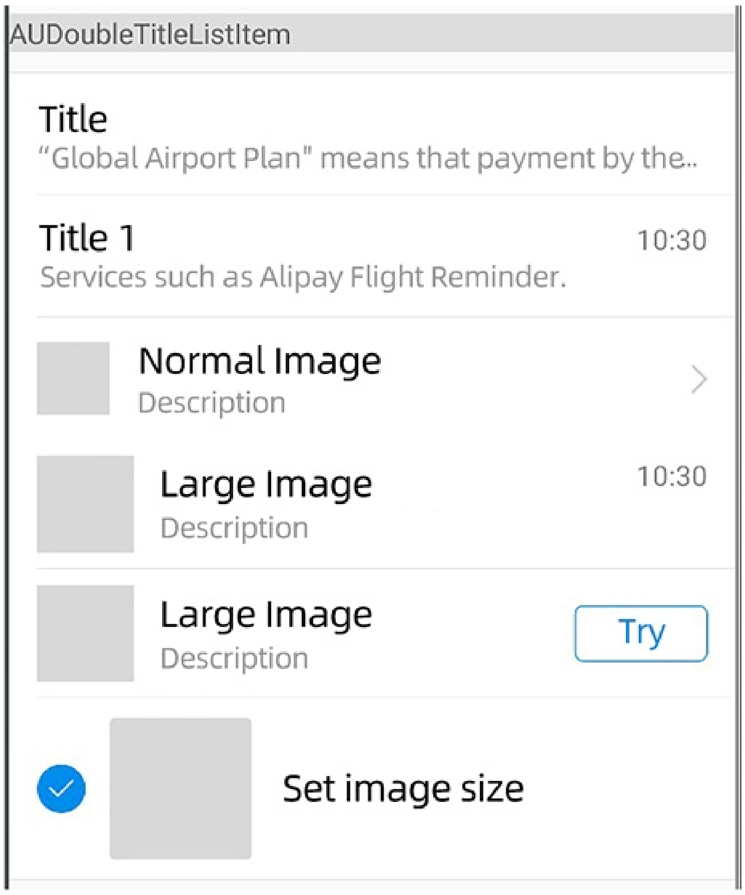
```
<com.alipay.mobile.antui.tablelist.AULineBreakListItem
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:listItemType="top"
    app:listLeftText="Main info"
    app:listRightText="The distance between the left part and the right part should be
30 px."/>

<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 3
0 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 3
0 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 3
0 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


AUDoubleTitleListItem

Title
“Global Airport Plan" means that payment by the..

Title 1
10:30
Services such as Alipay Flight Reminder.

Normal Image
Description

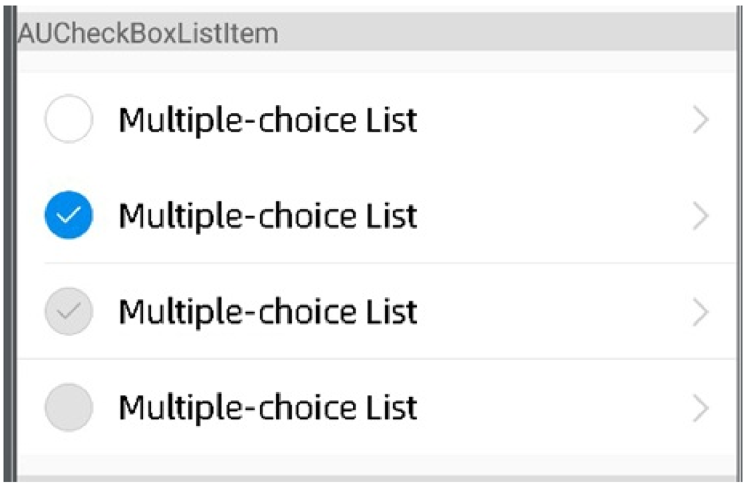
Large Image
10:30
Description

Large Image
Description
Try

Set image size

```
<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 overse
as 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

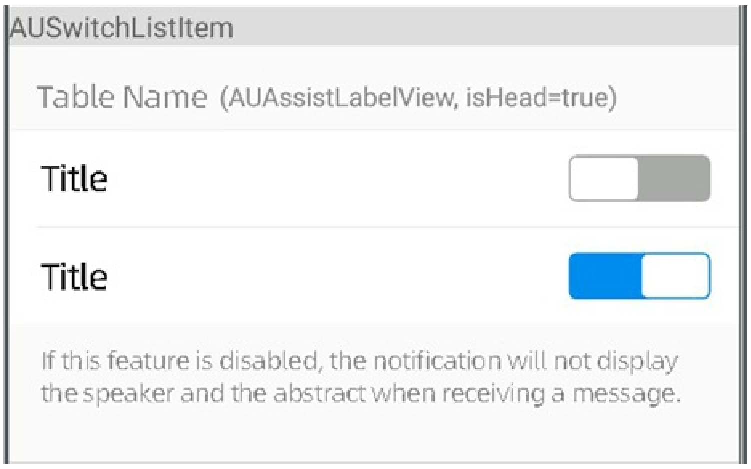

AUSwitchListItem

Table Name (AUAssistLabelView, isHead=true)

Title

Title

If this feature is disabled, the notification will not display the speaker and the abstract when receiving a message.

```
<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" />
```

1.2.5. Result page components

1.2.5.1. Progress page

AUFlowResultView displays a result page with a progress indicator. Each `FlowResult` represents a node. You can set a different type and supplementary text for each node.

API reference

```
    /**
     * Clears all FlowStepView instances.
     */
    public void clearFlows() {
        removeAllViews();
    }

    /**
     * Sets a list of FlowResult objects and generates the corresponding FlowStepView i
nstances.
     *
     * @param flowResultList
     */
    public void setFlows(List<FlowResult> flowResultList) {
```

FlowResult interface

```
    /**
     * Constructs a FlowResult object.
     *
     * @param resultStatus The node status. The value is one of the
ResultConstant.RESULT_STATUS_ENUM_XX constants.
     * @param statusIcon The status icon. The type is the ResultStatusIcon enum.
     * @param mainInfoText The main text.
     * @param subTitles A list of subtitles.
     */
    public FlowResult(int resultStatus, ResultStatusIcon statusIcon, String
mainInfoText,
            List<String> subTitles);

    /**
     * Constructs a FlowResult object.
     *
     * @param resultStatus The node status. The value is one of the
ResultConstant.RESULT_STATUS_ENUM_XX constants.
     * @param statusIconId The resource ID of the status icon.
     * @param mainInfoText The main text.
     * @param subTitles A list of subtitles.
     */
    public FlowResult(int resultStatus, int statusIconId, String mainInfoText,
            List<String> subTitles);
```

Code example

```
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 successful", Arrays.asList("Supplementary description", "Supplementary
description")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_OK,
ResultStatusIcon.PENDING,
        "Label text", Arrays.asList("Supplementary description", "Supplementary
description")));
flows.add(new FlowResult(ResultConstant.RESULT_STATUS_ENUM_NORMAL,
ResultStatusIcon.PENDING,
        "Label text", Arrays.asList("Supplementary description", "Supplementary
description")));
flowResultView.setFlows(flows);
```

1.2.5.2. Net error page

AUNetErrorView, formerly known as APFlowTipView, displays a blank page when a network error occurs.

Dependencies

For more information, see Quick start.

API reference

```
/**
     * Sets the simple mode.
     * @param isSimple
     */
    public void setIsSimpleType(boolean isSimple);

    /**
     * Sets the network error pattern.
     * @param type
     */
    public void resetFlowTipType(int type);

    /**
     * Sets the button properties.
     *
     * @param text
     * @param clickListener
     */
    public void setAction(String text, OnClickListener clickListener) ;

    /**
     * Removes the action button.
     */
    public void setNoAction();

    /**
     * Sets the tip message.
     *
     * @param text
     */
    public void setTips(String text) ;

    /**
     * Sets the secondary tip message.
     * @param text
     */
    public void setSubTips(String text) ;

    /**
     * Gets the action button.
     * @return
     */
    public AUButton getActionButton();

    /**
     * Gets the image view.
     * @return
     */
    public AUImageView getImageView() ;
```

Custom attributes

Attribute	Description
netErrorType	The state of the network error. Valid values: `signalError` , `empty` , `warning` , and `overflow` .
isSimpleMode	Specifies whether to use the simple mode. This is a boolean property.

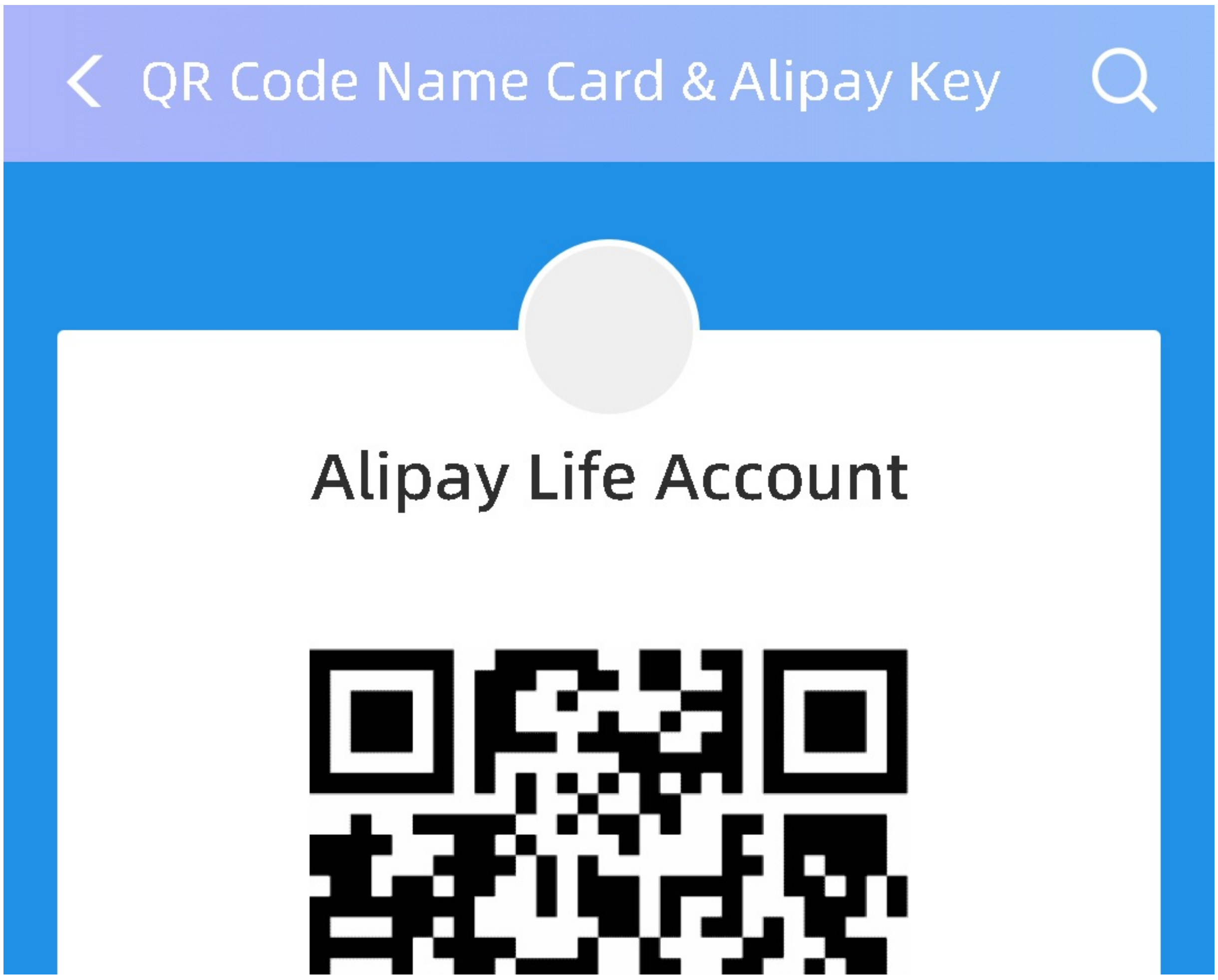
Code example

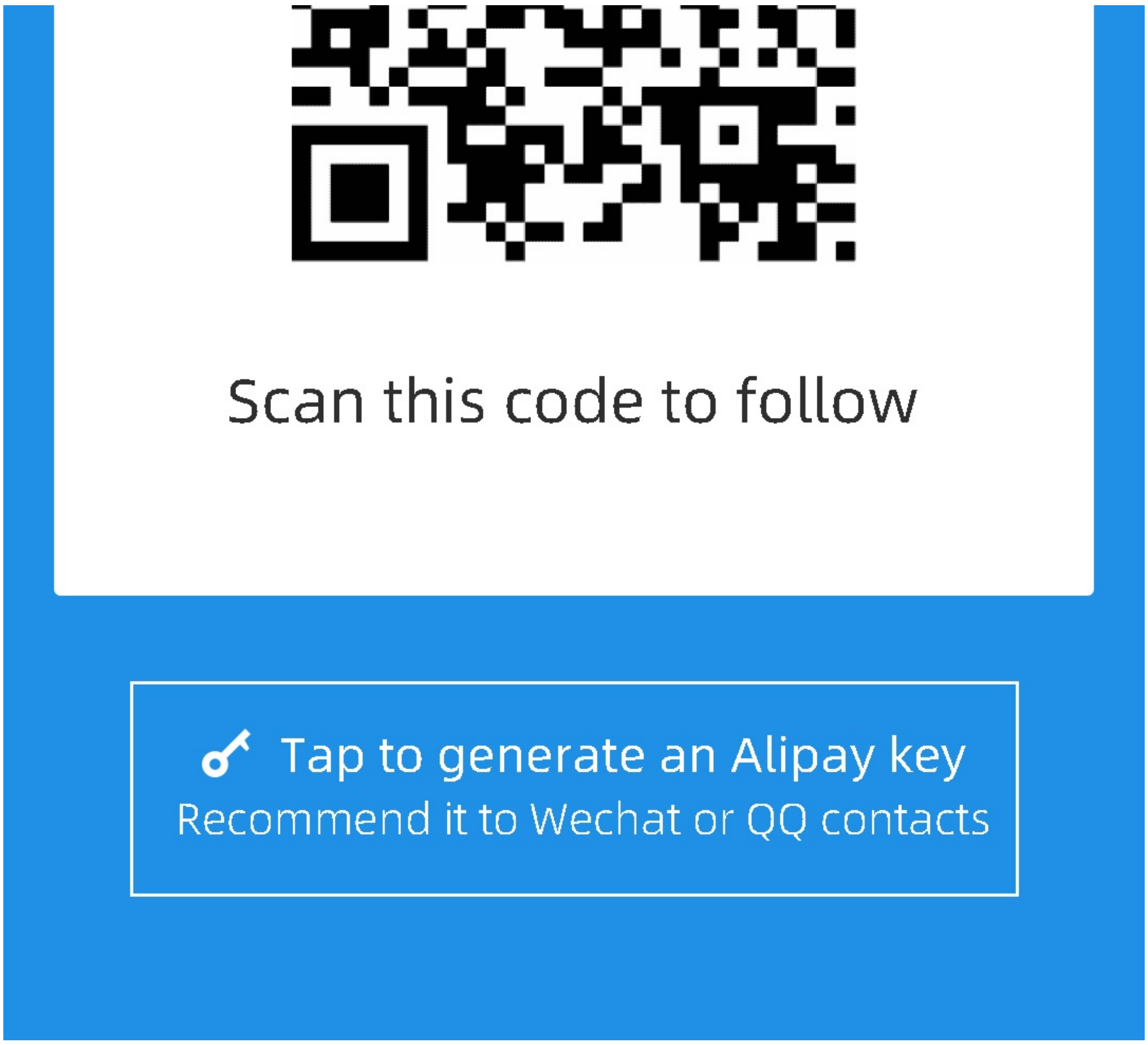
```
<com.alipay.mobile.antui.basic.AUNetErrorView
        android:id="@+id/net_error"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:netErrorType="signalError"/>
```

1.2.5.3. QR code page

You can use AUQRCodeView to display a QR code page.

Preview




Scan this code to follow
Tap to generate an Alipay key
Recommend it to Wechat or QQ contacts

API reference

```
/**
 * Sets the profile picture name.
 * @param name
 */
public void setAvatarName(CharSequence name)

/**
 * Sets the QR code information.
 * @param title
 * @param description
 */
public void setCodeInfo(CharSequence title, CharSequence description)

/**
 * Sets the QR code title.
 * @param title
 */
public void setCodeTitle(CharSequence title)
```

```
/**
 * Sets the QR code description.
 * @param description
 */
public void setCodeDescription(CharSequence description)

/**
 * Sets the button information with a security token icon.
 * @param title
 * @param content
 */
public void setButtonInfo(CharSequence title, CharSequence content)

/**
 * Sets the button information.
 * @param title
 * @param content
 * @param isToken Specifies whether to display the security token icon.
 */
public void setButtonInfo(CharSequence title, CharSequence content, boolean isToken)

/**
 * Sets the button title.
 * @param title
 */
public void setButtonTitle(CharSequence title)

/**
 * Sets whether the button has an icon.
 * @param isToken
 */
public void setButtonToken(boolean isToken)

/**
 * Sets the button visibility.
 * @param isVisible
 */
public void setButtonVisibility(boolean isVisible)

/**
 * Sets the button content.
 * @param content
 */
public void setButtonContent(CharSequence content)

/**
 * Gets the profile picture ImageView.
 * @return
 */
public AUImageView getAvatarImage()

/**
 * Gets the profile picture name View.
 * @return
 */
public AUTextView getAvatarName()

/**
 * Gets the QR code ImageView.
 * @return
 */
public AUImageView getCodeImage()

/**
 * Gets the QR code title.
 * @return
 */
public AUTextView getCodeTitle()

/**
 * Gets the QR code description.
 * @return
 */
public AUEmptyGoneTextView getCodeDescription()

/**
 * Gets the button.
 * @return
 */
public AULinearLayout getButton()

/**
 * Gets the button title.
 * @return
 */
public AUTextView getButtonTitle()

/**
 * Gets the button content.
 * @return
 */
public AUEmptyGoneTextView getButtonContent()
```

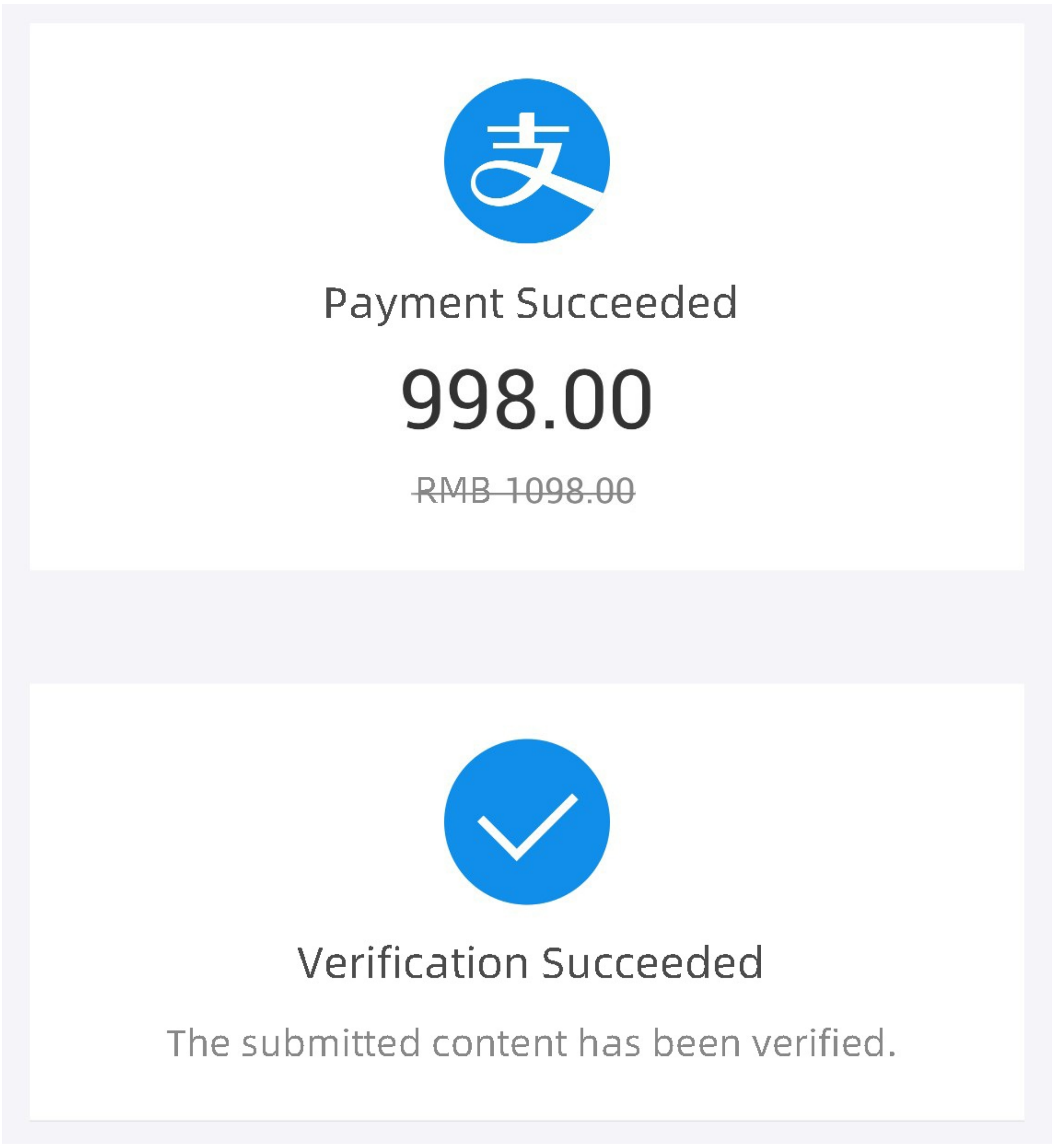
Code example

```
AUQRCodeView codeView = new AUQRCodeView(this);
codeView.setAvartarName("Lifestyle Account Name");
codeView.setCodeInfo("Scan the QR code with Alipay to join this Lifestyle Circle","This
QR code will expire on November 05, 2017");
codeView.setButtonInfo("Tap to generate a security token","Recommend the Lifestyle Acco
unt to your WeChat and QQ friends");
codeView.getCodeImage().setImageResource(R.drawable.qr_default);
```

1.2.5.4. Result page

AUResultView provides a result page with an icon and three levels of text.

Preview


Payment Succeeded

998.00

~~RMB 1098.00~~

Verification Succeeded

The submitted content has been verified.

API reference

```
    /**
     * Sets the icon.
     *
     * @param iconRes The resource ID of the icon.
     */
    public void setIcon(@DrawableRes int iconRes);

    /**
     * Sets the main title text.
     *
     * @param text The text content.
     */
    public void setMainTitleText(CharSequence text);

    /**
     * Sets the subtitle text.
     *
     * @param text The text content.
     */
    public void setSubTitleText(CharSequence text);

    /**
     * Sets the tertiary title text.
     *
     * @param text The text content.
     */
    public void setThirdTitleText(CharSequence text);

    /**
     * Sets the tertiary title text with a strikethrough effect.
     *
     * @param text The text content.
     * @param strikeThrough Specifies whether to display the text with a strikethrough.
     */
    public void setThirdTitleText(CharSequence text, boolean strikeThrough);
```

Custom properties

Property	Description	Type
icon	Icon	reference
mainTitleText	Main title text	string, reference
subTitleText	Subtitle text	string, reference
thirdTitleText	Tertiary title text	string, reference

Code example

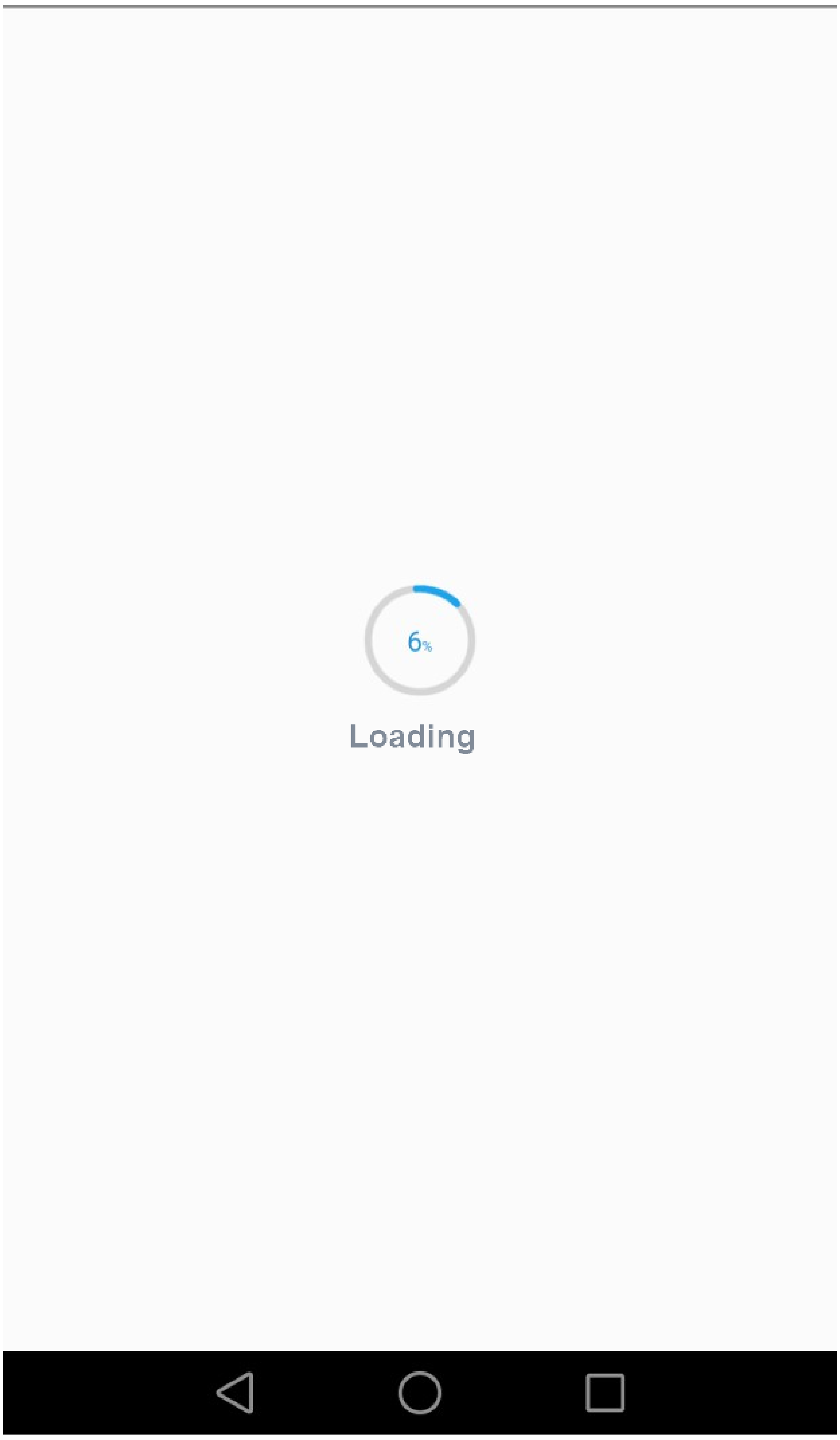
XML example:

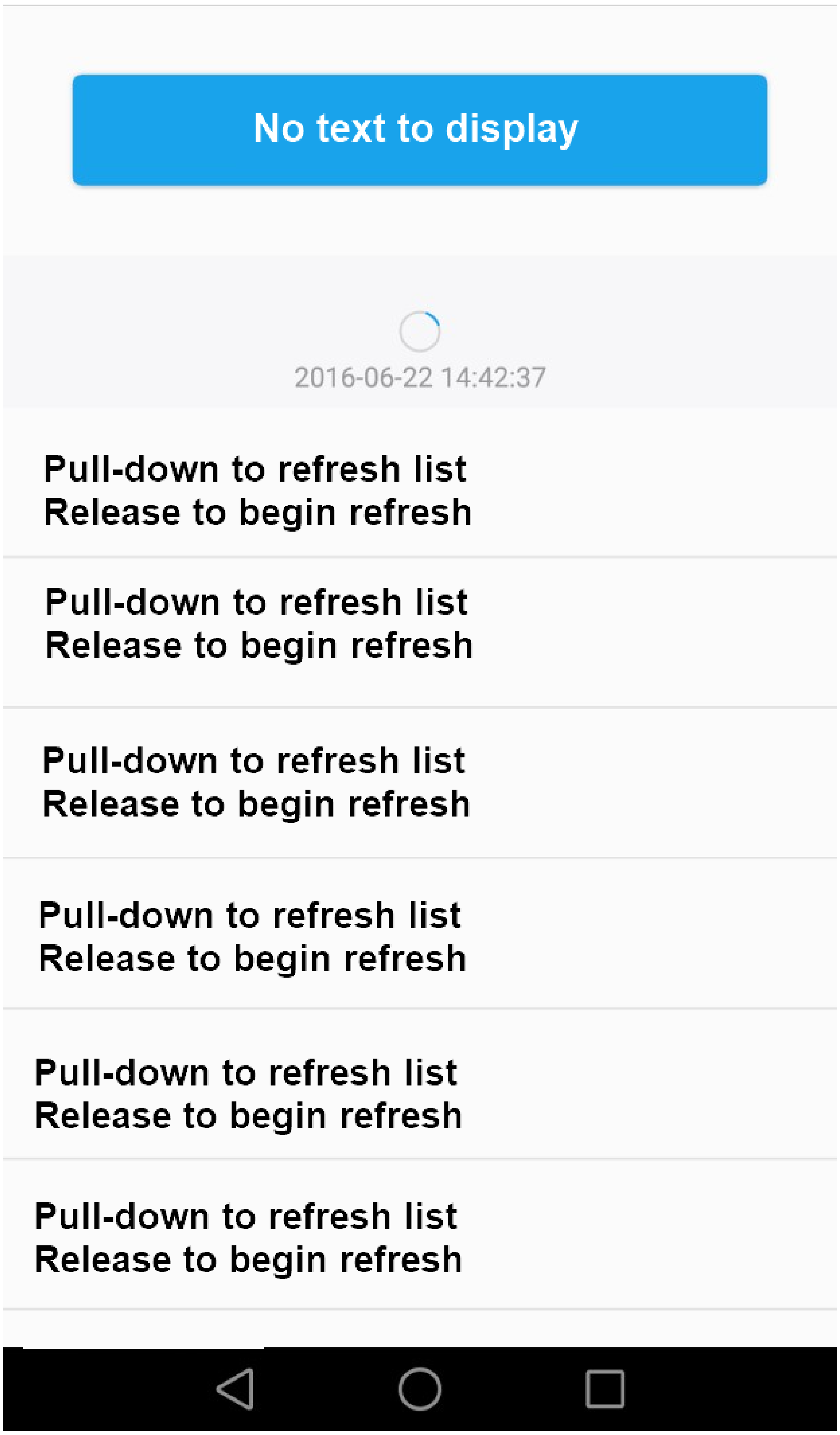
```
<com.alipay.mobile.antui.status.AUResultView
    android:id="@+id/result_view2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:icon="@drawable/icon_result_alipay"
    app:mainTitleText="Payment successful"
    app:subTitleText="998.00"
    app:thirdTitleText="CNY 1098.00"/>
```

1.2.6. Loading component

The AULoadingView component provides a loading page that displays a progress indicator, loading status, and descriptive text.

Previews


6%
Loading



API reference

AULoadingView

```
/**
* Constructor
* @param context The page context that contains the antu dependency.
*/
public AULoadingView(Context context)
/**
* Sets the progress.
* @param curentProgress The progress.
*/
public void setCurrentProgress(int curentProgress)
```

AUPullLoadingView

```
/**
 * Constructor
 * @param context The page context that contains the antu dependency.
 */
public AUPullLoadingView(Context context)

    /**
 * Sets the progress pattern.
 * @param drawable
 */
public void setProgressDrawable(Drawable drawable)

/**
 * Sets the rebound pattern.
 * @param mIndicatorUpDrawable
 */
public void setIndicatorUpDrawable

/**
 * Sets the loading text.
 * @param loadingText
 */
public void setLoadingText(String loadingText)

/**
 * Sets the drag text.
 * @param indicatorText
 */
public void setIndicatorText(String indicatorText)
```

AUDragLoadingView

```
    /**
     * Constructor
     * @param context The page context that contains the antu dependency.
     */
    public AUDragLoadingView(Context context)
    /**
     * Sets the loading text.
     * @param text
     */
    public void setLoadingText(CharSequence text)
```

Code examples

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("Modified text...");
        }
    });
```

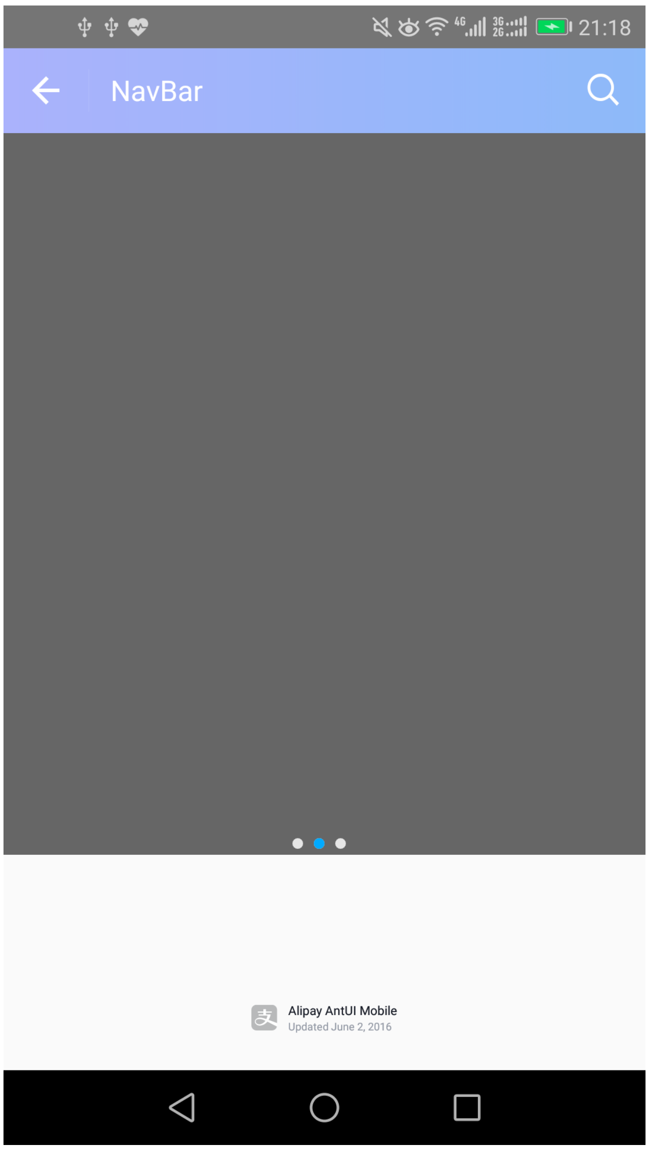
1.2.7. Navigation component

1.2.7.1. Carousel component

The AUBannerView component creates an image carousel.

Demonstration

The default setup includes an AUTitleBar control with a white background:



Code sample

```
        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);
```

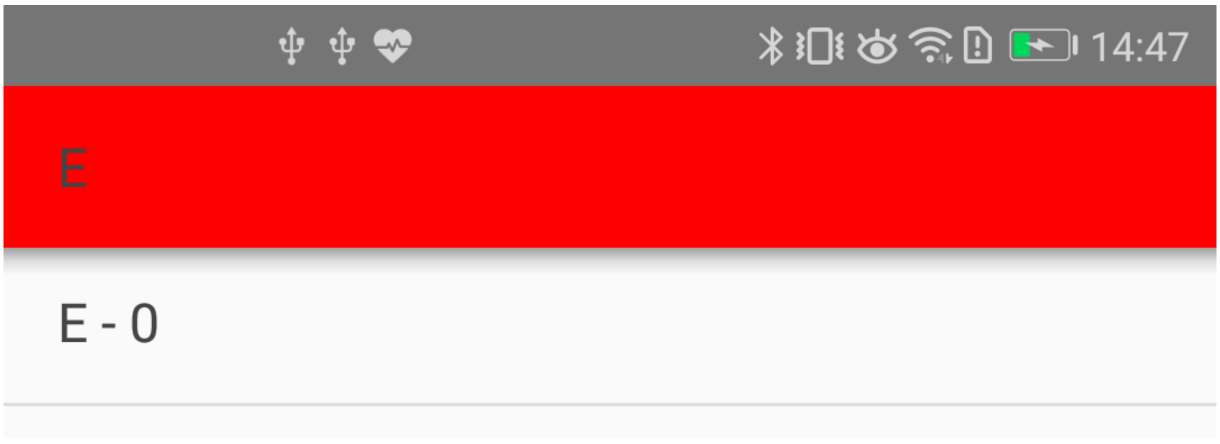
1.2.7.2. List component

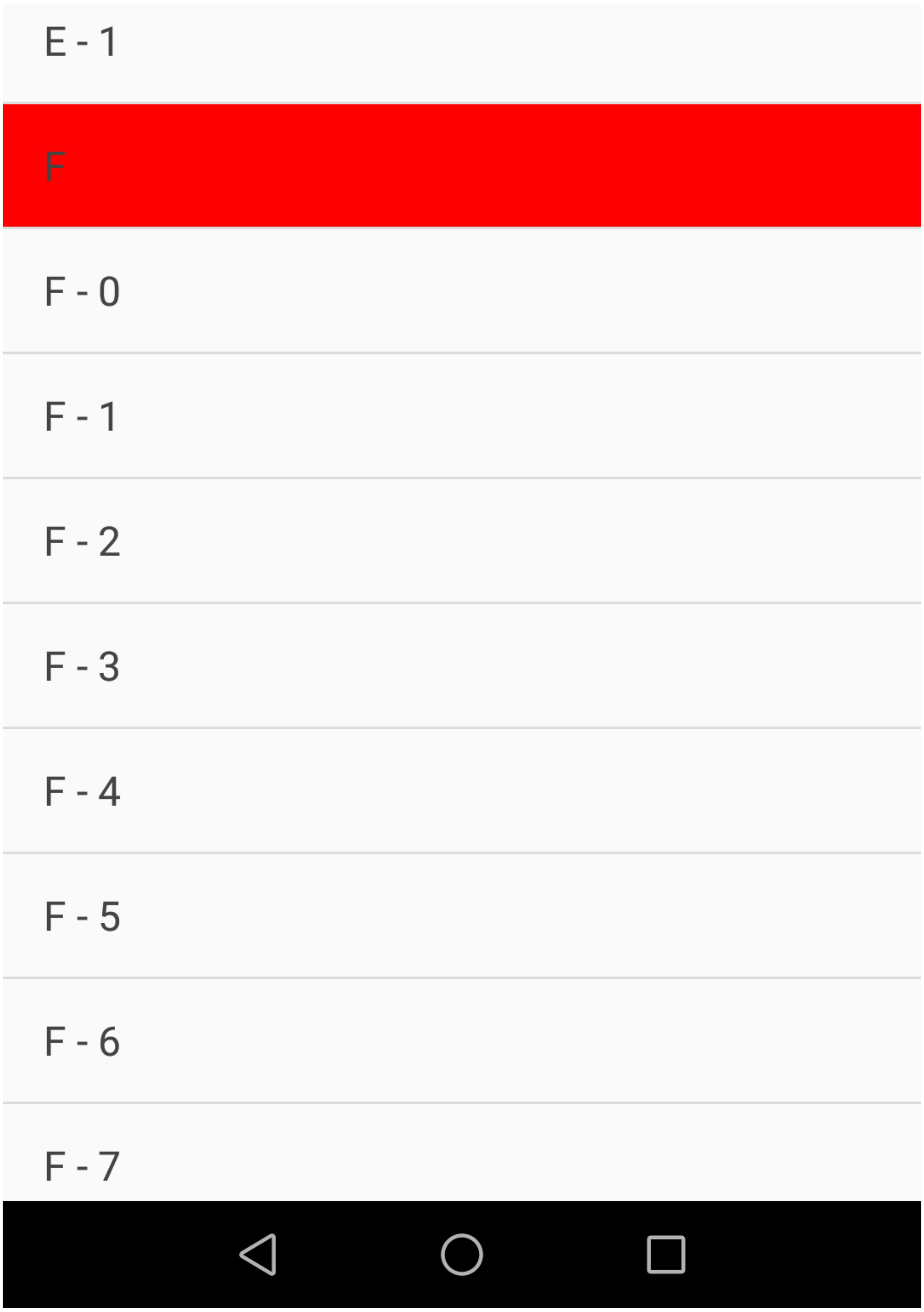
AUPinnedSectionListView provides a grouped ListView that pins the title of each group to the top during scrolling.

Note

To use this control, the data model must differentiate items by type. Otherwise, it functions as a standard ListView.

Preview




E - 1

F

F - 0

F - 1

F - 2

F - 3

F - 4

F - 5

F - 6

F - 7

Code sample

```
        public class PinnedSectionActivity extends Activity{
```

```
        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("Hello");
        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);
                add(section);

                final int itemsNumber = (int) Math.abs((Math.cos(2f*Math.PI/3f * section
sNumber / (i+1f)) * 25f));
                for (int j=0;j<itemsNumber;j++) {
                    Item item = new Item(Item.ITEM,
section.text.toUpperCase(Locale.ENGLISH) + " - " + j);
```

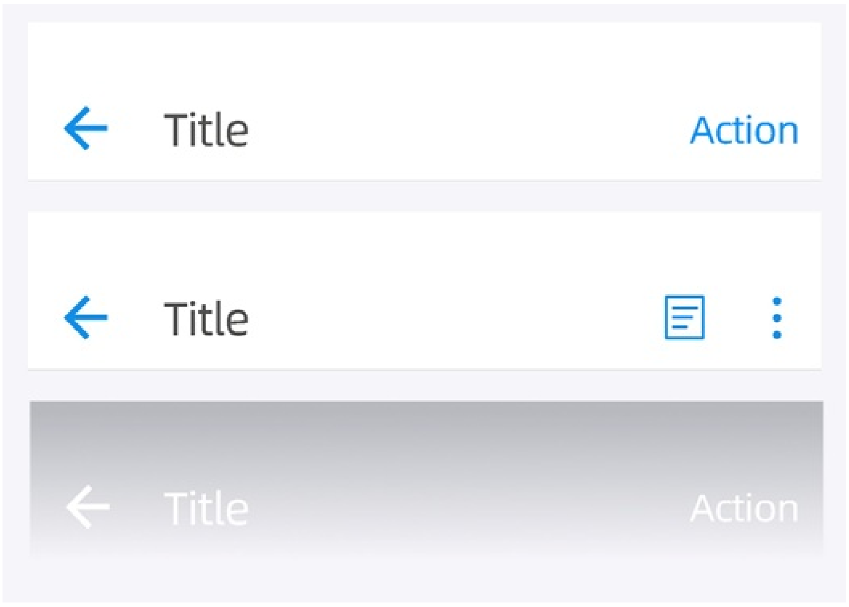
```
                        item.sectionPosition = sectionPosition;
                        item.listPosition = listPosition++;
                        add(item);
                    }

                    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;
            }

        }
    }
```

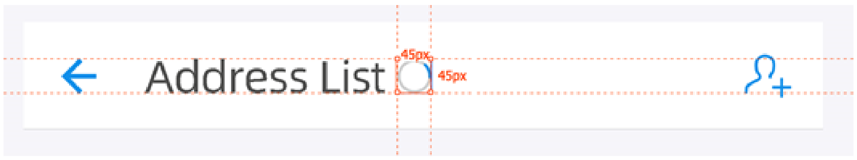
1.2.7.3. Title bar component

AUTitleBar is a title bar component that includes a back button, a title, a progress bar, a left button (with text and an icon), and a right button (with text and an icon).

Preview

By default, the AUTitleBar control has a white background:





API reference

```
    /**
     * Sets the drawable for a button.
     * @param iconView
     * @param resId
     */
    public void setBtnImage(AUIconView iconView, int resId) ;

    /**
     * Sets the size and color of a button.
     * @param iconView
     * @param size
     * @param color
     */
    public void setIconFont(AUIconView iconView, int size, int color);


    /**
     * Gets the back button.
     * @return
     */
    public AUIconView getBackButton() ;

    /**
     * Gets the left button.
     * @return
     */
    public AURelativeLayout getLeftButton() ;

    /**
     * Gets the right button.
     * @return
     */
    public AURelativeLayout getRightButton() ;

    /**
     * Gets the progress spinner.
     * @return
     */
    public AUProgressBar getProgressBar() ;

    /**
     * Gets the title text view.
     * @return
     */
    public AUTextView getTitleText() ;

    /**
     * Gets the title container.
     * @return
     */
    public AURelativeLayout getTitleContainer() ;


    /**
     * Gets the title bar area.
     * @return
     */
    public AURelativeLayout getTitleBarRelative() ;

    @Override
    public void setBackgroundDrawable(Drawable backgroundDrawable) ;

    /**
     * Sets the rotation resource for the progress bar.
     * @param progressDrawable
     */
    public void setProgressBarDrawable(Drawable progressDrawable) ;

    /**
     * Sets the text and style of the title. To use default values, set parameters to n
ull or 0.
     * @param text
     * @param textSize
     * @param textColor
     */
    public void setTitleText(String text, int textSize, int textColor) ;

    /**
     * Sets the title text.
     * @param text
     */
    public void setTitleText(String text) ;

    /**
     * Sets the resource, size, and color of the back button. If a parameter is null or
0, the default value is used.
     * @param drawable
     * @param size
     * @param color
     */
    public void setBackBtnInfo(Object drawable, int size, int color);

    /**
     * Sets the resource for the back button.
     * @param drawable
     */
    public void setBackBtnInfo(Object drawable);

    /**
     * Sets the resource, size, and color of the left button. If a parameter is null or
0, the default value is used.
     * @param drawable
     * @param size
     * @param color
     * @param isText
     */
    public void setLeftBtnInfo(Object drawable, int size, int color, boolean isText);

    /**
     * Sets the resource for the left button.
     * @param drawable
     */
    public void setLeftButtonIcon(Drawable drawable);

    public void setLeftButtonIcon(String unicode);

    public void setLeftButtonText(String text);

    /**
     * Sets the color and size of the left button.
```

```
      * Sets the color and size of the left button.
      * @param size
      * @param color
      * @param isText
      */
     public void setLeftButtonFont(int size, int color, boolean isText);

     /**
      * Sets the resource, size, and color of the right button. If a parameter is null o
r 0, the default value is used.
      * @param drawable
      * @param size
      * @param color
      * @param isText
      */
     public void setRightBtnInfo(Object drawable, int size, int color, boolean isText) ;

     /**
      * Sets the resource for the right button.
      * @param drawable
      */
     public void setRightButtonIcon(Drawable drawable);

     public void setRightButtonIcon(String unicode);

     public void setRightButtonText(String text);

     /**
      * Sets the color and size of the right button.
      * @param size
      * @param color
      * @param isText
      */
     public void setRightButtonFont(int size, int color, boolean isText);

     /**
      * Starts the progress bar rotation.
      */
     public void startProgressBar();

     /**
      * Stops the progress bar and makes it disappear.
      */
     public void stopProgressBar() ;

     /***
      * Handles the default scroll gradient. The default height is used for totalHeight.
      * @param currentHeight The current height.
      */
     public void handleScrollChange(int currentHeight);


     /***
      * Handles the default scroll gradient.
      *
      * @param totalHeight   The total height.
      * @param currentHeight The current height.
      */
     public void handleScrollChange(int totalHeight, int currentHeight) ;

     /**
      * Sets the icon color to white for use with a transparent background.
      */
     public void setColorWhiteStyle();

     /**
      * Sets the icon color for use with a white background.
      */
     public void setColorOriginalStyle();

     /**
      * Hides the back button.
      */
     public void setBackButtonGone();


     /**
      * Adds a search box (read-only). This is for visual adjustment only.
      * @param search
      */
     public void setTitle2Search(String search);

     /**
      * Transforms the search box into a title.
      */
     public void setSearch2Title();

     /**
      * Sets the search box to have black text on a white background.
      */
     public void setSearchColorOriginalStyle();

     /**
      * Sets the search box to have white text on a transparent background.
      */
     public void setSearchColorTransStyle();

     /**
      * Adds a red dot to the left icon.
      * @param flagView
      */
     public void attachFlagToLeftBtn(AUWidgetMsgFlag flagView);

     /**
      * Adds a red dot to the right icon.
      * @param flagView
      */
     public void attachFlagToRightBtn(AUWidgetMsgFlag flagView) ;
     /**
      * Adds a red dot to the targetView.
      * @param targetView
      * @param flagView
      */
     public void attachFlagView(AURelativeLayout container, View targetView,
 AUWidgetMsgFlag flagView);
```

Custom properties

Property name	Description	Type
backgroundDrawable	The background of the entire title bar.	reference
backIconColor	The color of the back arrow.	color, reference
titleText	The title text.	string, reference
titleTextSize	The font size of the title.	dimension, reference
titleTextColor	The font color of the title.	color, reference
leftIconResid	The PNG or JPG ID of the left icon.	reference
leftIconUnicode	The Unicode for the left icon.	string, reference
leftIconColor	The color of the left icon.	color, reference
leftIconSize	The size of the left icon.	dimension, reference
leftText	The text on the left.	string, reference
leftTextColor	The color of the text on the left.	color, reference
leftTextSize	The size of the text on the left.	dimension, reference
rightIconResid	The PNG or JPG ID of the right icon.	reference
rightIconUnicode	The Unicode for the right icon.	string, reference

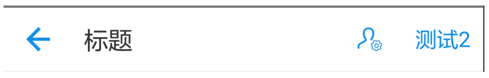
rightIconColor	The color of the right icon.	color, reference
rightIconSize	The size of the right icon.	dimension, reference
rightText	The text on the right.	string, reference
rightTextColor	The color of the text on the right.	color, reference
rightTextSize	The size of the text on the right.	dimension, reference

Code examples

In the following examples, the reference path for the `aui` prefix is `xmlns:aui="http://schemas.android.com/apk/res/com.alipay.mobile.antui"` .

Basic usage

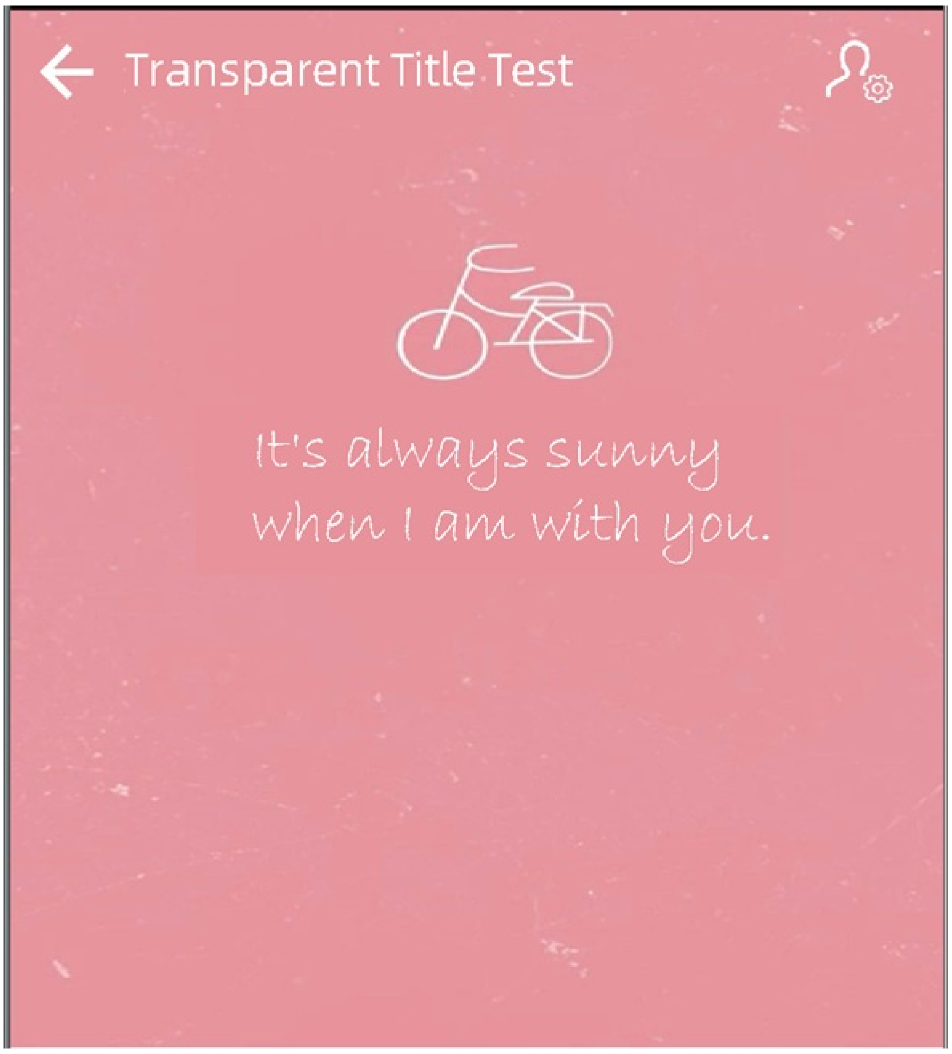
```
<com.alipay.mobile.antui.basic.AUTitleBar
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        aui:aui_titleText="Title"
        aui:aui_titleTextSize="@dimen/AU_TEXTSIZE2"
        aui:aui_titleTextColor="#f64219"
        aui:leftIconUnicode="@string/iconfont_user_setting"
        aui:rightText="Test2"/>
```



Transparent on scroll

```
<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" />
```

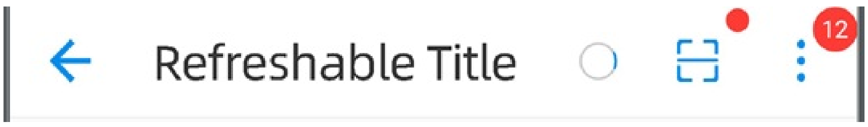
```
titleBar.handleScrollChange(testImg.getMeasuredHeight(), 0);
testScroll.setScrollViewListener(new AUScrollViewListener() {
            @Override
            public void onScrollChanged(ScrollView scrollView, int x, int y, int oldx,
int oldy) {
                titleBar.handleScrollChange(y);
            }
        });
```



Title with 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="Title with Refresh"
        aui:leftIconUnicode="@string/iconfont_scan"
        aui:rightIconUnicode="@string/iconfont_more"/>
```

```
AUTitleBar processBar = (AUTitleBar) findViewById(R.id.progress_title);

        processBar.startProgressBar();

        WidgetMsgFlag j = new WidgetMsgFlag(this);
        j.showMsgFlag();
        processBar.attachFlagToLeftBtn(j);

        WidgetMsgFlag i = new WidgetMsgFlag(this);
        i.showMsgFlag(12);
        processBar.attachFlagToRightBtn(i);
```



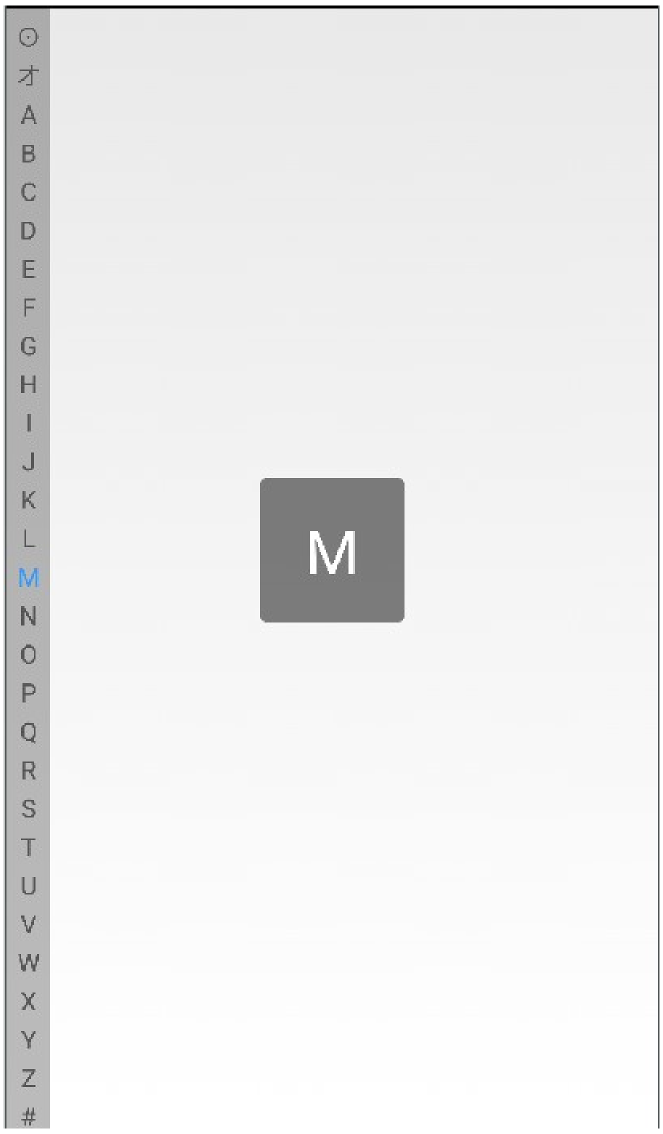
1.2.8. Other component

1.2.8.1. Index component

The AUBladeView index component works with a ListView to categorize items alphabetically. Clicking or sliding to a letter on the index, which appears on the left or right side of the page, triggers an event for that letter. The default index includes the letters A to Z. You can also add one or two custom characters at the top.

Preview

The following figure shows an example. The two characters above 'A' are custom characters, and the default index contains the letters A to Z.



API reference

```
    /**
      * Sets a listener for letter selection.
      */
         public void setOnItemClickListener(OnItemClickListener listener)

         public interface OnItemClickListener {

         /**
          * Sets a listener for letter selection.
          * @param clickChar The clicked or selected letter.
          */
         void onItemClick(String clickChar);

         /**
          * Event triggered when a finger is lifted. Ignore this method if you have no s
pecial requirements.
          */
         void onClickUp();
     }
```

Custom properties

Property name	Description	Type
top1Text	The first custom text character.	reference
top2Text	The second custom text character.	reference
showSelectPop	Specifies whether to display a pop-up layer when you slide or click an item.	boolean

Code example

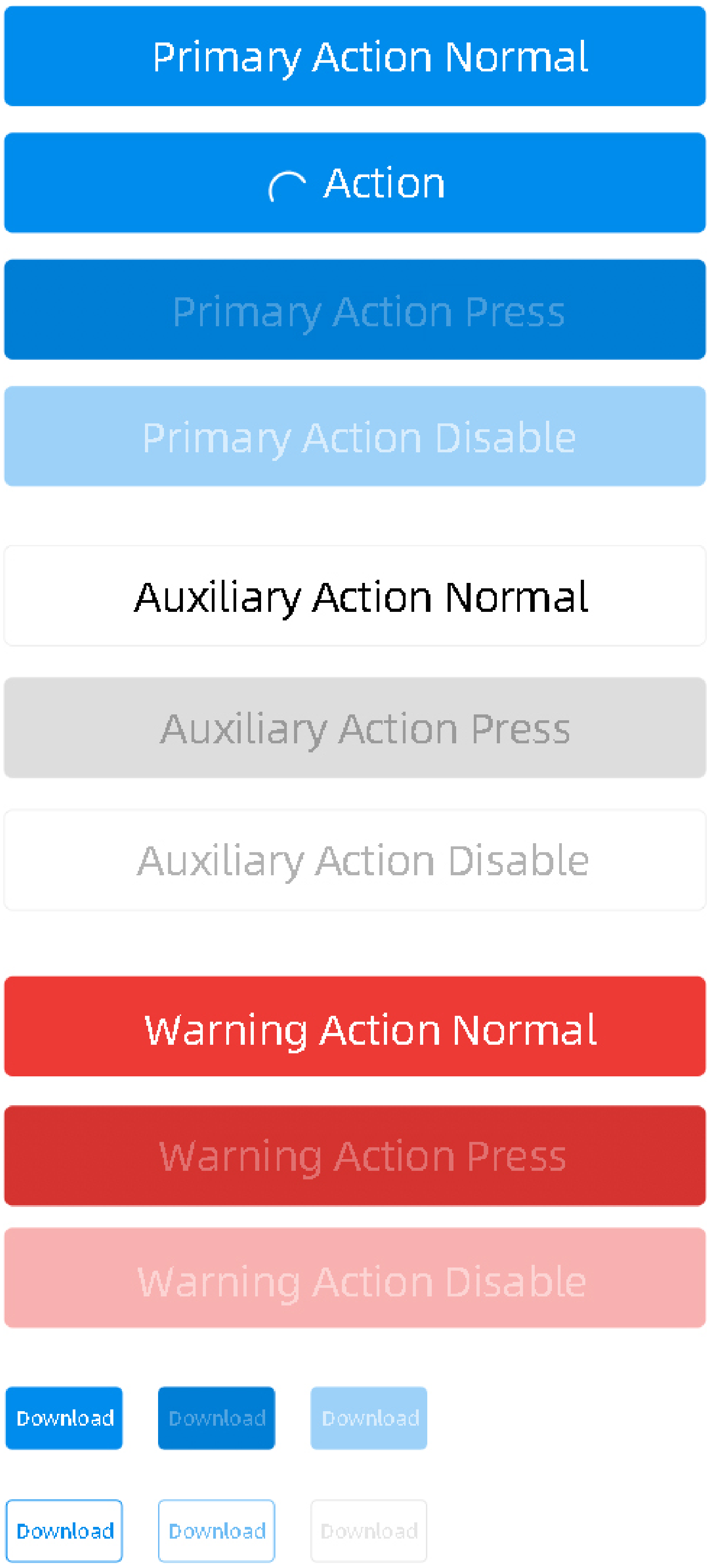
```
 <com.alipay.mobile.antui.basic.AUBladeView
        android:layout_width="24dp"
        android:layout_height="wrap_content"
        app:top1Text="⊙"
        app:top2Text="オ"/>
```

As shown in the Screenshot, the top1Text and top2Text properties are optional.

1.2.8.2. Button component

The AUButton widget provides buttons in different styles.

Preview

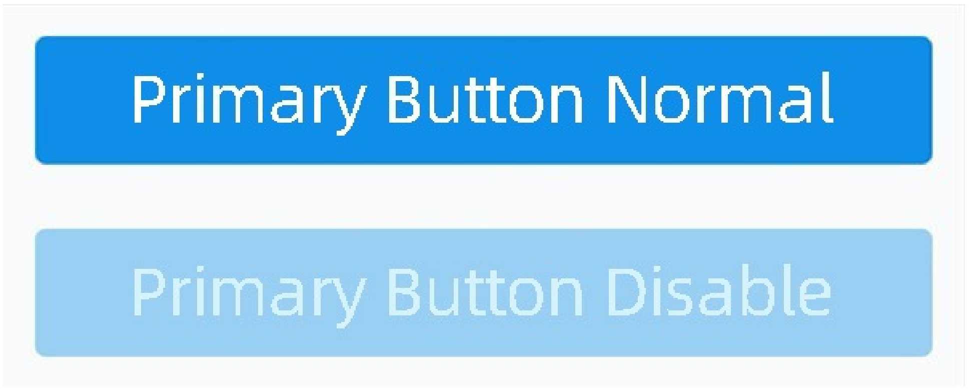


Style interface

Property name	Description
mainButtonStyle	Primary page button
subButtonStyle	Secondary page button
warnButtonStyle	Warning button
assMainButtonStyle	Auxiliary primary button
assButtonStyle	Auxiliary secondary button
listButtonStyle	List button

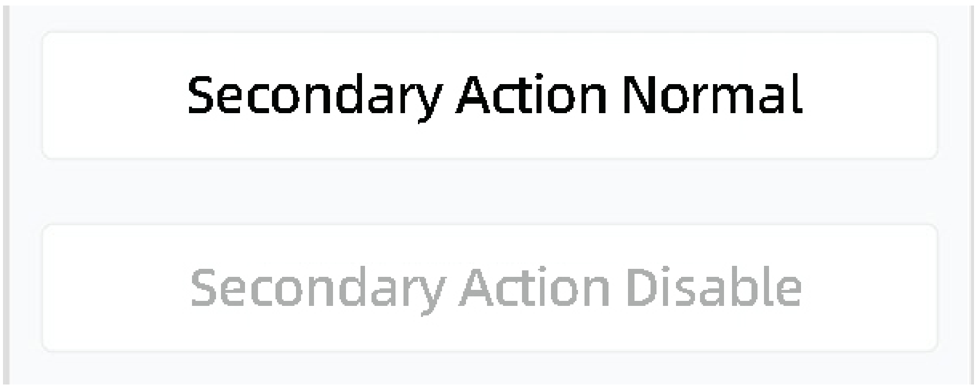
Code examples

- **Primary button**



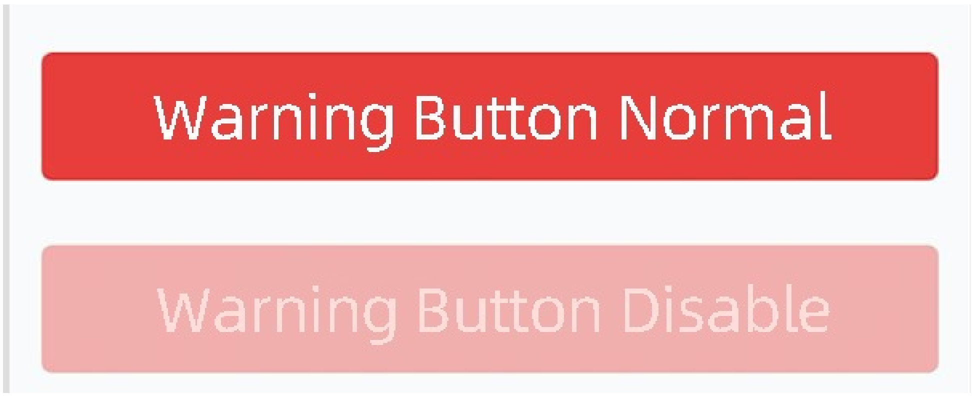
```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/mainButtonStyle"
    android:layout_margin="12dp"
    android:clickable="true"
    android:text="Primary Page Button Normal" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/mainButtonStyle"
    android:layout_margin="12dp"
    android:enabled="false"
    android:text="Primary Page Button Disable"
    app:dynamicThemeDisable="true" />
```

- **Secondary button**



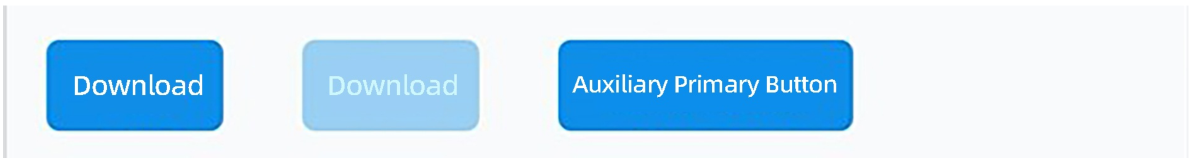
```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/subButtonStyle"
    android:layout_margin="12dp"
    android:clickable="true"
    android:text="Secondary Page Operation Normal" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/subButtonStyle"
    android:layout_margin="12dp"
    android:enabled="false"/>
```

- **Warning button**



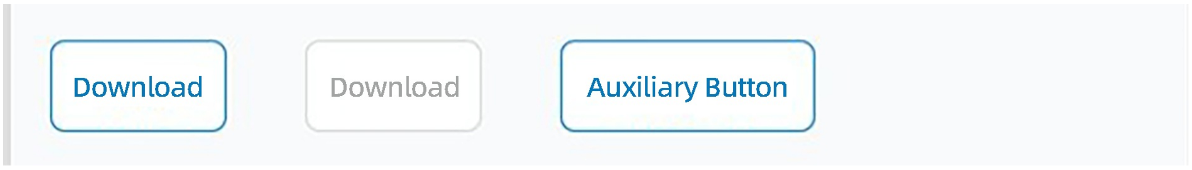
```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/warnButtonStyle"
    android:layout_margin="12dp"
    android:clickable="true"
    android:text="Warning Button Normal" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/warnButtonStyle"
    android:layout_margin="12dp"
    android:enabled="false"
    android:text="Warning Button Disable" />
```

- **Auxiliary primary button**



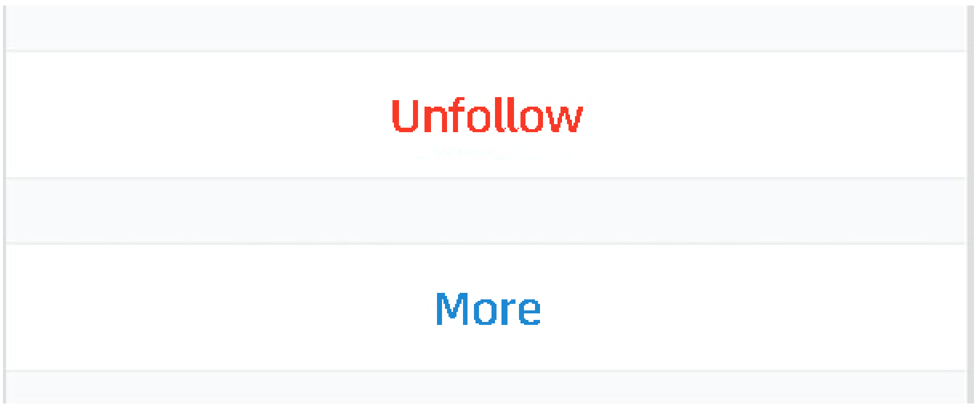
```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assMainButtonStyle"
    android:layout_margin="12dp"
    android:clickable="true"
    android:text="Download" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assMainButtonStyle"
    android:layout_margin="12dp"
    android:enabled="false"
    android:text="Download" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assMainButtonStyle"
    android:layout_margin="12dp"
    android:text="Auxiliary Primary Button" />
```

- **Auxiliary secondary button**



```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assButtonStyle"
    android:layout_margin="12dp"
    android:clickable="true"
    android:text="Download" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assButtonStyle"
    android:layout_margin="12dp"
    android:enabled="false"
    android:text="Download" />

<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/assButtonStyle"
    android:layout_margin="12dp"
    android:text="Auxiliary Button" />
```

- **List button**

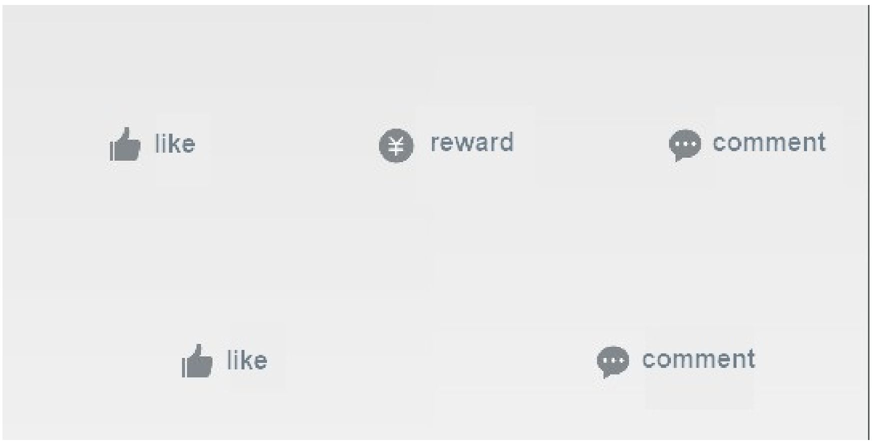


```
<com.alipay.mobile.antui.basic.AUButton
    style="@com.alipay.mobile.antui:style/listButtonStyle"
    android:layout_marginTop="12dp"
    android:layout_marginBottom="12dp"
    android:clickable="true"
    android:text="Unfollow" />

<com.alipay.mobile.antui.basic.AUButton
    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" />
```

1.2.8.3. Action bar component

The AUCardOptionView action bar component is a composite View that provides operations such as likes, comments, and rewards. It inherits from AULinearLayout and supports integration through XML layouts.

Preview



API reference

```
/**
     * Sets the information for the entire view.
     * @param itemArrayList
     * @param textVisible
```

```
        @param textVisible
      */
    public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, boolean
textVisible)

 /**
      * Sets the information for the entire view.
      * @param itemArrayList
      * @param textType If set to CardOptionView.TEXT_NOT_CHANGE, the text is always dis
played and does not change to a number.
      */
    public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, String textType)

/**
      * Sets the information for the entire view.
      * @param itemArrayList
      */
    public void setViewInfo(ArrayList<CardOptionItem> itemArrayList)

/**
      * Sets the information for the entire view.
      * @param itemArrayList
      * @param height
      * @param textVisible
      */
    public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height, boolea
n textVisible)

    /**
      * Sets the information for the entire view.
      * @param itemArrayList
      * @param height
      */
    public void setViewInfo(ArrayList<CardOptionItem> itemArrayList, int height)

/**
      * Increments the counter of a child view.
      * @param childView
      */
    public void unitIncrease(View childView)

/**
      * Decrements the counter of a child view.
      * @param childView
      */
    public void unitDecrease(View childView)

/**
      * Gets the count.
      * @param position
      * @return
      */
    public int getCount(int position)

/**
```

```
      * Returns a View object.
      * @param type
      * @return
      */
     public View getChildView(String type)

/**
      * Sets a listener.
      * @param cardOptionListner
      */
     public void setCardOptionListner(CardOptionClickListner cardOptionListner) {
         this.mListner = cardOptionListner;
     }
```

Custom properties

This component is a standard ViewGroup and does not have custom properties.

Code example

```
    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.CardOptionClickListner() {
            @Override
            public void onCardOptionClick(View v, AUCardOptionView.CardOptionItem
optionItem, int position) {
                mAUCardOptionView.unitIncrease(v);
            }
        });
```

1.2.8.4. Check icon component

The AUCheckIcon component is an IconView that functions as a checkbox.

API reference

```
/**Checked state*/
public static final int STATE_CHECKED = 0x01;
/**Unchecked state*/
public static final int STATE_UNCHECKED = 0x02;
/**Cannot be unchecked state*/
public static final int STATE_CANNOT_UNCHECKED = 0x03;
/**Cannot be checked state*/
public static final int STATE_CANNOT_CHECKED = 0x04;

/**
 * Sets the state of the check icon.
 * @param state
 */
public void setIconState(int state);

/**
 * Gets the state of the check icon.
 * @return
 */
public int getIconState() ;
```

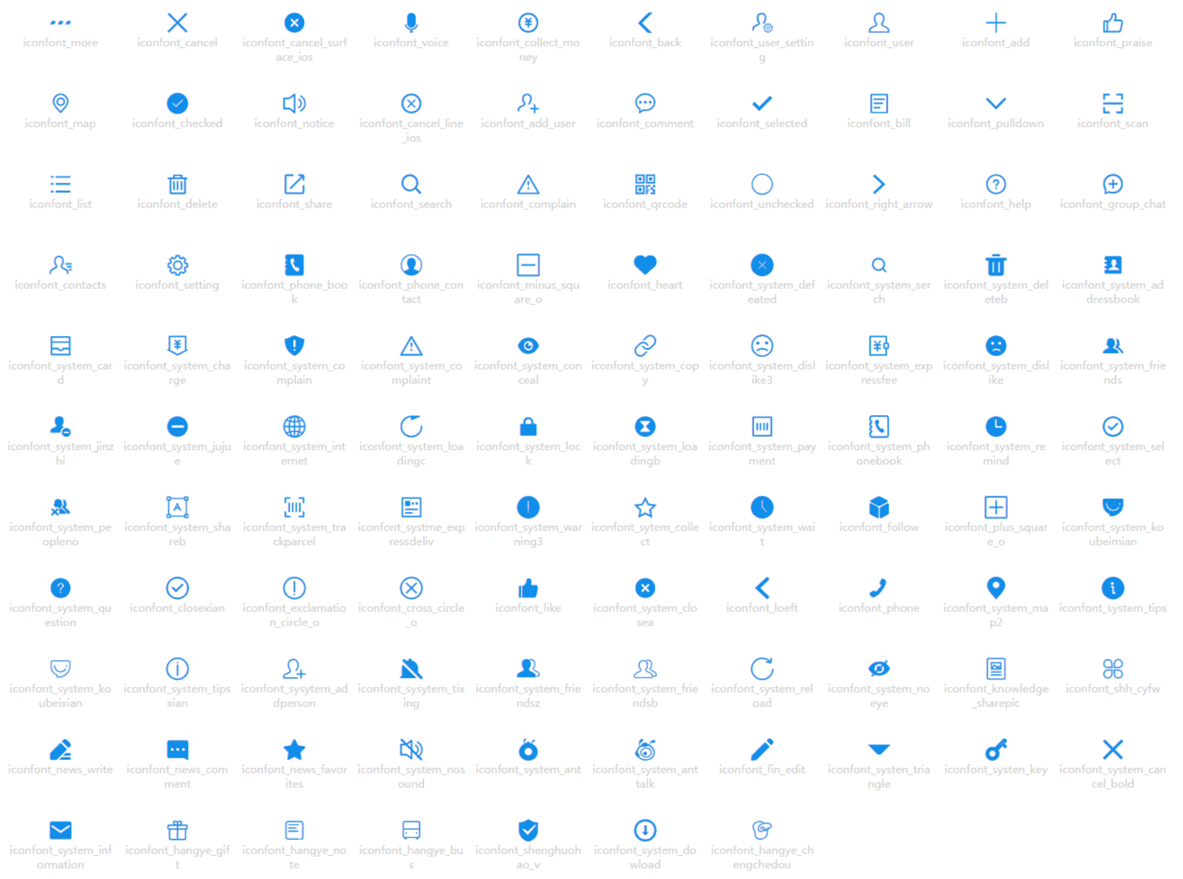
1.2.8.5. Icon component

AUIconView is an iconfont vector graphic control that combines the features of TextView and ImageView.

The iconfont control functions similarly to a TextView and uses a TrueType Font (TTF) file. An iconfont is a single font file that contains multiple icons, and each icon is mapped to a unique Unicode code.

Each iconfont collection is a single TTF file. You can load multiple TTF files, and each file is identified by a name. The default name for the AntUI TTF file is `auiconfont`.

Preview



Icon resources

Resource ID	Example 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
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	Checked
com.alipay.mobile.antui.R.string.iconfont_notice	Notice
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
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	Complaints
com.alipay.mobile.antui.R.string.iconfont_qrcode	QR code
com.alipay.mobile.antui.R.string.iconfont_unchecked	Unchecked
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	Contacts
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	Phone contacts

API reference

```
/**
 * Sets the image resource ID.
 * @param resId
 * @return
 */
@Override
public AUIconView setImageResource(int resId) {
    if (resId == 0) {
        return this;
    }
    clearView();
    initImageView();
    imageView.setImageResource(resId);
    this.addView(imageView);
    return this;
}

/**
 * Sets the image resource as a drawable.
 * @param drawable
 * @return
 */
@Override
public IconfontInterface setImageDrawable(Drawable drawable)

/**
 * Sets the iconfont color.
 * @param color
 * @return
 */
public AUIconView setIconfontColor(int color)

/**
 * Sets the iconfont color as a ColorStateList.
 * @param color
 * @return
 */
public AUIconView setIconfontColorStates(ColorStateList color)
```

```
  public AUIconView setIconfontColorStates(ColorStateList color)


  /**
   * Sets the size of the view in pixels (px).
   *
   * @param size
   */
  public AUIconView setIconfontSize(float size)


  /**
   * Sets the iconfont resource or text for the view.
   * @param text
   * @return
   */
  @Override
  public AUIconView setIconfontUnicode(String text)
```

Code examples

- 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
confont_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)
```

1.2.8.6. Refresh component

The AURefreshListView is a ListView component that supports pull-to-refresh and loading more items.

API reference

```
/**
 * A listener for the pull-to-refresh status.
 *
 * @param onPullRefreshListener
 */
public void setOnPullRefreshListener(OnPullRefreshListener onPullRefreshListener)

/**
 * A listener for the load-more status.
 *
 * @param onLoadMoreListener
 */
public void setOnLoadMoreListener(OnLoadMoreListener onLoadMoreListener)

/**
 * Starts a pull-to-refresh action in your code.
 */
public void startRefresh()

/**
 * Ends the pull-to-refresh action.
 */
public void finishRefresh()

/**
 * Updates the status of the load-more footer.
 *
 * @param isShowLoad
 * @param hasMore
 */
public void updateLoadMore(boolean isShowLoad, boolean hasMore)
```

Code examples

```
<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", "Load more list items");
            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() {

    }
});
```

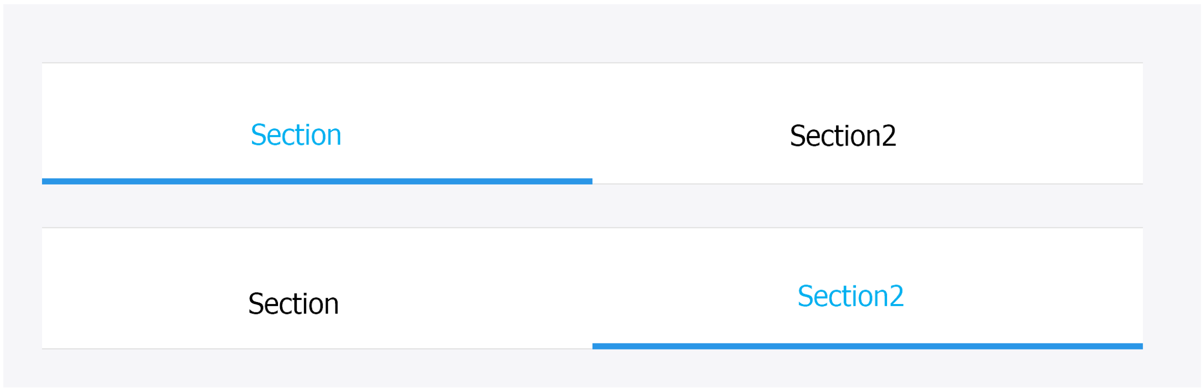
1.2.8.7. Switch bar component

AUSegment replaces the APSwitchTab control. The code has been refactored, but it retains the original interfaces and supports smooth scrolling.

In versions **10.0.20** and later, this component supports scrollable tab switching. Each tab has 14 dp of space on its left and right sides.

- If the total width of all tabs exceeds the initial width, you can scroll to switch between them.
- If the total width of all tabs is less than the initial width, the tabs are spaced evenly by default.

Preview



API reference

```
  /**
   * Resets the tab view.
   */
  public void resetTabView(String[] tabNameArray)

  /**
   * Adjusts the position of the bottom selection line.
   * This method is typically called in the onPageScrolled callback of a ViewPager.
   *
   * @param position The starting position.
   * @param positionOffset The offset from the starting position, as a percentage.
   */
 public void adjustLinePosition(int position, float positionOffset)

  /**
   * Selects a tab without adjusting the bottom line position.
   * This is suitable for tab switching in a ViewPager.
   * The bottom selection line is adjusted by calling adjustLinePosition in the onPageS
crolled callback of the ViewPager.
   *
   * @param position The position of the tab.
   */
 public void selectTab(int position)

  /**
   * Selects a tab and adjusts the bottom line position.<br>
   * This is used for tab switching scenarios that do not involve a ViewPager. The tran
sition animation time between tabs is 250 ms.
   *
   * @param position The target tab to select.
   */
  public void selectTabAndAdjustLine(int position)

  /**
   * Selects a tab and adjusts the bottom line position.<br>
   * This is used for tab switching scenarios that do not involve a ViewPager. You can
specify the transition animation time between tabs.<br>
   * If a new animation starts before the previous one finishes, the previous animation
stops immediately. The new animation starts from the final position of the previous one
```

```
.
   *
   * @param position The target position.
   * @param during The transition animation time between tabs.
   */
  public void selectTabAndAdjustLine(int position, int during)

  /**
   * Sets the tab switch listener.
   *
   * @param tabSwitchListener
   */
  public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

  /**
   * Adds a red dot at a specified position.
   * @param view The red dot view.
   * @param position
   */
  public void addTextRightView(View view, int position)

  /**
   * Adds a red dot at a specified position.
   * @param view The red dot.
   * @param params The relative layout parameters of the red dot.
   * @param position
   */
  public void addTextRightView(View view, RelativeLayout.LayoutParams params, int posit
ion)
```

API reference for scrollable tab switching

To enable the scroll feature, set the custom `scroll` property to `true`. For example, add `app:scroll="true"` in the layout file. Scrollable tabs support only the following four APIs.

```
  /**
   * Sets the tab switch listener.
   * @param tabSwitchListener
   */
  public void setTabSwitchListener(TabSwitchListener tabSwitchListener)

  /**
   * Sets the data source.
   * @param list
   */
  public void init(List<ItemCategory> list)
  /**
   * Sets the selected tab.
   * @param position
   */
  public void setCurrentSelTab(int position)

  /**
   * The space on the left and right of each tab is fixed at 14 dp.
   * If the total width of all tabs is less than the initial width, this method determi
nes whether to space them evenly.
   * By default, tabs are spaced evenly. Set this parameter to false to disable even sp
acing.
   * @param divideAutoSize
   */
  public void setDivideAutoSize(boolean divideAutoSize)
```

Custom properties

The `scroll` property is available in versions 10.0.20 and later.

Property	Usage	Type
tabCount	Number of tabs	integer
tab1Text	Text for Tab1	string, reference
tab2Text	Tab 1 text	string, reference
tab3Text	Text for Tab3	string, reference
tab4Text	Text for Tab4	string, reference
tabTextArray	Text array for tabs	string, reference
uniformlySpaced	Whether to space tabs evenly	boolean

tabTextColor	Text color	reference, color
tabTextSize	Text size	dimension
buttomLineColor	Bottom line color	color, reference
scroll	Whether scrolling is supported	boolean

Code example

XML example:

```
<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="Center Text"
        app:tab3Text="Right Text"
        app:tabCount="3"/>
```

1.2.8.8. TabBar item component

The AUTabBarItem label component is used for each item in the TabBar of the mPaaS framework.

Custom properties

Property name	Description	Type
topIconSid	Icon	reference
topIconSize	Icon size	dimension
textColor	Text color	color, reference

Code example

XML example:

```
<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"/>
```

1.3. Native based - iOS component library

1.3.1. Getting Started

This topic describes how to integrate the common UI component library into an iOS client.

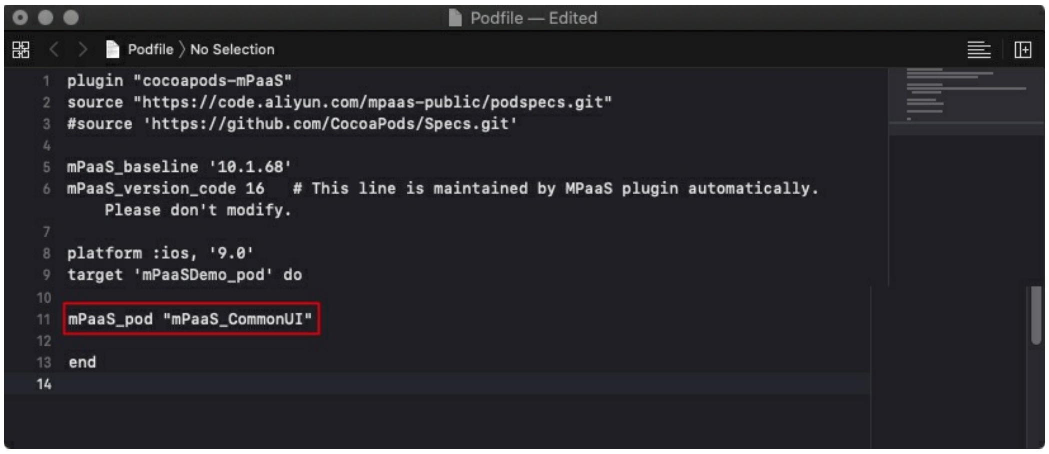
Prerequisites

Your project is connected to mPaaS. For more information, see Connect an existing project using CocoaPods.

Add the SDK

Use the cocoapods-mPaaS plugin to add the SDK for the Common UI component library. The procedure is as follows:

1. In your Podfile, add the dependency for the Common UI component using `mPaaS_pod "mPaaS_CommonUI"` .



```
plugin "cocoapods-mPaaS"
source "https://code.aliyun.com/mpaas-public/podspecs.git"
#source 'https://github.com/CocoaPods/Specs.git'

mPaaS_baseline '10.1.68'
mPaaS_version_code 16   # This line is maintained by MPaaS plugin automatically.
    Please don't modify.

platform :ios, '9.0'
target 'mPaaSDemo_pod' do

mPaaS_pod "mPaaS_CommonUI"

end
```

2. Run `pod install` to complete the integration.

Use the SDK

You can use the Common UI SDK with baseline 10.1.60 or later. For more information, see the official Common UI demo.

1.3.2. Basic components

1.3.2.1. Activity Indicator base class

- The activity indicator base class AUActivityIndicatorView is the UIActivityIndicatorView version in mPaaS.
- To facilitate subsequent extension, use AUActivityIndicatorView instead of UIActivityIndicatorView of the system in all mPaaS apps.
- Since current AUActivityIndicatorView is completely inherited from UIActivityIndicatorView without additional properties and methods, this topic does not describe APIs and code examples.

1.3.2.2. 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.

1.3.2.3. Checkbox control

This topic provides a preview of the checkbox control and a code example.

- `AUCheckBox` is the checkbox control.
- `AUCheckBox` is a migration from `APCheckbox` in `APCommonUI`. You should use `AUCheckBox`.

API reference

```
/**
 Checkbox style

 - AUCheckBoxStyleDefault:   The default style, similar to a web checkbox.
 - AUCheckBoxStyleCheckmark: The checkmark style for a table view.
 */
typedef NS_ENUM(NSInteger, AUCheckBoxStyle) {
        AUCheckBoxStyleDefault,
        AUCheckBoxStyleCheckmark
};



/**
 The checkbox control.
 */
@interface AUCheckBox : UIControl

/**
 Initializes an AUCheckBox instance with a specific style.

 @param style The checkbox style.

 @return An AUCheckBox instance.
 */
- (instancetype)initWithStyle:(AUCheckBoxStyle)style;

/**
 The property that indicates whether the checkbox is selected.
 */
@property(nonatomic, assign, getter = isChecked) BOOL checked;

/**
 The property that indicates whether the checkbox is disabled.
 */
@property(nonatomic, assign, getter = isDisabled) BOOL disabled;

/**
 The checkbox style. This property is read-only and can be set only during initializati
on.
 */
@property (nonatomic, assign, readonly) AUCheckBoxStyle style;


@end
```

Code example

```
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);
}
```

1.3.2.4. Base image class

The AUImage class is the mPaaS version of UIImage. To support future extensions, all mPaaS applications must use AUImage instead of the system's UIImage class.

The base image class fully inherits from UIImage and adds no new properties or methods. Therefore, this document does not include interface descriptions or code examples.

1.3.2.5. 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 AntUI) can be used.

1.3.2.6. Footer base class

The footer control mainly contains the text link component, the copyright component, and the combination of the text link and copyright.

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 at
Button:(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) AULabel *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) AUTextLinkView *textLinkView;        // The text link.
@property (nonatomic ,strong) AUCopyrightView *copyrightInfoView;  // Copyright text.

//
- (instancetype)initWithFrame:(CGRect)frame params:(paramsBlock)params;

@end
```

Code sample

- Text link:

```
 AUTextLinkView *textLinkBtns = [[AUTextLinkView 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];
```

1.3.2.7. mPaaS custom loading controls

This topic describes the custom loading control in mPaaS and provides code examples.

- AULoadingIndicatorView is the custom loading control provided by mPaaS.

- It is migrated from APActivityIndicatorView in APCommonUI. Therefore, you should use AULoadingIndicatorView.

API reference

```
typedef enum{
        AULoadingIndicatorViewStyleTitleBar,   // Loads in the navigation bar.
Diameter: 36 px. Ring width: 3 px.
        AULoadingIndicatorViewStyleRefresh,    // Loads when a list is refreshed.
Diameter: 48 px. Ring width: 4 px.
        AULoadingIndicatorViewStyleToast,      // Loads in a toast message. Diameter: 72
px. Ring width: 6 px.
        AULoadingIndicatorViewStyleLoading,    // Loads on a page. Diameter: 90 px.
Ring width: 6 px.
}AULoadingIndicatorViewStyle;




/**
 mPaaS custom loading control
 */
@interface AULoadingIndicatorView : UIView

@property (nonatomic, assign) BOOL    hidesWhenStopped;    // Specifies whether to hide
the control when the animation stops.
@property (nonatomic, strong) UIColor *trackColor;         // The color of the ring.
@property (nonatomic, strong) UIColor *progressColor;      // The color of the
indicator.
@property (nonatomic, assign) float progressWidth;         // The width of the ring. Th
e default value is 2 when you customize the circle size.
@property (nonatomic, assign) CGFloat progress;            // The ratio of the loading
indicator's arc length to the ring. The default value is 0.1.

/**
 *  A circular loading box.
 *  Note: To customize the circle size instead of using a default style, use initWithFr
ame: for initialization. The ring width defaults to 2. Set progressWidth to adjust the
width.
 *
 *  @param style  The current loading type.
 *
 */
- (instancetype)initWithLoadingIndicatorStyle:(AULoadingIndicatorViewStyle)style;

/**
 Starts the animation.
 */
- (void)startAnimating;

/**
 Stops the animation.
 */
- (void)stopAnimating;

/**
```

```
 Checks whether the animation is running.

 @return YES if the animation is running. NO if the animation is not running.
 */
- (BOOL)isAnimating;


@end
```

Code example

```
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];
```

1.3.2.8. 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.

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:(AUButtonStyle)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 AU
ButtonStyle.
     *  @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:(AUButtonStyle)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 t
he 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 th
is parameter is set to nil or an empty string, text is not displayed and the loading ic
on is centered.
     @param currentVC       The current VC that is used to remove the mask after the lo
ading is complete.
     */
    - (void)startLoadingWithTitle:(NSString *)loadingTitle currentViewController:(UIVie
wController *)currentVC;


    /**
     Stop the rotation of the loading icon.
     */
    - (void)stopLoading;
```

Custom properties

Property	Description
AUButtonStyleNone	The default style.

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

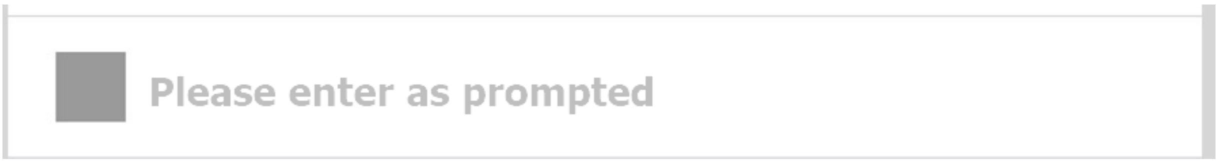
```
AUButton *button = [AUButton buttonWithStyle:AUButtonStyle2     title:@"AUButtonStyle2"
target:self action:@selector(onButtonClicked:)];
   button.frame = CGRectMake(XX, XX,XX, XX);

    AUButton *buttonDisable = [AUButton buttonWithStyle:AUButtonStyle1];
   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];
```

1.3.3. Input components

1.3.3.1. 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

```
AUImageInputBox *imageInputBox = [AUImageInputBox inputboxWithOriginY:startY inputboxTy
pe:AUInputBoxTypeNone];
imageInputBox.textField.placeholder = @"Enter as promoted";
[imageInputBox setIconImage:image];
[self.view addSubview:imageInputBox];
```

1.3.3.2. 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.

API description

```
// The multi-line text input box.

@interface AUParagraphInputBox : UIView

@property (nonatomic, strong) UITextView *textView;      // Imput 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, AUCommonUIGetScreenWidth(), 10);
_paragraphInputBox.maxInputLen = 1240;
_paragraphInputBox.textView.delegate = self;
[_paragraphInputBox setPlaceHolder:@"Please enter text here"];
[self.view addSubview:_paragraphInputBox];
```

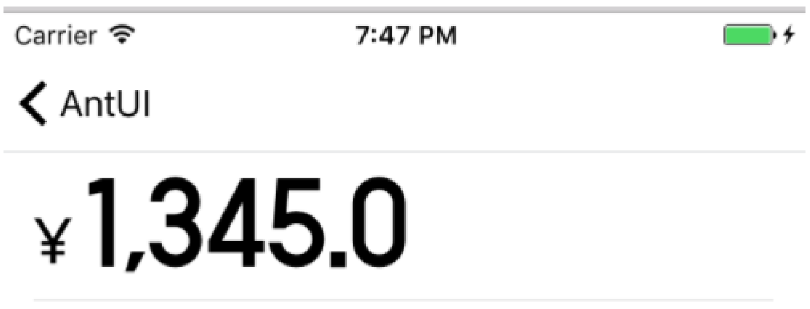
1.3.3.3. Simplified amount input box

The AUAmountEditTextField is a simplified amount input box that works with the AUAmountLabelText amount display component.

AUAmountEditTextField

AUAmountEditTextField does not include logic for input validation or pre-processing. You can implement these features by setting a delegate.

Preview



API reference

```
NS_ASSUME_NONNULL_BEGIN

@interface AUAmountEditTextField : UITextField

@end


/**
  A simplified amount input component with a "¥" symbol and an underline.
  The font size of the input content scales with the content length.
 */
@interface AUAmountEditText : UIView


/**
    The amount input box. Modify its properties or set a delegate as needed.
    When a clear event occurs, it calls [amountTextField
sendActionsForControlEvents:UIControlEventEditingChanged].
 */
@property(nonatomic,strong) AUAmountEditTextField *amountTextField;


/**
 Exposed for AUAmountLabelText to adjust the font size when the inputText length change
s.
 Do not use this method in your application.

 @param textLength The length of inputText.
 @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 fontWithCustomName:kAmountNumberFontName size:45.0];
_amountTextField.contentVerticalAlignment= UIControlContentVerticalAlignmentCenter;
_amountTextField.inputView = [AUNumKeyboards
sharedKeyboardWithMode:AUNumKeyboardModeCommon];
_amountTextField.rightViewMode = UITextFieldViewModeWhileEditing;
_amountTextField.rightView = self.rightView;// The clear button is implemented using ri
ghtView.
```

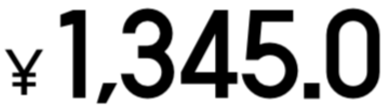
Code example

```
field = [[AUAmountEditText alloc] init];// Defaults to screen width and a height of 70.
field.amountTextField.delegate = self;
[view addSubview:field];
```

AUAmountLabelText

AUAmountLabelText is an amount display component that works with AUAmountEditTextField.

Preview



API reference

```
NS_ASSUME_NONNULL_BEGIN

/**
 An amount display component that works with AUAmountEditTextField.
 */
@interface AUAmountLabelText : UIView

@property (nonatomic, copy) NSString *amountText;// The amount. Does not include the cu
rrency symbol. For example: "80.01".

@end

NS_ASSUME_NONNULL_END
```

Code example

```
label = [[AUAmountLabelText alloc] init];// Defaults to screen width and a height of 64
px.
label.amountText = @"1,345.0";
[view addSubview:label];
```

1.3.3.4. Amount input box

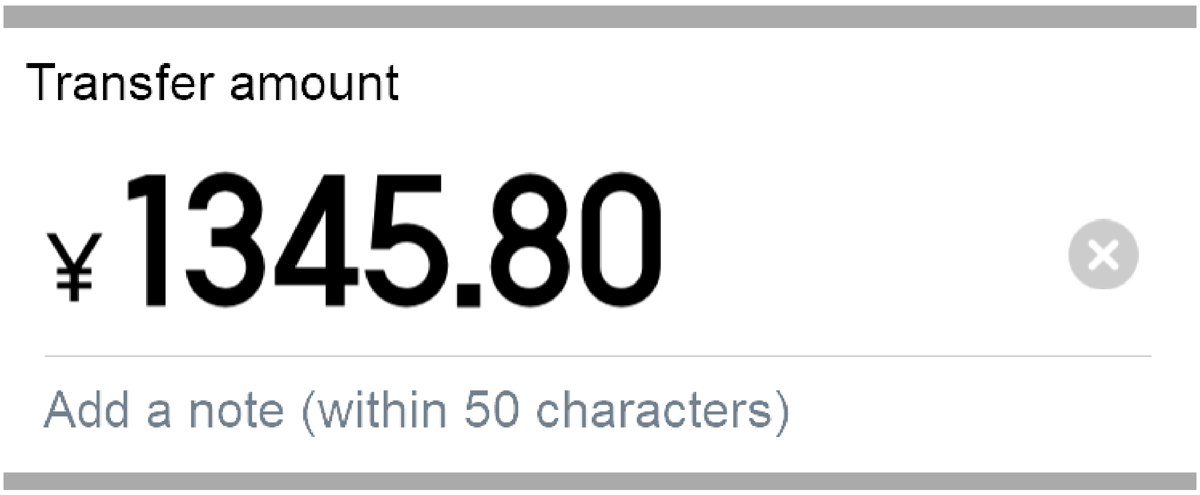
AUAmountInputBox is a composite amount input box that consists of AUAmountInputField and AUAmountInputFieldFooterView.

AUAmountInputBox

AUAmountInputBox supports a plain text title and a footer, which can be either plain text or an input box.

This component does not include logic for input validation or pre-processing. You can implement this logic by setting a delegate in your application.

Preview



API reference

```
NS_ASSUME_NONNULL_BEGIN

/**
 An amount input box with combined features.
 Supports setting a title (plain text) and adding a footer (plain text or an input box)
.
 Does not include logic for input validation or pre-processing. Implement this logic by
setting a delegate in your application.
*/
@interface AUAmountInputBox : UIView

/**
 The AUAmountInputBox initialization method.

 @param views @[AUAmountInputField,AUAmountInputFieldFooterView]
 @return An AUAmountInputBox object.
 */
+ (AUAmountInputBox *)amountInputBoxWithViews:(NSArray *) views;

@end

NS_ASSUME_NONNULL_END
```

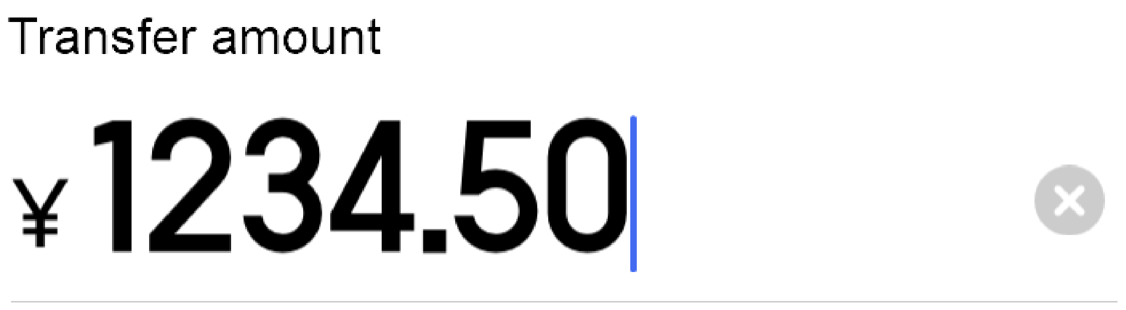
Code example

```
AUAmountInputField *inputField = [AUAmountInputField amountInputWithTitle:@"Transfer Am
ount"];
AUAmountInputFieldFooterView *footerView = [AUAmountInputFieldFooterView
footerWithInput:@"Add a remark (50 characters max)"];
AUAmountInputBox *inputBox = [AUAmountInputBox amountInputBoxWithViews:[NSArray arrayWi
thObjects:inputField,footerView,nil]];
inputField.textField.delegate = self;
footerView.inputTextField.delegate = self;
[_scrollView addSubview:inputBox];
```

AUAmountInputField

This is a composite extension of AUAmountEditText that supports a title.

Preview



API reference

```
NS_ASSUME_NONNULL_BEGIN


/**
 A composite extension of AUAmountEditText that supports setting a title.
 */
@interface AUAmountInputField : UIView

- (AUAmountEditTextField *)textField;

+ (AUAmountInputField *)amountInputWithTitle:(NSString *) title;

@end

NS_ASSUME_NONNULL_END
```

Code example

See the code example for AUAmountInputBox.

AUAmountInputFieldFooterView

Note

AUAmountInputFieldFooterView is the footerView for AUAmountInputBox. It supports two types: plain text and input box.

Preview

Dependencies

AUAmountInputFieldFooterView has the following dependency:

```
pod 'AntUI'
```

API reference

```
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
```

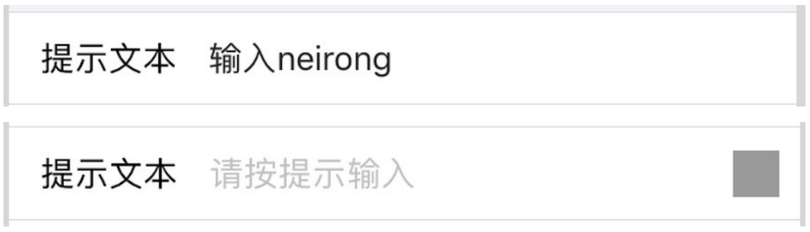
Code example

See the code example for AUAmountInputBox.

1.3.3.5. Standard input box

AUInputBox is a standard input box that features a title on the left and an image button on the right.

Preview



API reference

```
typedef NS_ENUM(NSInteger, AUInputBoxType)
{
        AUInputBoxTypeMobileNumber,     // Mobile phone number
        AUInputBoxTypeCreditCard,       // Credit card
        AUInputBoxTypeBankCard,         // Debit card
        AUInputBoxTypeAmount,           // Amount
        AUInputBoxTypeIDNumber,         // ID card number
        AUInputBoxTypeNotEmpty,         // Not empty
        AUInputBoxTypeAlipayAccount,    // mPaaS application account
        AUInputBoxTypeNone              // No validation
};

typedef enum AUInputBoxStyle
{
        AUInputBoxStyleNone, // No background image
        AUInputBoxStyleiOS6, // Background image with rounded corners
        AUInputBoxStyleiOS7  // Background image without rounded corners
} AUInputBoxStyle;


/**
 A standard input box that can have a title and an image button.
 */
@interface AUInputBox : UIView

#pragma mark - AUInputBox properties

// Text box
@property(strong, nonatomic)   AUTextField      *textField;
@property(strong, nonatomic)   NSString         *textFieldText;
@property(strong, nonatomic)   NSString         *textFieldFormat;
@property(assign, nonatomic)   CGFloat          horizontalMargin;
@property(assign, nonatomic)   CGFloat          textFieldHorizontalMargin;


// Button
@property(strong, nonatomic)   UIButton         *iconButton;
@property(assign, nonatomic)   BOOL             hidesButtonWhileNotEmpty;
@property(assign, nonatomic)   BOOL             hidesButton;


// The label on the left of the input box
@property(nonatomic, readonly) UILabel          *titleLabel;
@property(nonatomic, assign)   CGFloat          titleLabelWidth;


// Style, validator, background image, and text box type
@property(assign, nonatomic)   AUInputBoxStyle  style;
@property(readonly, nonatomic) UIImageView      *backgroundImage;
@property(assign, nonatomic)   AUInputBoxType   inputBoxType;

#pragma mark - AUInputBox static methods
/**
 *  Creates an input box component.
 *  @param  originY The Y-coordinate of the input box.
 *  @param  type The type of the text box.
 *  @return The input box component.
 */
+ (instancetype)inputboxWithOriginY:(CGFloat)originY inputboxType:(AUInputBoxType)type;

/**
 *  Creates an input box component with an icon button.
 *  @param  originY The Y-coordinate of the input box.
 *  @param  icon The icon on the button. The size is 44 × 44.
 *  @param  type The type of the text box.
 *  @return The input box component with a button.
 */
+ (instancetype)inputboxWithOriginY:(CGFloat)originY buttonIcon:(UIImage *)icon inputbo
xType:(AUInputBoxType)type;
```

```
/**
 *  @return The height of the control. The default value is 44. For iPhone 6 Plus, the
value is 47.
 */
+ (float)heightOfControl;


#pragma mark - AUInputBox instance methods

- (instancetype)initWithFrame:(CGRect)frame inputboxType:(AUInputBoxType)type;

- (void)buildIconButton:(UIImage *)icon;

/**
 *  Adds spaces to the text based on the specified format.
 *  @param  text The text content.
 *  @return The text with spaces added.
 */
- (NSString *)formatText:(NSString *)text;

/**
 *  Adds an icon to an input box that was initialized without one.
 *  @param icon The icon for the button.
 *
 */
- (void)setRightButtonIcon:(UIImage *)icon;

/**
 * Checks the validity of the input.
 */
- (BOOL)checkInputValidity;

/**
 * Filters the text to allow only numbers and limits the maximum length.
 * The parameters are the corresponding delegate parameters. maxLength is the maximum l
ength.
 */
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withMaxLe
ngth:(int)maxLength;

/**
 * Limits the maximum length.
 * @maxLength The maximum length, which does not include spaces in the format.
 */
- (BOOL)shouldChangeRange:(NSRange)range replacementString:(NSString *)string withForma
tStringMaxLength:(int)maxLength;
```

Code examples

- Standard input box:

```
  AUInputBox *inputBox = [AUInputBox inputboxWithOriginY:startY
inputboxType:AUInputBoxTypeNone];
  inputBox.titleLabel.text = @"Prompt text";
  inputBox.textField.placeholder = @"Enter text as prompted";
  [self.view addSubview:inputBox];
```

- Image button:

```
  AUInputBox *iconInputBox = [AUInputBox inputboxWithOriginY:startY buttonIcon:image
inputboxType:AUInputBoxTypeNone];
  iconInputBox.titleLabel.text = @"Prompt text";
  iconInputBox.textField.placeholder = @"Enter text as prompted";
  [self.view addSubview:iconInputBox];
```

1.3.3.6. 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.

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(NSInteger, AUSearchTitleStyle) {
        AUSearchTitleStyleDefault = 0,    // Search box with black text, which is
applicable when a light color background is used.
        AUSearchTitleStyleMiddleAlign,    // Search box with centered black text, which
is applicable when a light color background is used.
        AUSearchTitleStyleContent,        // Search box with white text, which is applic
able when a deep color background is used.
};

@class AUSearchTitleView;

@protocol AUSearchTitleViewDelegate <NSObject>
```

```
@optional

// The search bar control.
- (void)didPressedTitleView:(AUSearchTitleView *)titleView;

// The voice icon of the search bar control.
- (void)didPressedVoiceButton:(AUSearchTitleView *)titleView;

@end




/**
The search bar control. (The width is the same as that of the screen by default.)
*/
@interface AUSearchTitleView : UIView

@property(nonatomic, assign)AUSearchTitleStyle style;          // The background style o
f 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<AUSearchTitleViewDelegate> delegate;

@property(nonatomic,strong) UIImage *searchIconImage;          // The search icon.
@property(nonatomic,strong) UIColor *normalBackgroundColor;    // The background color o
f 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 va
lue 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 titleView of a
navigationItem, the system specifies the spacing between the titleView and the views on
its left and right in an adaptive manner. To meet the visual requirements, the system s
ets 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 *titleView = [[AUSearchTitleView alloc]
initWithSearchStyle:AUSearchTitleStyleDefault];
    titleView.placeHolder = @"The search bar style";
    titleView.placeHolderColor = [UIColor blackColor];
    titleView.normalBackgroundColor = [UIColor orangeColor];
    titleView.isShowVoiceIcon = YES;
    titleView.delegate = self;
    self.navigationItem.titleView = titleView;
```

- Used in a common view

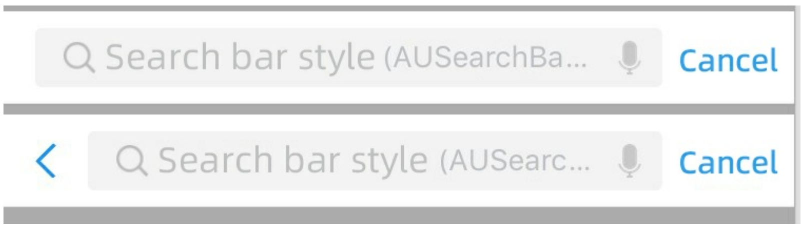
```
titleView = [[AUSearchTitleView alloc]
initWithSearchStyle:AUSearchTitleStyleMiddleAlign];
    titleView.placeHolder = @"The AUSearchTitleStyleMiddleAlign style";
    titleView.isShowVoiceIcon = YES;
    titleView.delegate = self;
    [self.view addSubview:titleView];
```

1.3.3.7. Search bar component

The dependency of AUSearchBar is as follows:

- AUSearchBar 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 repla
cementText:(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 YE
S).
```

```
- (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 t
he height is 44.)
*/
@interface AUSearchBar : UIView

@property (nonatomic, strong) NSString *text;                               // The search bo
x text.
@property (nonatomic, assign) BOOL isSupportHanziMode;                      // Whether to su
pport search while input. Default value: YES.
@property (nonatomic, assign) AUSearchBarStyle style;                       // The style of
the search box.
@property (nonatomic, assign) BOOL shouldShowVoiceButton;                   // Whether to di
splay the Voice button. Default value: NO.
@property (nonatomic, strong, readonly) UITextField *searchTextField;    // The search b
ox.
@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];
```

1.3.3.8. Verification code input box

AUTextCodeInputBox is a verification code input control.

API description

```
/**
The SMS verification code input box with a countdown timer.
*/
@interface AUTextCodeInputBox : AUSecurityCodeBox

/**
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.
*/
- (AUTextCodeInputBox *)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.
*/
- (AUTextCodeInputBox *)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

```
AUTextCodeInputBox *smsInputBox = [[AUTextCodeInputBox alloc] initWithOriginY:startY in
terval:60];
[smsInputBox.actionButton addTarget:self action:@selector(onSmsButtonClicked:) forContr
olEvents:UIControlEventTouchUpInside]; // The callback for processing the event that th
e button on the right is tapped.
[self.view addSubview:smsInputBox];
```

1.3.4. Item component

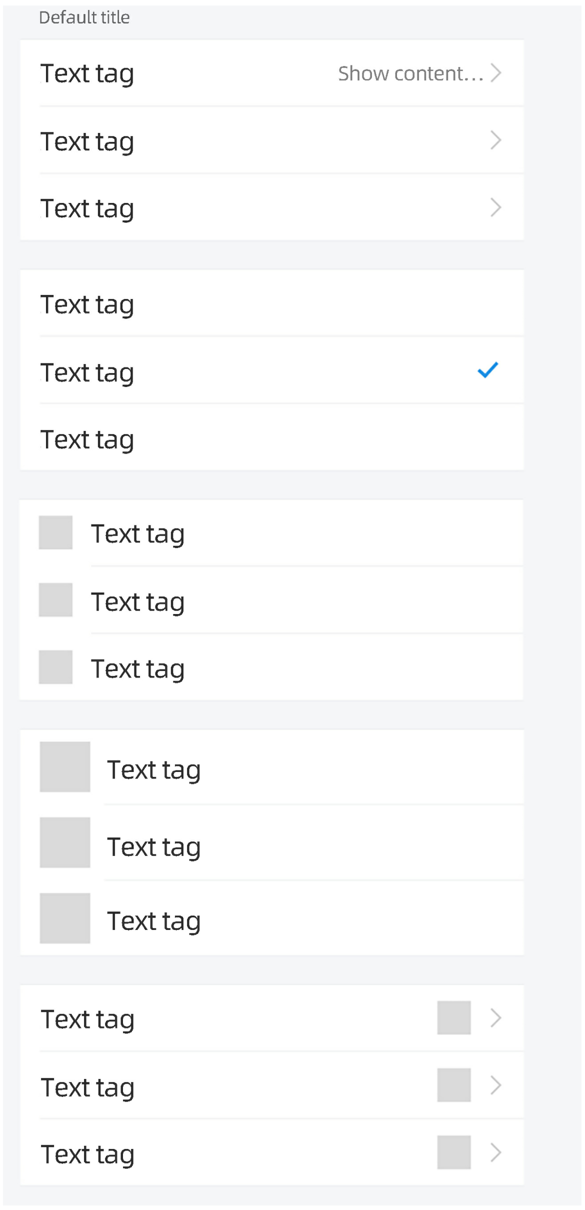
AUListItem is a series of controls designed based on the new UED requirements. It cannot be used interchangeably with the APTableView control in the original APCommonUI because most UED styles are different.

AUListItem contains four ListItems. The following table lists the elements supported by them respectively.

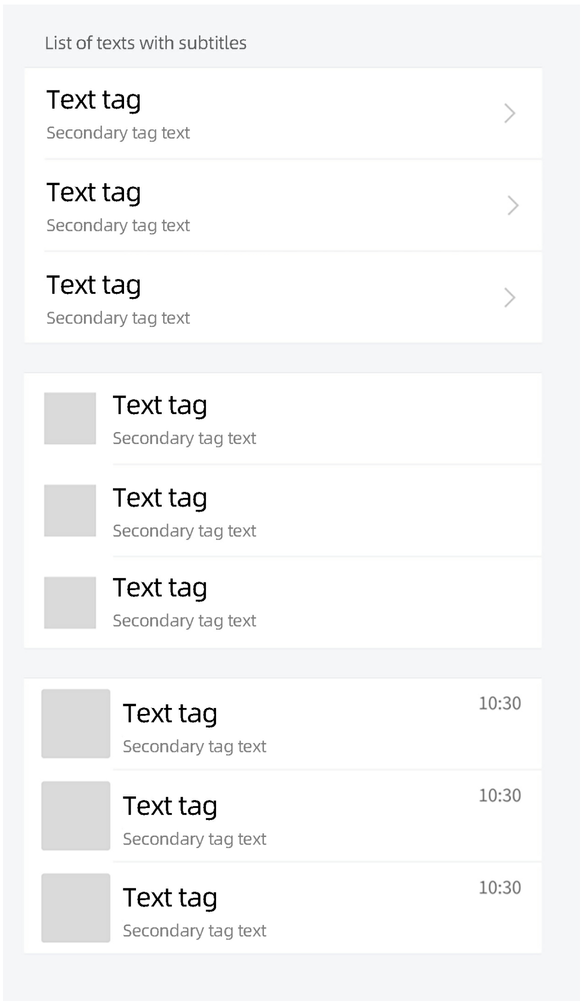
AUListItem	Title	Subtitle	Left icon	Right icon	Left icon in a customized size	Show a check mark when selected	Rightmost assistant arrow
AUSingleTitleListItem	YES	YES	YES	YES	YES	YES	YES
AUDoubleTitleListItem	YES	YES	YES	-	YES	-	YES
AUCheckBoxListItem	YES	-	-	-	-	-	YES
AUSwitchListItem	YES	-	-	-	-	-	-

Sample images

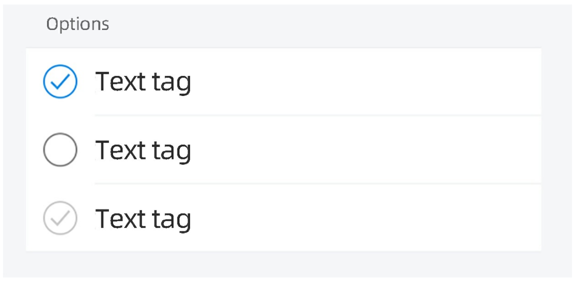
- AUSingleTitleListItem



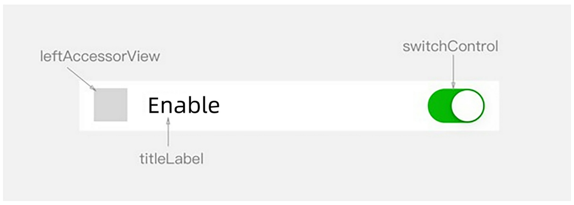
- AUDoubleTitleListItem



- AUCheckBoxListItem



- AUSwitchListItem



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 inform
ation 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 th
e 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 AUMultiListItemDelagate.

*/
@protocol AUMultiListItemDelagate <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
image.
@property (nonatomic, assign) BOOL    showAccessory;       // Whether to display the au
xiliary icon.
@property (nonatomic, assign) NSInteger subtitleLines;     // Set the number of subtitl
e lines.

@end


/**
Data items that can be set and accessed in AUMultiListBottomAssistDelagate.

*/
@protocol AUMultiListBottomAssistDelagate <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 AUCouponsItemDelagate.

*/
@protocol AUCouponsItemDelagate <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 descrip
tion.
@property (nonatomic, assign) NSInteger totalWidth;        // Set the width of the card
.

@end


/**
The rich text protocol of TTTAttributeLabelDelagate.

*/

@protocol TTTAttributeLabelDelagate <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 ric
h-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(^)(AUL
istItemModel*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 t
he 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 ot
her.
*/
@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(NSInteger, 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, T
TTAttributeLabelDelagate>

@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, TTTA
ttributeLabelDelagate>*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:(AUDoubl
eTitleListItemStyle)style;


/**
 Return a height aaccording 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)checkboxValueDidChanged:(AUCheckBox *)item;// Take the tag of the cell as the t
ag of the item.

@end

@interface AUCheckBoxListItem : AUBaseListItem<AUCheckBoxListItemModelDelegate>

@property(nonatomic, assign, getter = isChecked) BOOL checked;// Set the check box to t
he selected state.
@property(nonatomic, assign, getter = isDisableCheck) BOOL disableCheck;// Specify whet
her 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 cel
l.

// 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

Property	Purpose	Type
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 a
e 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 no
supported";
                                                            model.leftImage = [UIImage im
geNamed:@"AntUI.bundle/ilustration_ap_expection_limit.png"];
                                                            model.leftimageSize = CGSizeM
ke(100, 100);
                                                            model.showAccessory = YES;

                                                            // The supported properties a
e 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/ilustration_ap_expection_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/ilustration_ap_expection_limit.png"];
```

AUCheckBoxListItem

- The recommended usage is as follows:

```
AUCheckBoxListItem* cell = [[AUCheckBoxListItem alloc]
initWithReuseIdentifier:identifierChecbkox
model:^(AUListItemModel<AUCheckBoxListItemModelDelegate> *model) {
                                                            model.title = @"Selected by d
fault";
                                                            model.showAccessory = NO;
                                                            // Only the preceding two pro
erties 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(switchValueDidChanged:)
forControlEvents:UIControlEventValueChanged];
return switchCell;
```

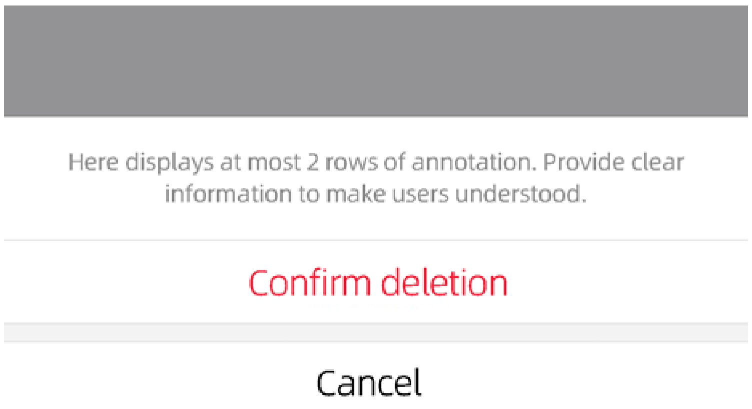
1.3.5. Pop-up window component

1.3.5.1. Action sheet

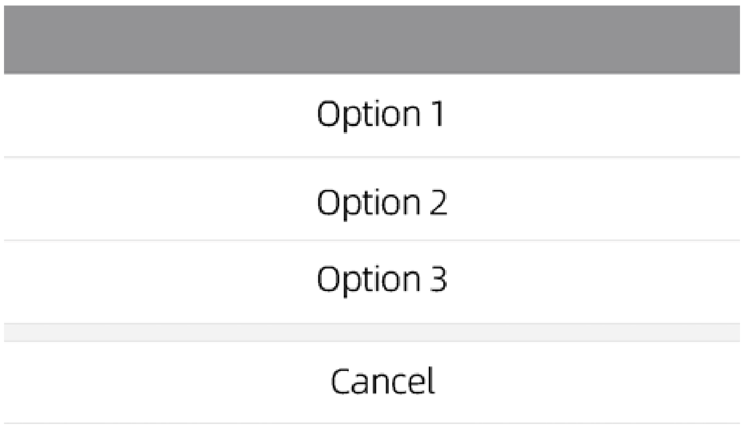
AUActionSheet is migrated from APActionSheet. The style is slightly adjusted, supporting the common layout with the deletion button and the common sheet layout.

Sample images

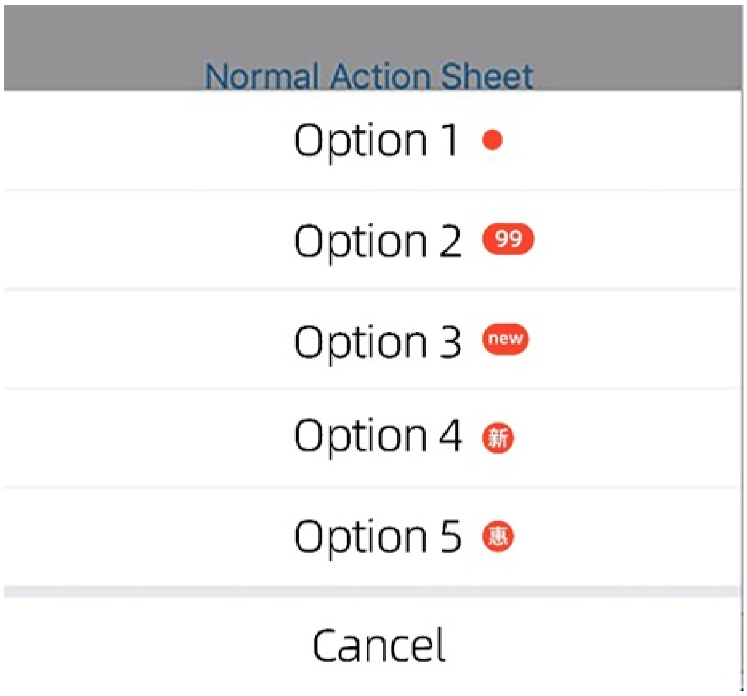
- Common layout with the delete button:


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(NSInteger, 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 APActionSheet, 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 des
tructiveButtonTitle:(NSString *)destructiveButtonTitle otherButtonTitles:(NSString *)ot
herButtonTitles, ... NS_REQUIRES_NIL_TERMINATION;

/**
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 items                    The customOption data list (with custom title colors and
badges).
@return                         The AUActionSheet instance.
*/
- (instancetype)initWithTitle:(NSString *)title
delegate:(id<UIActionSheetDelegate>)delegate
cancelButtonTitle:(NSString *)cancelButtonTitle
destructiveButtonTitle:(NSString *)destructiveButtonTitle
items:(NSArray<AUActionSheetItem *> *)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:(AUActionSheetButtonType)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)inde
- (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_AVAI
LABLE_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
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
more (...) 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
lines 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];
```

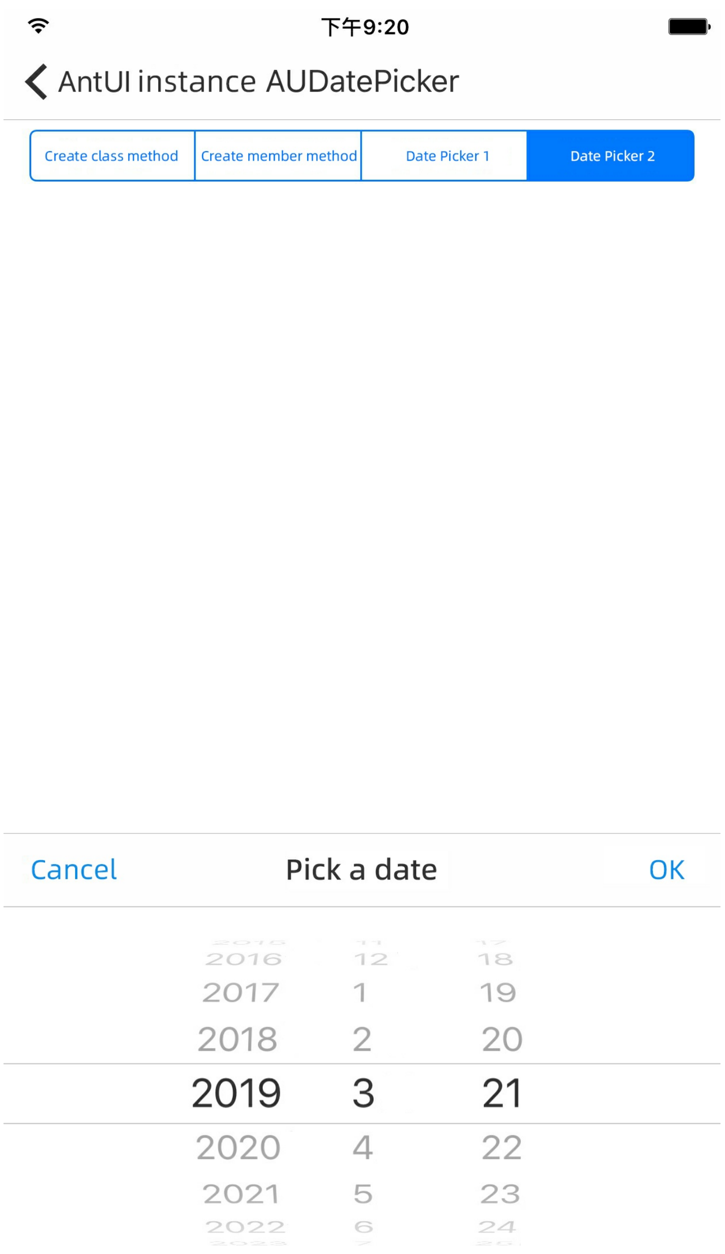
- Add a badge to an option:

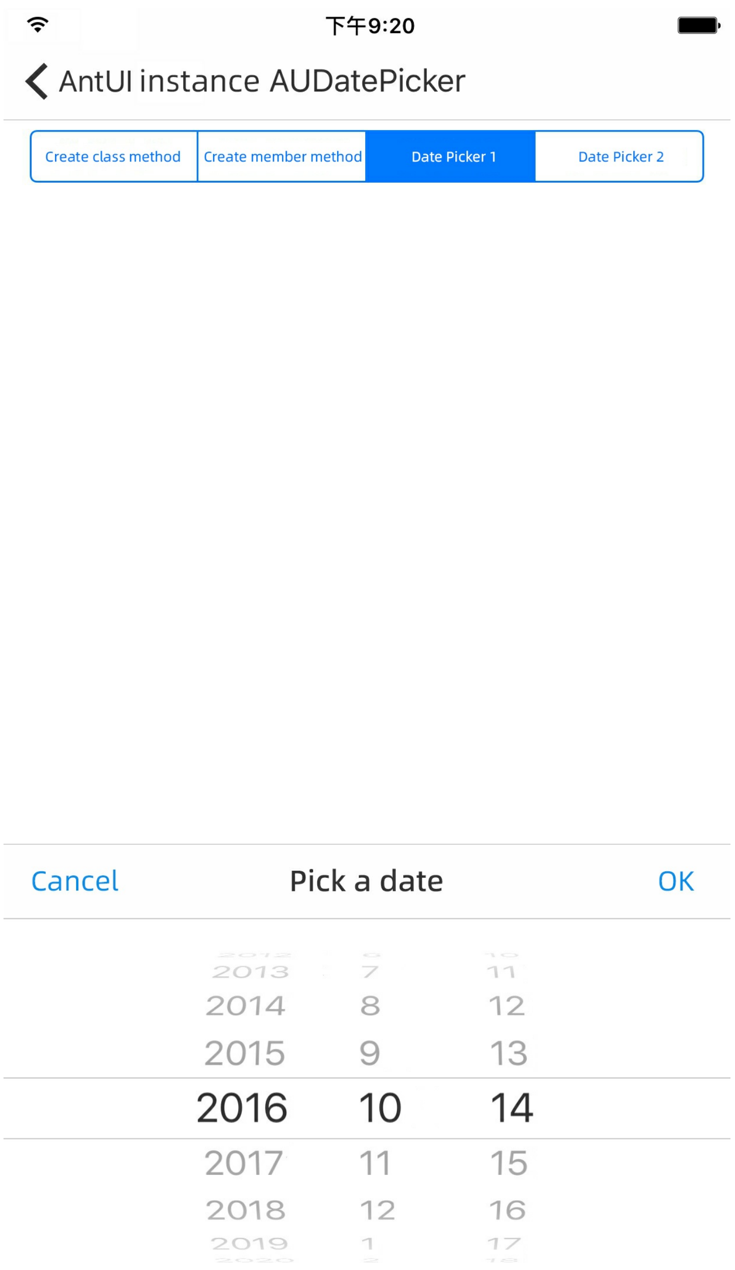
```
    AUActionSheet *actionSheet = [[AUActionSheet alloc] initWithTitle:nil
                                                               delegate:self
                                                      cancelButtonTitle:@"Cancel"
                                                 destructiveButtonTitle:nil
                                                      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];
```

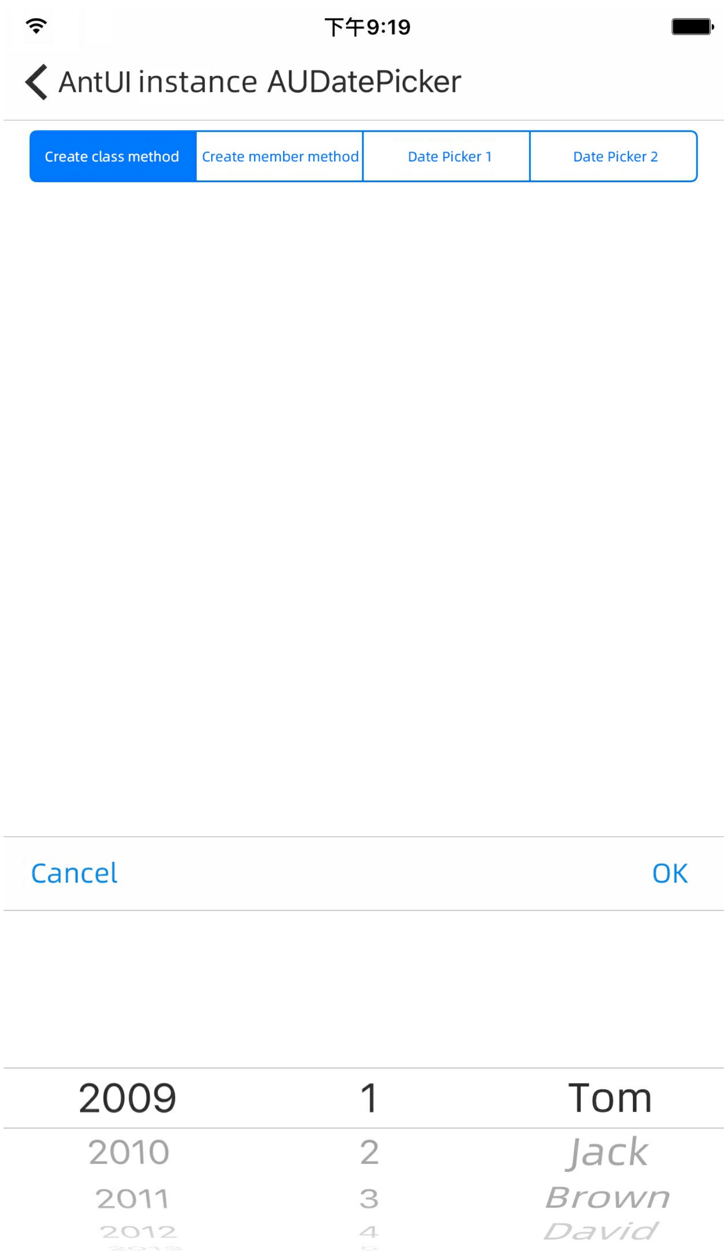
1.3.5.2. Date picker component

AUDatePicker is a date selection control.

Sample images


下午9:20
AntUI instance AUDatePicker
Create class method
Create member method
Date Picker 1
Date Picker 2
Cancel
Pick a date
OK
2016 12 18
2017 1 19
2018 2 20
2019 3 21
2020 4 22
2021 5 23
2022 6 24


下午9:20
AntUI instance AUDatePicker
Create class method
Create member method
Date Picker 1
Date Picker 2
Cancel
Pick a date
OK
2013 7 11
2014 8 12
2015 9 13
2016 10 14
2017 11 15
2018 12 16
2019 1 17



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 picke
r 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 defaut. call the show method if showing t
he 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)setAUDatePickerDate:(NSDate *)date animated:(BOOL)animated; // if animated is Y
ES, animate the wheels of time to display the new date

@end
```

Sample code

```
//
//  APPickerViewViewController.m
//  UIDemo
//

#import "APPickerViewViewController.h"
#import "AUDatePicker.h"

@interface APPickerViewViewController ()
<AUDatePickerDelegate,UIPickerViewDelegate,UIPickerViewDataSource>
@property(nonatomic,strong)AUDatePicker* apPickerView;
@property(nonatomic,strong)AUDatePicker* apPickerView2;
```

```
@property(nonatomic,strong)AUDatePicker* apPickerView3;
@property(nonatomic,strong)AUDatePicker* 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:UIControlEventValueChanged];
    segmentControl.selectedSegmentIndex = 0;
    [segmentControl setFrame:CGRectMake(15, 70, AUCommonUIGetScreenWidth() - 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,
(AUCommonUIGetScreenWidth()-self.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 = [AUDatePicker 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) wi
thTitle: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 * curretntDate = [NSDate date];
    NSDate * minxDate = [NSDate dateWithTimeInterval:-(3600*24*3000)
sinceDate:curretntDate];
    NSDate * maxDate = [NSDate dateWithTimeInterval:3600*24*3000
sinceDate:curretntDate];
    [self.apPickerView3 setTimeDateminDate:minxDate MaxDate:maxDate];
    [self.apPickerView3 setCurrentDate:curretntDate];
    [self.view addSubview:self.apPickerView3];

    // Time picker 2.
    self.apPickerView4 = [AUDatePicker pickerViewWithTitle:@"Select time"];
    self.apPickerView4.tag = 1003;
    self.apPickerView4.isDatePicker = YES;
    [self.apPickerView4 setTimeDateminDate:minxDate MaxDate:maxDate];
    [self.apPickerView4 setCurrentDate:curretntDate];
    NSDate * selectDate =[NSDate dateWithTimeInterval:3600*24*888
sinceDate:curretntDate];
    [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.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.m
onthArray objectAtIndex:index]]];

    index = [pickerView.pickerView selectedRowInComponent:2];
    result = [result stringByAppendingString:[NSString stringWithFormat:@"  %@",[self.n
ameArray 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
```

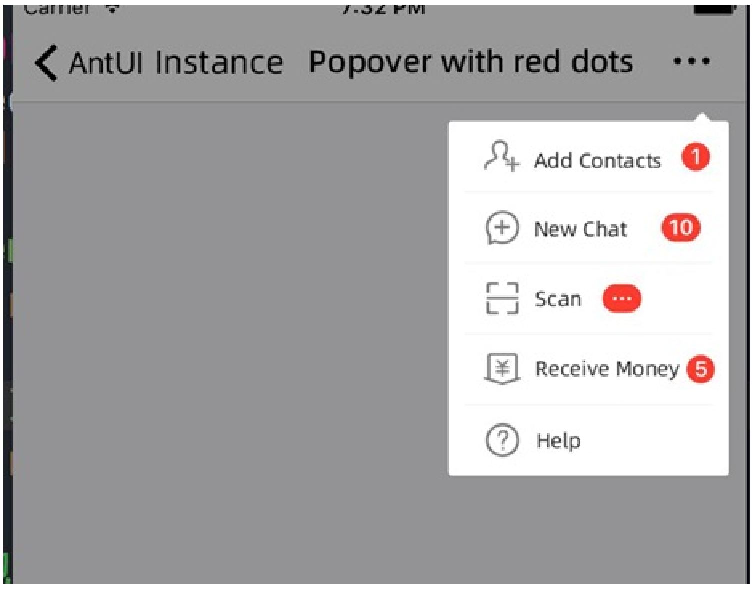
1.3.5.3. Menu component

The floating layer menu provides a menu containing icons and an option list.

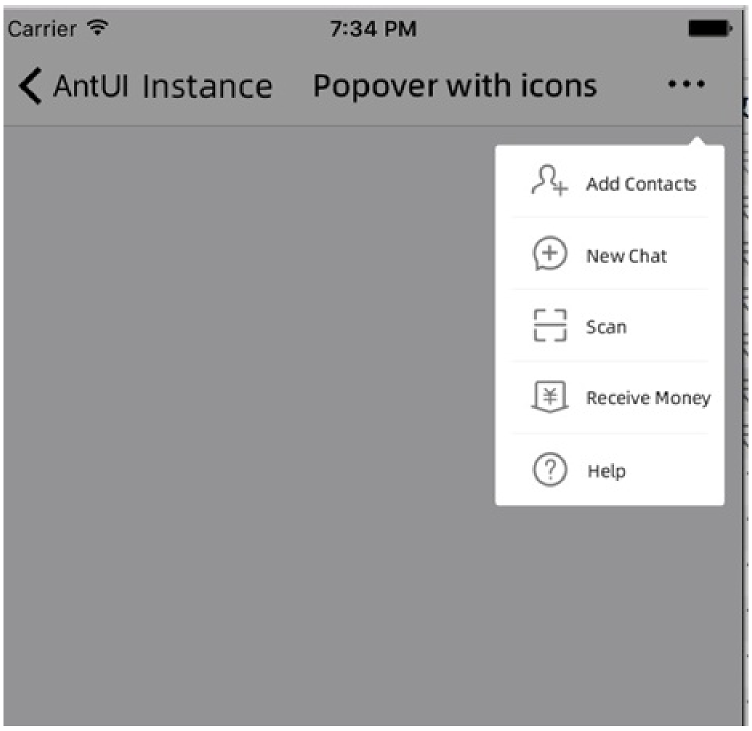
When using it, you need to change the APNavPopview and APNavItemView in the original AntUI frameWork to AUFloatMenu and AUNavItemView.

Sample images

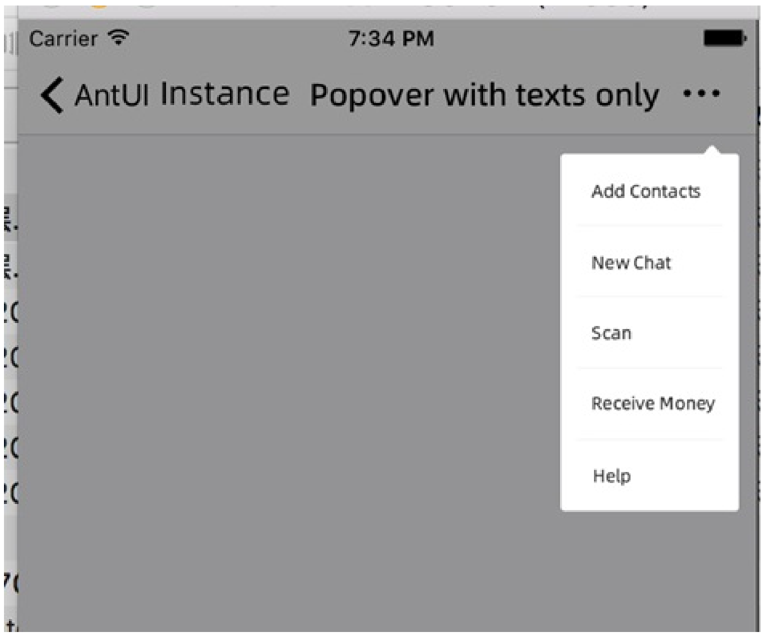
- Popover with red dots



- 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 APExtUIPopViewDissmissedNotification =
@"APExtUIPopViewDissmissedNotification";



@class AUNavItemView;
/*!
 @class        AUFloatMenu
 @abstract     UIView
 @ Discussion floatViewMenu The floating layer.
 */
@interface AUFloatMenu : UIView<UIGestureRecognizerDelegate>
@property(nonatomic, assign) CGFloat marginToRight;  // The right margin of a white pop
view, 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 APNavItemView obj
ect.
 *
 * @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 orignY The y-coordinate value of the floating menu on the screen.
 * @param items The array of displayed content, which is generally an APNavItemView obj
ect.
 *
 * @return The floating menu view.
 */
+(AUFloatMenu *)showAtPostion:(CGPoint)position startOrignY:(CGFloat)orignY items:(NSAr
ray<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. Af
ter 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)NSString *itemTitle;

@property(nonatomic,strong,readonly)UIFont *titleFont;

/**
 * The normal state.
 */
@property(nonatomic,strong)UIImage *nomarlStateIconImage;

/*
 * IconFont Name: If iconFont needs to be set, the AUNavItemView.h but not AUFloatMenu.
h API is invoked.
 */
@property(nonatomic,strong)NSString *nomarlStateIconFontName;

/**
 * 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 pr
ocessed.
 */
- (void)onClicked;

/**
 Return the size of the icon view.

 @return size
 */
- (CGSize) iconViewSize;
```

```
@end
```

Sample code

- Example of a popover with red dots:

```
//
//  APNavPopViewViewController.m
//  UIDemo
//

#import "APNavPopViewViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopViewViewController ()

@end

@implementation APNavPopViewViewController

- (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;
    [[AURegisterManager shareInstance] 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:

```
//
//  APNavPopViewViewController.m
//  UIDemo
//

#import "APNavPopViewNoneRedViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopViewNoneRedViewController ()

@end
```

```
@implementation APNavPopViewNoneRedViewController

- (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 targe
t:self action:@selector(onClick:)];
    self.navigationItem.rightBarButtonItem = rightItem;
    [[AURegisterManager shareInstance] 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:

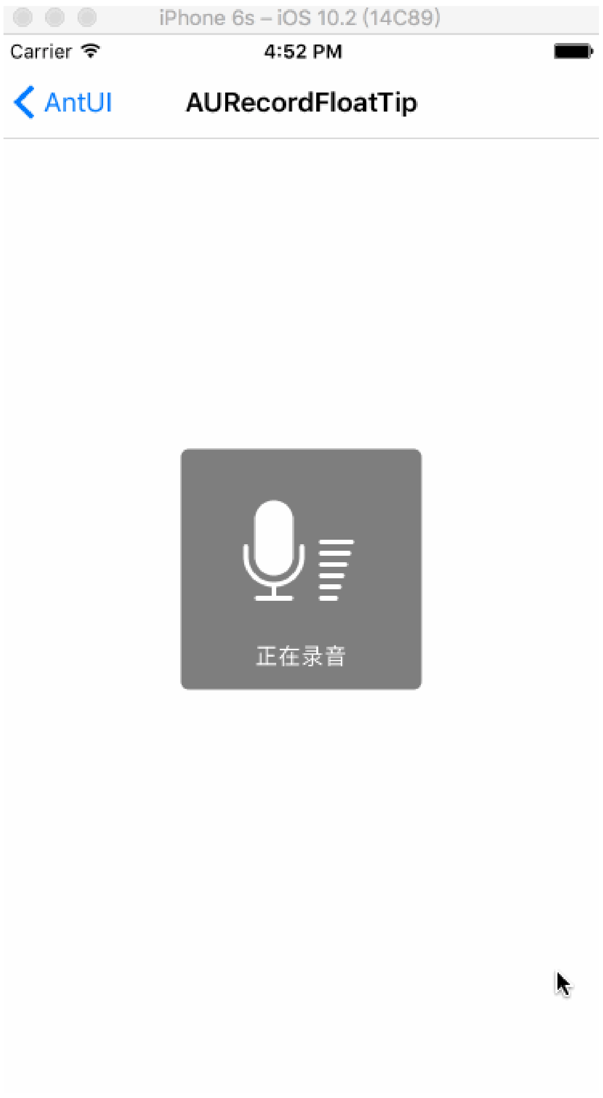
```
//
//  APNavPopViewViewController.m
//  UIDemo
//

#import "APNavPopViewOnlyViewController.h"
#import "AUUtils.h"
#import "AUNavItemView.h"
#import "AUFloatMenu.h"
#import "AntUIShellObject.h"
#import "AUIconView.h"

@interface APNavPopViewOnlyViewController ()

@end

@implementation APNavPopViewOnlyViewController

- (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 targe
t:self action:@selector(onClick:)];
    self.navigationItem.rightBarButtonItem = rightItem;
    [[AURegisterManager shareInstance] 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
```

1.3.5.4. 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)showRecodingInView:(UIView *)view;

// Hide the floating layer.
- (void)dismissRecordView;


@end
```

Code sample

```
AURecordFloatTip *_tipView = [[AURecordFloatTip alloc] init];
    [_tipView showRecodingInView:self.view];
```

1.3.5.5. Image dialog

AUImageDialog is a dialog box with images. The dialog box has rounded corners, and the style can be customized. The window level of AUImageDialog is: `self.windowLevel = UIWindowLevelAlert - 1`.

API description

```
// The index corresponding to button clicking.
    typedef NS_ENUM(NSInteger, AUImageDialogButtonIndex) {
            AUImageDialogButtonIndex_Close  = -2,
            AUImageDialogButtonIndex_Link   = -1,
            AUImageDialogButtonIndex_Action =  0
    };



    /**
     The image dialog box is a special-style dialog box meeting the UED requirements, t
he 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 cros
s 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 c
annot pass the assert check.
     */
    @interface AUImageDialog : AUDialogBaseView

    /**
     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 AUDialogDelegate-compliant protocol object.
     @return            The AUImageDialog instance.
     */
    - (instancetype)initWithImage:(UIImage *)image
                                                title:(NSString *)title
                                            message:(NSString *)message
                                        delegate:(id<AUDialogDelegate>)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 AUImageDialog instance.
     */
    - (instancetype)initWithImage:(UIImage *)image
                                                title:(NSString *)title
                                                message:(NSString *)message
                                            delegate:(id<AUDialogDelegate>)delegate
                                    buttonTitles:(NSString *)buttonTitle, ...
NS_REQUIRES_NIL_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 AUImageDialog 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 AUImageDialog 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 t
he 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 a
ction or a link).

     @param buttonTitle     The common button title.
     @param actionBlock     The callback of the button.
     */
    - (void)addButton:(NSString *)buttonTitle actionBlock:
(AUDialogActionBlock)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:
(AUDialogActionBlock)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:
(AUDialogActionBlock)actionBlock;

    /**
     Hide the close button in the upper right corner.
     */
    - (void)setCloseButtonHidden:(BOOL) hidden;
```

API for a large image style AUImageDialog

```
/**
 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 AUImageDialog (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 AUDialogDelegate-compliant protocol object.
 @return            The AUImageDialog instance.
 */
- (instancetype)initWithLargeImage:(UIImage *)image
                             title:(NSString *)title
                           message:(NSString *)message
                          delegate:(id<AUDialogDelegate>)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"];
  AUImageDialog *dialog = [[AUImageDialog alloc] initWithImage:image title:@"Goda Tak
eshi" 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 t
he candidate control set. A delegate, such as the title delegate, may not be implemen
ted as a single control." 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:CGSizeM
ake(100, 100)];
  AUImageDialog *dialog = [[AUImageDialog alloc] initWithLargeImage:image
title:@"Title in a line" message:@"Illustrate the status and prompt a solution. No mo
re than two lines." delegate:self];
  [dialog addButton:@"Cancel" actionBlock:nil];
  [dialog addButton:@"OK" actionBlock:nil];
  [dialog resetCloseIconColor:[UIColor redColor]];
  [dialog show];
```

1.3.5.6. 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`.

API description

```
@interface AUInputDialog : AUDialogBaseView

/// 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, t
he index 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 mes
sage 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 actio
n", 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];
```

1.3.5.7. 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.

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 f
unction 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 promp
t.
#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. Other
wise, 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 o
f the parent view in the x-axis.
@property (nonatomic, assign) CGFloat yOffset; // Set the offset to the central point o
f 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 (^)())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 (^)())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.
*/
+ (AUToast *)presentModelToastWithin:(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.
*/
+ (AUToast *)presentModalToastWithin:(UIView *)superview
withIcon:(AUToastIcon)icon
text:(NSString *)text
duration:(NSTimeInterval)duration
logTag:(NSString*)logTag
completion:(void (^)())completion;


/**
* Show the modal Toast.
* Different from the common Toast, the modal Toast has a transparent background layer,
 * 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.
*/
+ (AUToast *)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

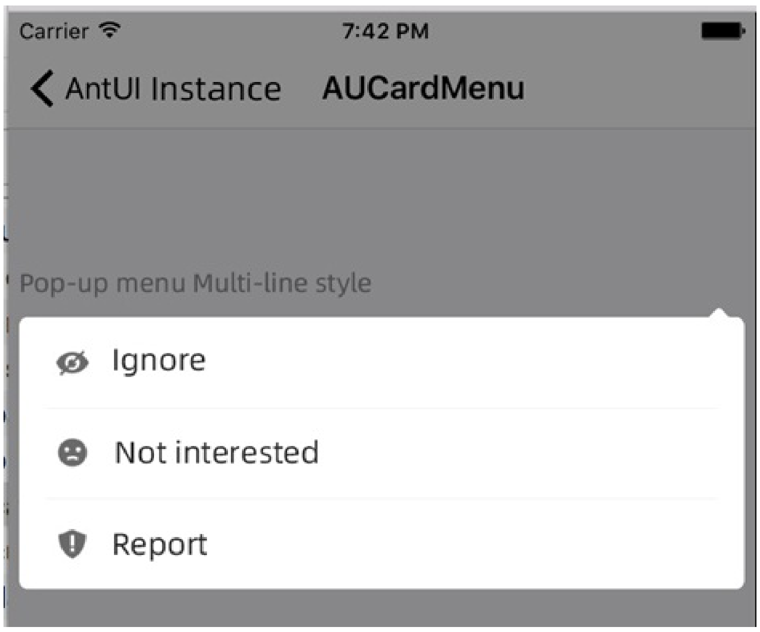
```
[AUToast presentToastWithin:self.view withIcon:AUToastIconNetFailure text:@"System Busy
" logTag:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconSuccess text:@"Success" logTa
g:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconFailure text:@"Failure" logTa
g:@"demo"];
[AUToast presentToastWithin:self.view withIcon:AUToastIconAlert text:@"Alarm"
logTag:@"demo"];

// Loading.
[AUToast presentToastWithin:self.view withIcon:AUToastIconLoading text:nil
logTag:@"demo"];

// Set the scenario where the progress appears.
AUToast *toast = [AUToast presentToastWithin:self.view withIcon:AUToastIconProgress tex
t:@"Loading" logTag:@"demo"];
toast.origin = point;
[toast setProgressPrefix:@"~~~"];
[toast setProgressText:0.5];

// The modal Toast.
[AUToast presentModalToastWithin:weakSelf.view withIcon:AUToastIconLoading text:@"Modal
toast,There are so many longest texts,is too long" duration:3 logTag:@"demo" completion
:NULL];
[AUToast presentModalToastWithin:weakSelf.view withIcon:AUToastIconLoading text:@"Modal
toast,There are so many longest texts,is too long" duration:3 delay:2  logTag:@"demo" c
ompletion:NULL];
```

1.3.5.8. Card menu

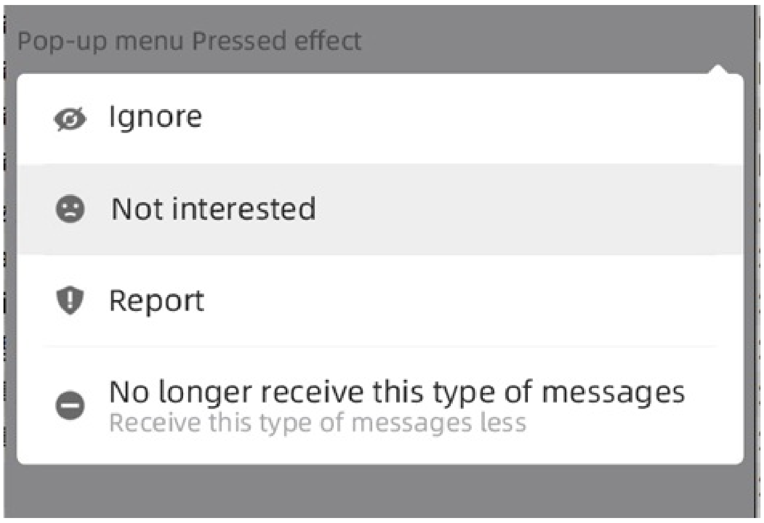
The card menu component is a pop-up menu that appears when a user clicks a card on the client page. In iOS, you can replace `beeviews:BEEPopMenuView` with `AUCardMenu.h` .

Previews

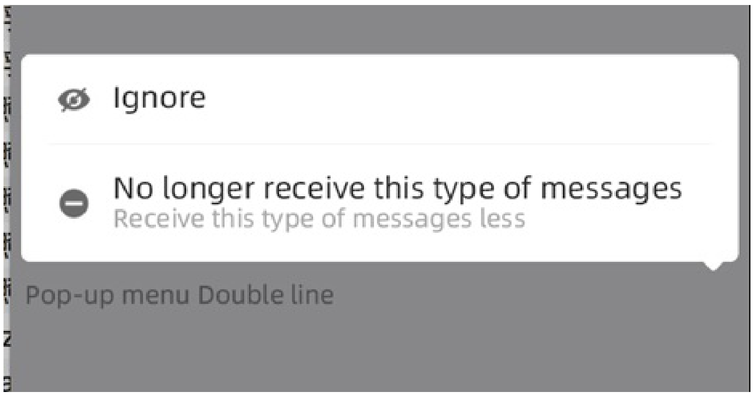
- Pop-up menu / Multi-line style



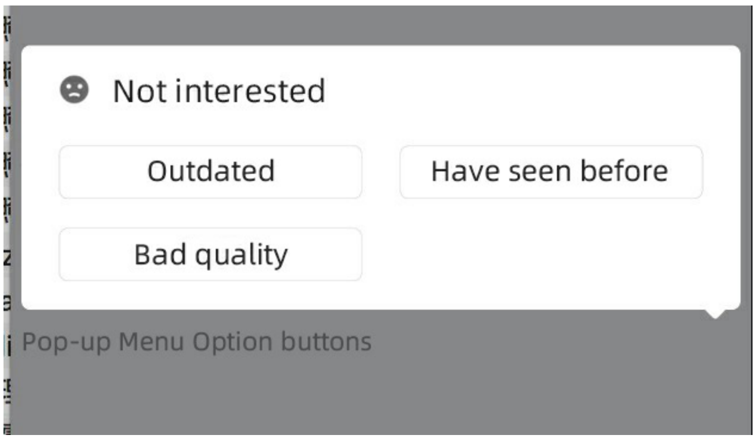
- Pop-up menu / Pressed effect


Pop-up menu Pressed effect
Ignore
Not interested
Report
No longer receive this type of messages
Receive this type of messages less

- Pop-up menu / Double line


Ignore
No longer receive this type of messages
Receive this type of messages less
Pop-up menu Double line

- Pop-up menu + Option buttons



API reference

- **AUCardMenu.h**

```
//
//  AUCardMenu.h
//  AntUI
//


@class AUMultiStyleCellView;
@class AUWindow;
/*!
 @class       AUCardMenu
 @abstract    AUWindow
 @discussion  A pop-up menu with a mask layer and a directional arrow.
*/

@interface AUCardMenu : AUWindow
{

}

/**
 *  To respond to click events, implement the AUMultiStyleCellDelegate protocol for c
ellView.
 *  In your view controller, assign popMenuView.cellView.delegate = self;
 */
@property (nonatomic, strong) AUMultiStyleCellView *cellView;

/**
 *

Initialization method (recommended).


 *
 *  @param data      An array that stores CellDataModel object models.
 *  @param location  The base point of the directional arrow.
 *  @param offset    The offset of the directional arrow relative to the base point.
 *
```

```
    @return self
 */

- (instancetype)initWithData:(NSArray *)data
                    location:(CGPoint)location
                      offset:(CGFloat)offset;

/**
 * Displays the pop-up menu.
 *
 *  @param superView  The superview of PopMenuView.
 */
- (void)showPopMenu:(UIView *)superView;

// Hides the pop-up menu. Also call this in the dealloc method.
- (void)hidePopMenu;

// Note: Animated show and hide methods must be used in pairs. If you show the menu with an animation, you must hide it with an animation. If you show it without an animation, you must hide it without one.

// Shows the menu with an animation.
- (void)showPopMenu:(UIView *)superView animation:(BOOL) isAnimation;

// Hides the menu with an animation.
- (void)hidePopMenuWithAnimation:(BOOL)isAnimation;

@end
```

- **AUCellDataModel.h**

```
//
//  AUCellDataModel.h
//  AntUI
//

#import <Foundation/Foundation.h>

/*!
 @class       AUMultiStyleCellView
 @abstract    UIView
 @discussion  A subview in the menu.
 */

@interface AUCellDataModel : NSObject

@property (nonatomic, strong) NSString *iconUrl;
@property (nonatomic, strong) NSString *titleText;
@property (nonatomic, strong) NSString *descText;
@property (nonatomic, strong) NSString *checkMarkUrl;  // Check mark
@property (nonatomic, strong) NSString *indicatorUrl;  // Right indicator arrow
@property (nonatomic, strong) NSArray *buttonsArray;   // NSArray<NSString>
@property (nonatomic, strong) NSDictionary *extendDic; // Extension field for busines
s use.
@property (nonatomic, assign) BOOL selectedState;      // The selected state of the c
urrent model. The default value is NO, which indicates that the model is not selected
.

@end
```

- **AUMultiStyleCellView.h**

```
//
//  AUMultiStyleCellView.h
//  AntUI
//

#import <UIKit/UIKit.h>
#import "AUCellDataModel.h"

@class AUMultiStyleCellView;
@protocol AUMultiStyleCellDelegate <NSObject>

@optional
/**
 *  Click event callback.
 *
 *  @param dataModel The data model that corresponds 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


/**
 *  A cell view that combines multiple styles.
 *  1. Icon + Main title
 *  2. Icon + Main title + Subtitle below the main title
 *  3. Icon + Main title + Multi-row, multi-column button controls with borders
 */

@interface AUMultiStyleCellView : UIView

@property (nonatomic, weak) id<AUMultiStyleCellDelegate> delegate;
@property (nonatomic, strong) NSArray *cellDataArray;

// If cellDataArray is empty, this is the same as calling the initWithFrame method.
- (instancetype)initWithFrame:(CGRect)frame
                cellDataArray:(NSArray *)cellDataArray
                     isUpward:(BOOL)isUpward;

// Handles the selected state of the current cellView.
- (void)updateSelectedState;


@end
```

Code example

```
//
```

```
//  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 buttonWithType:UIButtonTypeCustom];
    [button setFrame:CGRectMake(0, 100, self.view.width, 100)];
    [button setTitle:@"Pop-up menu/Multi-line combination 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 buttonWithType:UIButtonTypeCustom];
    [button2 setFrame:CGRectMake(0, 220, self.view.width, 100)];
    [button2 setTitle:@"Pop-up menu/Press 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 buttonWithType:UIButtonTypeCustom];
    [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 buttonWithType:UIButtonTypeCustom];
    [button4 setFrame:CGRectMake(0, 420, self.view.width, 100)];
    [button4 setTitle:@"Pop-up menu + Selection 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 = @"I'm not interested";
    model.buttonsArray = @[@"Outdated",@"Already seen",@"Poor quality"];
    model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard
@""};


    AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model]
location:CGPointMake(button.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopMenu: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 = @"Ignore";
    //    model.buttonsArray = @[@"Hello",@"Do you stutter?",@"I'm not hungry",@"How ar
e you?",@"I'm fine"];
    model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard
@""};

    AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
    model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
    model4.titleText = @"Stop receiving this type of message";
    model4.descText = @"Receive fewer messages of this type";
    model4.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"CCard
@""};
    AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model,model4]
location:CGPointMake(button.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopMenu: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",@"How ar
e you?",@"I'm fine"];
    model.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"CCard
@""};
    AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
    model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
    model2.titleText = @"I'm not interested";
    model2.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"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":@"2016092615150327202000000911282916069500009026 88",@"CCard
@""};
    AUCellDataModel * model4 = [[AUCellDataModel alloc] init];
    model4.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_reject.png";
    model4.titleText = @"Stop receiving this type of message";
    model4.descText = @"Receive fewer messages of this type";
    model4.extendDic =
@{@"type":@"reject",@"cardId":@"2016092615150327202000000911282916069500009026 88",@"CCard
@""};
    AUCardMenu *tmpView=[[AUCardMenu alloc]initWithData:@[model,model2,model3,model4] l
ocation:CGPointMake(button.width - 20, button.centerY) offset:13];
```

```
ocation:CGPointMake(button.width - 20, button.centerY) offset:13];
    tmpView.cellView.delegate=self;
    [tmpView showPopMenu: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",@"How are yo
u?",@"I'm fine"];
    model.extendDic =
@{@"type":@"reject",@"cardId":@"20160926151503272020000009112829160695000090268 8",@"CCard
@""};
    AUCellDataModel * model2 = [[AUCellDataModel alloc] init];
    model2.iconUrl = @"APCommonUI_ForDemo.bundle/hc_popmenu_dislike.png";
    model2.titleText = @"I'm 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 showPopMenu:button animation:YES];
    self.popMenuView=tmpView;


}


- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

- (void)hidePopMenu
{
    if (self.popMenuView) {
        [self.popMenuView hidePopMenuWithAnimation:YES];
        self.popMenuView.cellView.delegate = nil;
        self.popMenuView = nil;

    }
}


#pragma mark --- AUMultiStyleCellDelegate
```

```
/**
 *  Click event callback.
 *
 *  @param dataModel The data model that corresponds 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
{
    [self hidePopMenu];
}
- (void)DidClickCellButton:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath
*)indexPath
{
    [self hidePopMenu];
}
- (void)DidClickCellView:(AUCellDataModel *)dataModel ForRowAtIndexpath:(NSIndexPath *)
indexPath cellView:(AUMultiStyleCellView *)cellView
{
    [self hidePopMenu];
}


- (void)dealloc
{
    self.popMenuView = nil;

}

/*
#pragma mark - Navigation

// In a storyboard-based application, you will often want to do a little preparation be
fore 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
```

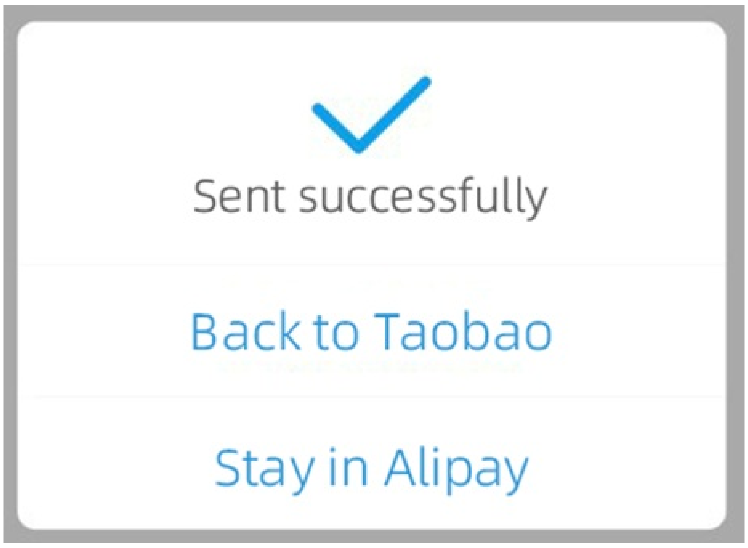
1.3.5.9. Operation result dialog

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 : AUDialogBaseView

/**
 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 AUDialogDelegate-compliant protocol object.
 @return            The AUImageDialog instance.
 */
- (instancetype)initWithImage:(UIImage *)image
                      message:(NSString *)message
                     delegate:(id<AUDialogDelegate>)delegate;

/**
 The method of dialog box initialization with the button title.
 @param image       The image.
 @param describe    The message details.
 @param delegate    The AUDialogDelegate-compliant protocol object.
 @param buttonTitle The list of button title parameters.
 @return            The AUImageDialog instance.
 */
- (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 AUImageDialog 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 actio
n.

 @param buttonTitle The common button title.
 @param actionBlock The callback of the button.
 */
- (void)addButton:(NSString *)buttonTitle actionBlock:(AUDialogActionBlock)actionBlock;

@end
```

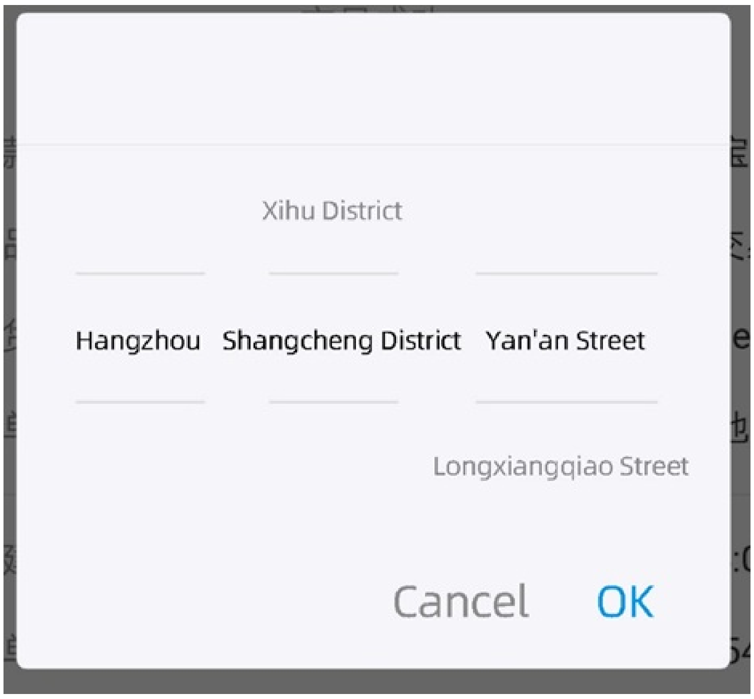
Sample code

```
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];
```

1.3.5.10. Cascade picker

AUCascadePicker is a multi-level cascade picker control that supports up to three levels.

Preview



API reference

```
// Sets the selected item for the picker.
@interface AUCascadePickerSelectedRowItem : NSObject

@property (nonatomic, strong) NSString *selectedLeftTitle;    // The title of the selec
ted item in the first sublist.
@property (nonatomic, strong) NSString *selectedMiddleTitle;  // The title of the selec
ted item in the second sublist.
@property (nonatomic, strong) NSString *selectedRightTitle;   // The title of the selec
ted item in the third sublist.

@end

@interface  AUCascadePickerRowItemModel : NSObject
```

```
@property (nonatomic, strong) NSString *rowTitle;
@property (nonatomic, strong) NSArray<AUCascadePickerRowItemModel *> *rowSubList;

@end

// The data model required for the filter interaction.
@interface AUCascadePickerModel : NSObject

@property (nonatomic,strong) AUCascadePickerSelectedRowItem            *preSelected;   /
/ The pre-selected item.
@property (nonatomic, strong) AUCascadePickerSelectedRowItem           *selectedItem;
// The list of selected data recorded within 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

// Callback for the "Cancel" & "Done" buttons at the top.
@protocol AUCascadePickerDelegate <AUPickerBaseViewDelegate>
/*
 * Callback for when Cancel is tapped.
 */
- (void)cancelPickerView:(AUCustomDatePicker *)pickerView;

/*
 * Callback for when Done is tapped. The selected item can be returned through selected
RowInComponent.
 */
- (void)selectedPickerView:(AUCustomDatePicker *)pickerView

@end
```

JSAPI reference

API: antUIGetCascadePicker

Example

```
AlipayJSBridge.call('antUIGetCascadePicker',
{
    title: 'nihao',// The title of the cascade picker.
    selectedList:[{"name":"Hangzhou",subList:[{"name":"Shangcheng District"}]}],
    list: [
        {
            name: "Hangzhou",// The item name.
            subList: [
                {
                    name: "Xihu District",
                    subList: [
                        {
                            name: "Gucui Street"
                        },
                        {
                            name: "Wenxin Street"
                        }
                    ]
                },
                {
                    name: "Shangcheng District",
                    subList: [
                        {
                            name: "Yanan Street"
                        },
                        {
                            name: "Longxiangqiao Street"
                        }
                    ]
                }
            ]// The cascaded sublist of data.
        }
    ]// The cascaded data list.
},
function(result){
    console.log(result);
});
```

Input parameters

Property	Type	Description	Required	Version
title	String	The title of the cascade control.	No	10.1.2

selectedList	Json	The selected state. Specifies the selected sub-item. The format is the same as the request parameter format. Example: `[{"name":"Hangzhou",subList:[{"name":"Shangcheng District"}]}]` .	No	10.1.2
list	Json	The data list for the picker.	Yes	10.1.2
name	String	The name of an item in the list.	Yes	10.1.2
subList	Json	The sub-item list. This is a sublist within the list.	No	10.1.2
fn	Function	The callback function that is executed after a selection is made.	No	10.1.2

Output parameters

Property	Type	Description	Version
success	bool	Indicates whether the selection was completed. Returns false if the operation was canceled.	10.1.2
result	Json	The selection result. Example: `[{"name":"Hangzhou",subList:[{"name":"Shangcheng District"}]}]` .	10.1.2

Code example

```
    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:@"Layer 1: %d", i];
        NSMutableArray *array = [[NSMutableArray alloc] init];
        for (int j=0; j<7; j++)
        {
            if (i == 0)
            {
                break;
            }
            AULinkagePickerRowItemModel *item1 = [[AULinkagePickerRowItemModel alloc] i
nit];
            item1.rowTitle = [NSString stringWithFormat:@"Layer 2: %d", j];
            NSMutableArray *array1 = [[NSMutableArray alloc] init];
            for (int k=0; k<5; k++) {
                AULinkagePickerRowItemModel *item2 = [[AULinkagePickerRowItemModel alloc
] init];
                item2.rowTitle = [NSString stringWithFormat:@"Layer 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;


    AULinkagePickerSelectedRowItem *item = [[AULinkagePickerSelectedRowItem alloc] init
];
    item.selectedLeftTitle = @"Layer 1: 0";
    item.selectedMiddleTitle = @"Layer 2: 0";
    item.selectedRightTitle = @"Layer 3: 0";

    model.selectedItem = item;

self.linkagePickerView = [[AULinkagePickerView alloc] initWithPickerModel:model];
self.linkagePickerView.linkageDelegate = self;
[self.linkagePickerView show];
```

1.3.5.11. 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`.

API description

```
/**
The common dialog box style. It is the same as the system style but does not contain th
e 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:(AUDialogActionBlock)actionBlock;
```

```
/**
The method of closing the dialog box. It is similar to the
dismissWithClickedButtonIndex method of APAlertView.
*/
- (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:@"Me
ssage 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:@"Me
ssage details" delegate:delegate buttonTitles:@"OK", nil];
  [dialog show];

  The delegate protocol is AUDialogDelegate, similar to UIAlertViewDelegate.
```

- Create a dialog box in a simple way.

```
 NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitle(NSString *title)
  NS_INLINE AUNoticeDialog *AUNoticeDialogWithTitleAndMessage(NSString *title, NSStri
ng *message)
```

Use APAlertView and UIAlertView

This section describes how to modify APAlertView and UIAlertView to AUNoticeDialog.

Most APIs of AUNoticeDialog support APAlertView 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 APAlertView to create a dialog box.

```
  - (instancetype)initWithTitle:(NSString *)title
                      message:(NSString *)message
                     delegate:(id<AUDialogDelegate>)delegate
            cancelButtonTitle:(NSString *)cancelButtonTitle
            otherButtonTitles:(NSString *)otherButtonTitles, ...
NS_REQUIRES_NIL_TERMINATION;
```

Change the class name from `[[APAlertView 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 APAlertView 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 `removeAllAlerviews` method to `dismissAll`.

    For example, change `[APAlertView removeAllAlerviews]` to `[AUNoticeDialog dismissAll]`.
- To use the input box feature of APAlertView or UIAlertView, replace the corresponding methods with those of AUInputDialog. The operations are the same as those of AUNoticeDialog.

> **Note**
>
> The class file is AUInputDialog.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" m
essage:@"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 c
amel-case format. Note: Some controls may be implemented in one platform but not in t
he 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];
```

1.3.5.12. Custom date picker

ACustomDatePicker is a custom date selection control and currently supports the following modes:

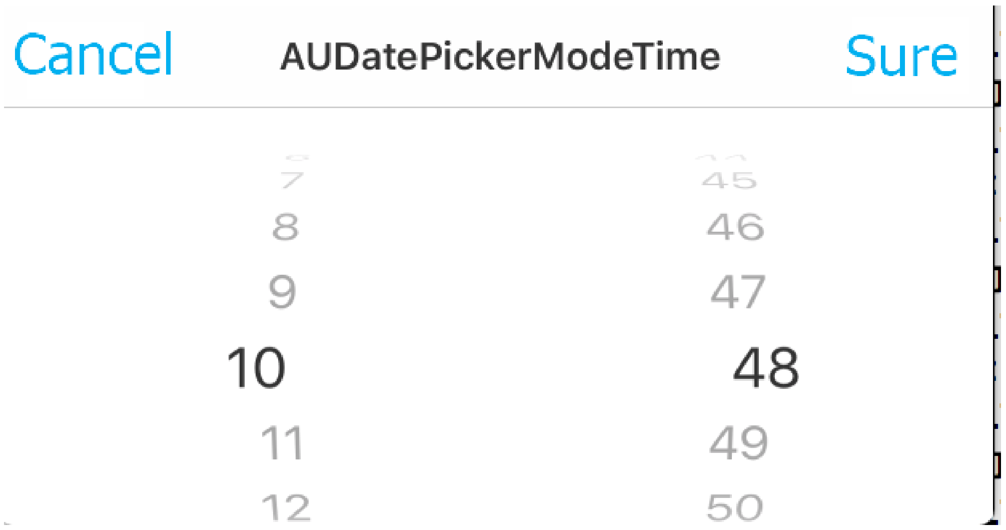
- `AUDatePickerModeTime` : hour/minute, 24-hour clock
- `AUDatePickerModeDate` : year/month/day
- `AUDatePickerModeDateAndTime` : month/day/day of week/hour/minute, 24-hour clock

> **Note**
>
> The year is defined based on minimumDate and is 2000 (leap year) by default. Therefore, February 29 exists.

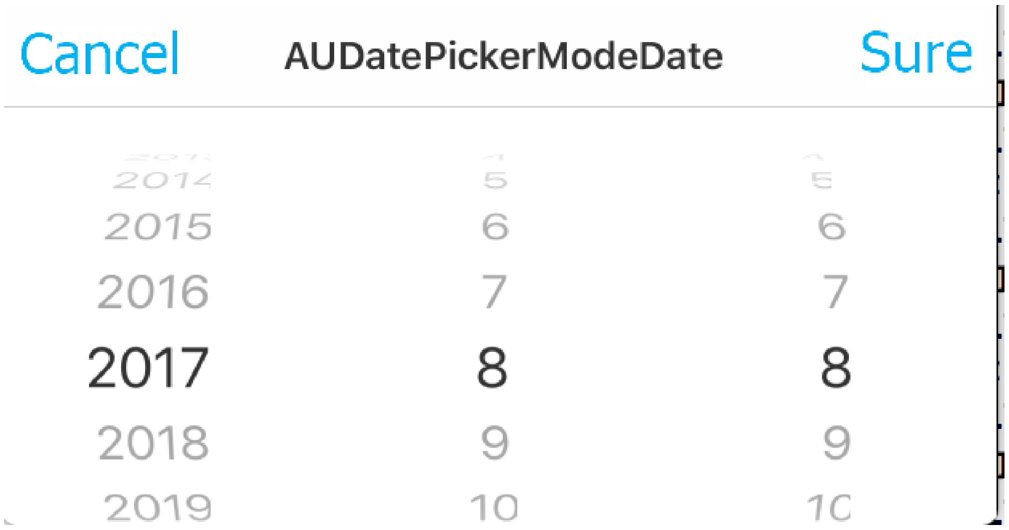
- `AUDatePickerYear` : year
- `AUDatePickerYearMonth` : year/month

Sample images

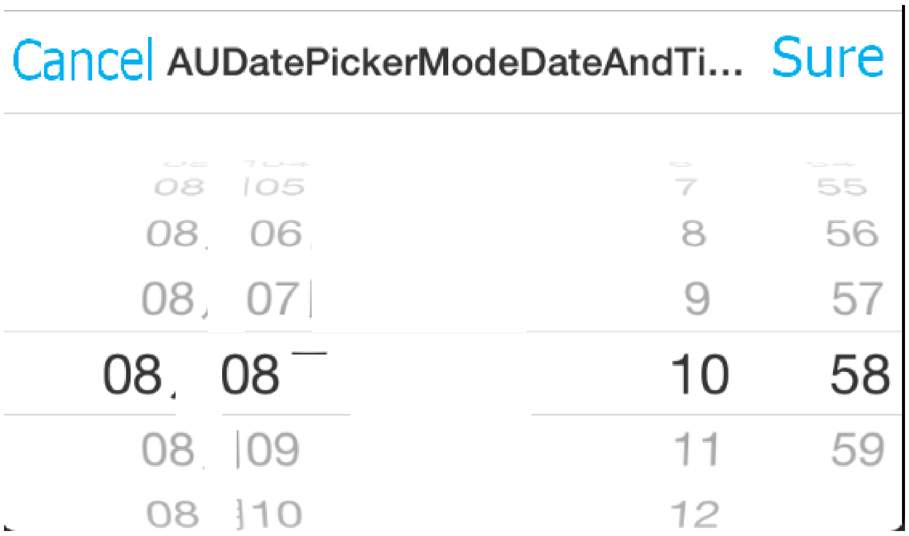
- AUDatePickerModeTime



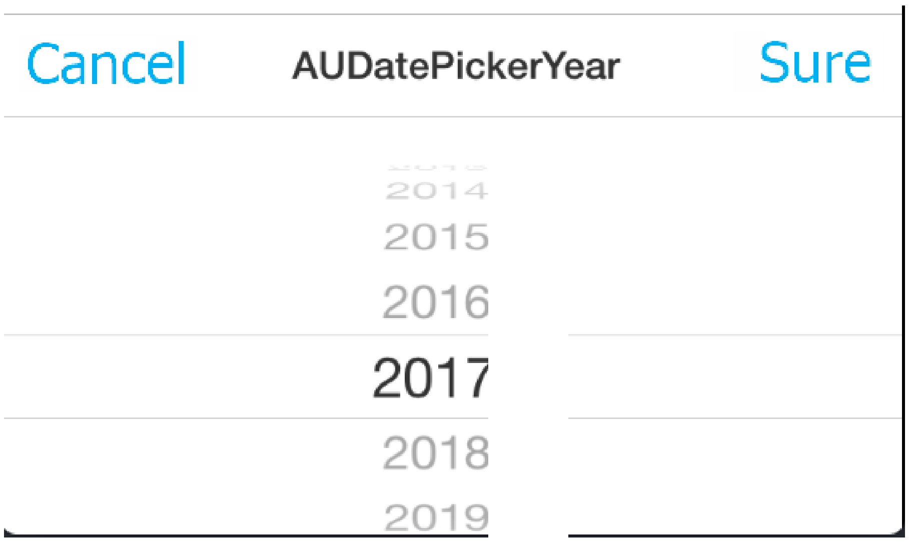
- AUDatePickerModeDate



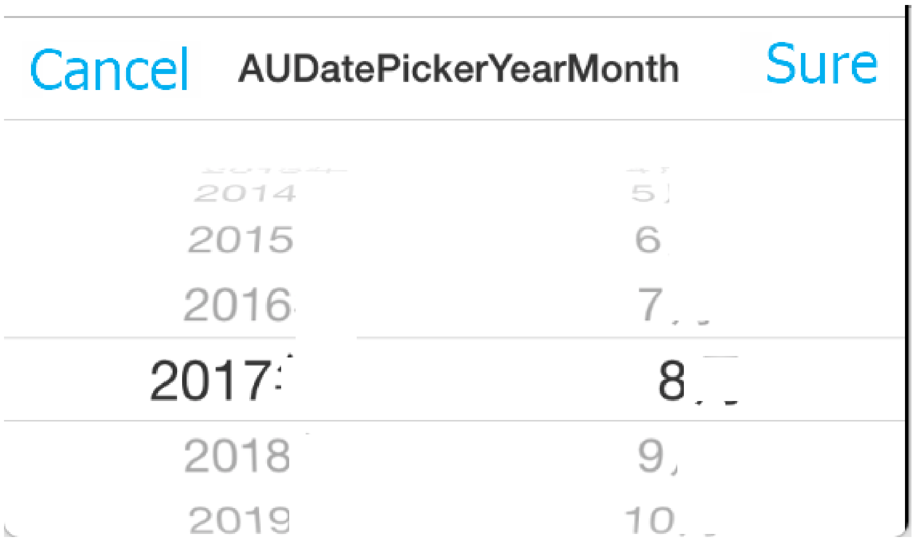
- AUDatePickerModeDateAndTime



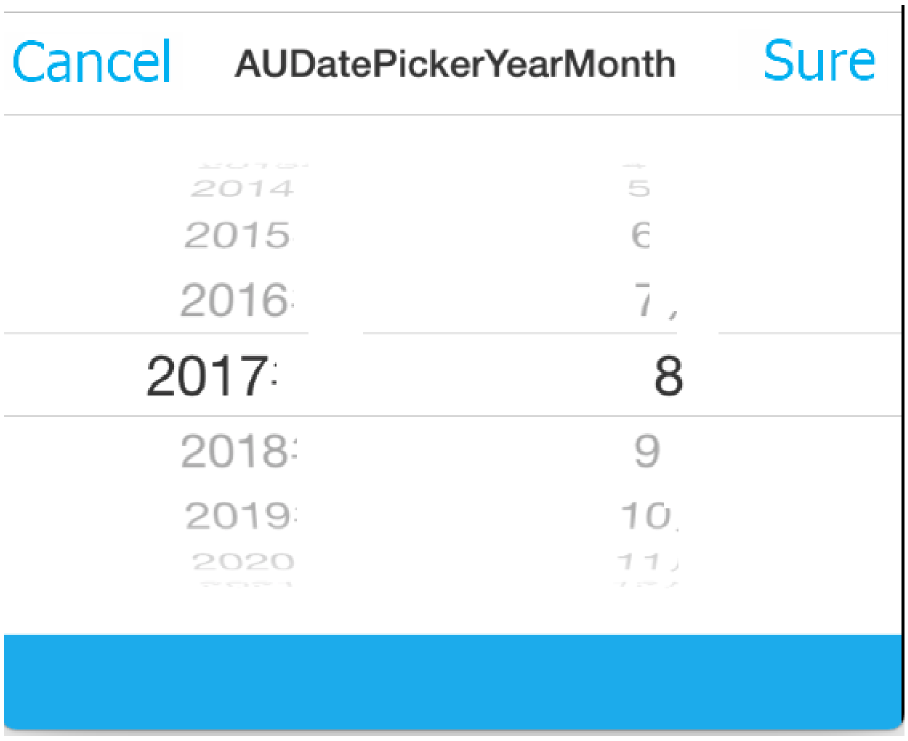
- AUDatePickerYear



- AUDatePickerYearMonth



- With a custom bottom view



API description

AUCustomDatePicker.h

```
// The custom bottom view.
@property (nonatomic,strong) UIView *bottomView;


/**
 * Create a picker, in AUDatePickerModeDate mode by default.
 *
 */
+ (AUCustomDatePicker *)pickerViewWithTitle:(NSString *)title;

+ (AUCustomDatePicker *)pickerViewWithTitle:(NSString *)title pickerMode:
(AUCustomDatePickerMode)mode;


/**
 * Set an available date range.
 @param minDate The earliest time (included), which is 00:00:00 on January 1, 2000 by d
efault.
 @param maxDate The latest time (included), which is 23:59:59 on December 31, 2050 by d
efault.
 */
- (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, AUCommonUI
GetScreenWidth(), 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];

  }
```

1.3.6. Loading components

1.3.6.1. Pull-up refresh control

The Pull-up refresh component (AUDragLoadingView) and the pull-down refresh component (AUPullLoadingView) provide loading styles when pulling up or down on a page.

The following controls must be replaced with the pull-up (AUDragLoadingView) or pull-down (AUPullLoadingView) component.

CommonUI includes four components: ODRefreshControl, APCircleRefreshControl, EGORefreshTableHeaderView, and APNextPagePullView.

API reference

- **AUDragLoadingView.h**

```
//
//  AUDragLoadingView.h
//  AntUI
//

#import <AntUI/AntUI.h>

@interface AUDragLoadingView : AUPullLoadingView

@end
```

- **AUPullLoadingView.h**

  For more information, see Pull-down refresh component.

Code example

```
//
//  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];
    // Do any additional setup after loading the view.
    self.edgesForExtendedLayout = UIRectEdgeNone;
//    self.navigationItem.rightBarButtonItem = [APUtil
getBarButtonWithTitle:RightBarButtonTitle target:self];

    self.tableView = [[UITableView alloc]initWithFrame:self.view.bounds
style:UITableViewStylePlain];
    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:@"Release"];
        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:(NSIndexP
ath *)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
```

1.3.6.2. Pull-down refresh component

The pull-up refresh component (AUDragLoadingView) and the pull-down refresh component (AUPullLoadingView) provide loading styles when pulling up or down on a page.

You can replace all non-customized controls with the pull-down (AUPullLoadingView) or pull-up (AUDragLoadingView) components.

CommonUI includes four components: ODRefreshControl, APCircleRefreshControl, EGORefreshTableHeaderView, and APNextPagePullView.

API reference

- **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       AURefreshLoadingView
   @abstract    UIView
   @discussion  Migrated from the third-party EGORefreshTableHeaderView. A view that
supports pull-down-to-refresh and pull-up-to-load-more features.
   */

  @interface AUPullLoadingView : UIView {

      __weak id _delegate;
      AUEGOPullRefreshState _state;

      UILabel *_lastUpdatedLabel;
      UILabel *_statusLabel;
  //    APActivityIndicatorView *_activityView;
      AUEGOPullDirection _pullDirection;

      BOOL    isAutoPullFlag;
  }
  @property(nonatomic, strong) AULoadingIndicatorView *activityView;

  /**
   *  Sets the initial status text for pull-up-to-load. The default text is "Pull up
to load more". This text is displayed by default.
   *
   *  @param tip The content of the tip.
   *
   */
  - (void)setPullUp:(NSString *)tip;

  /**
   *  Sets the initial status text for pull-down-to-refresh. The default text is "Pul
l down to refresh". This text is not displayed by default.
   *
   *  @param tip The content of the tip.
   *
   */
  - (void)setPullDown:(NSString *)tip;
  - (void)setPullDown:(NSString *)tip;

  /**
   *  Sets the text displayed during the loading process after you release the contro
l. The default text is "Loading...".
   *  Note: This text is not displayed for pull-down-to-refresh by default, but it is
displayed for pull-up-to-load.
   *
   *  @param tip The content of the tip.
   *
   */
  - (void)setLoading:(NSString *)tip;

  /**
   *  Sets the text that prompts the user to release the control. The default text is
"Release to refresh".
   *
   *  @param tip The content of the tip.
   *
   */
  - (void)setRelease:(NSString *)tip;

  /**
   *  Specifies whether to display the text for the last refresh time. This text is n
ot displayed by default.
   *
   *  @param isOpen Set to YES to display the text.
   *
   */
  - (void)ShowLastPullDate:(BOOL)isOpen;

  /**
   *  Specifies whether to display the loading status text.
   *
   *  Default: The text is not displayed for pull-down-to-refresh, but the text "Load
ing..." is displayed for pull-up-to-load.
   *
   *  @param isShow Set to YES to display the text.
   *
   */
  - (void)ShowStatusLabel:(BOOL)isShow;

  - (void)setDateFormat:(NSDateFormatter *)dateFromatter;

  - (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;
```

```
 /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 pull-up refresh control.

Code example

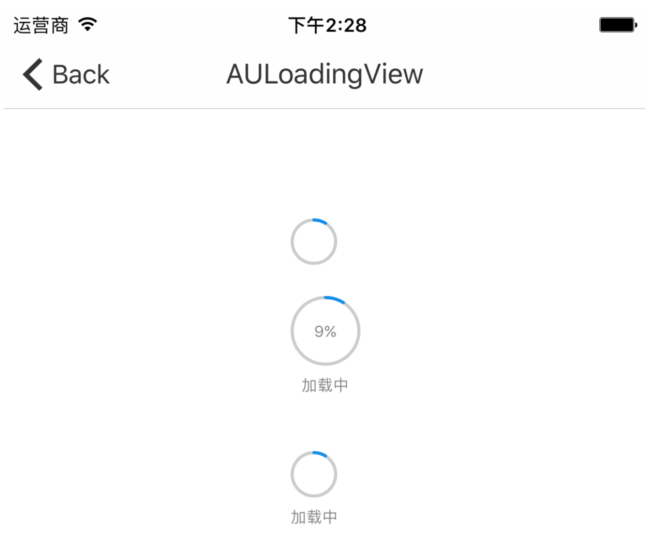
```
//
//  APRefreshTableViewController.m
//  UIDemo
//

#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];
    // Do any additional setup after loading the view.
    self.edgesForExtendedLayout = UIRectEdgeNone;
//    self.navigationItem.rightBarButtonItem = [APUtil
getBarButtonWithTitle:RightBarButtonTitle target:self];

    self.tableView = [[UITableView alloc]initWithFrame:self.view.bounds
style:UITableViewStylePlain];
    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) {
        AUPullLoadingView *view = [[AUPullLoadingView alloc]
        AUDragLoadingView *view = [[AUDragLoadingView alloc]
initWithFrame:CGRectMake(0, 0, self.view.bounds.size.width, 48)];
        view.delegate = self;
        [view setPullUp:@"Pull up to load more information"];
        [view setRelease:@"Release"];
        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:(NSIndexP
ath *)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;
    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
```

1.3.6.3. Loading component

AULoadingView is a loading control that displays a page with a progress indicator, the loading progress, and text.

Preview



API definition

AULoadingView.h

```
//
//  AULoadingView.h
//  AntUI
//

#import <UIKit/UIKit.h>

/**
  A loading control that displays a number in the center.
 */
@interface AULoadingView : UIView

@property (nonatomic,assign) BOOL isShowProgressPer; // Specifies whether to display th
e progress percentage. The default value is NO.
@property (nonatomic,assign) BOOL isShowLoadingText; // Specifies whether to display th
e loading text. The default value is NO.


/**
  Sets the progress percentage.

 @param progress The percentage value.
 */
- (void) setProgressPer:(CGFloat) progress;


@end
```

Code example

```
//
//  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 be
fore 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
```

1.3.7. Result page component

1.3.7.1. Result page component

AUResultView displays status views that contain images.

Screenshots

API reference

```
/**
 Displays a result view for a status.
 */
@interface AUResultView : UIView

/**
 The image at the top.
 */
@property (nonatomic, strong) UIImage *icon;

/**
 The medium-sized black title at the top of the text area.
 */
@property (nonatomic, strong) NSString *mainTitleText;

/**
 The large-sized black title in the middle.
 */
@property (nonatomic, strong) NSString *midTitleText;

/**
 The gray message at the bottom.
 */
@property (nonatomic, strong) NSString *bottomMessage;

/**
 Specifies whether to add a strikethrough to the bottom message.
 */
@property (nonatomic, assign) BOOL messageThrough;

/**
 The expected height of the view. This value is available after initialization.
 */
@property (nonatomic, assign, readonly) CGFloat expectHeight;

/**
 Instantiates a ResultView.

 @param icon The image.
 @param mainTitleText The first title.
 @param midTitleText The large title in the middle.
 @param bottomMessage The gray message at the bottom.
 @param messageThrough Specifies whether to add a strikethrough.
 @return An AUResultView instance.
 */
- (instancetype)initWithIcon:(UIImage *)icon mainTitleText:(NSString *)mainTitleText midTitleText:(NSString *)midTitleText bottomMessage:(NSString *)bottomMessage messageThrough:(BOOL)messageThrough;
```

Code example

```
UIImage *image = AUBundleImage(@"alipay-60");
AUResultView *resultView = [[AUResultView alloc] initWithIcon:image
                                                 mainTitleText:@"Payment successful"
                                                  midTitleText:@"998.00"
                                                 bottomMessage:@"CNY 1098.00"
                                                messageThrough:YES];
resultView.frame = CGRectMake(marginX, originY, AUCommonUIGetScreenWidth()-2*marginX, r
esultView.expectHeight);
[self.view addSubview:resultView];
```

1.3.7.2. Exception page component

The AUNetErrorView control displays exception views for empty pages and is available in two styles.

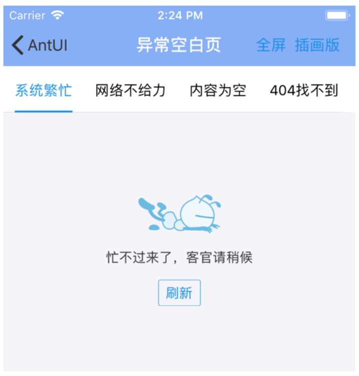
- Minimalist (half-screen) style: The default style, which includes five variations.
- Illustration (full-screen) style: Includes five variations.

The main difference between the styles is the prompt image. See the following examples.

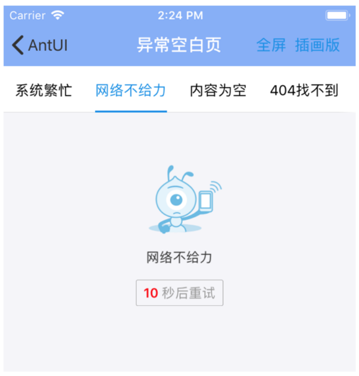
Examples

- Minimalist (half-screen) style

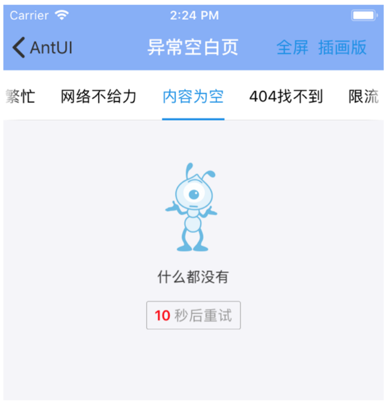
AUNetErrorTypeAlert



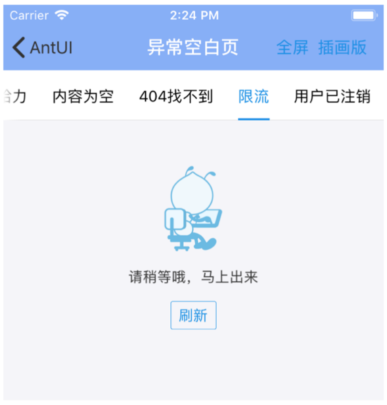
AUNetErrorTypeNetworkError



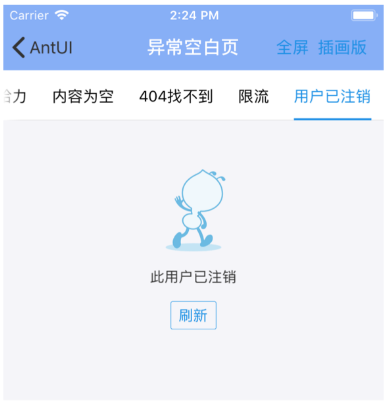
AUNetErrorTypeEmpty



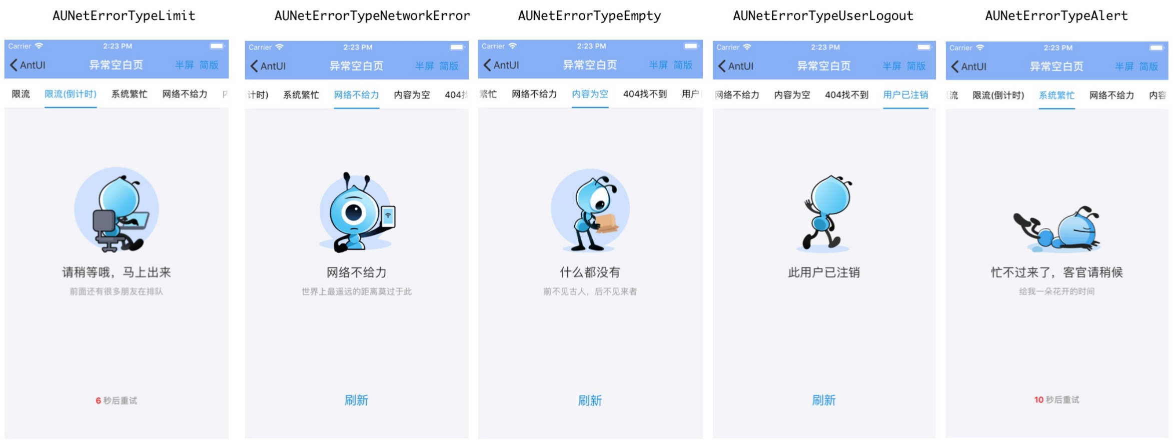
AUNetErrorTypeLimit



AUNetErrorTypeUserLogout



- Illustration (full-screen) style



API reference

```
    typedef NS_ENUM(NSInteger, AUNetErrorType) {

    AUNetErrorTypeLimit,        // Throttling
    AUNetErrorTypeAlert,        // System busy (system error), alert
    AUNetErrorTypeNetworkError, // Network error
    AUNetErrorTypeEmpty,        // Empty content
    AUNetErrorTypeNotFound,     // 404 Not Found (uses the same image as
AUNetErrorTypeAlert)
    AUNetErrorTypeUserLogout,   // User logged off

    AUNetErrorTypeFailure __attribute__((deprecated)) = AUNetErrorTypeNetworkError,
    AUNetErrorTypeError __attribute__((deprecated)) = AUNetErrorTypeNetworkError,
// Network error, connection failed
    AUNetErrorTypeSystemBusy __attribute__((deprecated)) = AUNetErrorTypeAlert,
// Alert
    APExceptionEnumNetworkError __attribute__((deprecated)) =
AUNetErrorTypeNetworkError,     // Network error, connection failed
    APExceptionEnumEmpty __attribute__((deprecated)) = AUNetErrorTypeEmpty,
// Empty content
    APExceptionEnumAlert __attribute__((deprecated)) = AUNetErrorTypeAlert,
// Alert
    APExceptionEnumLimit __attribute__((deprecated)) = AUNetErrorTypeLimit,
// Throttling
    APExceptionEnumNetworkFailure __attribute__((deprecated)) =
AUNetErrorTypeNetworkError,  // Network error
};


typedef NS_ENUM(NSInteger, AUNetErrorStyle) {
    AUNetErrorStyleMinimalist,    // Minimalist style
    AUNetErrorStyleIllustration,   // Illustration style

    APExceptionStyleIllustration __attribute__((deprecated)) =
AUNetErrorStyleIllustration,     // Illustration style
    APExceptionStyleMinimalist __attribute__((deprecated)) =  AUNetErrorStyleMinimalist
// Minimalist style
};
```

```
    /**
     A control that displays exception views on empty pages.

     It has two prompt styles:
            1. Minimalist style (default): Includes 3 types.
            2. Illustration style: Includes 7 types.

     The main difference between the styles and types is the image.
     */
    @interface AUNetErrorView : UIView

    @property(nonatomic, strong, readonly) UIButton *actionButton;      // The default
text is "Refresh".
    @property(nonatomic, strong, readonly) UIImageView *iconImageView;  // Icon view.
    @property(nonatomic, strong, readonly) UILabel *infoLabel;          // Main prompt
```

```
text label.
    @property(nonatomic, strong, readonly) UILabel *detailLabel;        // Detailed
prompt text label.

    @property(nonatomic, strong) NSString *infoTitle;                   // Main prompt t
ext.
    @property(nonatomic, strong) NSString *detailTitle;                 // Secondary pro
mpt text.

    /**
     *  Initializes an exception view and sets its style and type.
     *  (If target and action are empty, the refresh button is not displayed.)
     *
     *  @param frame The coordinates of the view. Required.
     *  @param style The exception style, either illustration or minimalist. Required.
     *  @param type The exception type. Required.
     *  @param target The object that handles the refresh event.
     *  @param action The method that handles the refresh event.
     *
     *  @return APExceptionView
     */
    - (id)initWithFrame:(CGRect)frame
                            style:(AUNetErrorStyle)style
                            type:(AUNetErrorType)type
                        target:(id)target
                        action:(SEL)action;

    /**
     *  Initializes an exception view and displays it on a specified view.
     *  (If target and action are empty, the refresh button is not displayed.)
     *
     *  @param parent The superview of the view. Required.
     *  @param style The exception style, either illustration or minimalist. Required.
     *  @param type The exception type. Required.
     *  @param target The object that handles the refresh event.
     *  @param action The method that handles the refresh event.
     *
     *  @return APExceptionView
     */
    + (id)showInView:(UIView *)parent
                    style:(AUNetErrorStyle)style
                    type:(AUNetErrorType)type
                    target:(id)target
                    action:(SEL)action;

    /**
     * Dismisses the exception view.
     */
    - (void)dismiss;

    /**
 *  Countdown timer. For throttling only.
 *  If completeBlock is nil and no click response event is set for the actionButton, th
e countdown feature does not work.
```

```
 *  If completeBlock is not nil, the completeBlock is executed directly when the countd
own ends, and the actionButton is hidden.
 *  If you use getActionButton to add a response event to the button, make sure to add
the event before calling this method.
 */
- (void)setCountdownTimeInterval:(NSInteger)startTime  // Countdown start time
                   completeBlock:(void (^)(void))completeBlock; // Called after the coun
tdown ends

    @end
```

Code example

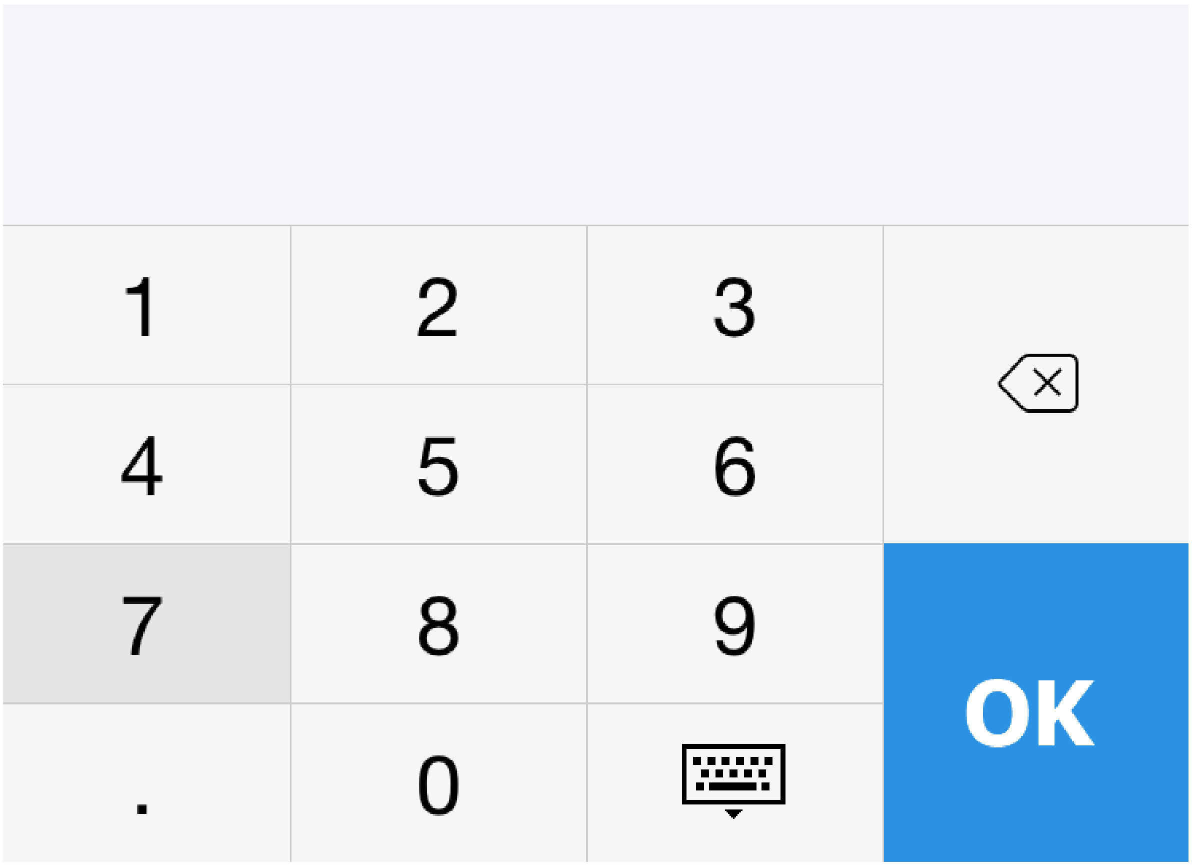
```
    netErrorView = [[AUNetErrorView alloc] initWithFrame:CGRectMake(0,
CGRectGetMaxY(label.frame) + 5, self.view.width, 300) style:AUNetErrorStyleIllustration
type:AUNetErrorTypeError target:self action:@selector(pressedNetErrorView)];
    netErrorView.detailTitle = @"The type is AUNetErrorTypeError";
    [self.view addSubview:netErrorView];

    // Set the countdown timer
    [netErrorView setCountdownTimeInterval:10 completeBlock:^{
            NSLog(@"Countdown finished");
    }];
```

1.3.8. Numeric keypad component

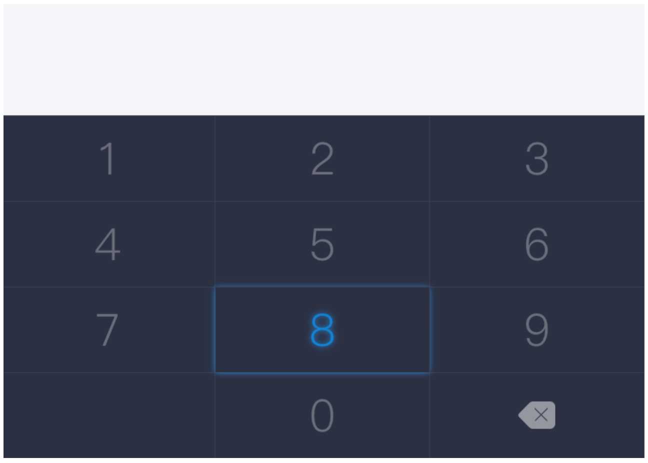
AUNumKeyboards is a custom numeric keypad component.

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 exte
rnally.
 */
@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 characte
rs.
 */
@property (nonatomic, strong) NSString *submitText;
```

Code sample

```
UITextField *numTextField = ...
numTextField.inputView = [AUNumKeyboards
sharedKeyboardWithMode:AUNumKeyboardModeCommon] ; // The chat mode parameter: AUNumKeyb
oardModeChat.
[self.view addSubview:numTextField];
```

1.3.9. Guidance component

1.3.9.1. Tooltip component

AUPopTipView is a component that displays a tooltip.

Preview

API reference

```
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
```

Code example

```
// Show the tooltip.
AUPopTipView *popTipView = [AUPopTipView showFromView:button
                                            fromPoint:CGPointZero
                                               toView:self.view
                                             animated:YES
                                             withText:@"This is a tooltip."
                                          buttonTitle:@"Close"]; // If this parameter is
omitted, the button on the right is not displayed.

// Hide the tooltip.
[popTipView dismiss:YES];
```

1.3.9.2. Guide pop-up bar component

AUPopBar is a pop-up guide bar component.

Preview

API reference

```
@interface AUPopBar : AUView

AU_UNAVAILABLE_INIT

+ (instancetype)showInViewBottom:(UIView *)view
                        animated:(BOOL)animated
                        withText:(NSString *)text
                            icon:(UIImage *)icon
                     buttonTitle:(NSString *)buttonTitle
                     actionBlock:(BOOL(^)())actionBlock;

- (void)dismiss:(BOOL)animated;

@end
```

Code example

```
// Display
AUPopBar *popBar = [AUPopBar showInViewBottom:weakSelf.view animated:YES withText:@"Add
'City Services' to the home page" icon:[UIImage imageNamed:@"ap_scan"]
buttonTitle:@"Add Now" actionBlock:^{
                    NSLog(@"Clicked");
                    return YES;
                }];

// Hide
[popBar dismiss:YES];
```

1.3.10. 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.

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 imag
e.
  //@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) superV
iew:self.view isArchViewUp:YES];
_menu.delegate = self;
[_menu showMenu];
```

1.3.11. Navigation components

1.3.11.1. Vertical tab

AUVerticalTabView is a vertical tab-based component.

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 AntDN
A.
 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)selectedCallback
                                          height:(CGFloat)height
                                        business:(NSString *)business;

@property(nonatomic, strong) NSArray <id<AUVerticalTabViewDataProtocol>>*
verticalTabViewDatas;
@property(nonatomic, assign) NSUInteger selectedIndex;//default 0
@property(nonatomic, copy) AUVerticalTabSelectedCallback selectedCallback;

@end
```

Code sample

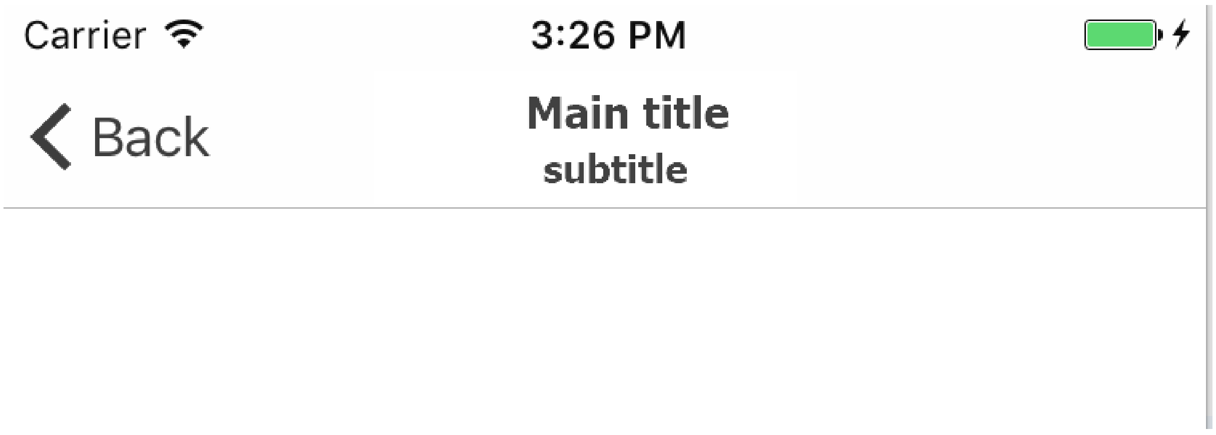
```
// The external data object implementation AUVerticalTabViewDataProtocol, which returns
the required tabName.
@interface DemoVerticalTabData : NSObject <AUVerticalTabViewDataProtocol>

- (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];
```

1.3.11.2. Double title

AUDoubleTitleView is a view control for displaying a two-line title that consists of a main title and a subtitle in a navigation bar.

Preview



API reference

```
    /**
 A titleView for a navigation bar that contains two lines.
 */
@interface AUDoubleTitleView : UIView

/**
 *  Creates a titleView with a main title and a subtitle.
 *
 *  @param title          The main title.
 *  @param detaileTitle   The subtitle.
 *
 *  @return An initialized APTitleView control.
 */
- (UIView *)initWithTitle:(NSString *)title detailTitle:(NSString *)detaileTitle;

/**
 *  Updates the text of the main title.
 *
 *  @param title          The text of the main title.
 *
 */
- (void)updateTitle:(NSString *)title;

/**
 *  Updates the text of the subtitle.
 *
 *  @param detailTitle          The text of the subtitle.
 *
 */
- (void)updateDetailTitle:(NSString *)detailTitle;


/**
 Updates the font of the main title.

 @param titleFont The font of the main title.
 */
- (void)updateTitleFont:(UIFont *)titleFont;

/**
 *  Updates the font of the subtitle.
 *
 *  @param detailTitleFont      The font of the subtitle.
 *
 */
- (void)updateDetailTitleFont:(UIFont *)detailTitleFont;

@end
```

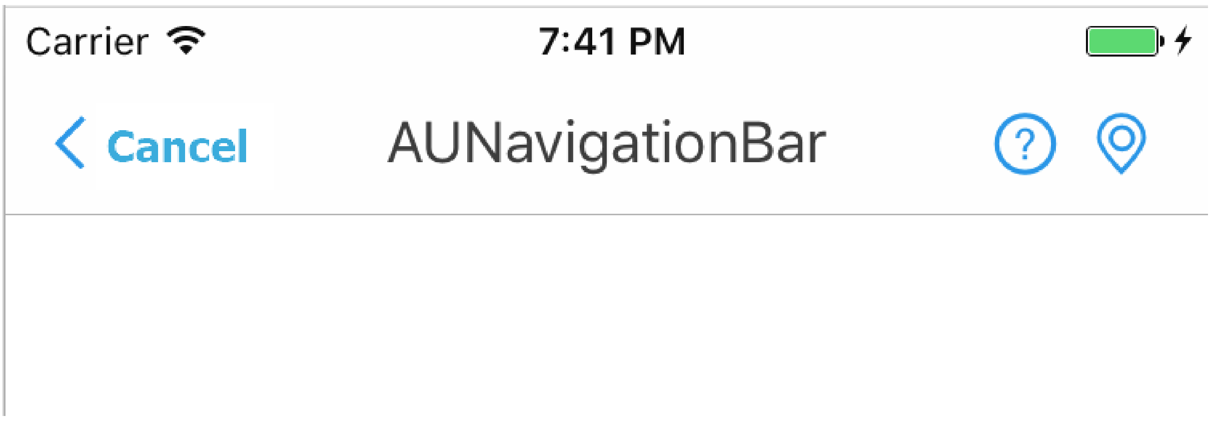
Code example

```
self.navigationItem.titleView = [[AUDoubleTitleView alloc] initWithTitle:@"Main Title"
detailTitle:@"Subtitle"];
```

1.3.11.3. Navigation bar

AUNavigationBar is a navigation bar control of mPaaS. It extends the UINavigationBar control and provides default mPaaS navigation bar styles. To facilitate subsequent extension, use AUNavigationBar rather than UINavigationBar 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:NSClassFromString(@"AUNavigationBar")  toolbarClass:nil];
 */
@interface AUNavigationBar : UINavigationBar

@end


/**
 The UINavigationBar extension that defines default UINavigationBar styles.
 */
@interface UINavigationBar (AUNavigationBarExtensions)

/**
 *  Return the default color of the title of framework navigation bar. The default valu
e is #000000.
 *
 *  @return
 */
+ (UIColor*)getNavigationBarTitleDefaultColor;

/**
 *  Return the color of the item of framework navigation bar. The default value is #108
EE9.
 *
 *  @return
 */
+ (UIColor*)getNavigationBarButtonItemDefaultColor;

/**
 *  Return the color of the item of framework navigation bar. The default value is #fff
ff.
 *
 *  @return
 */
+ (UIColor*)getNavigationBarDefaultColor;

/**
 *  Get the color of the bottom line of the navigation bar. The default value is #e1e1e
1.
 *
 *  @return
 */
+ (UIColor*)getNavigationBarBotLineColor;

/**
 * Note:
 * 1. The base class DTViewController sets the default style of navigation bar in ViewW
illAppear.
 * 2. Business personnel can call the system APIs or the following APIs to change the s
tyle 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 whe
n ViewWillDisappear is called after the modification.
 * 5. If VC defines the homepage in the UITabBarController container, do not use setNav
igationBarDefaultStyle 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 botto
m 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
animation will flash white. Currently, this issue has not been resolved. Do not call th
is 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 transl
ucent 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 mee
t some UI requirements. Call this method to set the color of the separation line to pre
vent 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 aft
er 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 setBackg
roundColor to set the color of navigation bar, call this method to eliminate the defaul
t effect.
 *  Otherwise, the color will be overlaid with the default color and cause a color devi
ation.
 */
- (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)setNavigationBarMaskLayerWithtColor:(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" targe
t:self action:@selector(cancel)];
cancelItem.backButtonTitle = @"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:UIBar
ButtonItemStylePlain target:self action:@selector(rightBarItemPressed)];
AUBarButtonItem *rightItem2 = [[AUBarButtonItem alloc] initWithImage:image2 style:UIBar
ButtonItemStylePlain target:self action:@selector(rightBarItemPressed)];
self.navigationItem.rightBarButtonItems = @[rightItem1, rightItem2];
```

1.3.11.4. 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.

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 *backButtonTitle;           // The back button ti
tle (no title by default).
@property(nonatomic, strong) UIColor *backButtonTitleColor;       // The title color 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 *titleView;                   // Customized titlevie
w.

@property(nonatomic, strong) NSString *rightItemTitle;            // The right item
title.
@property(nonatomic, strong) UIColor *rightItemTitleColor;        // The color of the r
ight 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.
 */
 + (AUCustomNavigationBar *)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
action:(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 a
ction:(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 ac
tion:(SEL)action;


@end
```

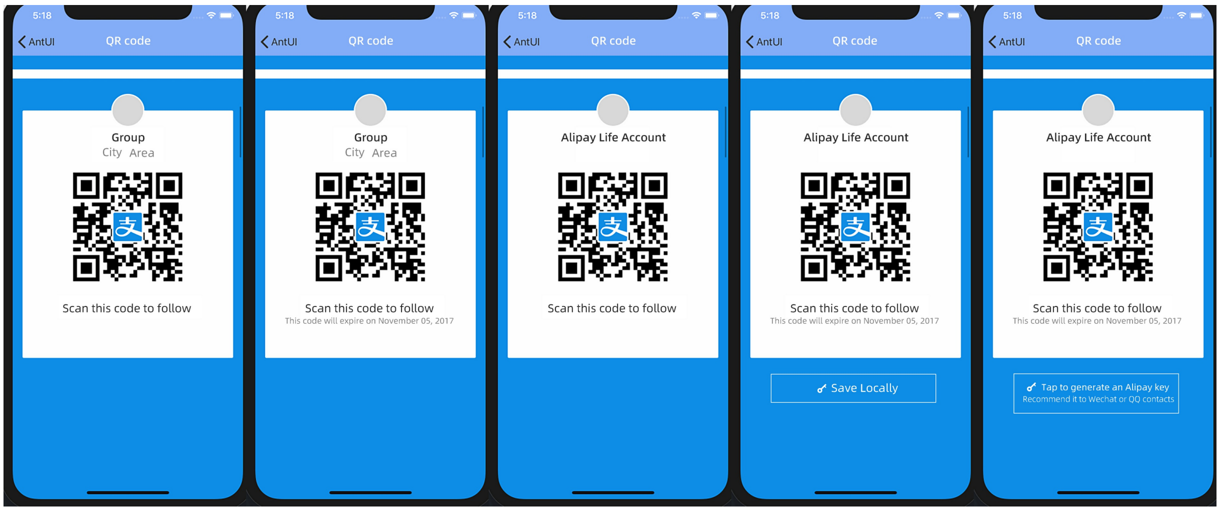
Sample code

```
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_COLO
R_LINK];
navBar.backButtonTitle = @"Bill";
navBar.rightItemImage = [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;
}
```

1.3.12. 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


5:18
AntUI
QR code
Group
City Area
Alipay Life Account
Scan this code to follow
This code will expire on November 05, 2017
Save Locally
Tap to generate an Alipay key
Recommend it to Wechat or QQ contacts

API description

```
// The data model object.
@interface QRDataModel : NSObject

@property (nonatomic, strong) id topLeftIcon;                // An image, a URL, or clou
dID 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 descripti
on of 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
left corner.
@property (nonatomic, strong) UILabel *topTitleLabel;         // Description text for t
he title on the top.
@property (nonatomic, strong) UIImageView *qrCodeView;        // The QR code image.
@property (nonatomic, strong) UILabel *bottomTitleLabel;      // Main description text
at the bottom.
@property (nonatomic, strong) UILabel *bottomMessageLabel;    // Secondary description
text at the bottom.
@property (nonatomic, strong) QRActionButton *actionButton;   // The action button at t
he bottom.

// 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

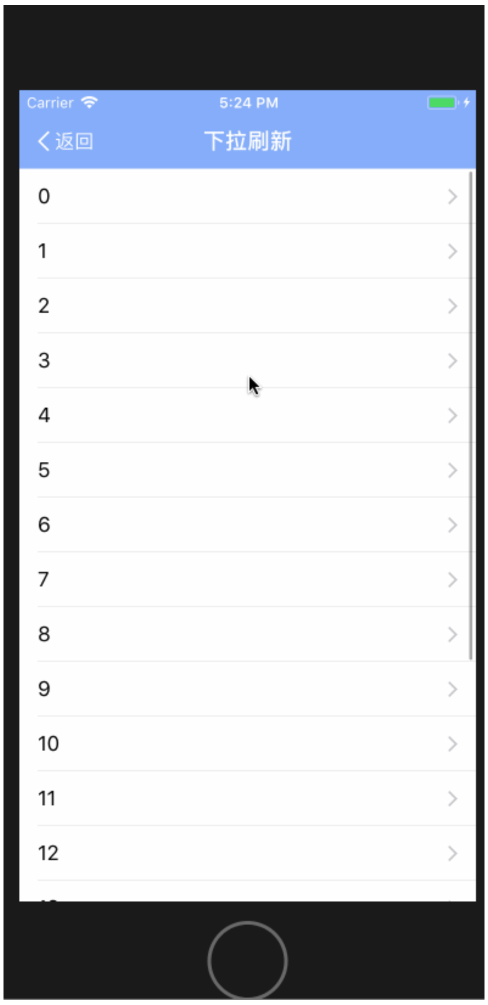
The following sample shows the code of a standard style QR code.

```
    AUQRCodeView *qrCodeView = [[AUQRCodeView alloc] initWithFrame:frame
model:^(QRDataModel *model) {
        model.topLeftIcon = [UIImage imageWithColor:[UIColor colorWithRGB:0xbbbbbb] siz
e: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];
```

1.3.13. Pull-to-refresh component

AURefreshView is the new pull-to-refresh component in the Ant style. It currently supports two color schemes, as shown in the preview.

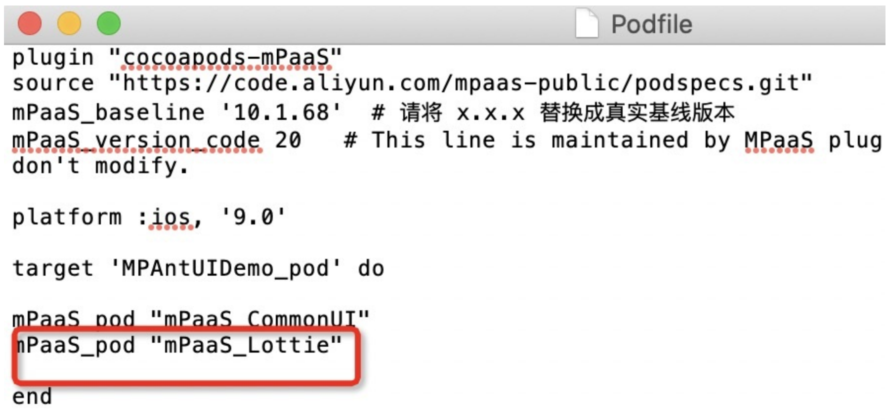
Preview



Add the component

Add the Common UI and Lottie component SDKs using the cocoapods-mPaaS plugin. The procedure is as follows:

1. In the Podfile, use the `mPaaS_pod "mPaaS_CommonUI"` and `mPaaS_pod "mPaaS_Lottie"` commands to add dependencies for the **Common UI** and **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` on the command line to complete the process.

API reference

```
typedef NS_ENUM(NSUInteger, AURefreshViewState) {
    AURefreshViewStateNomal = 0,           // The list returns to its initial position.
    AURefreshViewStateBeginPulling = 1,    // The user starts to pull down.
    AURefreshViewStateLoading = 2,         // An RPC load is triggered. The list's
contentInset is set to the default position.
    AURefreshViewStateFinishedLoading = 3, // The RPC load is complete. The list's cont
entInset is about to return to its original default value.
    AURefreshViewStateBeginResetting = 4,  // The list's contentInset is starting to re
turn to its original default inset.
};
typedef NS_ENUM(NSUInteger, AURefreshViewType) {
    AURefreshViewDefault,       // The refresh style within the page.
    AURefreshViewTypeFeature1   // Used for a title bar with a background, such as on t
he home page or wealth tab.
};
@protocol AURefreshViewDelegate;
/**
 The mPaaS pull-to-refresh animation view.
 */
@interface AURefreshView : UIView
@property (nonatomic, readonly) AURefreshViewState state;
@property (nonatomic, weak) id <AURefreshViewDelegate> delegate;
/**
 The Lottie control for the pull-to-refresh animation.
 */
@property (nonatomic, strong) UIView /*LOTAnimationView */ *lottieAnimationView;
/* Specifies the parent view for the pull-to-refresh control. The initial default heigh
t is the height of the scrollView. By default, the refreshView is added to the parent s
crollView.
 * The default initial frame is (0, 0 - scrollView.height, scrollView.width, scrollView
.height). */
```

```
- (instancetype)initWithSuperView:(UIScrollView *)scrollView
                              type:(AURefreshViewType)type
                           bizType:(NSString *)bizType;
// The text for the pull-to-refresh control.
- (void)setupLabelText:(NSString *)text;
// Call the following methods in the UIScrollView delegate.
- (void)auRefreshScrollViewWillBeginDragging:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidScroll:(UIScrollView *)scrollView;
- (void)auRefreshScrollViewDidEndDragging:(UIScrollView *)scrollView;
// Call the following method to end the animation and collapse the list.
- (void)auRefreshScrollViewDidFinishedLoading:(UIScrollView *)scrollView;
// You must first scroll to the initial position and then call this method for an autom
atic pull-to-refresh. Otherwise, the scroll behavior will be abnormal.
- (void)autoPullRefreshScrollView:(UIScrollView *)scrollView;
//
- (void)pauseAnimation;
// The page expands.
- (void)resumeAnimation;
@end
@protocol AURefreshViewDelegate <NSObject>
@optional
// This protocol is triggered when the view is pulled down to the default position, whi
ch is the height of the (Lottie)View.
- (void)auRefreshViewDidTriggerloading:(AURefreshView *)view;
// The pull-to-refresh is complete and the view is reset.
- (void)auRefreshViewDidDidFinishAnimation:(AURefreshView *)view;
@end
```

Code examples

Standard style

The following code shows an example of a pull-to-refresh component with the standard style.

```
_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 create a custom style, use Lottie and rewrite the category of `AUThemeManager`. The following code shows an example:

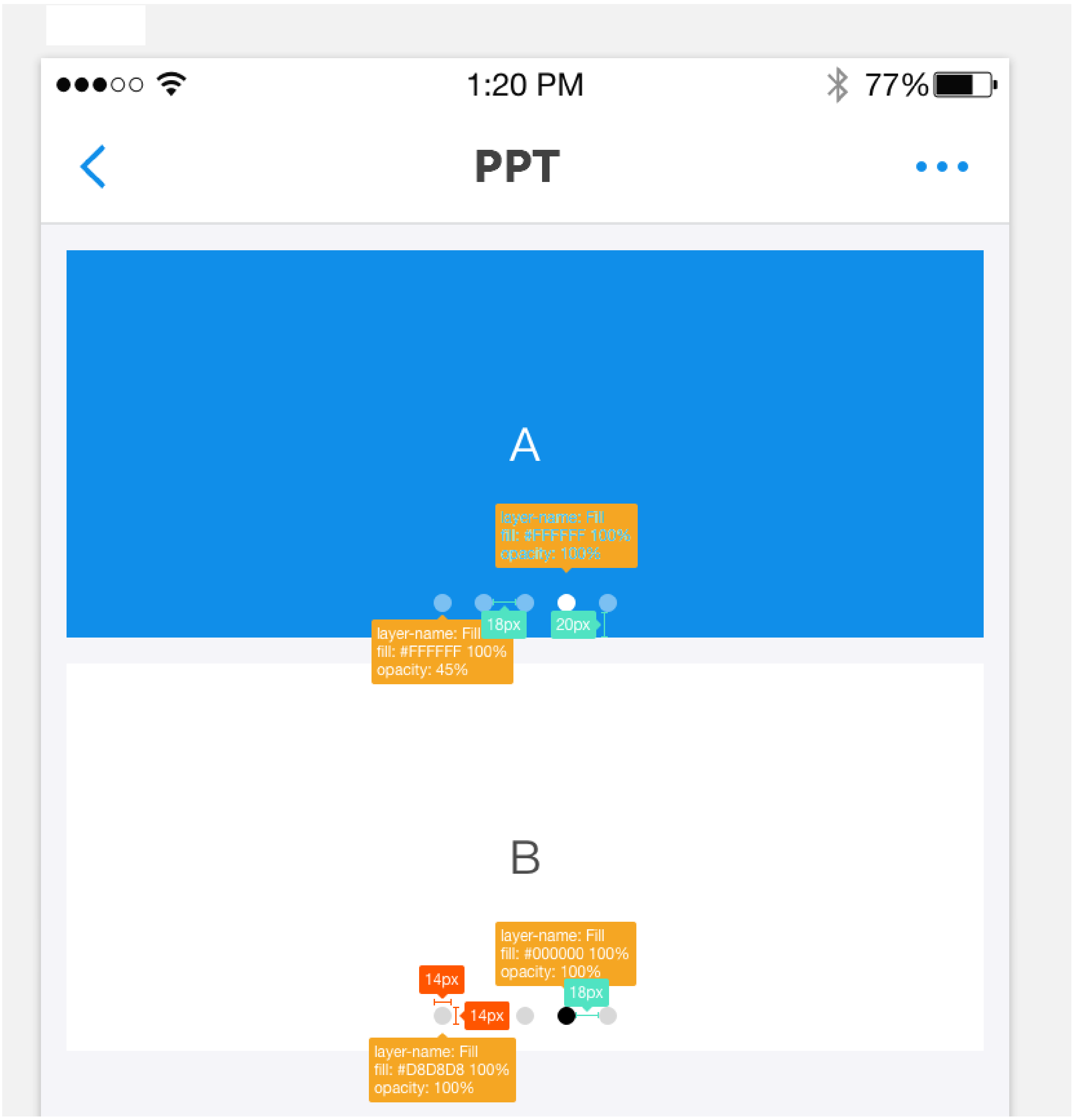
```
+ (NSString *)au_defaultTheme_refresh_lottie_path{

    NSString *path = [[NSBundle mainBundle] pathForResource:@"ani" ofType:@"json"];
    return path;

}
```

1.3.14. Other components

1.3.14.1. 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 N
O.
@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 a
rea 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 p
age, which starts from page 0.
@property (nonatomic, weak)     id<AUBannerViewDelegate> delegate;      // The data sou
rce 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, se
t 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;
```

```
 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,
50, 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];
        [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:@"%@.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 ye
llowColor]];
    }
    // The light color.
    else {
        array = @[RGB(0xfFFFFF),RGB_A(0xeFFFFF, 0.7),RGB(0xcFFFFF),RGB_A(0xeFFFFF, 0.5)
,RGB_A(0xeFFFFF, 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;
//}
```

1.3.14.2. Segment component

AUSegment provides a scrollable tab bar.

AUSegment is not fully interchangeable with APSegmentedControl in APCommonUI. This is because APSegmentedControl only encapsulates the system's UISegmentedControl and does not provide other features.

Dependencies

AUSegment has the following dependencies:

```
import "AUSegmentedControlItem.h"
```

API reference

```
@protocol AUSegmentedControlDelegate <UIScrollViewDelegate>
```

```
// Callback for AUSegment tap events.
@optional

- (void)didSegmentValueChanged:(AUSegment*)segmentControl;

- (void)didSelectSegmentItemModel:(AUSegmentItemModel*)selectedItemModel;

@end

// Default height of the segment.
#define     AUSegmentHeight     AU_SPACE13

/**
    A segmented control component.
 */
@interface AUSegment : UIScrollView

/**
 Initializes the component.

 @param frame The frame.
 @param titles An array that contains all title strings.

 @return An AUSegment instance.
 */
- (instancetype)initWithFrame:(CGRect)frame titles:(NSArray<NSString*> *)titles;

/**
 Disables the init method.
 */
- (instancetype)init NS_UNAVAILABLE;

/**
 Disables the initWithFrame method.
 */
- (instancetype)initWithFrame:(CGRect)frame NS_UNAVAILABLE;

/**
 The AUSegmentedControlDelegate.
 */
@property (nonatomic, weak)     id <AUSegmentedControlDelegate> delegate;

/**/

/**
 The title array.
 */
@property (nonatomic, strong)   NSMutableArray *titles;

/**
 * The menu font.
 */
@property (nonatomic, copy) UIFont *titleFont;
```

```
/**
 The index of the currently selected segment.
 */
@property (nonatomic, assign)   NSInteger selectedSegmentIndex;


/**
 The color of the selected item, including the text and slider.
 */
@property (nonatomic, copy)     UIColor *selecedColor;

/**
 * The horizontal margin on the left and right of each text menu.
 * The default value is 21 pixels.
 * If the red dot style is used, the fixedItemWidth property does not take effect and t
he menus do not have a fixed width.
 */
@property(nonatomic, assign)    NSInteger textHorizontalPadding;

/**
 * Specifies whether to use a fixed width for menus.
 * The default value is YES for compatibility with older menu styles.
 * If this property is set to YES, the horizontalPadding property does not take effect
and all menus have the same fixed width.
 */
@property (nonatomic, assign) BOOL fixedItemWidth;

/**
 * Specifies whether to automatically scroll the selected menu item to a proper positio
n. The item is centered by default. If there is not enough space, the item is aligned t
o the edge.
 * The default value is NO.
 */
@property (nonatomic, assign) BOOL autoScroll;

/**
 * Specifies whether to automatically update the indicator bar to the index of the curr
ently selected item after a tap.
 * The default value is YES.
 */
@property (nonatomic, assign) BOOL autoChangeSelectedIndex;

/*
 * The model array.
 */
@property(nonatomic, strong) NSMutableArray<AUSegmentItemModel *> *itemModels;

/**
 Inserts multiple items at specified indexes.

 @param array The array of titles to insert.
 @param indexes The indexes at which to insert the items.
 */
- (void)insertTitleArray:(NSArray<NSString*> *)array atIndexes:(NSIndexSet *)indexes;
```

```
/**
 Appends multiple items to the end.

 @param array The array of titles to add.
 */
- (void)addTitleArray:(NSArray<NSString*>*)array;

/**
 * Automatically scrolls to the item at the specified index. Note: This only scrolls th
e item into view. The selection indicator does not change.
 * By default, this is the same as selectedSegmentIndex, which is the selected item.
 */
- (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;       // Specifies whether to reserve
space for a red dot on the current item. If no space is reserved, the interface may fli
cker when the red dot is displayed.
@property(nonatomic, strong) NSDictionary *extendInfo; // Extension field.

@end


@interface AUSegment (ItemModel)



/**
 * The second version of the initializer function.
 * @param frame The frame.
 * @param menus The item array.
 */
- (instancetype) initWithFrame:(CGRect)frame menus:(NSArray<AUSegmentItemModel *>*)menu
s;

/**
 Updates the control items.

 @param items The array of items to update. Use this to add, delete, or update all exis
ting model data.
 */
- (void)updateItems:(NSArray<AUSegmentItemModel *>*)it
- (void)updateItems:(NSArray<AUSegmentItemModel *>*)items;

/**
 Updates the item model at a specific index.

 @param model The new item model.
 @param index The index of the item to update.
 */
- (void)updateItemModel:(AUSegmentItemModel *)model
                atIndex:(NSInteger)index;

@end


// Displays an action icon button on the right. A plus sign (+) is displayed by default
.
@interface AUSegment (AUActionIcon)

- (void)showActionIcon:(BOOL)showIcon target:(id)target action:(SEL)action;

@end
```

Custom properties

Property	Purpose	Type
titles	The title array of the segment.	NSArray
selectedSegmentIndex	The selected segment.	NSInteger
delegate	Implements the AUSegmentedControlDelegate protocol.	id
autoScroll	Specifies whether to automatically scroll the selected menu item to a proper position. The item is centered by default. If there is not enough space, the item is aligned to the edge.	BOOL
fixedItemWidth	Specifies whether to use a fixed width for menus.	BOOL
textHorizontalPadding	The horizontal margin on the left and right of each text menu.	BOOL
titleFont	A custom menu font.	UIFont

Code examples

- Segmented control without red dots:

```
      NSArray *testArray1 =
@[@"tab1",@"tab2",@"tab3",@"tab4",@"tab5",@"tab6",@"tab7",@"tab8"];
      AUSegment *segment = [[AUSegment alloc] initWithFrame:CGRectMake(0, 300, self.v
iew.width, 44) titles:testArray1];
      segment.delegate = self;
      [self.view addSubview:segment];

          // Callback
          - (void)didSegmentValueChanged:(AUSegment*)segmentControl {
      NSLog(@"AUSegmented switched");
  }
```

- Segmented control with red dots:

```
  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, sel
f.view.width, 44) menus:array];
  [self.view addSubview:segment2];
  [segment2 autoScrollToIndex:6];
  segment2.backgroundColor = [UIColor whiteColor];
  [segment2 showActionIcon:YES target:self action:@selector(clickActionIcon:)];
```

1.3.14.3. Icon component

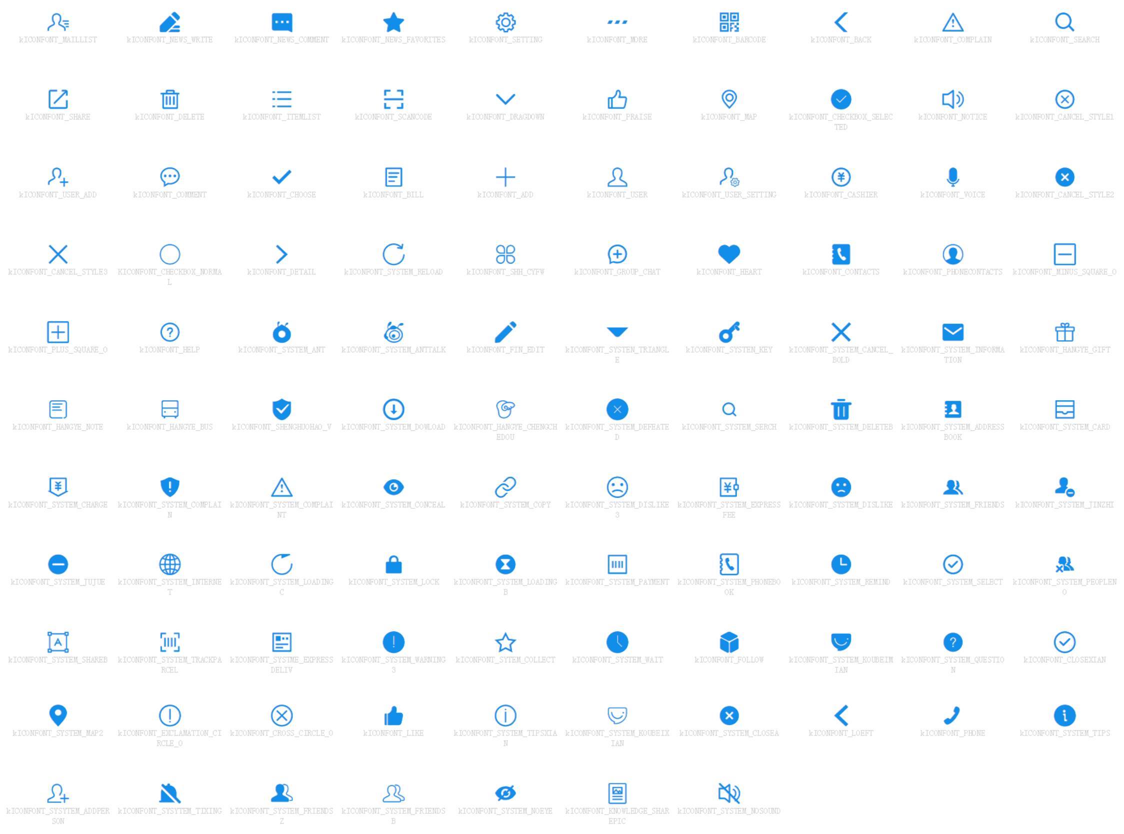
The iconfont image control is an image object rendered from a string using the `drawrect` feature. It behaves like an image view.

- AUIconView is an iconfont vector image control. You can use it similarly to a `UIImage`.
- The iconfont image control is an image object rendered from a string using the `drawrect` feature. It behaves like an image view.

> **Note**
>
> Only square vector images are supported.

- An iconfont is a font file that contains a collection of images. Each image is mapped to a unique Unicode code. To render an iconfont, you can set the text to the corresponding Unicode code and call the `drawInRect` method of the string.
- Each iconfont collection is a TTF font file. You can load multiple TTF files, each identified by a name. The default font is the AntUI TTF font, named `auiconfont`.

Preview



API reference

```
// Default iconfont name for AntUI
#define kICONFONT_FONTNAME              (@"auiconfont")
// Default iconfont path for AntUI
#define kICONFONT_FONTPATH              (@"APCommonUI.bundle/iconfont/auiconfont")





/**
 An iconfont image control. You can use it like an image view.
 It is an image object drawn by the drawrect feature of a string.
 Note: Only square vector images are supported.

 An iconfont is a loaded font. A single font can contain multiple images, and each imag
e has a Unicode code.
 You can set the text to the corresponding Unicode code and call the drawInRect method
of the string to render the iconfont.

 Each iconfont collection is a TTF font file. You can load multiple
 TTF font files. Each TTF font file has a name. The default is the AntUI TTF font.
 The name is auiconfont.
 */
@interface AUIconView : UIImageView

@property (nonatomic, strong) UIColor *color;       // The color of the vector image. T
he default is Ant Blue.
@property (nonatomic, strong) NSString *name;       // The name of the vector image.
@property (nonatomic, strong) NSString *fontName;   // The name of the vector font libr
ary.


/**
 Initialization method.

 @param frame The view frame.
 @param name  The name of the iconfont image.

 @return An AUIconView instance.
 */
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name;

/**
 Initialization method.
 (If this iconfont font is already loaded, you do not need to pass fontPath for renderi
ng.)

 @param frame    The view frame.
 @param name     The name of the iconfont image.
 @param fontName The name of the iconfont font.

 @return An AUIconView instance.
 */
- (instancetype)initWithFrame:(CGRect)frame name:(NSString *)name fontName:(NSString *)
fontName;


/**
 Gets the size of the iconView.

 @return The size of the iconfont if it is an iconfont, or the size of the image if it
is a normal image view.
 */
- (CGSize)iconViewSize;

@end


@interface UIImage (AUIconFont)

/**
 Registers an iconfont. This method only needs to be called once.

 @param fontName The name of the iconfont font.
 @param fontPath The path of the iconfont, such as @"AntUI.bundle/iconfont/auiconfont"
 @param fontPath The path of the iconfont, such as @"AntUI.bundle/iconfont/auiconfont".
 */
+ (void)registerIconFont:(NSString *)fontName fontPath:(NSString *)fontPath;

/**
 Gets a square vector image (equal width and height).

 @param name  The name.
 @param width The size.
 @param color The color of the image. If you pass nil, the default is Ant Blue.

 @return A square vector image.
 */
+ (UIImage *)iconWithName:(NSString *)name
                                        width:(CGFloat)width
                                        color:(UIColor *)color;



/**
 Gets a square vector image (equal width and height).

 @param name         The name.
 @param fontName     The name of the vector font.
 @param width        The size.
 @param color        The color of the image. If you pass nil, the default is Ant Blue.

 @return A square vector image.
 */
+ (UIImage *)iconWithName:(NSString *)name
                                fontName:(NSString *)fontName
                                        width:(CGFloat)width
                                        color:(UIColor *)color;

@end
```

Code example

```
// 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 separately
self.image = [UIImage iconWithName:self.name fontName:self.fontName width:width color:s
elf.color];
```

1.3.14.4. 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.

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:(N
SString*)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 *)in
dexPath
{
    return 44;
}// Use the estimatedHeight methods to quickly calcuate 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:(NSIn
teger)section
{
    NSDictionary * test = [self.sectionArr objectAtIndex:section] ;

    return [[test allKeys] firstObject];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexP
ath *)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 ;
    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 be
fore 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:(N
SString*)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 in
Section: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
objectOrNilAtIndex: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
```

1.3.14.5. 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.

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 each segment is 26 px.
    #define AUTitleBarSegment_DefaultSegmentWidth       90      // The default width of each segment is 90 px.

    @interface AUTitleBarSegment : UISegmentedControl

    @end
```

Sample code

```
AUTitleBarSegment *titleBatSegment = [[AUTitleBarSegment alloc] initWithItems:@[@"Label", @"Label"]];
    self.navigationItem.titleView = titleBatSegment;
```

1.3.14.6. 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 AUSwitch 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 b
utton.

/**
*  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 APBarButtonItem
*/
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title target:(id)target act
ion:(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 APBarButtonItem
*/
+ (AUBarButtonItem *)backBarButtonItemWithTitle:(NSString *)title maxWordsCount:(NSInte
ger)count target:(id)target action:(SEL)action;

@end
```

Code sample

```
// Define a backBarItem.
// The backBarItem contains a back icon by default.
AUBarButtonItem *cancelItem = [AUBarButtonItem backBarButtonItemWithTitle:@"Back" targe
t:self action:@selector(cancel)];
cancelItem.backButtonTitle = @"Cancel";
self.navigationItem.leftBarButtonItem = cancelItem;

AUBarButtonItem *rightItem1 = [[AUBarButtonItem alloc] initWithImage:image1 style:UIBar
ButtonItemStylePlain target:self action:@selector(rightBarItemPressed)];
```

1.3.14.7. Adaptation and dependency

As the shell of AntUI, AntUIShell is mainly used to implement third-party protocols in AntUI. It can be embedded to an mPaaS app and reduce external dependencies of AntUI.

API description

AntUIShellObject.h

```
//
//  AntUIShellObject.h
//  AntUIShell
//

#import <Foundation/Foundation.h>
#import <AntUI/AntUI.h>

@interface AntUIShellObject : NSObject<AUThirdPartyAdapter>

@end
```

Code sample

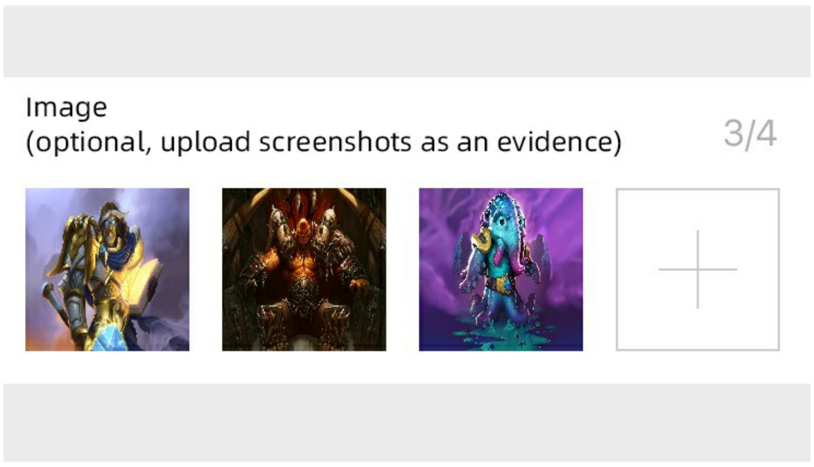
```
//
//  AntUIShellObject.m
//  AntUIShell
//

#import "AntUIShellObject.h"
#import <APMonitor/APMonitor.h>
#import <APMultimedia/APMultimedia.h>
#import <MPBadgeService/MPBadgeService.h>

@implementation AntUIShellObject


#pragma mark ----AUThirdPartyAdapter
/************************************************************/
// The image protocol APMultimedia.
/*
 API adaptation for third-party to download image.
 It wraps mulimedia 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  [[APImageManager manager] getImage:identifier business:business zoom:size o
riginalSize:originSize progress:progress completion:complete];

}

/*
API adaptation for third-party to download UIImageView image.
 It is implemented by a third party.
 */
- (void)thirdPartypFromImageView:(UIImageView *)fromImgView
                 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(fromImgView && [fromImgView isKindOfClass:[UIImageView class]]) {
        [fromImgView 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
// 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 up
on logout 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
```

1.3.14.8. 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



Dependency

```
Currently, this component is not added to the AntUI baseline.
```

API description

```
@protocol AUImagePickerDataProtocol <NSObject>

- (UIImage *)image;

@end

@interface AUImagePickerSkeleton : UIView

- (AUImagePickerSkeleton *)initWithTitle:(NSString *)title
              maxNumberOfImages:(NSUInteger)maxNumberOfImages;

@property(nonatomic, assign, readonly) NSUInteger maxNumberOfImages;
@property(nonatomic, weak) id<AUImagePickerDelegate> delegate;
@property(nonatomic, strong, readonly) NSArray<id<AUImagePickerDataProtocol>> *imagePic
kerDatas;

- (void)updateImagePickerDatas:(NSArray <id<AUImagePickerDataProtocol>>*) datas;

@end

@protocol AUImagePickerDelegate <NSObject>

@required
- (void)imagePickerAddButtonClick:(AUImagePickerSkeleton *)imagePicker;

@optional
- (void)imagePickerImageClick:(AUImagePickerSkeleton *)imagePicker
                   clickData:(id<AUImagePickerDataProtocol>)clickData;
@end
```

Sample code

```
- (void)viewDidLoad {
    [super viewDidLoad];
    self.datas = [[NSMutableArray alloc] init];
    self.picker = [[AUImagePickerSkeleton alloc] initWithTitle:@"Image(optional, upload
screenshots as an evidence)"maxNumberOfImages: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:(AUImagePickerSkeleton *)imagePicker
{
    AUImagePickerData *data = [AUImagePickerData new];
    data.originalImage = [self getImageWithCount:[self.datas count]];
    [self.datas addObject:data];
    [self updatePickerAndResize];
}

-(void)imagePickerImageClick:(AUImagePickerSkeleton *)imagePicker
                   clickData:(id<AUImagePickerDataProtocol>)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";
    }
    [AUToast presentModalToastWithin:self.view
                            withIcon:AUToastIconNone
                                text:msg
                            duration:1
                              logTag:@"demo"
                          completion:NULL];

}
```