Ant Technology

Mini Program User Guide

Document Version: 20230209

蚂蚁集团 ANT GROUP

Legal disclaimer

Ant Group all rights reserved©2022.

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

Trademark statement

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

Disclaimer

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

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	Danger: Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	Warning: Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	Notice: If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	Note: You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `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.Mini Program Development 14
1.1. Activate mPaaSAbout Mini Program 14
1.2. Integrate Mini Program into Android 15
1.2.1. Quick start 15
1.2.2. Advanced guide 18
1.2.2.1. Access real-device preview and debugging function 18
1.2.2.2. Customize navigation bar 19
1.2.2.3. Customize bi-directional channels 19
1.2.2.4. Extend API permissions 21
1.2.2.5. Customize startup loading page 23
1.2.2.6. Extend upper-right pop-up menu 25
1.2.2.7. Set entrance/exit animation 26
1.2.2.8. Specify the page to redirect to when starting the ... 27
1.2.2.9. Pass startup parameters to Android Mini Program 28
1.2.2.10. Preset an Android mini program in the client 29
1.2.2.11. Realize multiple instantiation of Android mini prog... 31
1.2.2.12. Customize View in Android Mini Program 32
1.2.2.13. Customize rendering parameters in custom View 34
1.2.2.14. Send custom message to custom View 34
1.2.2.15. Send custom event to the mini program 35
1.2.3. Instructions on upgrading Mini Program 35
1.2.4. Tutorial 36
1.2.4.1. Overview 36
1.2.4.2. Create a native project in Android Studio 36
1.2.4.3. Create an App in the mPaaS console 40
1.2.4.4. Integrate Mini Program to project through Native A... 41

1.2.4.5. Initialize configuration 44
1.2.4.6. Create and release a mini program 46
1.2.4.7. Start mini program 51
1.2.4.8. Access real-device preview and debugging 52
1.2.4.9. Customize bi-directional channel 55
1.2.4.10. Customize startup loading page 59
1.2.4.11. Custom navigation bar 62
1.3. Integrate Mini Program into iOS 72
1.3.1. Quick start 72
1.3.2. Advanced guide 84
1.3.2.1. Preview and debug iOS Mini Program on real device 84
1.3.2.2. Customize navigation bar of iOS Mini Program 85
1.3.2.3. Custom bi-directional channel for iOS Mini Program 87
1.3.2.4. Customize startup loading page of iOS Mini Progra... 88
1.3.2.5. Customize error page for iOS mini program 90
1.3.2.6. iOS Mini Program supports custom View 91
1.3.2.7. API permission extended configuration for iOS Mini ... 94
1.3.2.8. Preset an iOS mini program in the client 96
1.3.2.9. Add extended information for iOS Mini Program 98
1.3.2.10. Pass startup parameters to iOS Mini Program 98
1.3.3. Mini Program upgrade instructions 99
1.4. Develop mini program 100
1.4.1. Quick start 100
1.4.2. Advanced guide 102
1.4.2.1. Real-device preview and debugging 102
1.4.2.2. Extensions 103
1.4.2.3. Unregister custom event 107
1.5. OpenAPI in the console 108

1.5.1. Overview and Preparation 108
1.5.2. Interface description 110
1.6. Mini program base library 135
1.7. Frameworks 138
1.7.1. Overview 138
1.7.2. App 139
1.7.3. Page 143
1.7.4. View layer 146
1.7.5. Event 153
1.7.6. Style 155
1.7.7. Global configuration for mini program 156
1.7.7.1. Introducing global configuration for Mini Program 156
1.7.7.2. app-json global configuration 157
1.7.7.3. app acss global style 159
1.7.7.4. app.js registering Mini Program 159
1.7.7.5. getApp method 161
1.7.8. Mini Program page 161
1.7.8.1. Page introduction 161
1.7.8.2. Page configuration 162
1.7.8.3. Page structure 163
1.7.8.4. Page style 163
1.7.8.5. Page registration 163
1.7.8.6. getcurrentpages method 170
1.7.9. AXML 170
1.7.9.1. AXML introduction 170
1.7.9.2. data binding 170
1.7.9.3. Conditional rendering 173
1.7.9.4. List rendering 173

1.7.9.5. Template 175
1.7.9.6. Reference 176
1.7.10. SJS syntax reference 177
1.7.10.1. SJS introduction 177
1.7.10.2. Variable 178
1.7.10.3. Comments 179
1.7.10.4. Operator 179
1.7.10.5. Statement 181
1.7.10.6. Data type 183
1.7.10.7. Basic library 188
1.7.10.8. esnext 190
1.7.11. ACSS syntax reference 191
1.7.12. Event system 192
1.7.12.1. Event introduction 192
1.7.12.2. Event object 193
1.7.13. Custom components 195
1.7.13.1. Custom component introduction 195
1.7.13.2. Create custom component 195
1.7.13.3. Component configuration 195
1.7.13.4. Component template and style 196
1.7.13.5. Component object 199
1.7.13.6. Component lifecycle 202
1.7.13.7. mixins 204
1.7.13.8. Obtain component instances by ref 204
1.7.13.9. Use custom component 205
1.7.13.10. Publish custom component 206
1.7.14. Optimization 207
1.8. Basic components 209

1.8.1. Component overview 209
1.8.2. Component FAQ 211
1.8.3. View container 211
1.8.3.1. View 211
1.8.3.2. Swiper 212
1.8.3.3. Scroll view 213
1.8.3.4. Cover view 214
1.8.3.5. Movable view 215
1.8.4. Basic elements 216
1.8.4.1. Text 216
1.8.4.2. Icon 216
1.8.4.3. Progressbar 217
1.8.4.4. Rich text 218
1.8.5. Form component 220
1.8.5.1. button 220
1.8.5.2. form 222
1.8.5.3. label 223
1.8.5.4. input 224
1.8.5.5. textarea 227
1.8.5.6. radio 228
1.8.5.7. checkbox 230
1.8.5.8. switch 231
1.8.5.9. slider 232
1.8.5.10. picker-view 233
1.8.5.11. Bottom scroll selector 234
1.8.6. Navigator 236
1.8.7. Media component 236
1.8.7.1. image 237

1.8.7.2. video 242
1.8.8. Canvas 246
1.8.9. Map 248
1.8.10. Open components 253
1.8.10.1. web-view 253
1.9. Component extension 255
1.9.1. Overview 255
1.9.2. Navigation layout 256
1.9.2.1. List 256
1.9.2.2. Tabs 258
1.9.2.3. Vertical tabs 260
1.9.2.4. Card 261
1.9.2.5. Grid 262
1.9.2.6. Steps 263
1.9.2.7. Footer 264
1.9.2.8. Flex 265
1.9.2.9. Pagination 267
1.9.2.10. Collapse 268
1.9.3. Floating layer 270
1.9.3.1. Popover 270
1.9.3.2. Filter 271
1.9.3.3. Modal 272
1.9.3.4. Popup 273
1.9.4. Result class 274
1.9.4.1. PageResult 274
1.9.4.2. Message 275
1.9.5. Reminders 276
1.9.5.1. Tips 276

1.9.5.2. Notice 278
1.9.5.3. Badge 279
1.9.6. Form class 280
1.9.6.1. InputItem 280
1.9.6.2. PickerItem 283
1.9.6.3. AmountInput 284
1.9.6.4. SearchBar 286
1.9.6.5. AMCheckBox 287
1.9.7. Gesture class 288
1.9.7.1. Swipe action 288
1.9.8. Others 290
1.9.8.1. Calendar 290
1.9.8.2. Stepper 291
1.9.8.3. AMIcon 292
1.10. API 293
1.10.1. Overview 293
1.10.2. API list 293
1.10.3. UI 301
1.10.3.1. Navigation bar 301
1.10.3.2. tabBar 307
1.10.3.3. Route 313
1.10.3.4. Feedback 321
1.10.3.5. Pull down refresh 329
1.10.3.6. Contact 331
1.10.3.7. Choose city 332
1.10.3.8. Choose date 339
1.10.3.9. Animation 341
1.10.3.10. Canvas 344

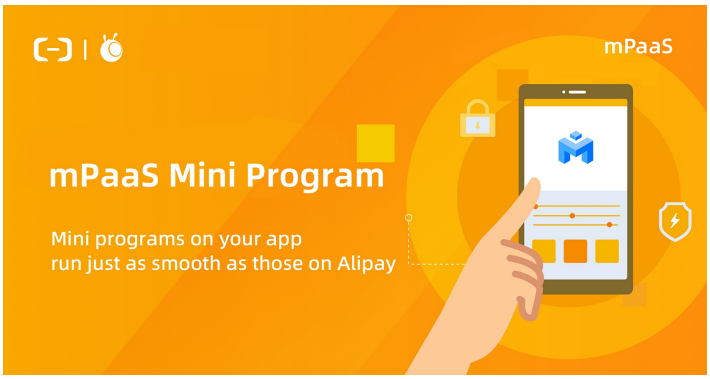
1.10.3.11. Keyboard 361
1.10.3.12. Scroll 362
1.10.3.13. Node query 362
1.10.3.14. Option selector 366
1.10.3.15. Multilevel select 368
1.10.3.16. Set background color 369
1.10.3.17. Set page pulldown 370
1.10.3.18. Settings 370
1.10.4. Multimedia 371
1.10.4.1. Image 371
1.10.4.2. Video 378
1.10.5. Cache 380
1.10.6. File 384
1.10.7. Location 387
1.10.8. Network 393
1.10.9. Device 405
1.10.9.1. Can I use 405
1.10.9.2. Obtain base library version 405
1.10.9.3. System info 406
1.10.9.4. Network status 409
1.10.9.5. Clipboard 410
1.10.9.6. Shake 411
1.10.9.7. Vibrate 412
1.10.9.8. Accelerometer 414
1.10.9.9. Gyroscope 415
1.10.9.10. Compass 415
1.10.9.11. Make phone call 416
1.10.9.12. User captures screen 417

1.10.9.13. Screen brightness 417
1.10.9.14. Add phone contacts 421
1.10.9.15. Scan code 427
1.10.9.16. Bluetooth API overview 428
1.10.9.17. Bluetooth API list 432
1.10.9.18. Bluetooth API error codes 460
1.10.9.19. Bluetooth API FAQ 461
1.10.10. Data security 462
1.10.11. Share 464
1.10.12. Mini program version type 468
1.10.13. Custom analysis 469
1.10.14. Custom API 469
1.10.15. Mini program jumping 470
1.10.16. Webview component control 471
1.11. Design guide 472
1.11.1. Principles 472
1.11.1.1. Clarity 472
1.11.1.2. Efficiency 474
1.11.1.3. Security 478
1.11.2. Visual guidelines 480
1.11.2.1. Color 480
1.11.2.2. Font 481
1.11.2.3. Icon 481
1.11.3. Component guidelines 482
1.11.3.1. Navigation 482
1.11.3.2. Enter information 484
1.11.3.3. Display information 491
1.11.3.4. Feedback 495

1.11.3.5. Gesture 501
1.11.3.6. Differentiated design 501
1.11.3.7. Combine components 503
1.12. FAQ 505
1.12.1. FAQ of using console 505
2.Mini Program Release 506
2.1. Introduction to Mini Program Release 506
2.2. Configure Mini Program package 506
2.3. Create a Mini Program package 506
2.4. Release a Mini Program package 508
2.5. Manage Mini Program package 508
2.6. Mini Program permission control 509
3.Mini Program Analytics 513
3.1. About Mini Program Analytics 513
3.2. Data overview 513
3.3. Mini program management 514
3.4. User analytics 515
3.5. Page analytics 516
3.6. Sharing analytics 517
4.Mini program monitoring 518

1.Mini Program Development

1.1. Activate mPaaSAbout Mini Program



Introduction

Mini Program is derived from Alipay Mini Framework and inherits Alipay Mini Framework's ease of development, cross-platform approach, and native performance, allowing you to embed mini programs in your own Apps and many other Apps, such as DingTalk, Taobao, and Alipay, and quickly build packages. Based on mPaaS Mini Program, you can fast optimize the release package size, and save traffic and storage space. The service iteration will no longer be restricted by version release, namely, you can fast release the mini programs and implement iteration on demand. With the unified development standards, you even can develop a mini program once and deploy it to multiple platforms.

Features

- **Unify development standards to realize "develop once, deploy anywhere"**

  Inheriting the native IDE of Alipay Mini Program, Mini Program provides the one-stop capability integrating development, debugging and release, greatly speeds the requirement iteration, and guarantees the release quality. As a brand-new mobile development mode, mPaaS Mini Program deeply integrates the HTML5's ease of development, cross-platform approach, and native performance, allowing you to code once and reuse them on multiple platforms.

- **Realize dynamic App release and update**

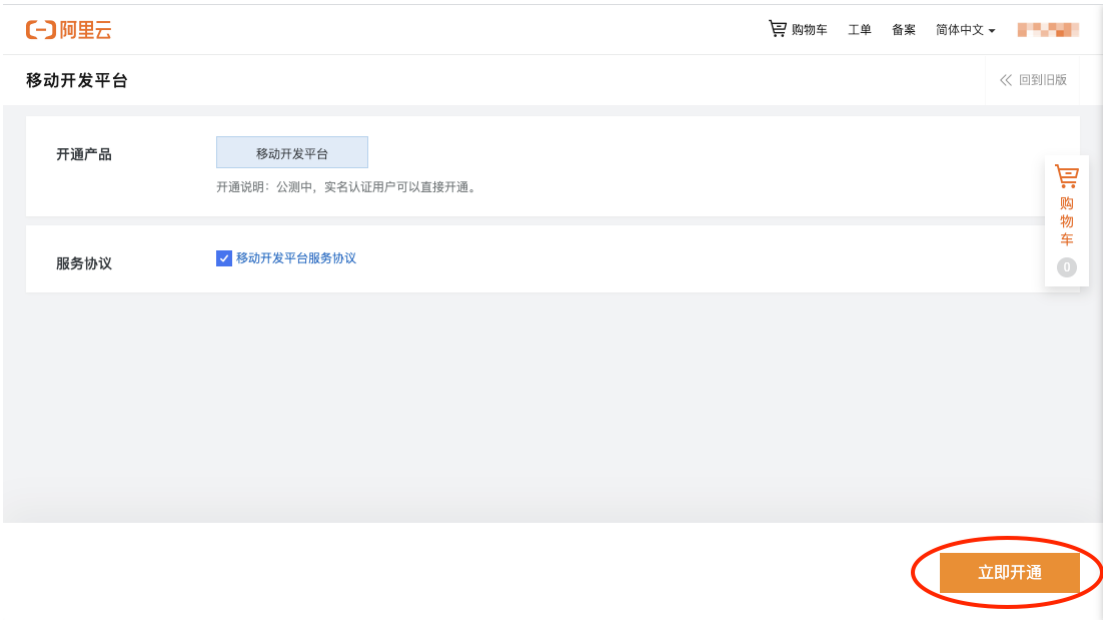
  New App versions, mini program packages, and switch configurations can be easily released through the mPaaS Mini Program. Mini Program publishing service supports official release and gray release. You can effectively verify if content to be released has any potential risks. Meanwhile, it supports publishing the App from multiple dimensions, such as whitelist, model, city, and OS version, thus realizing dynamic management of the overall application.

How to use



1. Activate Mini Program.

   Activate mPaaS, create an mPaaS App, and download the corresponding configuration file.



2. Introduce Mini Program.

   Use the mPaaS IDE plugin to add the Mini Program framework to you project. For more information, refer to:

   - Access Mini Program to Android client
   - Access Mini Program to iOS client

3. Develop and release a mini program.

   Use the Mini Program IDE to implement development and test, and run the mini program in your project. For more information, see Develop Mini Program.

1.2. Integrate Mini Program into Android

1.2.1. Quick start

You can access Mini Program SDK to Android client through Native AAR, mPaaS Inside or Portal & Bundle accessing method. For more information, see the Accessing method introduction.

Note: Mini Program is only available in 10.1.60 and higher baseline versions.

This section introduces how to use Mini Program in combination with the official Demo.

If you encounter problems during accessing the Mini Program, please scan the QR code with DingTalk to join the Mini program question answering group for consultation.

> **Note**
>
> If you encounter problems during accessing the Mini Program, please search the group number 31591197 with DingTalk to join the Mini program question answering group for consultation.

Prerequisites

Native AAR

1. Add the mPaaS to your project.
2. Add the Mini Program dependency. Install the Mini Program component through mPaaS Component Management (AAR) in the project.

mPaaS Inside

1. Complete the mPaaS Inside access process.
2. Add the Mini Program dependency. Install the Mini Program component through mPaaS Component Management in the project. For more information, see Manage component dependencies.

Portal & Bundle

1. Complete the mPaaS Portal & Bundle access process.
2. Add the Mini Program dependency. Install the Mini Program component through mPaaS Component Management in the Portal and Bundle projects. For more information, see Manage component dependencies.

Procedure

The procedure of accessing Mini Program contains the following steps:

1. Initialize configuration
   i. Initialize mPaaS
   ii. Configure mini program signature verification
   iii. Configure AndroidManifest

iv. Apply for UC kernel

2. Release a mini program
   i. Enter the mini program background
   ii. Configure virtual domain name
   iii. Create a mini program
   iv. Release the mini program
3. Start the mini program

1. Initialize configuration

1.1 Initialize mPaaS

If you access Mini Program through Native AAR or mPaaS Inside, you need to initialize mPaaS.You need to add the following code in `MyApplication`:

```
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS initialization callback
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // Initialize the mini program public resource package
                H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
            }
        });
    }

    @Override
    public void onCreate() {
        super.onCreate();
        // Initialize mPaaS
        QuinoxlessFramework.init();
    }
}
```

In `onPostInit`, the public resource packages are set as follows:

```
H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
```

If you cannot find the class `TinyAppCenterPresetProvider`, it may be that your baseline version is less than 10.1.68.7, please refer to Mini Program base library to process it.

1.2 Configure mini program signature verification

Create the file custom_config.json under the path `assets/config` in the Android project, and enter the following content in the file.

```
[
  {
    "value": "NO",
      "key": "h5_shouldverifyapp"
  }
]
```

As for the value, "NO" means disabling the Mini Program signature verification while "YES" for enabling the Mini Program signature verification. It defaults to "YES" in case of no input. In the development and debugging stage, it is suggested to enable signature verification. For specific operations, see Configure mini program packages.

Configure the time interval for requesting mini program packages

mPaaS supports configuring the time interval for requesting mini program packages. You can set the time interval globally or individually.

- Global setting: You can add the following code in custom_config.json:

  ```
  {
  "value": "{\"config\":{\"al\":\"3\",\"pr\":{\"4\":\"86400\",\"common\": \"864000\"},\"ur\":\"1800\",\"fpr\":{\"common\":\"3888000\"}},\"switch\":\"
  yes\ "}",
  "key": "h5_nbmngconfig"\
  }
  ```

  `\"ur\":\"1800\"` defines the global update interval, where 1800 is the default value, representing the time length in seconds. You can modify this value to set your global mini program package request interval which can be in the range of 0 ~ 86,400 seconds (namely 0 ~ 24 hours), and 0 means there is no limit on the request interval.

  Important: Do not modify other parameters at will.

- Individual setting: The interval setting only works on the current mini program package. You can log in to the mPaaS console, go to the Add Mini Program package > Extended information page, and fill in `{"asyncReqRate":"1800"}` to set the request time interval. For details, see the extended information in Create a mini program package.

Verify

Verify if the time interval configuration works. You can open a project that has accessed the Mini Program, and filter the keyword "H5BaseAppProvider" in the logcat log. If you can find the following information, it means the configuration has taken effect.

```
lastUpdateTime: xxx updateRate: xxx
```

1.3 Configure AndroidManifest

If you access Mini Program through Native AAR, you need to add the following configuration in `AndroidManifest.xml`.

```
<application>
    ...
<meta-data android:name="nebula.android.meta.enable" android:value="true"/>
  ...
</application>
```

1.4 Apply for UC kernel

Before using the Mini Program, you need to apply for and configure the UC kernel. For specific operations, refer to Apply for UC Kernel.

> **Note**
> Due to the change of product strategy, public application for UC Key will not be supported from December 1, 2022.

UC kernel endows Mini Program with the same-layer processing capability, such as embedding webview and embedding map, and gives it a better rendering experience.

2. Release a mini program

Before starting the mini program, you must release the mini program through the mPaaS console.

2.1 Enter the mini program background

Log in to the mPaaS console, and enter the **Mini Program** > **Release Mini Program** page from the left navigation pane.

2.2 Configure virtual domain name

If you are using Mini Program for the first time, please go to the **Mini Program** > **Release Mini Program** > **Manage configuration** page and configure the virtual domain name first. In principle, you need to use a second-level domain name managed by your company.

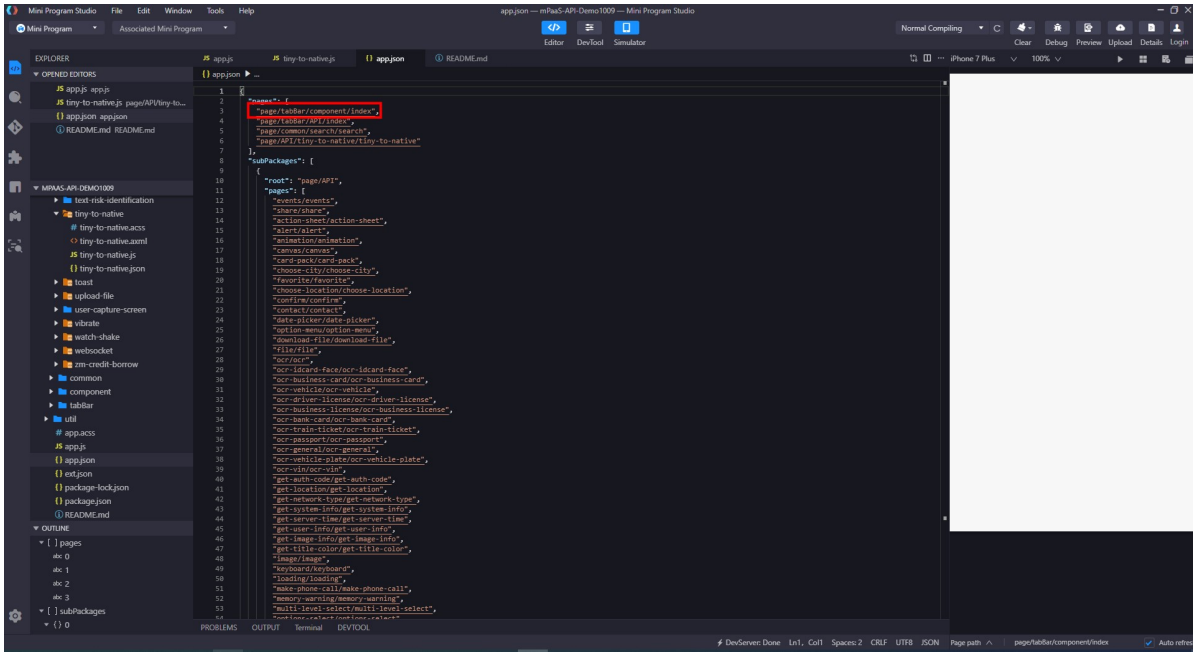
2.3 Create a mini program

Enter the mPaaS console, and complete the following steps:

1. Click the **Mini Program > Release Mini Program** menu from the left navigation pane.
2. On the mini program list page, click **New**.
3. In the **New Mini Program** window, fill in the mini program ID and name, then click **Submit**. The mini program ID is a 16-digit number, for example 2018080616290001.
4. In the mini program list, find the new mini program and click **Add**.
5. In the **Basic Information** column, complete the following configuration:
   - **Version**: Fill in the version number of the mini program package, for example 1.0.0.0.
   - **Client version range**: Select the applicable Android client version range for the mini program. Only when the App is in the version range, can the Mini Program start, otherwise it cannot start. The minimum version can be 0.0.0 while the maximum version can be left empty. In such case, the Mini Program can start in the Android client of all versions.

     Note: Ensure that the version is of the Android client, not the mini program.
   - **Icon**: Click Select a file to upload the icon of the mini program package. The icon is required when you create the mini program package for the first time. For example:
   - **File**: Upload the mini program package resource file which must be a .zip file. There is an mPaaS Mini Program demo for you (click here to download), you can upload it directly.

     Note: Before you upload the .zip file, you should rename the zip file and the file in the zip to the 16-digit ID of your mini program.
6. In the **Configuration information** column, complete the following configuration:
   - **Main entry URL**: Required. It is the first page of the mini program package, in the format of "/index.html#xxx/xxx/xxx/xxx", where "xxx/xxx/xxx/xxx" behind "#" is the first value in "pages" in the `app.json` of the mini program. As shown in the figure below, the main entrance of the mPaaS Mini Program demo is "/index.html#page/tabBar/component/index".



- As for the remaining configurations, keep the default setting.

7. Check **I have confirmed that the above information is accurate and will not be modified once submitted**.
8. Click **Submit**.

2.4 Release the mini program

1. Enter the **Mini Program > Release Mini Program > Manage official package** page from the left navigation pane.
2. On the mini program list page, select the target Mini Program and corresponding version, click **Create release task**.
3. In the release task creation column, complete the following settings:
   - **Release type**: select **Official**.
   - **Release description**: Optional
4. Click **OK** to complete the release task creation.

3 Start the mini program

After finishing the above steps, you can execute the following code in the Android project to start the demo mini program.

```
MPNebula.startApp("2018080616290001");
```

Note: The 2018080616290001 in the code refers to the mini program ID. The ID here is fore reference only, you should enter the actual program ID in practice.

1.2.2. Advanced guide

1.2.2.1. Access real-device preview and debugging function

Note: The real-device preview and debugging function is only supported in mPaaS 10.1.60 and above.

The procedure of accessing the real-device preview and debugging function is as follows:

1. Add the value of `h5_remote_debug_host` in HTML5 container configuration file (for example, `custom_config.json` in the demo project). The value is the server address for communication debugging.
   i. Open the Mini program IDE configuration file (`config.json`) downloaded from the mPaaS console, and find the field `debug_url`. An example of the configuration file is as follows:

   ```
   {
      "login_url":"https://mappcenter.cloud.alipay.com/ide/login",
      "uuid_url":"http://cn-hangzhou-mproxy.cloud.alipay.com/switch/uuid",
      "debug_url":"wss://cn-hangzhou-mproxy.cloud.alipay.com",
      "sign":"3decfd66c2924489204b4b0f38a9c228",
      "upload_url":"https://mappcenter.cloud.alipay.com/ide/mappcenter/mds"
   }
   ```

   ii. Add `h5_remote_debug_host` in the `custom_config.json` file, setting key as `h5_remote_debug_host` while value as `debug_url`, and adding `/host/` to the end. An example is as follows:

   ```
   [
   {
   "key": "h5_remote_debug_host",
   "value": "wss://cn-hangzhou-mproxy.cloud.alipay.com/host/"
   }
   ]
   ```

2. Set the virtual domain name used by the Mini program.
   i. On the mPaaS console, go to the **Mini Program > Release Mini Program > Manage configuration** page, and then obtain the domain name that you previously entered.



   ii. Open `MyApplication` in the project, and call the method `tinyHelper.setTinyAppVHost` before you start the mini program or the App starts to set the virtual domain name used by the mini program.

> **Note**
> You should replace `example.com` in the code sample with the actual virtual domain name.

```
MPTinyHelper tinyHelper = MPTinyHelper.getInstance();
tinyHelper.setTinyAppVHost("example.com");
```

3. Set the whitelist.

   When using the real-device preview and debugging function, you should configure the user ID on the client. Namely, the client will return the unique identifier of the App in the userId method according to the actual situation.

   Then, add the following code below the code block for setting the virtual domain name. The value you input in **Configure Whitelist** in the Mini Program IDE subsequently must be consistent with the userId configured here.

```
MPLogger.setUserId("your userId");
```

4. Access Code Scanner and parse the QR code for preview or debugging. Code sample of parsing the QR code and starting the mini program is as follows:
   - If your baseline version is 10.1.68.6 or higher, please use the following code to parse the QR code. You can view the baseline version in the **mPaaS > Component Management** menu.

```
//The first parameter refers to the uri of the QR code; the second one refers to the custom startup parameter. If no custom startup parameter is available, just fill new Bundle().

MPTinyHelper.getInstance().launchIdeQRCode(uri, new Bundle());
```

   - If your baseline version is earlier than 10.1.68.6, please use the following code to parse the QR code.

```
// uri corresponds to the content of QR code
String scheme = uri.getScheme();
if ("mpaas".equals(scheme)) {
        Bundle params = new Bundle();
        String appId = uri.getQueryParameter("appId");
        for (String key : uri.getQueryParameterNames()) {
            if (!"appId".equalsIgnoreCase(key)) {
               params.putString(key, uri.getQueryParameter(key));
            }
        }
        LauncherApplicationAgent.getInstance().getMicroApplicationContext()
            startApp(null, appId, params);
}
```

1.2.2.2. Customize navigation bar

Prerequisites

To activate the function of customizing navigation bar, you should add the following code in the `custom_config.json` file.

```
{
    "value": "NO",
    "key": "mp_ta_use_orginal_mini_nagivationbar"
}
```

Procedure

mPaaS Mini Program shares the same navigation bar with HTML5 Container. So, you can learn how to set the title bar with reference to the following documents:

- For 10.1.68 baseline
- For 10.1.60 and earlier baseline

1.2.2.3. Customize bi-directional channels

If the existing Mini Program APIs or events cannot meet your development requirements, you can extend on demand.

If the existing Mini Program APIs or events cannot meet your development requirements, you can extend on demand.

Mini Program calls native custom API

1. Customizes and registers API in client.
   - Customize API:

```
public class MyJSApiPlugin extends H5SimplePlugin {

      /**
       * Customize API
       */
      public static final String TINY_TO_NATIVE = "tinyToNative";

      @Override
      public void onPrepare(H5EventFilter filter) {
          super.onPrepare(filter);
          // Add in onPrepare
          filter.addAction(TINY_TO_NATIVE);
      }

      @Override
      public boolean handleEvent(H5Event event, H5BridgeContext context) {
          String action = event.getAction();
          if (TINY_TO_NATIVE.equalsIgnoreCase(action)) {
              JSONObject params = event.getParam();
              String param1 = params.getString("param1");
              String param2 = params.getString("param2");
              JSONObject result = new JSONObject();
              result.put("success", true);
              result.put("message", "parameters received by client:" + param1 + ", " + param2 + "\n Return the current package name of Demo:" + c
ontext.getActivity().getPackageName());
              context.sendBridgeResult(result);
              return true;
          }
          return false;
      }
  }
```

   - Register API: Register the API once globally before starting the Mini Program.

```
/*
* 1st parameter: Defines the full path of the API
* 2nd parameter: BundleName, fill "" if you adopt native AAR or mPaaS Inside accessing mode
* 3rd parameter: Object on which the API works, you can fill "page" directly
* 4th parameter: The API which works. Input the customized API in the form of String[]
*/
MPNebula.registerH5Plugin(MyJSApiPlugin.class.getName(), "", "page", new String[]{MyJSApiPlugin.TINY_TO_NATIVE});
```

2. Mini Program calls API.

```
my.call('tinyToNative', {
  param1: 'p1aaa',
  param2: 'p2bbb'
}, (result) => {
  console.log(result);
  my.showToast({
    type: 'none',
    content: result.message,
    duration: 3000,
  });
})
```

Native project sends custom event to Mini Program

1. Mini Program registers event.

```
my.on('nativeToTiny', (res) => {
  my.showToast({
    type: 'none',
    content: JSON.stringify(res),
    duration: 3000,
    success: () => {

    },
    fail: () => {

    },
    complete: () => {

    }
  });
})
```

2. Client sends the event.

```
H5Service h5Service = MPFramework.getExternalService(H5Service.class.getName());
final H5Page h5Page = h5Service.getTopH5Page();
if (null != h5Page) {
    JSONObject jo = new JSONObject();
    jo.put("key", value);
    // native project sends the event method to the mini program
    // 1st parameter refers to the event name; 2nd one refers to the parameter, 3rd one defaults to null
    h5Page.getBridge().sendDataWarpToWeb("nativeToTiny", jo, null);
}
```

Unregister custom event

If the custom event is unnecessary and needs to be canceled, you can unregister the custom event. For details, see Unregister custom event.

1.2.2.4. Extend API permissions

In Mini Program, some special APIs, such as Location, Camera and Album, usually prompt users for authorization, and the APIs can be executed only when the users grant permission.

In Mini Program, some special APIs, such as Location, Camera and Album, usually prompt users for authorization, and the APIs can be executed only when the users grant permission.

The Mini Program container allows the following extensions for API call:

- Customize text prompt. The access party can control the text and presentation style.
- Allow the access party to read and write permission configuration.

Note: The extended configuration only works when the Mini Program permission control is enabled.

Permission configuration

For the APIs that need users' authorization, it is required to configure a permission key. It is allowed that multiple APIs share the same key. For example, photo selection and code scanning may correspond to the permission key of Camera.

The table below lists Mini Program's existing default keys and the corresponding APIs.

Permissions	key	API
Camera	camera	scan, chooseImage, chooseVideo
Album	album	saveImage, saveVideosToPhotosAlbum, shareTokenImageSilent
Location	location	getLocation, getCurrentLocation
Microphone	audioRecord	startAudioRecord, stopAudioRecord, cancelAudioRecord

The container reads the permission configuration classes passed by the access party to process the API call permission.

```
public static class PermissionConfig {
    public String action;    // API name
    public String key;       // Permission key
    public String desc;      // Text to be presented, it is allowed to add the placeholder "%s", and the container will automatically fill the mini pr
ogram name

    public PermissionConfig() {
    }
}
```

Note: When `action` is `chooseImage` or `chooseVideo`, the key configured by the access party doesn't work. The container has two special logics processing the above two APIs. However, text is configurable.

Permission loading configuration

To configure the permission loading, you need to use the API `TinyAppPermissionExternProvider` provided by the container. The interface classes are as follows:

```
package com.alipay.mobile.nebula.provider;

import android.content.Context;

import java.util.List;

public abstract class TinyAppPermissionExternProvider {

    public interface PermissionCheckCallback {
        void accept();

        void deny();
    }

    public abstract List<PermissionConfig> loadPermissionCheckConfig();

    public abstract void showPermissionDialog(Context context, String appId, PermissionConfig config, PermissionCheckCallback callback);

    public abstract boolean shouldHandlePermissionDialog();
}
```

The method `loadPermissionCheckConfig` loads the permission configuration to the container.

Custom presentation

Present the authorization information in a customized way, and allow users to make confirmation.

1. Use the method `shouldHandlePermissionDialog` to return `true`.
2. The container calls the method `showPermissionDialog`, and the access party presents its custom style there.
3. When the users accept or reject the invocation, call the corresponding method of `PermissionCheckCallback`.

See the sample code below:

```
package com.mpaas.demo.nebula;

import android.content.Context;

import com.alipay.mobile.antui.dialog.AUNoticeDialog;
import com.alipay.mobile.nebula.provider.TinyAppPermissionExternProvider;

import java.util.ArrayList;
import java.util.List;

public class TinyExternalPermissionCheckProvider extends TinyAppPermissionExternProvider {

    private PermissionConfig create(String action, String key, String desc) {
        PermissionConfig config = new PermissionConfig();
        config.action = action;
        config.key = key;
        config.desc = desc;
        return config;
    }

    private List<PermissionConfig> permissionConfigs = new ArrayList<>();

    public TinyExternalPermissionCheckProvider() {
        permissionConfigs.add(create("saveFile", "file", "%s want to use your file storage"));
        permissionConfigs.add(create("getFileInfo", "file", "%s want to use your file storage"));
    }

    @Override
    public List<PermissionConfig> loadPermissionCheckConfig() {
        return permissionConfigs;
    }

    @Override
    public void showPermissionDialog(Context context, String action, PermissionConfig permissionConfig, final PermissionCheckCallback permissionCheck
Callback) {
        AUNoticeDialog dialog = new AUNoticeDialog(context, "Authorization reminder", permissionConfig.desc, "Accept", "Reject");
        dialog.setPositiveListener(new AUNoticeDialog.OnClickPositiveListener() {
            @Override
            public void onClick() {
                permissionCheckCallback.accept();
            }
        });
        dialog.setNegativeListener(new AUNoticeDialog.OnClickNegativeListener() {
            @Override
            public void onClick() {
                permissionCheckCallback.deny();
            }
        });
        dialog.show();
    }

    @Override
    public boolean shouldHandlePermissionDialog() {
        return true;
    }
}
```

Read/write configuration

The following methods can be called to read the configuration:

```
MPTinyHelper.getInstance().getMiniProgramSetting(appId)
```

Note:

- The Mini Program configuration is stored in two dimensions: application and user, so make sure that the application has called the method `MPLogger.setUserId`.
- If an API has never been called, the authorization status of the key value corresponding to the API cannot be obtained.

The following methods can be called to write configuration:

```
MPTinyHelper.getInstance().updateMiniProgramSetting(appId, key, isAllowed);
```

See the sample code below:

```
package com.mpaas.demo.nebula;

import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CompoundButton;

import com.alipay.mobile.antui.basic.AUSearchBar;
import com.alipay.mobile.antui.tablelist.AUSwitchListItem;
import com.alipay.mobile.framework.app.ui.BaseFragmentActivity;
import com.alipay.mobile.nebula.util.H5Utils;
import com.mpaas.nebula.adapter.api.MPTinyHelper;

import java.util.Map;

public class PermissionDisplayActivity extends BaseFragmentActivity {

    private ViewGroup mScrollView;

    private AUSearchBar mSearchInputBox;

    private Map<String, Boolean> permissions;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_permission);
        mScrollView = (ViewGroup) findViewById(R.id.scrollview);
        mSearchInputBox = (AUSearchBar) findViewById(R.id.search);
        mSearchInputBox.getSearchButton().setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                mScrollView.removeAllViews();
                final String appId = mSearchInputBox.getSearchEditView().getText().toString();
                permissions = MPTinyHelper.getInstance().getMiniProgramSetting(appId);
                for (Map.Entry<String, Boolean> entry : permissions.entrySet()) {
                    AUSwitchListItem item = new AUSwitchListItem(PermissionDisplayActivity.this);
                    final String key = entry.getKey();
                    item.setLeftText(key);
                    item.getCompoundSwitch().setChecked(entry.getValue());
                    item.getCompoundSwitch().setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
                        @Override
                        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                            MPTinyHelper.getInstance().updateMiniProgramSetting(appId, key, isChecked);
                        }
                    });
                    mScrollView.addView(item, new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT, H5Utils.dip2px(PermissionDisplayActivit
y.this, 48)));
                }
            }
        });
    }
}
```

1.2.2.5. Customize startup loading page

When the mini program is started, if the mini program has not been downloaded to the device, the mini program container will launch the loading page (as shown in the figure below) to prompt the users to wait. After the mini program is installed on the device, the loading page will be closed and the users will be redirected to the mini program.

When the mini program is started, if the mini program has not been downloaded to the device, the mini program container will launch the loading page (as shown in the figure below) to prompt the users to wait. After the mini program is installed on the device, the loading page will be closed and the users will be redirected to the mini program.



Implement a custom loading page

For the Android mini program, mPaaS supports customizing the content of loading page. You can follow the steps below to configure the loading page:

1. Implement `MPTinyBaseIntermediateLoadingView` class. The View implemented by this class will be inserted to the Activity where the loading page is. See the sample code below:

```
package com.mpaas.demo.nebula;

import android.content.Context;
import android.util.AttributeSet;
import android.view.LayoutInflater;
import android.widget.TextView;

import com.mpaas.nebula.adapter.api.MPTinyBaseIntermediateLoadingView;

public class TinyStartupLoadingView extends MPTinyBaseIntermediateLoadingView {

    private TextView tvAppName;

    private TextView tvAppId;

    private TextView tvTips;

    public TinyStartupLoadingView(Context context) {
        super(context);
        init();
    }

    public TinyStartupLoadingView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public TinyStartupLoadingView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        LayoutInflater.from(getContext()).inflate(R.layout.activity_loading, this, true);
        tvAppName = (TextView) findViewById(R.id.app_name);
        tvAppId = (TextView) findViewById(R.id.app_id);
        tvTips = (TextView) findViewById(R.id.tv_tips);
    }

    /**
     *  Call upon initialization, and the App ID will be passed in. Other information, such as App name, icon and version, may be empty.
     */
    @Override
    public void initView(AppInfo info) {
        tvAppName.setText(info.appName);
        tvAppId.setText(info.appId);
        tvTips.setText("loading");
    }

    /**
     * Call when it failed to obtain the mini program
     */
    @Override
    public void onError() {
        tvTips.setText("fail");
    }

    /**
     * Call when fetching the mini program App information, including App ID, name, icon and version
     */
    @Override
    public void update(AppInfo info) {
        tvAppName.setText(info.appName);
        tvAppId.setText(info.appId);
    }
}
```

2. Before the mini program is started (for example, when the App initialization is ongoing), enable custom configuration. See the code sample below:

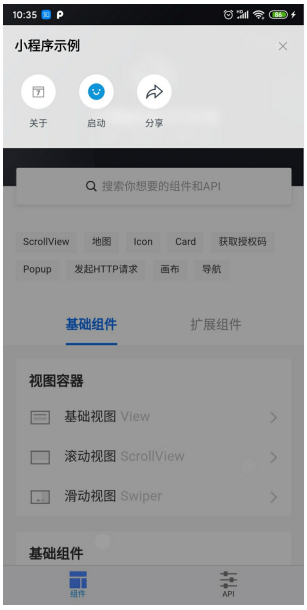
```
MPTinyHelper.getInstance().setLoadingViewClass(TinyStartupLoadingView.class);
```

3. If you need to operate on the Activity where the custom loading page is, such as interrupting the loading process and backing to the previous page, you can get the Activity through the base class method `getLoadingActivity()`. Pay attention to checking if the string is empty.

1.2.2.6. Extend upper-right pop-up menu

This section describes how to extend custom options in the pop-up menu and respond user's tapping event.

This section describes how to extend custom options in the pop-up menu and respond user's tapping event.



Procedure

1. Ensure that `mp_ta_showOptionMenu` configuration is enabled. See HTML5 container configuration for more information.
2. Use `TinyPopMenuItem.Builder` class to create `TinyPopMenuItem` object.
   - Builder class supports setting option name, icon (URL and drawable supported), and event callback event.
   - To call `setId` method of Builder class, you need to guarantee the uniqueness of ID.
3. The appId of the mini program and the path of the pop-up menu's page are carried in the second parameter of `onClick` method of the event callback object.
4. Set `TinyPopMenuProvider` instance object to the container before opening the mini program.

Code sample

```
H5Utils.setProvider(TinyPopMenuProvider.class.getName(), new TinyPopMenuProvider() {
    @Override
    public List<TinyPopMenuItem> fetchMenuItems(String appId) {
        List<TinyPopMenuItem> items = new ArrayList<>();
        TinyPopMenuItem urlItem = new TinyPopMenuItem.Builder()
                .setId("1000")
                .setIconUrl("https://gw-office.alipayobjects.com/basement_prod/3d46378a-6e4f-4aa1-820e-fd16da76b457.png")
                .setName("About")
                .setCallback(new TinyPopMenuItem.TinyPopMenuItemClickListener() {
                    @Override
                    public void onClick(Context context, Bundle bundle) {
                        String appId = bundle.getString("appId");
                        String path = bundle.getString("page");
                        Toast.makeText(context, "application ID=" + appId + ",page=" + path, Toast.LENGTH_LONG).show();
                    }
                })
                .build();
        items.add(urlItem);
        TinyPopMenuItem localItem = new TinyPopMenuItem.Builder()
                .setId("1001")
                .setIcon(getResources().getDrawable(R.drawable.smile))
                .setName("Start")
                .setCallback(new TinyPopMenuItem.TinyPopMenuItemClickListener() {
                    @Override
                    public void onClick(Context context, Bundle bundle) {
                        Toast.makeText(context, "Start" + bundle.toString(), Toast.LENGTH_LONG).show();
                    }
                })
                .build();
        items.add(localItem);
        return items;
    }
});
```

1.2.2.7. Set entrance/exit animation

The settings involved in this section is only applicable for entering and exiting from animation, not for the cross-page jumping in the mini program.

The settings involved in this section is only applicable for entering and exiting from animation, not for the cross-page jumping in the mini program.

Enable the function of entering and exiting from animation

Add the parameter `needAnimInTiny` after starting the Mini Program, and set the value as true. For example:

```
Bundle bundle = new Bundle();
bundle.putBoolean("needAnimInTiny", true);
MPNebula.startApp("2018080616290001", bundle);
```

Set animation entrance

Add the animation resource files (`tiny_fading_out.xml` and `tiny_push_up_in.xml`) in the main project.

- `tiny_fading_out.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<!--tiny_fading_out.xml-->
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="300">
    <translate android:fromYDelta="0%" android:toYDelta="100%" />
</set>
```

- `tiny_push_up_in.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<!--tiny_push_up_in.xml-->
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="300">
    <translate android:fromYDelta="100%" android:toYDelta="0%" />
</set>
```

Set animation exit

Add the animation resource files (`tiny_fading_in.xml` and `tiny_push_down_out.xml`) in the main project.

- `tiny_fading_in.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<!--tiny_fading_in.xml-->
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="300">
    <translate android:fromYDelta="100%" android:toYDelta="0%" />
</set>
```

- `tiny_push_down_out.xml` :

```
<?xml version="1.0" encoding="utf-8"?>
<!--tiny_push_down_out.xml-->
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="300">
    <translate android:fromYDelta="0%" android:toYDelta="100%" />
</set>
```

1.2.2.8. Specify the page to redirect to when starting the Mini Program

In certain scenarios, you need to specify the page to redirect to when starting the Mini Program. This topic describes how to implement this operation.

Prerequisites

You have accessed the mini program component by referring to Quick start.

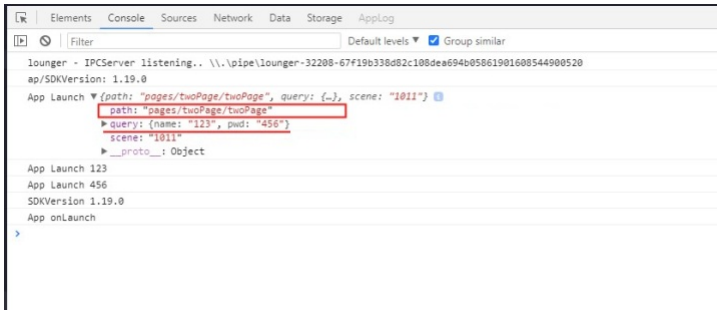
Procedure

1. Add parameter information of page redirection during startup on the client. The parameter passing method is as follows:

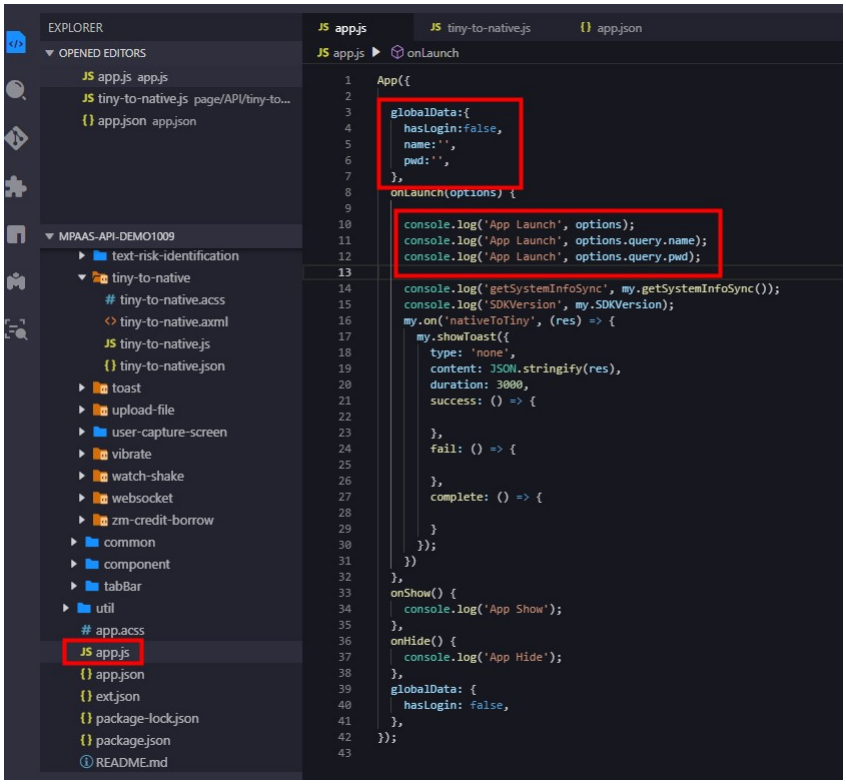
```
Bundle param = new Bundle();
String query = "name=123&pwd=456";
param.putString("query",query); // Set the parameter.
param.putString("page","pages/twoPage/twoPage"); // Set the path.
MPNebula.startApp(appId:"2020121620201216",param);
```

   - Description of Bundle parameters:
     - `query` : The parameter that is passed during Mini Program redirection, which is linked with `key=value` . Separate parameters with ampersand (&).
     - `page` : The redirection path of the Mini Program, which defaults to `pages/index/index` when not specified.
   - Description of startApp parameters:
     - `appId` : The app ID of the mini program, which can be viewed in the mPaaS console.
     - `param` : The Bundle object. You can pass request parameters to the Bundle object in the format of `key="query",value="Key-value pair"` . Separate parameters with ampersand (&).

2. Obtain the passed parameters in the Mini Program. To do this, the Mini Program obtains the parameters from the `options` parameter of the `onLaunch/onShow(options)` method.



The `app.js` storage script obtains the parameters passed from the client to the Mini Program and stores them to the `globalData` global variable. To use these parameters, the Mini Program directly obtains or updates the `globalData` value. After parameters such as `token` and `user_id` in the request header are passed from Native and stored in `globalData`, the Mini Program directly obtains the globalData value and use the parameters.



1.2.2.9. Pass startup parameters to Android Mini Program

In certain scenarios, you need to pass parameters to the default receiving page of the Mini Program, which is `pages/index/index`. This topic introduces the implementation process of this operation by passing `name` and `pwd` parameters as an example.

Prerequisites

You have accessed the mini program component by referring to Quick start.

Procedure

1. Add parameter information of page redirection during startup on the client. The parameter passing method is as follows:

```
Bundle param = new Bundle();
String query = "name="+Uri.encode("123")+"&pwd="+Uri.encode("456");
param.putSting("query",query); // set the parameter.
MPNebula.startApp(appId:"2020121620201216",param);
```

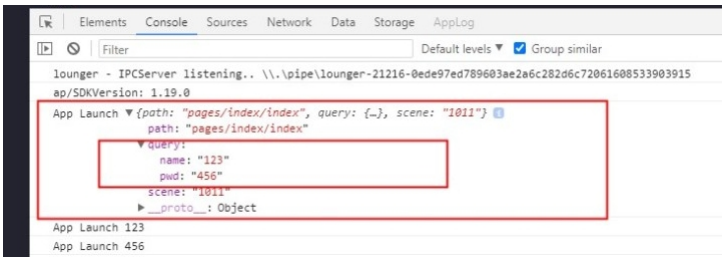
When the URL starts to pass parameters, the parameter passing field is `query`. When the Mini Program obtains the parameters, it parses the `query` field.

Description of startApp parameter:

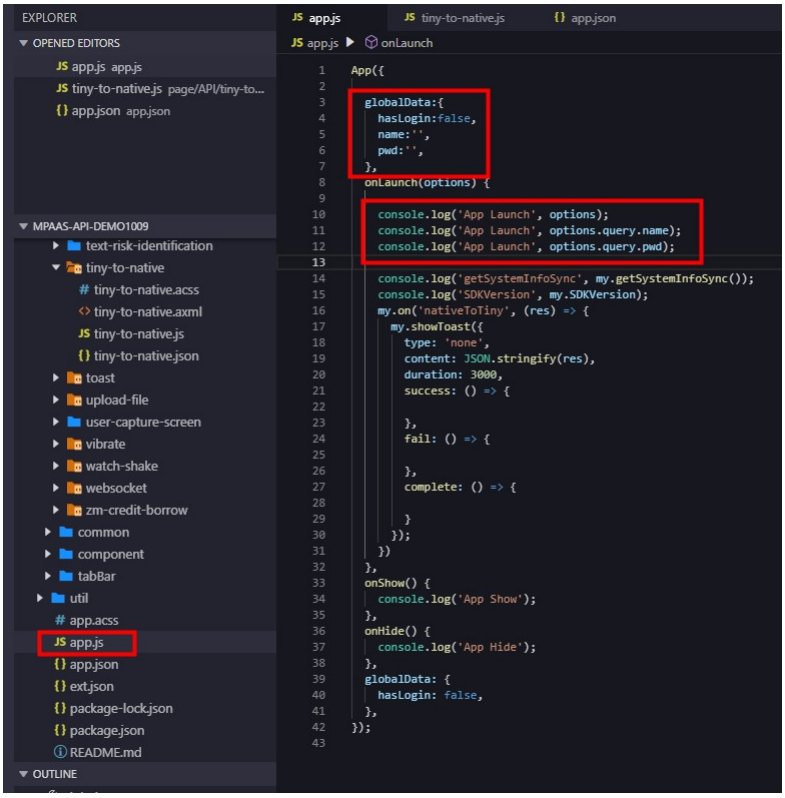
- `appId`: The Mini Program ID, which can be viewed in the mPaaS console.
- `param`: The Bundle object. You can pass request parameters to the Bundle object in the format of `key="query",value="Key-value pair"`. Separate parameters with ampersand (&).

> **Important**
>
> - The Mini Program framework decodes the `value` of the key-value pair for each custom input parameter with URI decode function. Therefore, you need to encode the encode the `value` of the input key-value pair with URI encode function.
> - The Mini Program framework does not process the `key` of the key-value pair for custom input parameters. Therefore, make sure not to set special characters for the `key`, otherwise the Mini Program framework may fail to recognize the custom parameters.

2.The Mini Program obtains the parameters. To do this, the Mini Program obtains the parameters from the `options` parameter of the `onLaunch/onShow(options)` method.



The `app.js` storage script obtains the parameters passed from the client to the Mini Program and stores them to the `globalData` global variable. To use these parameters, the Mini Program directly obtains or updates the `globalData` value. After parameters such as `token` and `user_id` in the request header are passed from Native and stored in `globalData`, the Mini Program directly obtains the globalData value and use the parameters.



1.2.2.10. Preset an Android mini program in the client

The traditional Mini Program technology can be easily affected by the network environment. Therefore, you may fail to pull the Mini Program package when the network condition is poor. However, you can avoid this problem by presetting the Mini Program. This topic introduces the principle and implementation process of presetting a Mini Program.

About presetting a Mini Program

Presetting a Mini Program is to pack the static resources of the Mini Program such as rendering, logic, and configuration into a compressed package. Then, the client downloads the Mini Program package to the local in advance and loads the resources locally. Presetting the Mini Program can minimize the impact of the network environment on the mPaaS Mini Program page. The benefits of using the preset package are as follows.

- **Improve the user experience**

  By embedding the static resources of the page into the App through the preset package and releasing the resources with the App, when you open the App for the first time, you can directly use the resources without relying on the network environment to download them.

- **Implement dynamic update**

  When a new version or emergency release is about to be launched for a Mini Program, you can conduct iterative development in the mini program IDE and release the new version in the mPaaS console. The Mini Program SDK integrated in the client will then automatically update the Mini Program to the latest version. This release method does not require App Store review, allowing users to receive updates in a timely manner.

The structure and usage

This topic describes how to preset a Mini Program from the following aspects:

- The structure of the mini program preset package
- The usage of the mini program preset package

The structure of the mini program preset package

The mini program preset package is a compression file with the `.amr` extension. By changing the extension from `.amr` to `.zip` and decompress the package, you can see HTML resources and JavaScript code. After the mini program container is loaded, these resources and code can be rendered in the UC kernel.

By taking the Android system as an example, the following figure shows the directory structure of a general resource package:

- Level-1 directory: It is generally the ID of the resource package, such as `2020121620201216_1.0.1.0.zip`.
- Level-2 and lower-level directories are custom resource files. They also specify the entry file that is opened by the current preset package by default, such as `/index.html`.



The usage of the mini program preset package

The process of using the mini program preset package is as follows.

1. Request the package information.

   This step is to request the mini program package from the server and store its information to the local database. The package information includes the download address and version number of the mini program package.

2. Download the mini program package.

   Download the mini program package from the server to your mobile phone.

3. Install the mini program package.

   Download directories and copy them to the installation directory on the mobile phone.

Prerequisites

- You have accessed the mini program component. For details about accessing the mini program component, see Get Started with Mini Programs.
- You have accessed the HTML5 container component. For details about accessing HTML5 containers, see Get Started with HTML5 Containers.

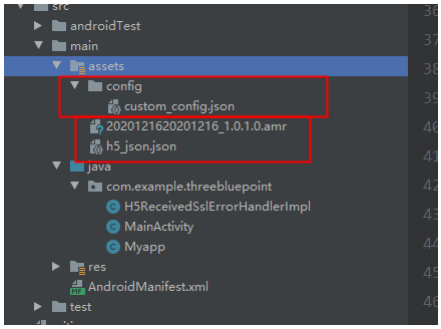
Procedure

1. Preset the mini program package.

   i. In the mPaaS console, release the Mini Program package and download the AMR and configuration files.

   ii. Place the downloaded AMR and configuration files to the assets directory of the mPaaS project.

iii. Add preset code in the project to call the preset code for App installation when the App starts. The code sample of presetting is as follows:

```
new Thread(new Runnable(){
@Override
public void run(){
 MPNebula.loadofflineNebula(jsonFileName: "h5_json.json",
                          new MPNebulaOfflineInfo(offLineFileName:"2020121620201216_1.0.1.0.amr",
                                                  addId:"2020121620201216",
                                                  version:"1.0.1.0"));
}
}).start();
```

Notes:

- This method is a blocking call, and therefore do not call the built-in preset package method in the main thread.
- This method can be called only once. If you call this method multiple times, only the first call is valid. For this reason, you need to pass in full preset package information at one time.
- If you need to build in multiple AMR packages, make sure that they already exist. Otherwise, other built-in preset packages cannot be imported.

2. Start the Mini Program. The code sample for starting the Mini Program is as follows:

```
/**
* Start the Mini Program.
*
* @param appId: The Mini Program ID.
*/
public static void startApp(String appId);
```

3. Update the Mini Program.

   By default, every time you open the App, the Mini Program SDK attempts to check if a newer version is available. The check interval is restricted to 30 minutes by default to minimize the stress on the server. To instantly check for the latest available version, call the following code to request an update. Generally, you can call the code after the App starts or the user logs in.

```
MPNebula.updateAllApp(new MpaasNebulaUpdateCallback(){
 @Override
 public void onResult(final boolean success, final boolean isLimit) {
     super.onResult(success, isLimit);
     runOnUiThread(new Runnable() {
         @Override
         public void run() {
             AUToast.makeToast(NebulaAppActivity.this,
                     success ? R.string.update_success : R.string.update_failure, 2000).show();
         }
     });
 }
});
```

4. Check the security signature.

   The Mini Program has a signature verification mechanism to prevent malicious programs from tampering with the mini program package downloaded to the device. To enable this mechanism, call the MPNebula API to configure verification parameters. If you are using the 10.1.60 baseline or later, you must enable container configuration additionally. For details, see Configure HTML5 Containers.

   Notes:

   - Be sure to call the MPNebula API before opening the mini program package for the first time. Otherwise, public key initialization will fail. For details about public and private keys, see Configure the mini program Package > Manage keys.
   - Signature verification is forced on mobile phones that are determined as rooted no matter signature verification is enabled or not on the client.

```
/**
* @param publicKey The public key for signature verification.
*/
public static void enableAppVerification(final String publicKey)
```

5. Delete the local Mini Program.

   Nebula provides the API for deleting local App information. After local App information is deleted, opening the App again requests the server to download and update local Mini Program information again.

```
public class MPNebula {
 // appId indicates the mini program app ID.
 public static boolean deleteAppInfo(String appId);
}
```

Note: The minimum baselines versions that supports this API are 10.1.68.8 for 10.1.68 series and 10.1.60.14 for 10.1.60 series.

1.2.2.11. Realize multiple instantiation of Android mini program

When receiving the push message, click on the notification bar to jump to the opened mini program page, and click the back button to the previous mini program page. When multiple instances are started by the mini program with the same AppId, you only need to pass different parameters.

Procedure

1. Configure the property of `h5_tiny_multiApp` for `assets`.

   Set the `h5_tiny_multiApp` property in `custom_config.json` under the `assets` folder.

```
[
{
 "value": "NO",
 "key": "h5_tiny_multiApp"
}
]
```

2. Configure startup parameter.

   You need to set `appClearTop` and `startMultApp` for mini program startup, the configuration is as follows:

```
Bundle param = new Bundle();
// multiple instances of the applet can be started after setting these two properties.
param.putBoolean("appClearTop", false);
param.putString("startMultApp", "YES");
MPNebula.startApp("2021042820210428",param);
```

1.2.2.12. Customize View in Android Mini Program

The custom View function of the Android Mini Program is only supported in mPaaS 10.1.68.29 and later versions.

The custom View function of the Android Mini Program is only supported in mPaaS 10.1.68.29 and later versions.

Upgrade baseline

1. Upgrade the baseline to 10.1.68.29 or later version, and add the Mini Program component to the project. Refer to mPaaS Upgrade Guide.
2. Use the customized appx base library. The code example is as follows:

```
 dependencies {
     ...
     implementation ('com.mpaas.tinyapp.commonres:tinyappcommonres:1.14.2-beta4.1') {
         force = true
     }
     ...
 }
```

Realize custom View

Customize a class to inherit `MPBaseEmbedView`, implement the `getView` method, then obtain the Android View and return it to the mini program.

```
public class MyTestEmbedView extends MPBaseEmbedView {
     @Override
    public View getView(int width, int height, String viewId, String mType, Map<String, String> params) {
        // Return the true Android view.
        return mRealView;
    }
}
```

Register custom View

Call the `MPEmbedViewHelper.registerEmbedView` to register a custom view before `QuinoxlessFramework` is called. The `registerEmbedView` is no time-consuming and does not affect startup performance.

```
MPEmbedViewHelper.registerEmbedView("com.mpaas.demo.nebula.MyTestEmbedView", "custom_map");
```

> **Note**
> - The parameter `"com.mpaas.demo.nebula.MyTestEmbedView"` represents the complete path of the custom View.
> - The parameter "custom_map" represents the type of the custom View, which also needs to be filled in on the mini program. Add a prefix is recommended.

Call custom View

The sample of of calling custom View by mini program is as follows:

```
<mpaas-component
  id="mpaas-map"
  type="custom_map"
  style="{{ width: 200, height: 200 }}"
/>
```

> **Note**
> - The `mpaas-component` is a fixed value, please do not modify it.
> - `id` is the ID of the custom View instance, please do not repeat it in a single mini program.
> - `type` is the type of custom View, it should be consistent with the third parameter of the client's registered custom View. Add a prefix is recommended.
> - `style`, in which input the width and height.

Callback of MPBaseEmbedView

All callback functions of custom View are as follows:

```
public class MPBaseEmbedView{
```

```
/**
 * The method is called when the custom View is instantiated. It is the first callback method to be called.
 *
 * @param context
 * @param h5Page
 */
public void onEmbedViewCreate(Context context, H5Page h5Page) {
}

/**
 * Get the embeded View instance.
 *
 * @param width  Customize the width of view.
 * @param height Customize the height of view.
 * @param viewId Customize view's self-increment id, which can be ignored.
 * @param mType  Fixed value "application/view", which can be ignored.
 * @param params Parameter
 * @return
 */
@Override
public View getView(int width, int height, String viewId, String mType, Map<String, String> params) {
}

/**
 * Call back when custom View is attached to webview.
 *
 * @param width  Customize the width of view.
 * @param height Customize the height of view.
 * @param viewId Custom view's self-increment id, which can be ignored.
 * @param mType  Fixed value "application/view", which can be ignored.
 * @param params Parameter
 */
@Override
public void onEmbedViewAttachedToWebView(int width, int height, String viewId, String mType, Map<String, String> params) {

}

/**
 * Call back when custom View is detached from webview.
 *
 * @param width  Customize the width of view.
 * @param height Customize the height of view.
 * @param viewId Customize view's self-increment id, which can be ignored.
 * @param mType  Fixed value "application/view", which can be ignored.
 * @param params Parameter
 */
@Override
public void onEmbedViewDetachedFromWebView(int width, int height, String viewId, String mType, Map<String, String> params) {

}

/**
 * Call back when the custom View is destoryed.
 *
 * @param width  Customize the width of view.
 * @param height Customize the height of view.
 * @param viewId Customize view's self-increment id, which can be ignored.
 * @param mType  Fixed value "application/view", which can be ignored.
 * @param params Parameter
 */
@Override
public void onEmbedViewDestory(int width, int height, String viewId, String mType, Map<String, String> params) {

}
/**
 * Call back when webview is resumed.
 */
@Override
public void onWebViewResume() {

}

/**
 * Call back when webview is paused.
 */
@Override
public void onWebViewPause() {

}

/**
 * Call back when webview is destroyed.
 */
@Override
public void onWebViewDestory() {

}
```

```
    /**
     * Called when receiving the context instruction of the mini program.
     *
     * @param actionType    Context name
     * @param data          Context parameter
     * @param bridgeContext callback bridge
     */
    @Override
    public void onReceivedMessage(String actionType, JSONObject data, H5BridgeContext bridgeContext) {

    }

    /**
     * The rendering instructions when custom View is created.
     *
     * @param data          Rendering data
     * @param bridgeContext Callback bridge, which can be ignored.
     */
    @Override
    public void onReceivedRender(JSONObject data, H5BridgeContext bridgeContext) {

    }
}
```

Sample code

- Android CustomView
- Mini Program CustomView

1.2.2.13. Customize rendering parameters in custom View

In Android Mini Program, the rendering parameter customization function of custom View is only supported in mPaaS 10.1.68.29 and later versions. If the current baseline version is lower than 10.1.68.29, please refer to mPaaS Upgrade Guide to upgrade the baseline version to 10.1.68.29.

Add custom rendering parameters in the tab of mini program

```
<mpaas-component
  id="mpaas-map"
  type="custom_map"
  style="{{ width: 200, height: 200 }}"
  color="#FFFF00FF"
  ...
/>
```

Note

- `color` is a custom rendering parameter, which can be named arbitrarily. But the name of the custom rendering parameter cannot start with 'on', and the type cannot be func.
- `id`, `type`, and `style` are the default fields, please do not use these fields as custom rendering parameters of custom views.

Receive and render custom rendering parameters

The client rewrites `onReceivedRender` to receive the rendering parameters from the mini program, and calls `Android View` to render the parameters.

```
public class MyTestEmbedView extends MPBaseEmbedView {
    ...
    @Override
    public void onReceivedRender(JSONObject jsonObject, H5BridgeContext h5BridgeContext) {
        LoggerFactory.getTraceLogger().debug(TAG, "onReceivedRender: " + jsonObject.toString());
        mRealView.render(jsonObject);
    }
    ...
}
```

1.2.2.14. Send custom message to custom View

The function of sending custom message to custom View is only supported in mPaaS 10.1.68.29 and later. When the current baseline version is lower than 10.1.68.29, please refer to mPaaS 10.1.68 upgrade guide to upgrade the baseline version to 10.1.68.29.

Create custom View context

`'mpaas-map'` is the value of `id` in the mini program tab, fill in your id here.

```
this.context = my.createMpaasCustomComponentContext('mpaas-map');
```

Send message to the client by context

`actiontype` is the name of the message event received by the client and `data` is the extension parameter.

```
this.context.postMessage({
      actionType: 'setColor',
      data: {
        "color": "#FF00FF00"
      }
    });
```

Client receives message and processes native View

In the `onReceivedMessage` method, the first parameter is the `actionType` passed in by the mini program, the second parameter is the extension parameter passed in, and the third parameter is `bridge` (can be ignored).

```
public class MyTestEmbedView extends MPBaseEmbedView {
    ...
    @Override
    public void onReceivedMessage(String actionType, JSONObject data, H5BridgeContext bridgeContext) {
        LoggerFactory.getTraceLogger().debug(TAG, "onReceivedMessage, actionType: " + actionType + ", data: " + data.toString());
        if("setColor".equals(actionType)){
            mRealView.setColor(Color.parseColor(data.getString("color")));
        }
    }
    ...
}
```

1.2.2.15. Send custom event to the mini program

The function of sending custom event to mini program through Android custom View is only supported in mPaaS 10.1.68.29 and later. When the current baseline version is lower than 10.1.68.29, refer to mPaaS Upgrade Guide to upgrade the version to 10.1.68.29.

Add custom event callback in the tag of mini program

Add custom event callback in `xxx.axml`.

```
<mpaas-component
  id="mpaas-map"
  type="custom_map"
  style="{{ width: 200, height: 200 }}"
  color="#FFFF00FF"
  onAnimationStart="onAnimationStart"
/>
```

`onAnimationStart` is a custom callback event, and the name of the custom event should start with `on`.

Handle custom events in JavaScript

```
onAnimationStart(data) {
  my.showToast({
    type: 'success',
    content: `onAnimationStart: ${JSON.stringify(data)}`,
  });
},
```

Trigger client custom View event

```
JSONObject data = new JSONObject();
data.put("sth", "start");
mMPBaseEmbedView.sendEventToTiny("onAnimationStart", data);
```

`mMPBaseEmbedView` is an instance of the class. The first parameter in the `sendEventToTiny` is the name of event callback, which needs to be consistent with the mini program; the second parameter is the event parameter.

1.2.3. Instructions on upgrading Mini Program

When the Mini Program SDK upgrades to 10.1.60, the following changes may occur.

When the Mini Program SDK upgrades to 10.1.60, the following changes may occur.

Changes on API

Add `MPTinyHelper` class, which is used to configure the information required for Mini Program API, real-device preview and debugging.

- Set App name: Obtain the App name through the `getSystemInfo` API.

  ```
  public void setAppName(String name)
  ```

- Set App version: Obtain the App version through the `getSystemInfo` API.

  ```
  public void setVersionName(String versionName)
  ```

- Set the virtual domain name of mini program: The real-device debugging depends on the virtual domain name to parse requests.

  ```
  public void setTinyAppVHost(String vhost)
  ```

Changes on project configuration

If you access Mini Program through the mPaaS Inside or Portal & Bundle method, you should:

- Ensure the version of `com.alipay.android:android-gradle-plugin` is 3.0.0.8.0 or higher.
- In the `build.gradle` file of the main project module, add `enableNebulaMetaInfo` and `useMetaInfoClass` in the portal filed. For example:

```
  portal {
      // Due to space limitation, the existing configurations are omitted here. Do not delete the configurations in practice. Do not add the portal configuration repeatedly in the Gradle file.
      enableNebulaMetaInfo true
      useMetaInfoClass true
  }
```

1.2.4. Tutorial

1.2.4.1. Overview

In the process of accessing mPaaS for development, the most common scenario is to access mPaaS based on the existing native Android project.The tutorial demonstrates the whole development process based on such scenario, from creating an Android native project, accessing mPaaS, accessing Mini Program for development to finally releasing a mini program and starting it.

Scenario

In the process of accessing mPaaS for development, the most common scenario is to access mPaaS based on the existing native Android project.The tutorial demonstrates the whole development process based on such scenario, from creating an Android native project, accessing mPaaS, accessing Mini Program for development to finally releasing a mini program and starting it.

Access method selection

mPaaS supports three access methods. If you want to easily access and use mPaaS just like using other SDK, you can adopt the Native AAR access method.Native AAR access method refers to using the native Android AAR packaging solution, which is close to Android technology stack. You don't have to learn mPaaS-related packaging knowledge, and can easily integrate the mPaaS to your project through the mPaaS Android Studio plugin or through Maven's pom and bom. This method greatly reduces access cost and makes it easier for you to use mPaaS. This method is suitable for the customers who have no special requirements on componentization (Portal & Bundle) access method, and want to quickly use mPaaS capabilities.The tutorial adopts Native AAR access method to access mPaaS.

What to learn

1. Create a native project in Android Studio: This project will realize the simple function of popping up a Toast by tapping text.
2. Create an App in the mPaaS console: This is the basic step for using the mPaaS console.
3. Integrate Mini Program to project through Native AAR: This access method is recommended in the tutorial.
4. Initialize configuration: Perform necessary project configuration after Mini Program is integrated.
5. Create and release a mini program: Create and release a mini program in the Mini Program IDE. You can download the Mini Program demo here.
6. Start the mini program: Replace the pop-up Toast related code with the code for starting the mini program. You can download the Android project demo mini program in the project.

1.2.4.2. Create a native project in Android Studio

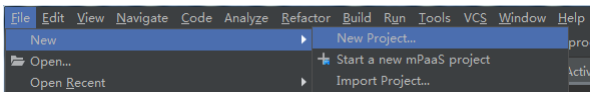
This section introduces how to create a native project where a Toast pops up upon click on text, and obtain an APK installation package. The procedure falls into four steps:

1. Create a project
2. Compile code
3. Create signature file and sign the project
4. Install App on mobile phone

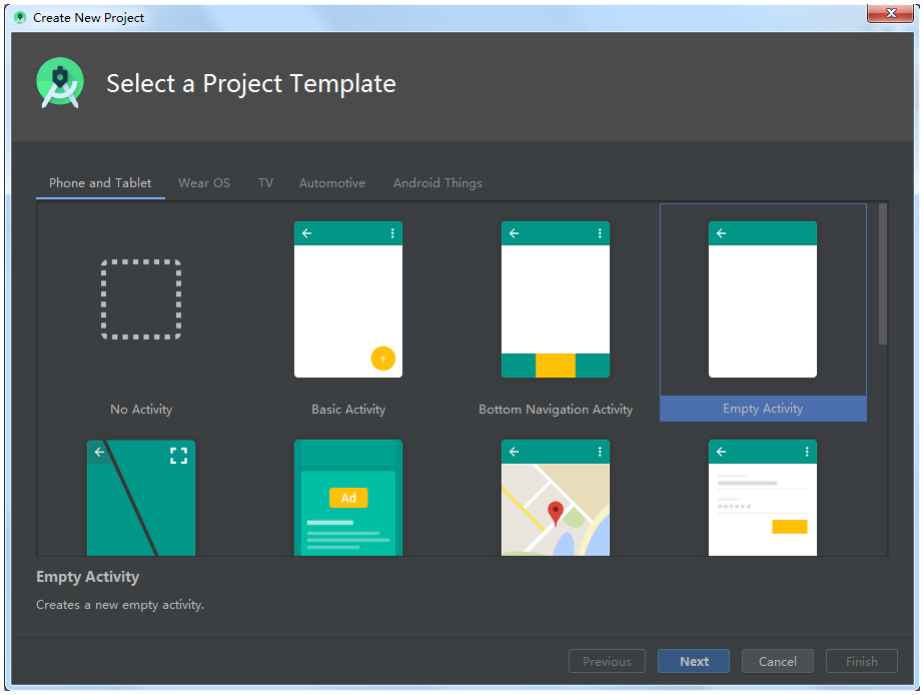
If you already have a native Android project which has been signed, you can skip this step, and directly Create an App in the mPaaS console.

Create a project

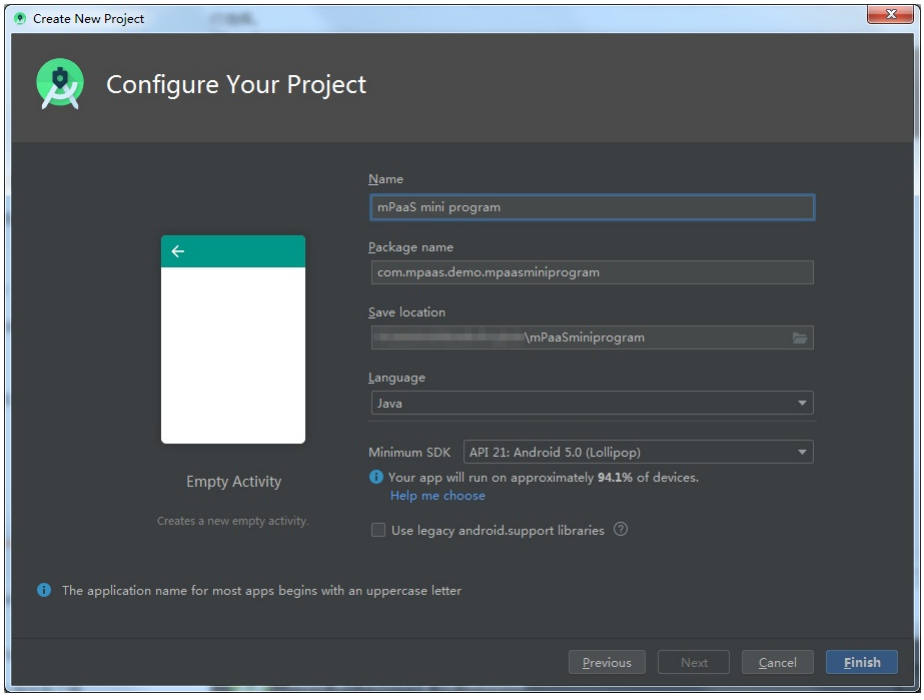
1. Open Android Studio, and click **File** > **New** > **New Project** to create a project.



2. In the pop-up window, select **Empty Activity**, and click **Next**.



3. Enter the **Name**, **Package name** (if any, you can use the default name), and **Save location**, with mPaaS mini program for example, and select **API 21: Android 5.0 (Lollipop)** as Minimum SDK.



4. Click **Finish** to complete creating the project.

Compile code

1. Open the file `activity_main.xml` under `res/layout`, and add a TextView with ID as "my_tv".

```
android:id="@+id/my_tv"
```

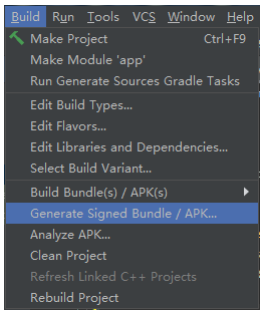
2. Open the class `MainActivity`, and add the following code to set the click event on text.

```
    findViewById(R.id.my_tv).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(MainActivity.this, "Hello mPaaS!", Toast.LENGTH_SHORT).show();
        }
    });
```

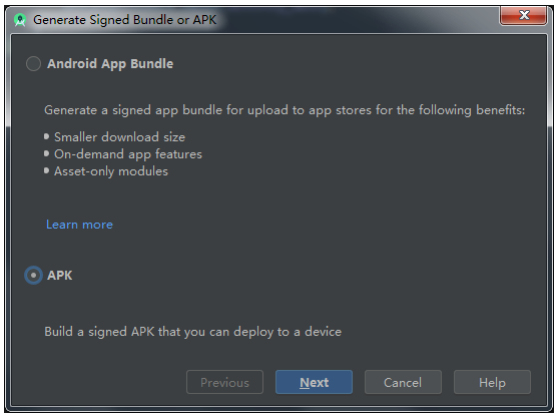
3. Compile and run the code. If the code runs successfully, it means you have completed the code compilation.

Create signature file and sign the project

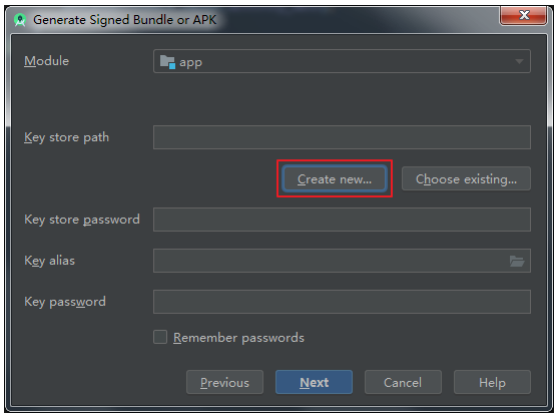
1. In the Android Studio, click **Build > Generate Signed Bundle / APK**.



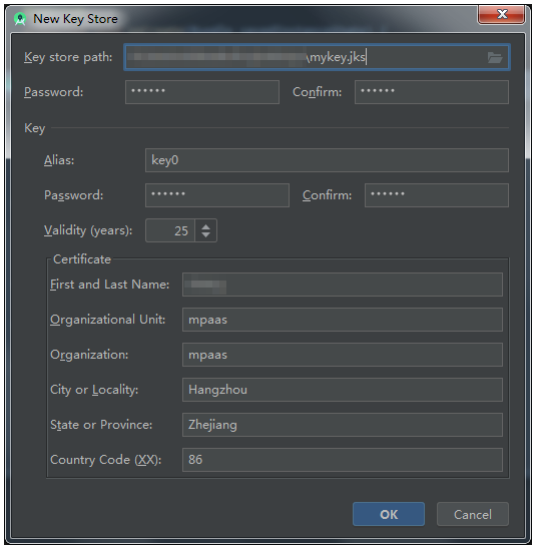
2. In the pop-up window, select **APK**, and click **Next**.



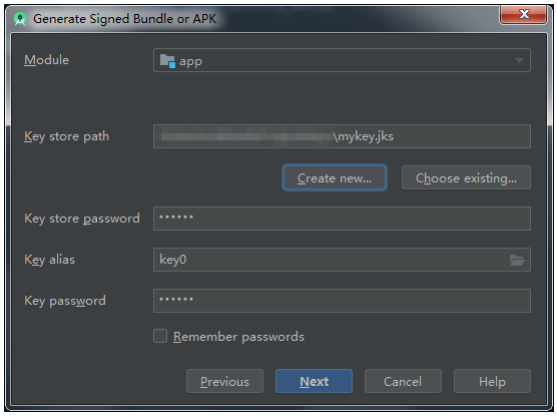
3. Select **Create new**.



4. Enter the corresponding information, and click **OK**. Then the signature is created. You can find the signature file in the **Key store path** you specified.

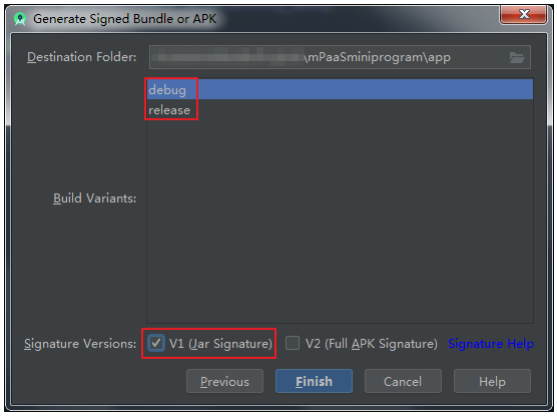


5. After filling the fields, click **Next** to sign the project.

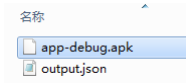


6. Select **Build Variants** on demand, and keep it in mind. When you use the encrypted file, the Build Variants must be consistent with that used in encrypted file generation.

   Check **V1 (Jar Signature)** as the signature version. The option V1 (Jar Signature) is required, while V2 (Full APK Signature) is optional.

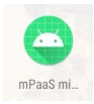


7. Click **Finish**. A moment later, you will find the signed APK installation package in the `debug` folder under `~\mPaaSminiprogram\app\debug`. In this tutorial, the installation package is named "app-debug.apk".

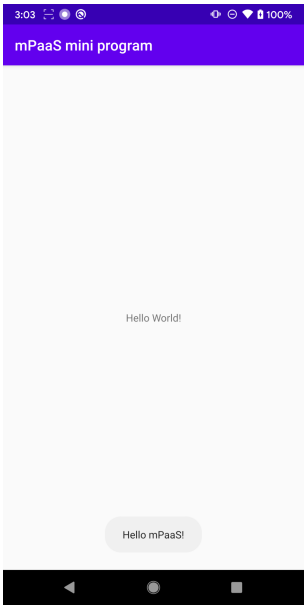


Install App on mobile phone

1. Connect your mobile to the computer, and turn on the USB debugging mode.
2. Run the project in Android Studio to install the App on your mobile.



3. Open the App on the mobile, click the text **Hello World!**, and a Toast "Hello mPaaS!" appears. It means the App has been successfully installed and the expected function is realized.

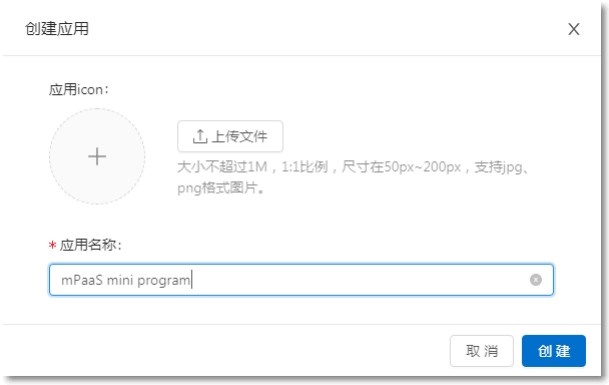


1.2.4.3. Create an App in the mPaaS console

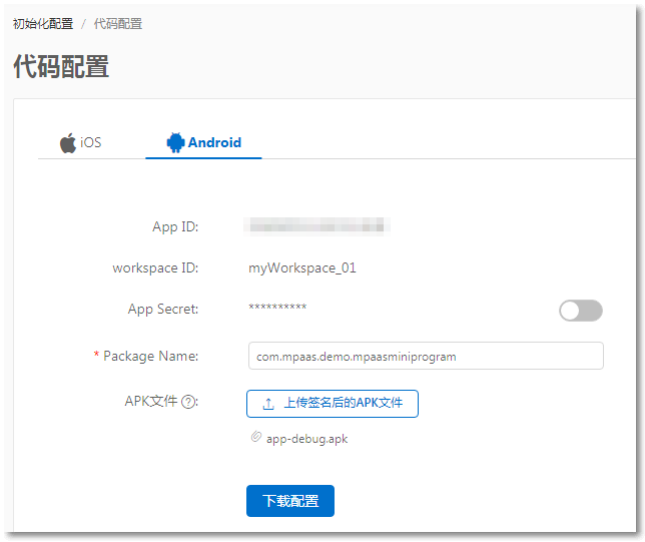
1. Log in to the mPaaS Console.
2. Click **+ Create an application** to create an mPaaS App.



3. Enter the App name, and click **Create**.



4. Enter the newly created App, click **Code Configuration > Android > Download now**, input the Package Name (for example: com.mpaas.demo.mpaasminiprogram), upload the compiled and signed APK package, and then click **Download configuration** to download the configuration file.



5. The downloaded configuration file is a compressed package. You will get a `.config` file and a `.jpg` encrypted picture after decompressing the package.

1.2.4.4. Integrate Mini Program to project through Native AAR

This section introduces how to integrate Mini Program to the project with Native AAR method.

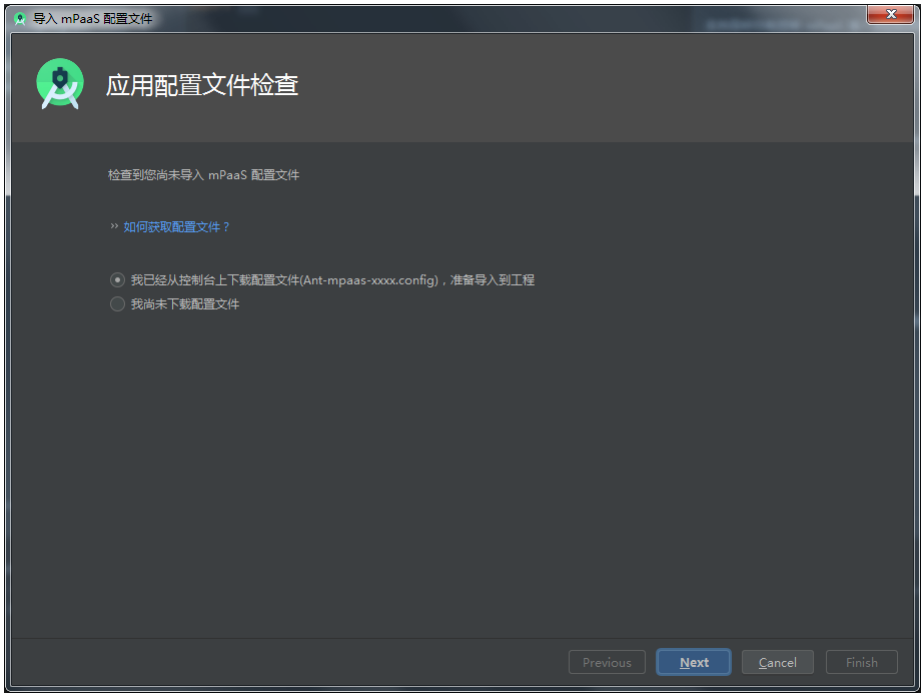
1. In the Android Studio, select **mPaaS** > **Native AAR** from the top menu bar.



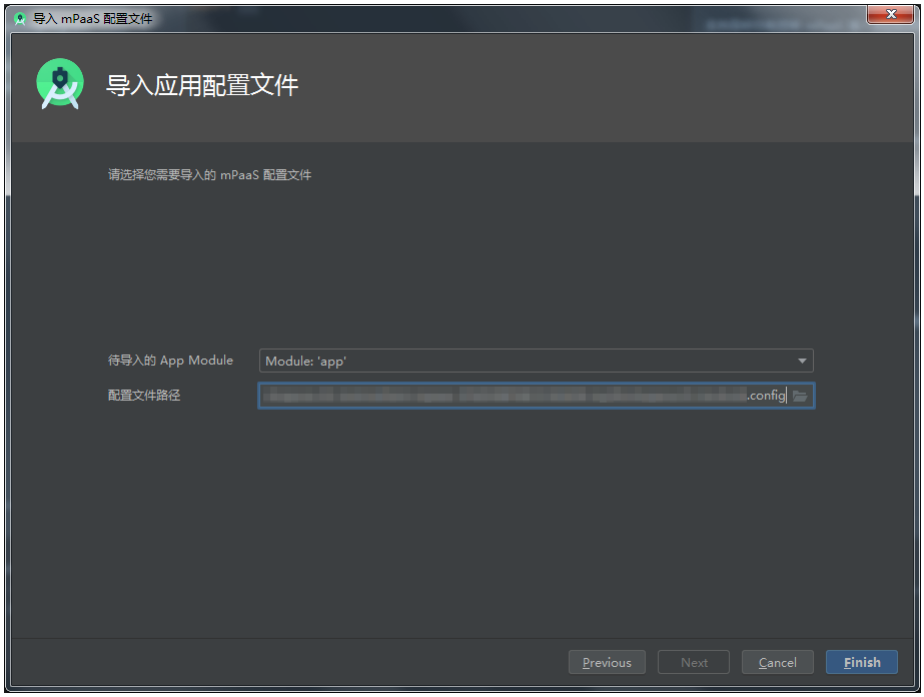
2. On the right pop-up panel, click **Import** under **Import App configuration**.



3. In the pop-up **Import mPaaS configuration file** window, select **I have downloaded the configuration file from mPaaS console**, and then click **Next**.



4. Select the downloaded configuration file, and then click **Finish** to import it.



5. When the configuration file is successfully imported, you will be prompted.



6. Click **Configure** under **Access/Upgrade Baseline** on the right panel.



7. In the pop-up **Select mPaaS Baseline Version** window, select 10.1.68, and then click **OK** to integrate mPaaS SDK.

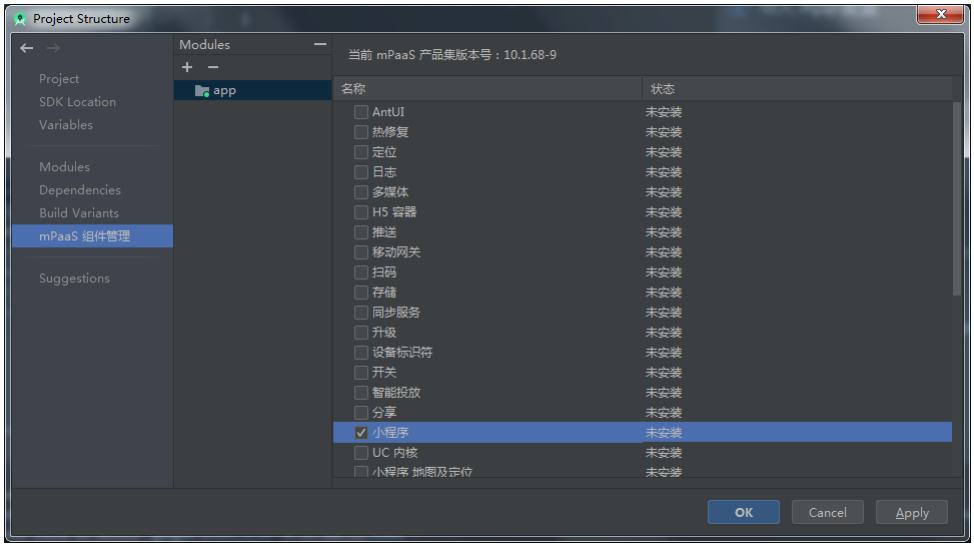
   Note: To upgrade the baseline, just click Configure again.



8. Click **Configure** under **Configure/Upgrade Component** on the right panel.



9. Check **Mini Program** in the component list, and then click **OK** to add the Mini Program to the project.



Now, you have successfully integrated Mini Program to the project through Native AAR.

1.2.4.5. Initialize configuration

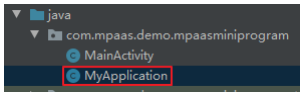
The configuration initialization covers the following four steps:

1. Initialize mPaaS
2. Configure Mini Program signature verification
3. Configure AndroidManifest
4. Apply for UC kernel

Initialize mPaaS

It is required to initialize mPaaS when you use the native AAR access method.

1. Create the `MyApplication` class in the project.



2. Add the following code in the class.

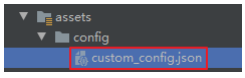
```
public class MyApplication extends Application {
 @Override
 protected void attachBaseContext(Context base) {
     super.attachBaseContext(base);
     // set mPaaS initialization callback
     QuinoxlessFramework.setup(this, new IInitCallback() {
         @Override
         public void onPostInit() {
             // Initialize mini program public resource package
             H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
         }
     });
 }
 @Override
 public void onCreate() {
     super.onCreate();
     // Initialize mPaaS
     QuinoxlessFramework.init();
 }
}
```

3. Open `AndroidManifest.xml`, and add the following code in `<application>` label to set `Application`.

```
android:name=".MyApplication"
```

Configure Mini Program signature verification

1. Create the file `custom_config.json` under `assets/config`.



2. Enter the following code in the file.

```
[
    {
        "value": "NO",
        "key": "h5_shouldverifyapp"
    }
]
```

If the value is "NO", it means disabling mini program signature verification while "YES" means enabling signature verification. It is "YES" by default if no value is filled. In the development and debugging stage, you can disable signature verification for quick access. It is recommended to enable signature verification before releasing the mini program.

For how to set mini program package signature verification, refer to Configure mini program package.

Configure AndroidManifest

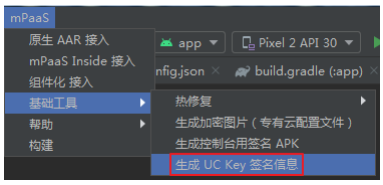
In this tutorial, the native AAR method is used to access, so the following configuration needs to be added to `AndroidManifest.xml`:

```
<application>
    ...
    <meta-data android:name="nebula.android.meta.enable" android:value="true"/>
  ...
</application>
```

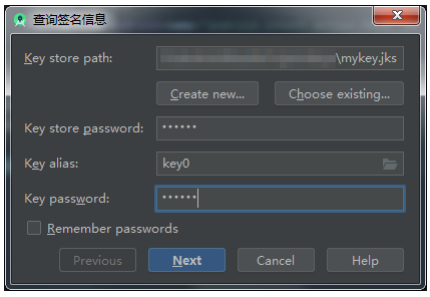
Apply for UC kernel

> **Note**
> From December 1, 2022, public application for UC Key will not be supported.

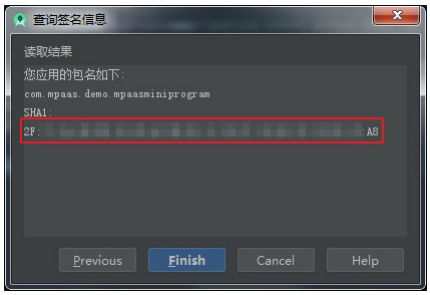
1. Click **mPaaS > Basic tool > Generate UC Key Signature** to open the **Query Signature** window.



2. In the **Query Signature** window, enter the necessary configuration information, and click **Next** to continue.



3. Copy the generated SHA1 string.



4. Log in to the mPaaS console, and please search for the group number 31591197 with DingTalk to join DingTalk group for further communication.
   i. Select the product to which the problem belongs. You can quickly search for mPaaS.



   ii. Select the problem type. You can select **Mini Program** or **Access Android**.
   iii. Select the expected solution **Create ticket**.

iv. On the ticket submission page, enter the following information to obtain the UC SDK Key.

- **Priority**: Required, select **Important** or **Normal** based on the actual situation.
- **Problem description**: Required, the following information is required.
  - **Summary**: For example, apply for UC SDK Key.
  - **Package Name**: In this tutorial, it is `com.mpaas.demo.mpaasminiprogram`.
  - **SHA1**: The SHA1 string obtained in above step.
- **Mobile**: Required.
- **Email**: Required.

5. Click **Submit**. After a while, the technical support staff will give you a feedback.



6. Fill in the Key you obtained in the previous step into the file `AndroidManifest.xml`.

```
<meta-data android:name="UCSDKAppKey" android:value="UC SDK Key information"/>
```

> **Note**
>
> The authorization information of UC SDK is associated with the package name and signature of the apk. Therefore, if UCWebView doesn't not work, you should check if the signature and package name are consistent with that used for applying for the key.

At this point, you have completed the initialization configuration.

1.2.4.6. Create and release a mini program

This section demonstrates the following operations:

1. Set virtual domain name in console
2. Create a mini program in console
3. Develop the mini program with IDE
4. Release the mini program

Set virtual domain name in console

1. Log in to the mPaaS console, enter the target mPaaS App, and then click **Mini Program > Release Mini Program** menu from the left navigation pane.
2. On the **Manage configuration** tab page, set the virtual domain name in the **Manage domain** area. In principle, you need to use a second-level domain name managed by your company.
3. Click **Save** to complete the virtual domain name configuration.

Create a mini program in console

1. In the mPaaS console, click **Mini Program > Release Mini Program** menu from the left navigation pane.
2. On the **Manage mini program packages** page, click **Create Mini Program App**.
3. In the **Create mini program** window, enter mini program ID and name, and click **OK**. Then, the App is created and shown in the left mini program package list.

   The mini program ID is a 16-digit number, for example 2020080120200801.
4. Upload the mini program code package here by clicking **Add** here.

Develop the mini program with IDE

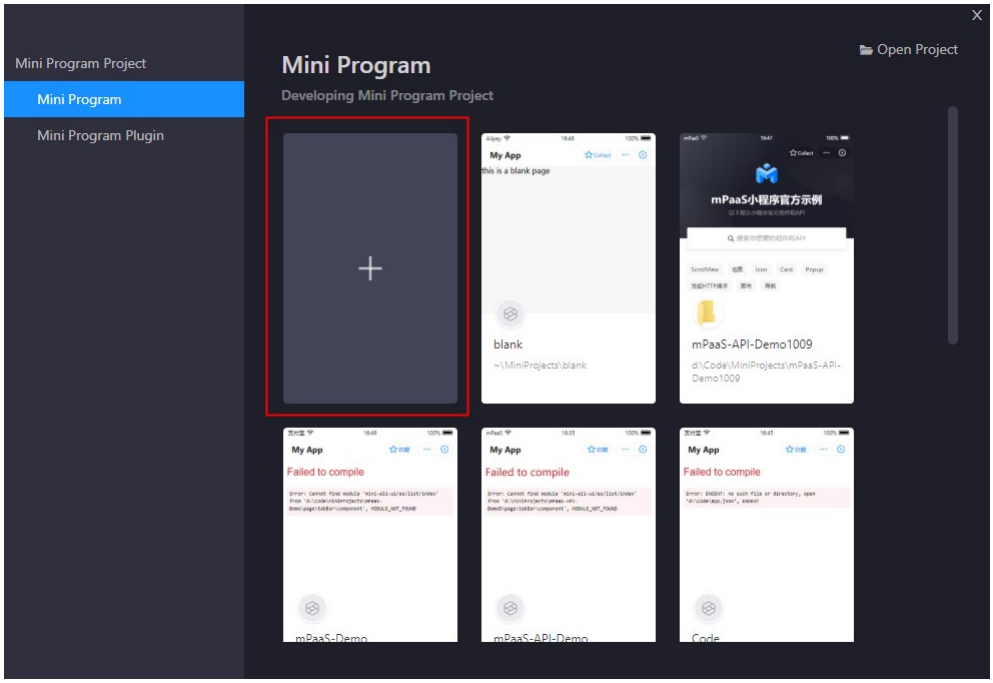
The procedure of developing the mini program with IDE falls into four steps:

1. Create a mini program
2. Download configuration file
3. Log in to Mini Program IDE
4. Develop mini program

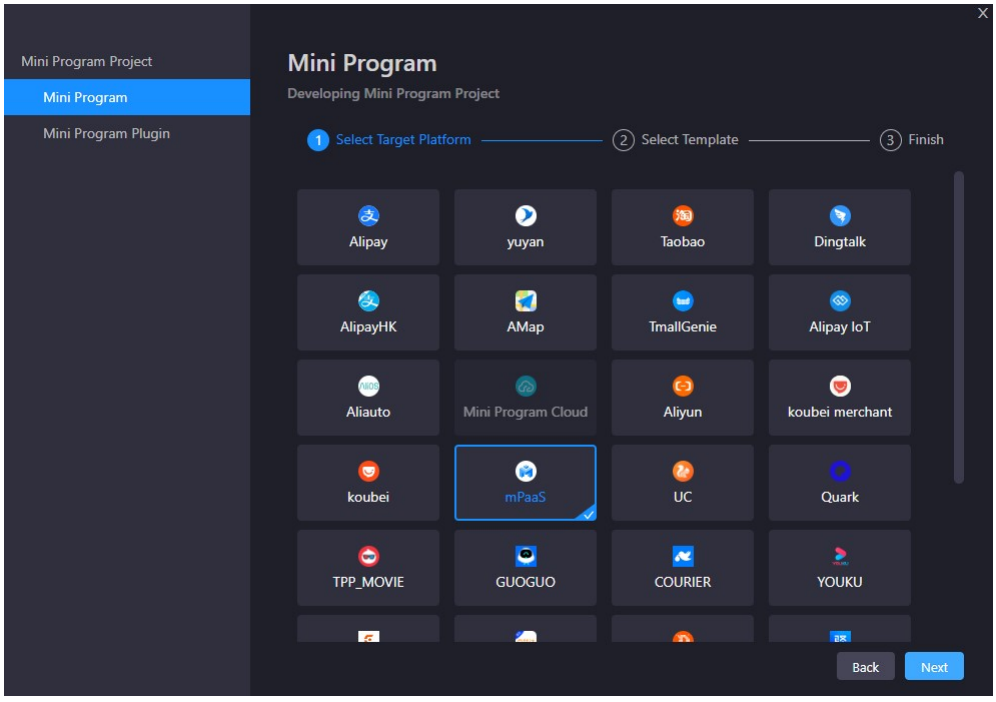
Create a mini program

1. Download the Mini Program development tool (IDE).

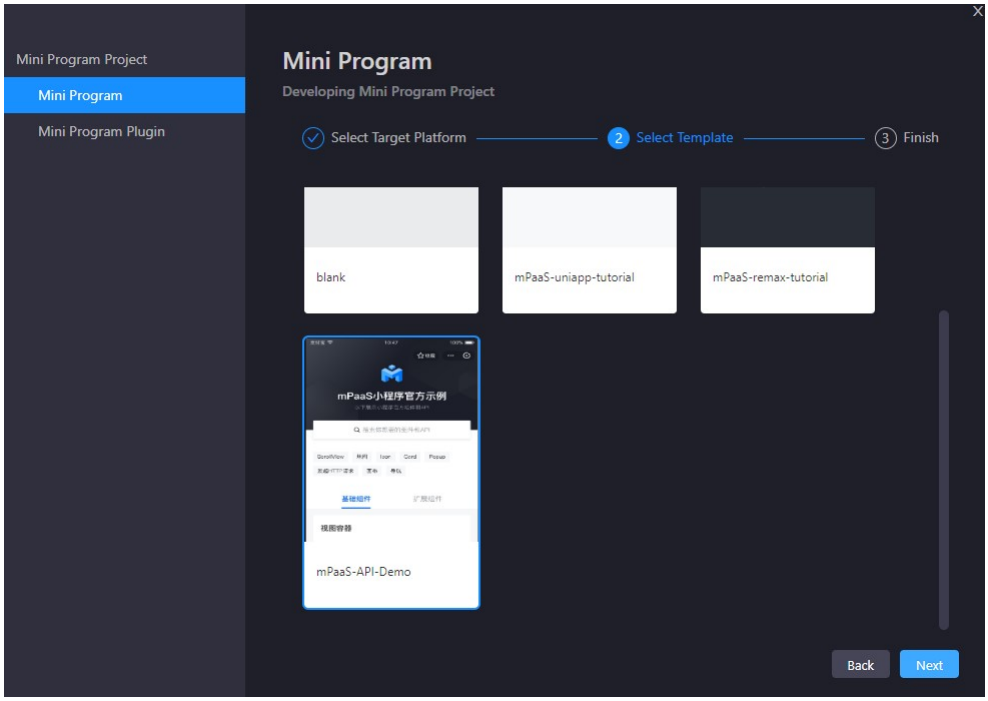
2. Create a mini program. Open the IDE, choose **Mini Program** on the left side menu, and then click the plus icon on the right to add a new project.



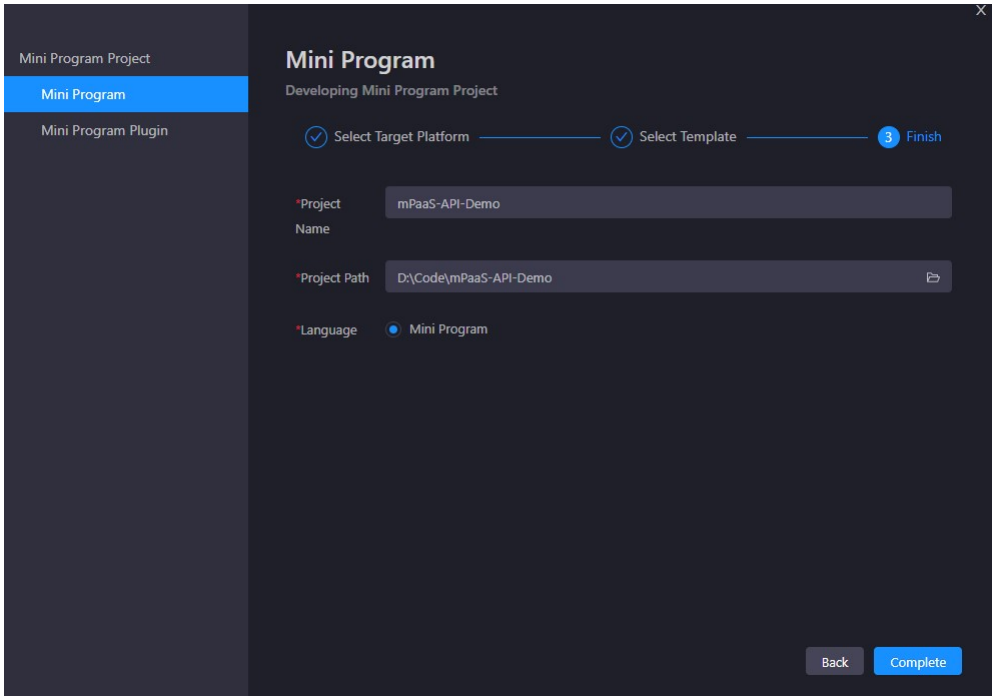
3. In the **Select Target Platform** step, choose **mPaaS** and click **Next**.



4. In the **Select Template** step, choose the mPaaS sample template and click **Next**.



5. Enter the project name, specify the project path, and click **Complete** to complete the Mini Program project creation.



Download configuration file

It is required to upload the IDE configuration file of the corresponding mini program downloaded from the mPaaS console every time you created a new environment.

1. Log in to the mPaaS console, go to the **Mini Program > Release Mini Program > Manage configuration** page, and click **Download configuration file** in the **Manage IDE configuration** area.

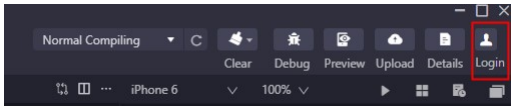
> **Note**
> The IDE configuration file is different from the configuration file of mPaaS application.

2. In the pop-up **Download configuration file** window, enter the **Dynamic password** which is the login password to Log in to the Mini Program IDE), and click **OK** to download the configuration file. Please keep the password in mind.

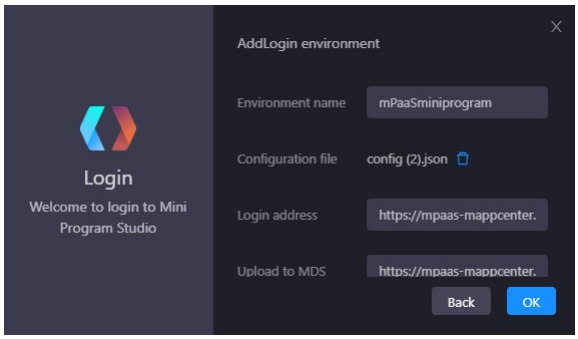
> **Note**
> The downloaded configuration file is named config.json by default.

Log in to Mini Program IDE

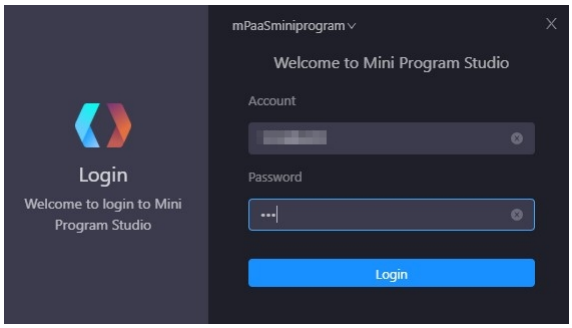
1. In the Mini Program IDE, click the upper-right **Log in**.



2. In the pop-up **New login environment** window, enter the environment name "mPaaSminiprogram", upload the **Mini Program IDE configuration file** (`config.json` file) downloaded from the mPaaS console, and then click **OK**.

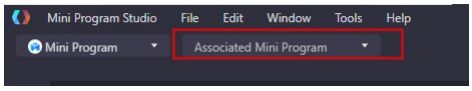


3. In the login window, enter the account and password to log in.
   - The account refers to the username for logging in to the mPaaS console.
   - The password refers to the dynamic password you set when downloading the IDE configuration file.

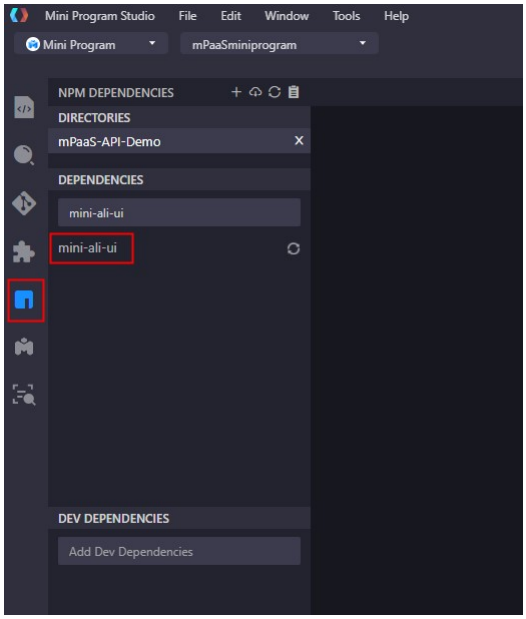


Develop mini program

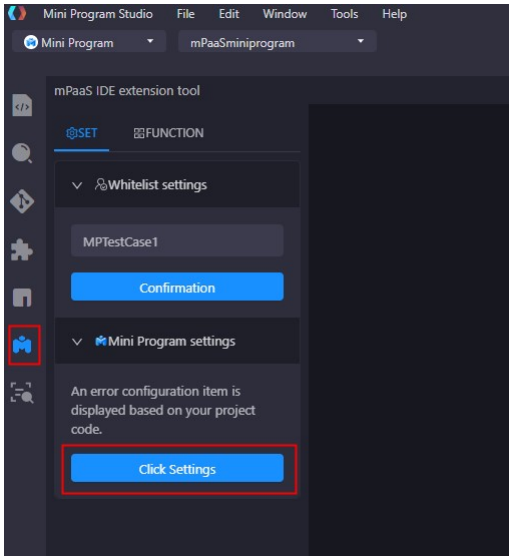
1. After you log in to the IDE, click **Associated mini program**, and select the mini program previously created in the mPaaS console from the drop-down menu.



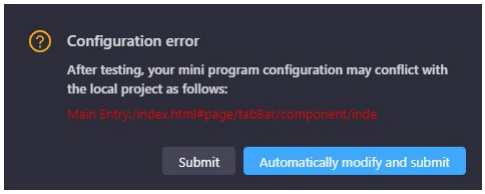
2. Click the dependency management icon on the left pane, and add the dependency `mini-ali-ui`.



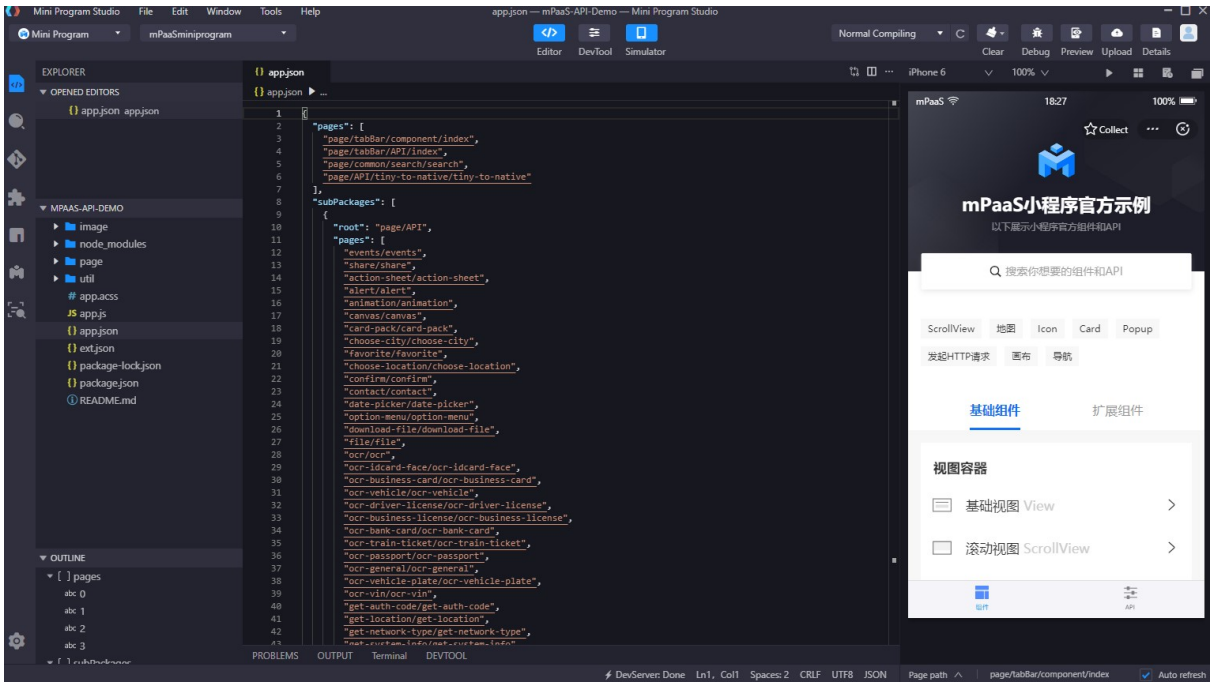
3. Click the toolbox icon on the left pane to set the mini program.



If any problem occurs in setting, for example main entrance configuration error, the system will prompt you the incorrect configuration items according to your project code. Click **Auto modify and submit**.

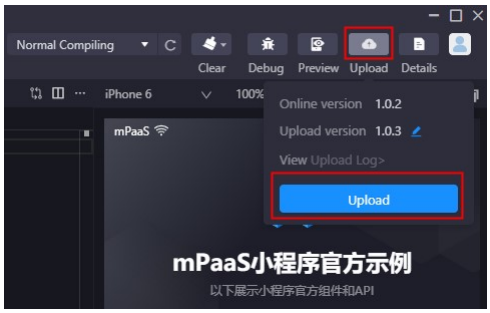


4. Start to compile the mini program code. You can also download the demo below, upload and release it in the mPaaS console directly.

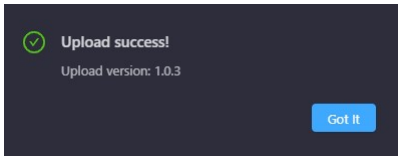


Release the mini program

1. In the Mini Program IDE, click **Upload** in the upper-right corner to upload the mini program package to the mPaaS console.



The IDE will prompts a tip once the package is successfully uploaded.



2. Log in to the mPaaS console, enter the target mPaaS App, and then click **Mini Program > Release Mini Program** menu from the left navigation pane. On the **Manage official package** tab page, you can find the uploaded package which is pending to be released.
3. Click **Create release**, select the release type as Official on the release task creation page, and click **OK**.

   Now, the mini program has been successfully released.

Code sample

Click to download the mPaaS mini program demo. You can directly upload and release the demo package in the mPaaS console.

> **Note**
>
> Before you upload the demo .zip file, you should rename the .zip file and the file in the zip to the 16-digit ID of your mini program.

1.2.4.7. Start mini program

This section demonstrates the following operations:

This section demonstrates the following operations:

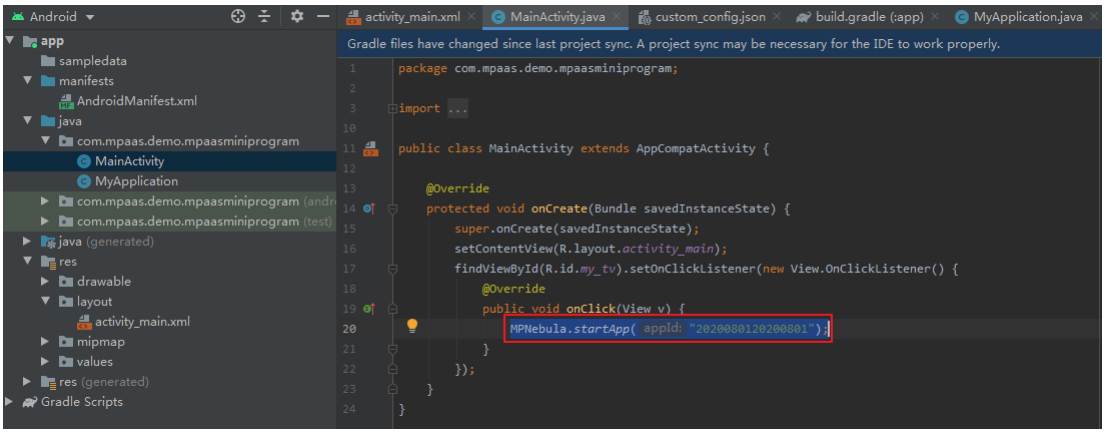
1. Start mini program
2. Run the mini program in real device

Start mini program

Starting the mini program simply needs one line of code.

Once the mini program package is uploaded to the mPaaS console, you can replace the "Hello mPaaS!" Toast related code (`Toast.makeText(MainActivity.this, "Hello mPaaS!", Toast.LENGTH_SHORT).show();`) in the `MainActivity` with the following code, so that the mini program starts upon a click on `TextView` .

```
MPNebula.startApp("2020080120200801");
```



Note: The mini program ID "2020080120200801" here is only for reference, please replace it with your real mini program ID in practice.

Run the mini program in real device

1. Connect your mobile to the computer, and turn on the USB debugging mode.
2. Run the project in Android Studio to update the App installed on your mobile.
3. Open the App on the mobile, click the text "Hello World" to start the mini program.

Now, you have successfully started the mini program and run it on the real device.

Code sample

Click to download the demo used in the tutorial.

1.2.4.8. Access real-device preview and debugging

This article guides you to use the real-device preview and debugging functions of the mini program. The process mainly falls into the following 5 steps:

1. Configure the debugging path of the mini program
2. Set the VHost and user whitelist of the mini program
3. Add mini program code scanning component
4. Realize the real-device preview and debugging function of the mini program
5. Use the preview function of the mini program
6. Use real-device debugging function of the mini program

> **Note**
> Only available in mPaaS 10.1.60 and later versions.

Procedure

Configure the debugging path of the mini program

1. Open the Mini program IDE configuration file downloaded form the mPaaS console. The following is an example of configuration file in the public cloud.

```
{
 "login_url":"https://mpaas-mappcenter.aliyuncs.com/ide/login",
 "uuid_url":"http://cn-hangzhou-mproxy.cloud.alipay.com/switch/uuid",
 "debug_url":"wss://cn-hangzhou-mproxy.cloud.alipay.com",
 "appId":"ONEX0D29541291400",
 "sign":"674f12adf7205358108839eb79f8a487",
 "tenantId":"AUDQKVYH",
 "upload_url":"https://mpaas-mappcenter.aliyuncs.com/ide/mappcenter/mds",
 "applist_url":"https://mpaas-mappcenter.aliyuncs.com/ide/mappcenter/mds/miniProgram/getAppListByApi",
 "workspaceId":"default"
}
```

2. Obtain the value of `debug_url` in the configuration file. Shown as below:

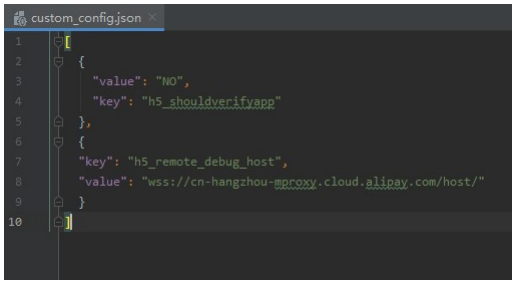
```
wss://cn-hangzhou-mproxy.cloud.alipay.com
```

3. Open the `custom_config.json` file in **assets** > **config** of the Android project.



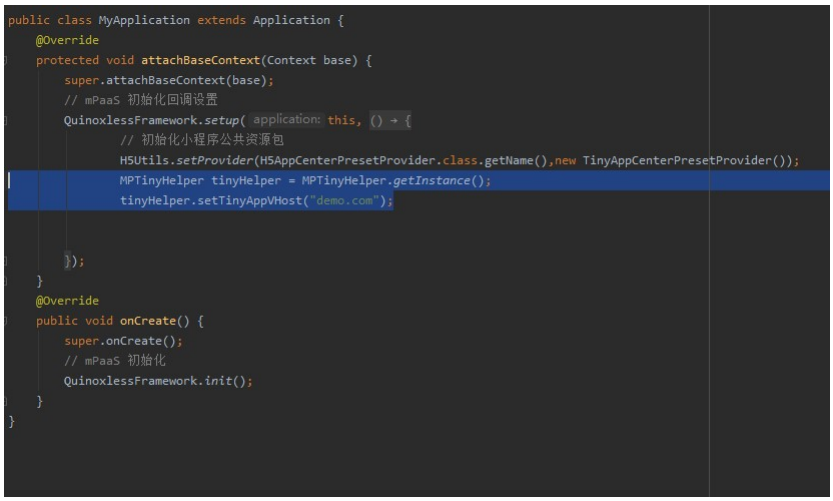
value: "NO" indicates that the mini program signature verification is disabled, while "YES" indicates that the it is enabled (it defaults to "YES" if you do not fill in). In the development and debugging stage, you can turn off the signature verification for quick access; before going online, it is recommended to turn on the signature verification. For the specific operation of the mini program package verification configuration, please refer to Configure mini program package.

4. Splice the obtained value of `debug_url` with `/host/` and fill it in the `value` of the `custom_config.json` file, as shown below:

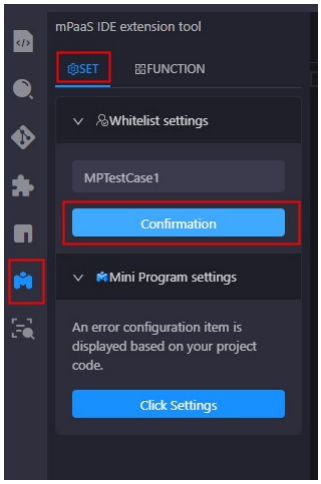


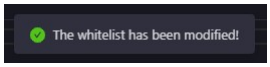
Set the VHost and user whitelist of the mini program

1. Open the `MyApplication` class, and add the following code to the mPaaS initialization callback. Call `tinyHelper.setTinyAppVHost` method before starting the application or mini program to set the virtual domain name used by the mini program. The value of VHost is consistent with the **Virtual domain name** in mPaaS console > **Mini program** > **Release mini program** > **Configuration management** > **Manage domain name**.



2. Open the mini program development tool, click **mPaaS toolbox** ( ) > **Settings** > **Whitelist** to add user whitelist, click **OK**, a pop-up window indicating setting success appears.





3. Open `MyApplication` class, and add the following code to the mPaaS initialization callback to set the user ID of whitelist.

```
public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS 初始化回调设置
        QuinoxlessFramework.setup( application: this, () -> {
                // 初始化小程序公共资源包
                H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                MPTinyHelper tinyHelper = MPTinyHelper.getInstance();
                tinyHelper.setTinyAppVHost("demo.com");
                MPLogger.setUserId("MPTestCase1");

        });
    }
    @Override
    public void onCreate() {
        super.onCreate();
        // mPaaS 初始化
        QuinoxlessFramework.init();
    }
}
```

Add mini program code scanning component

1. In Android Studio, choose **mPaaS** > **Native AAR Access**.
2. Click **Configure** in **Configure/Update Component** step.
3. Check **Mini program - scan** component, and click **OK**.

Realize the real-device preview and debugging function of the mini program

1. Add and click event to start the preview and debugging function of the mini program in the TestView of MainActivity.

```
public class MainActivity extends AppCompatActivity {

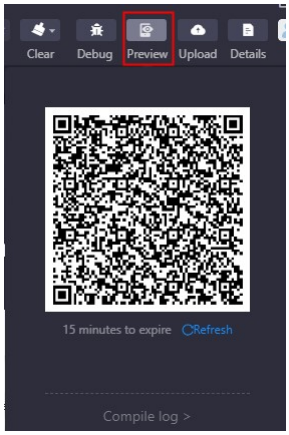
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.my_tv).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ScanRequest request = new ScanRequest();
                request.setScanType(ScanRequest.ScanType.QRCODE);
                MPScan.startMPaasScanActivity( activity: MainActivity.this, request, new ScanCallback() {
                    @Override
                    public void onScanResult(boolean b, Intent intent) {
                        if (null!=intent&&null!=intent.getData()){
                            //第一个参数为二维码的 uri，第二个参数为自定义启动参数。若无自定义启动参数则填 new Bundle()。
                            MPTinyHelper.getInstance().launchIdeQRCode(intent.getData(), new Bundle());
                        }
                    }
                });
            }

        });
    }
}
```

2. Click ▶ to run the application in the real device.
3. Tap **Hello World!** to activate code scanning function.
4. Tap **Always allow** in the pop-up menu.

Use the preview function of the mini program

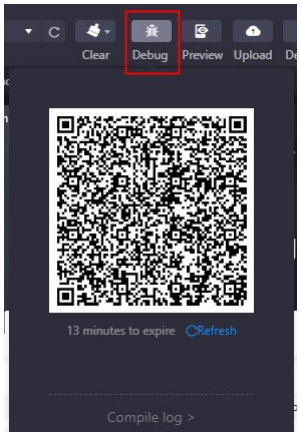
1. Click **Preview** in the navigation bar of mini program development tool to generate a QR code.



2. Use the code scanning function of the real device to scan this QR code to run the mini program in the mobile device.

Use real-device debugging function of the mini program

1. Click **Debug** in the navigation bar of mini program development tool to generate a QR code.



2. Use the code scanning function of the real device to scan this QR code. On the mobile device, you can see the remote debugging is connected and ready to enter the debugging mode.

1.2.4.9. Customize bi-directional channel

This article guides you to use the bi-directional channel function of the mini program. It contains the following 2 parts:

- Mini program calls native custom API
- Native project sends custom event to mini program

Mini program calls native custom API

1. Open Android Studio and create `MyJSApiPlugin` class (), make it inherit `H5SimplePlugin` to implement custom H5 API.

```
public class MyJSApiPlugin extends H5SimplePlugin {

    /**
     * Custom API
     */
    public static final String TINY_TO_NATIVE = "tinyToNative";

    @Override
    public void onPrepare(H5EventFilter filter) {
        super.onPrepare(filter);
        // onPrepare needs to be added
        filter.addAction(TINY_TO_NATIVE);
    }

    @Override
    public boolean handleEvent(H5Event event, H5BridgeContext context) {
        String action = event.getAction();
        if (TINY_TO_NATIVE.equalsIgnoreCase(action)) {
            JSONObject params = event.getParam();
            String param1 = params.getString("param1");
            String param2 = params.getString("param2");
            JSONObject result = new JSONObject();
            result.put("success", true);
            result.put("message", "Parameter received by the client:" + param1 + ", " + param2 + "\n Return the current package name of Demo:" +
context.getActivity().getPackageName());
            context.sendBridgeResult(result);
            return true;
        }
        return false;
    }
}
```

2. Register `MyJSApiPlugin` in `MyApplication`.

```
/*
 * 1st parameter: Defines the full path of the API
 * 2nd parameter: BundleName, fill "" if you adopt native AAR or mPaaS Inside accessing mode
 * 3rd parameter: Object on which the API works, you can fill "page" directly
 * 4th parameter: The API which works. Input the customized API in the form of String[]
 */
MPNebula.registerH5Plugin(MyJSApiPlugin.class.getName(), "", "page", new String[]{MyJSApiPlugin.TINY_TO_NATIVE});
```

```
public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS 初始化回调设置
        QuinoxlessFramework.setup( application: this, () -> {
                // 初始化小程序公共资源包
                H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                MPNebula.registerH5Plugin(MyJSApiPlugin.class.getName(), bundleName: "", scope: "page",new String[]{MyJSApiPlugin.TINY_TO_NATIVE});

        });
    }
    @Override
    public void onCreate() {
        super.onCreate();
        // mPaaS 初始化
        QuinoxlessFramework.init();
    }
}
```

3. Open the mini program development tool, add the following code into the `tiny-to-native.js` file under **page > API > tiny-to-native** to realize the call of the mini program.

```
Page({
tinyToNative() {
 my.call('tinyToNative', {
   param1: 'p1aaa',
   param2: 'p2bbb'
 }, (result) => {
   console.log(result);
   my.showToast({
     type: 'none',
     content: result.message,
     duration: 3000,
   });
 })
}
});
```

4. Click



to run the application on real device.

5. In the application running on the real device, tap **Hello World!** to start up the mini program.
6. Tap **API** tab (



) to open the API page.
7. Tap **Custom API** to open the custom API page.



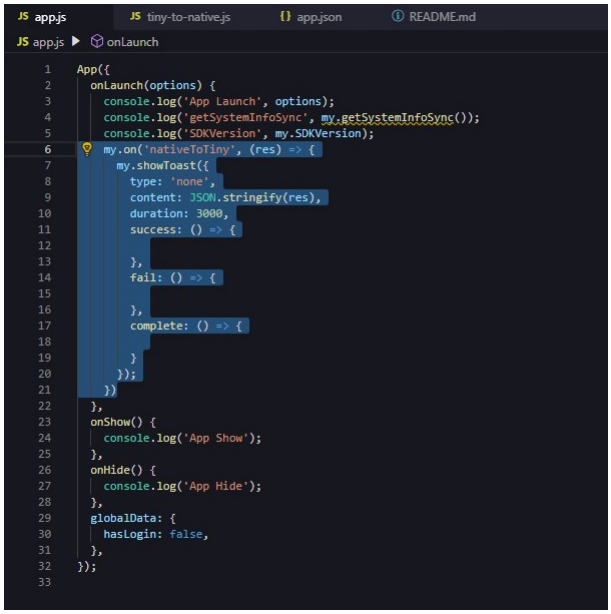
8. Tap **Tap to triger custom API** button, a toast pops up to show the parameter and the current Demo packagename received by the client.



Native project sends custom event to mini program

1. Open the `app.js` file in the mini program development tool (IDE) to add mini program registration event.

```
my.on('nativeToTiny', (res) => {
  my.showToast({
    type: 'none',
    content: JSON.stringify(res),
    duration: 3000,
    success: () => {
    },
    fail: () => {
    },
    complete: () => {
    }
  });
})
```



2. In Android Studio, modify the `my_tv` tap event in `MainActivity` to send event to mini program.

```
public class MainActivity extends AppCompatActivity {

 @Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);
     findViewById(R.id.my_tv).setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             MPNebula.startApp("2020092900000001");
             new Thread(new Runnable() {
                 @Override
                 public void run() {
                     for (int index = 0;index < 5;index++){
                         try {
                             Thread.sleep(5000);
                         } catch (InterruptedException e) {
                             e.printStackTrace();
                         }
                         final int i = index;
                         runOnUiThread(new Runnable() {
                             @Override
                             public void run() {
                                 H5Service h5Service = MPFramework.getExternalService(H5Service.class.getName());
                                 final H5Page h5Page = h5Service.getTopH5Page();
                                 if (null != h5Page) {
                                     JSONObject jo = new JSONObject();
                                     jo.put("key",i);
                                     // The method that Native send events to Mini Program
                                     // The first field is event name, the second is the parameter, the third is null by default
                                     h5Page.getBridge().sendDataWarpToWeb("nativeToTiny", jo, null);
                                 }

                             }
                         });
                     }
                 }
             }).start();

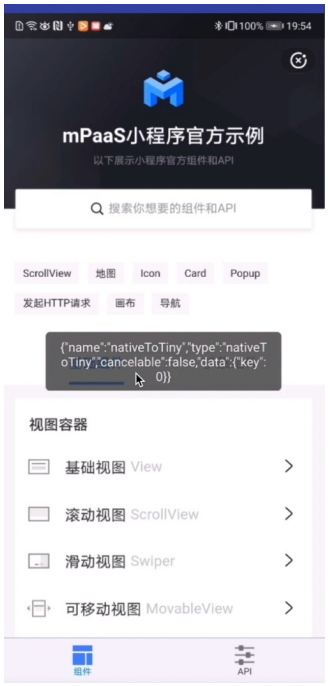
         }

     });
 }
}
```

3. Click

   to run the application on real device.
4. In the application running on the real device, tap **Hello World!** to start up the mini program.
5. The mini program receives an event in every 5 seconds, and the passed information is displayed in a toast.

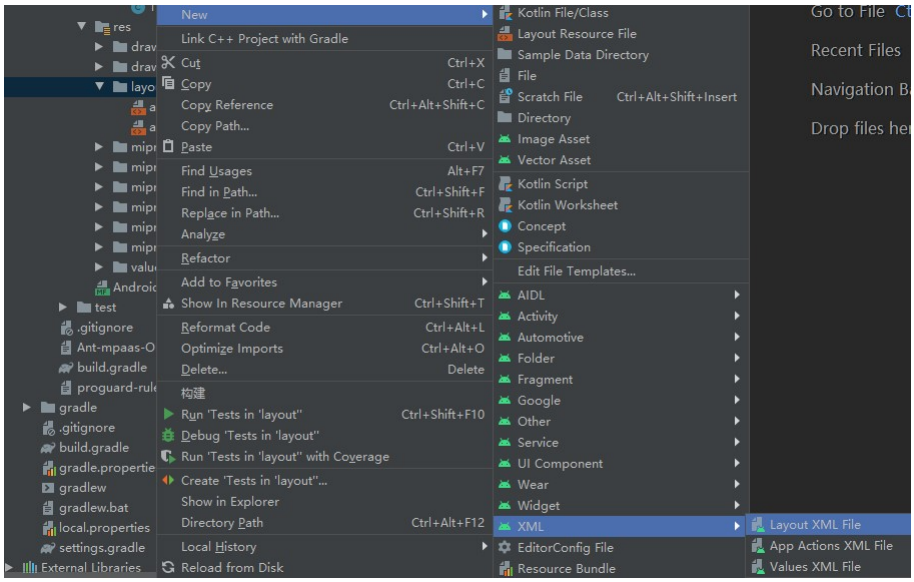


1.2.4.10. Customize startup loading page

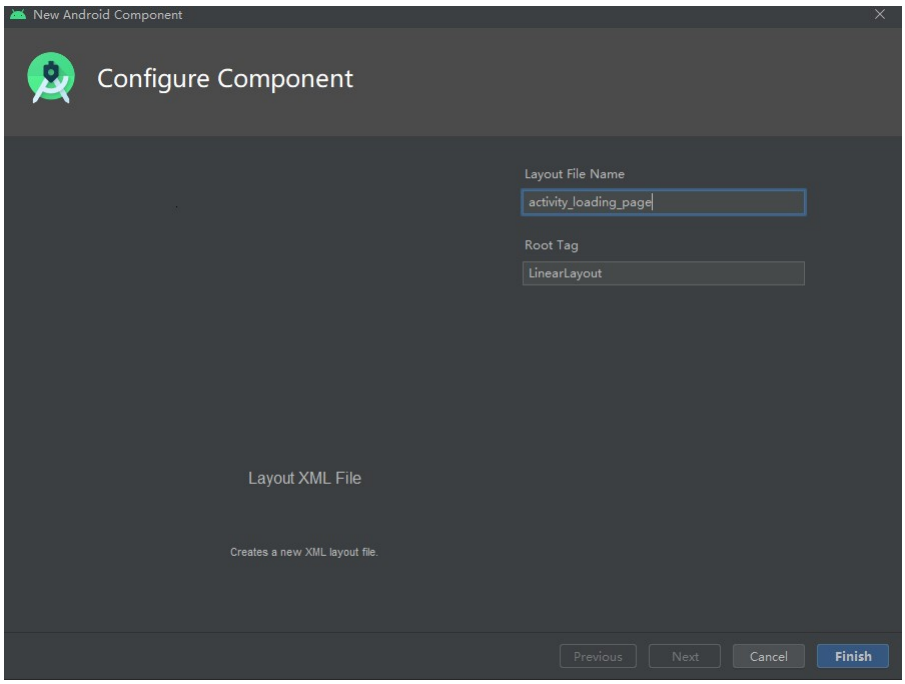
When you start up the mini program, if the mini program has not been downloaded to the device, the mini program container will launch a loading page to prompt the user to wait untill the mini program is installed on the device. After that, the loading page will be closed and the mini program will appear. This article guides you to customize the startup loading page of a mini program.

Procedure

1. Right click **layout** > **New** > **XML** > **Layout XML File** under **res**.



2. Enter the layout name in **Layout File Name**, click **Finish**.



3. Set loading layout in `activity_loading_page.xml`, the code is as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:orientation="vertical" android:layout_width="match_parent"
 android:layout_height="match_parent">
 <com.alipay.mobile.antui.basic.AUTitleBar
     android:id="@+id/title"
     android:layout_width="match_parent"
     android:layout_height="wrap_content" />
 <RelativeLayout
     android:background="@android:color/white"
     android:layout_width="match_parent"
     android:layout_height="match_parent">
     <LinearLayout
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:orientation="vertical"
         android:layout_centerInParent="true">
         <TextView
             android:id="@+id/tv_app"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"/>
         <com.alipay.mobile.antui.basic.AUProgressBar
             android:id="@+id/progress"
             style="?android:attr/progressBarStyleSmall"
             android:layout_gravity="center"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content" />
         <TextView
             android:id="@+id/tv_tips"
             android:visibility="gone"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content" />
     </LinearLayout>
 </RelativeLayout>
</LinearLayout>
```

4. Create `TinyStartupLoadingView` class and make it inherit `MPTinyBaseIntermediateLoadingView` to realize interface initialization and set the listening event of the back button.

```
public class TinyStartupLoadingView extends MPTinyBaseIntermediateLoadingView {

 private TextView tvAppName;

 private View progressBar;

 private TextView tvTips;

 public TinyStartupLoadingView(Context context) {
     super(context);
     init();
 }

 public TinyStartupLoadingView(Context context, AttributeSet attrs) {
     super(context, attrs);
     init();
 }

 public TinyStartupLoadingView(Context context, AttributeSet attrs, int defStyleAttr) {
     super(context, attrs, defStyleAttr);
     init();
 }

 private void init() {
     LayoutInflater.from(getContext()).inflate(R.layout.activity_loading_page, this, true);
     tvAppName = (TextView) findViewById(R.id.tv_app);
     progressBar =  findViewById(R.id.progress);
     tvTips = (TextView) findViewById(R.id.tv_tips);
     ((AUTitleBar)findViewById(R.id.title)).getBackButton().setOnClickListener(new OnClickListener() {
         @Override
         public void onClick(View v) {
             Activity host = getLoadingActivity();
             if (host != null) {
                 host.finish();
             }
         }
     });
 }

 /**
  * Called upon initialization, the appid of the mini program will be passed in. Other information, such as name, icon, and version, may be empty.
  */
 @Override
 public void initView(AppInfo info) {
     tvAppName.setText(info.appName);
 }

 /**
  * Called when it fails to obtain the mini program
  */
 @Override
 public void onError() {
     tvTips.setText("fail");
     tvTips.setVisibility(VISIBLE);
     progressBar.setVisibility(GONE);
 }

 /**
  * Called when the mini program information is obtained, including appid, name, icon, and version.
  */
 @Override
 public void update(AppInfo info) {
     tvAppName.setText(info.appName);
 }
}
```

5. Before starting up the mini program, enable the custom configuration. Add the following code to the click event listening of `MainActivity`:

```
MPTinyHelper.getInstance().setLoadingViewClass(TinyStartupLoadingView.class);
```

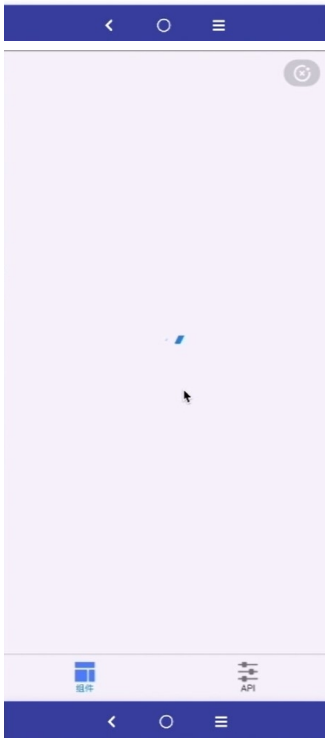
6. Click

   to run the application on real device.

7. Tap **Hello World!** to start up the mini program. The loading page after starting up the mini program is as follows:





1.2.4.11. Custom navigation bar

The mini program supports navigation bar customization, you can customize the style of the navigation bar, such as the position of the title, the style of the return button, etc. This article guides you to implement navigation bar customization in mini programs based on the 10.1.68 baseline.

This procedure falls into the following three steps:

1. Set the navigation bar of the mini program
2. Set the OptionMenu of the mini program
3. Run the mini program to view the navigation bar and OptionMenu after setting

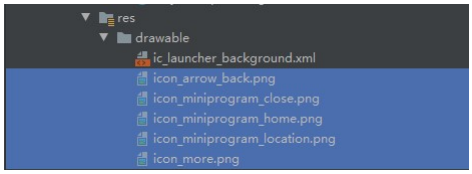
Procedure

Set the navigation bar of the mini program

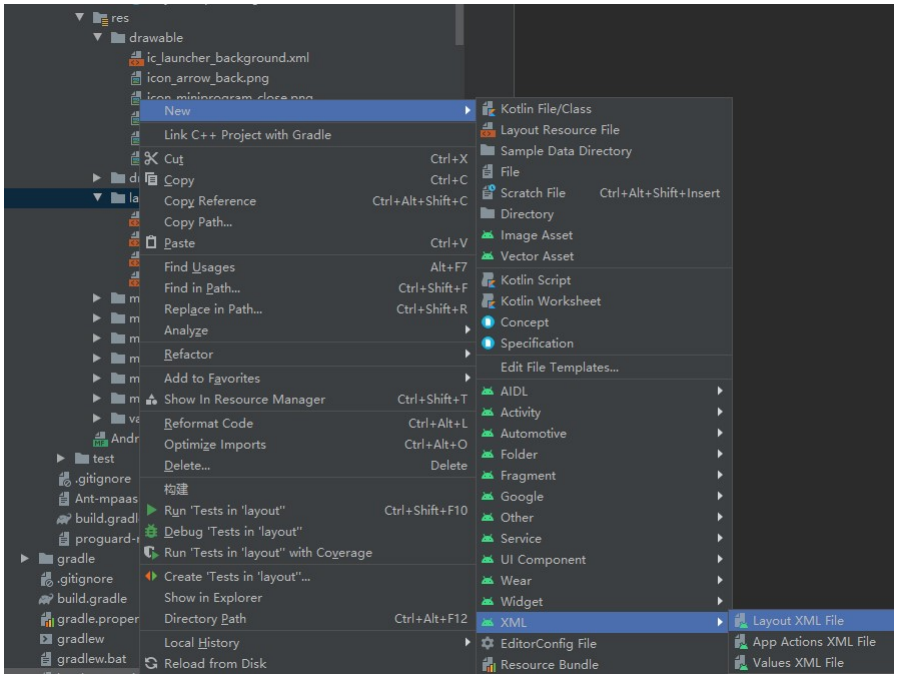
1. Open Android Studio, find the `custom_config.json` file under **assets** > **config** folder, and add the following code to disable the native navigation bar of the mini program.

```
{
    "value": "NO",
    "key": "mp_ta_use_orginal_mini_nagivationbar"
}
```

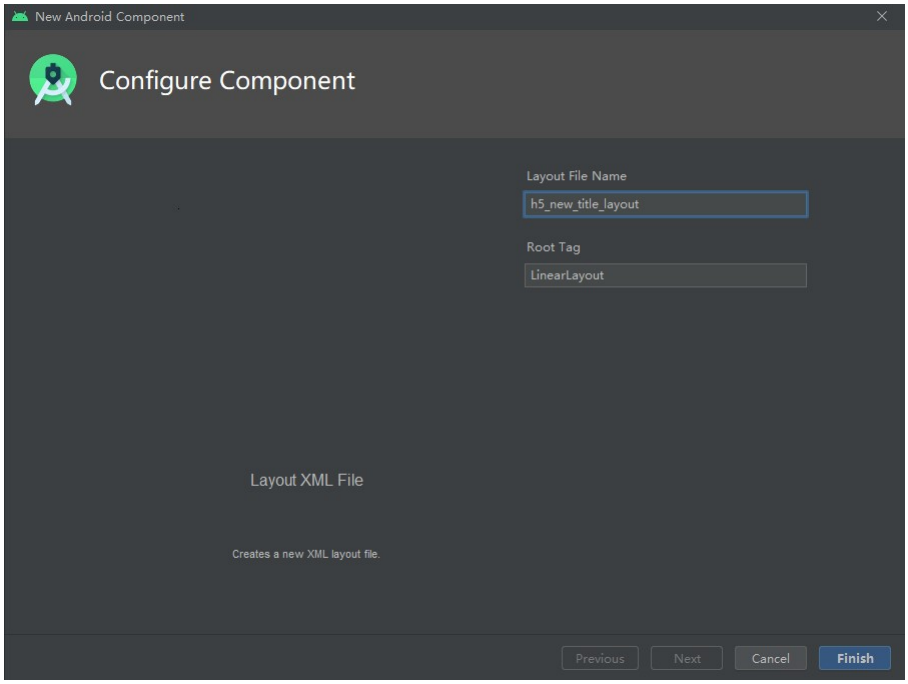
2. In **res** > **drawable** folder, add the image resource as shown in the following figure. Click here to download the image resource.



3. Right click the **res** > **layout** folder, select **New** > **XML** > **Layout XML File**, and press **Enter** key.



4. Enter the layout file name in **Layout File Name**, click **Finish**.



5. Add the following code in `h5_new_title_layout.xml` folder to set the layout of the navigation bar.

```
<?xml version="1.0" encoding="utf-8"?>
<com.alipay.mobile.nebula.view.H5TitleBarFrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/titlebar"
 android:layout_width="match_parent"
 android:layout_height="match_parent">

 <LinearLayout
     android:id="@+id/il_layout"
     android:layout_width="match_parent"
     android:layout_height="52dp"
     android:layout_gravity="center_vertical">

     <ImageView
         android:id="@+id/back"
         android:layout_width="26dp"
         android:layout_height="26dp"
         android:layout_gravity="center_vertical"
         android:layout_marginLeft="12dp"
         android:scaleType="centerInside"
         android:src="@drawable/icon_arrow_back" />

     <ImageView
         android:id="@+id/home"
         android:layout_width="26dp"
         android:layout_height="26dp"
         android:layout_gravity="center_vertical"
         android:layout_marginLeft="12dp"
         android:scaleType="centerInside"
         android:src="@drawable/icon_miniprogram_home"
         android:visibility="gone" />

     <LinearLayout
         android:layout_width="0dp"
         android:layout_height="match_parent"
         android:layout_weight="1"
         android:gravity="center_horizontal"
         android:orientation="vertical">

         <TextView
             android:id="@+id/mainTitle"
             android:layout_width="wrap_content"
             android:layout_height="match_parent"
             android:gravity="center"
             android:textColor="@android:color/white"
             android:textSize="20sp" />

         <TextView
             android:id="@+id/subTitle"
             android:layout_width="wrap_content"
             android:layout_height="match_parent"
             android:visibility="visible" />

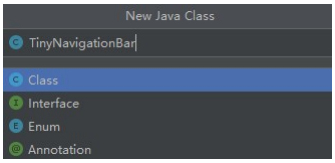
     </LinearLayout>

     <FrameLayout
         android:id="@+id/options1"
         android:layout_width="wrap_content"
         android:layout_height="match_parent"
         android:visibility="gone">

         <ImageView
             android:id="@+id/olimage"
             android:layout_width="26dp"
             android:layout_height="26dp"
             android:layout_gravity="center_vertical" />
     </FrameLayout>

     <LinearLayout
         android:id="@+id/options"
         android:layout_width="wrap_content"
         android:layout_height="match_parent"
         android:layout_gravity="center_vertical"
         android:layout_marginRight="12dp"
         android:orientation="horizontal"
         android:visibility="gone" />
 </LinearLayout>
</com.alipay.mobile.nebula.view.H5TitleBarFrameLayout>
```

6. Create `TinyNavigationBar` class.



7. Add the following code in `TinyNavigationBar` class to customize Title Bar.

```
public class TinyNavigationBar extends AbsTitleView {
    private H5TitleBarFrameLayout content;

    private TextView mainTitleView;

    private TextView subTitleView;

    private View btnBack;

    private View optionContainer;

    private View options1;

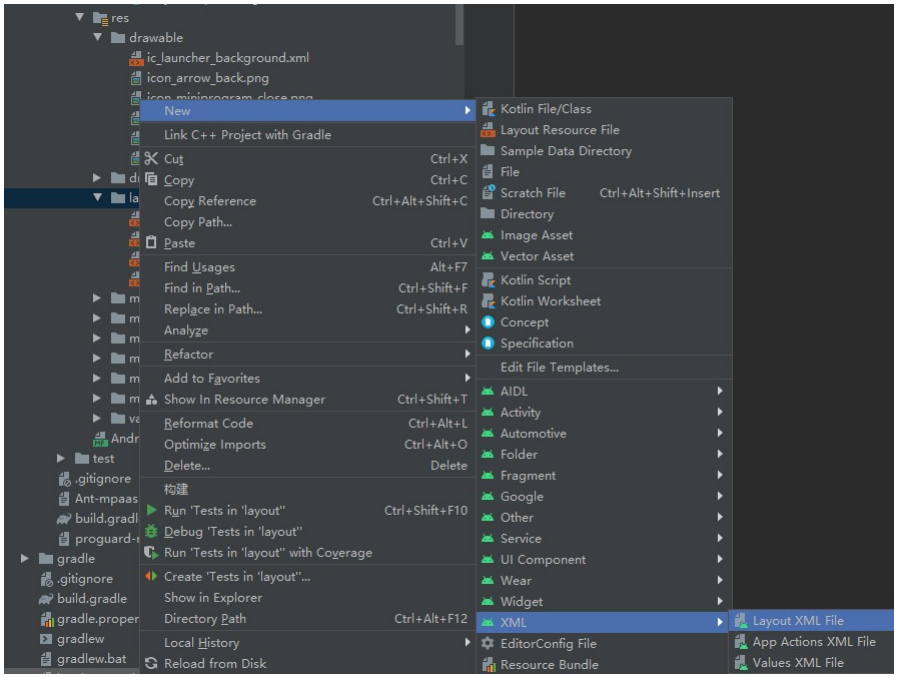
    private View btHome;

    private Context context;
    public TinyNavigationBar(Context context) {
        ViewGroup parent = null;
        this.context = context;
        if (context instanceof Activity) {
            parent = (ViewGroup) ((Activity) context).findViewById(android.R.id.content);
        }
        content = (H5TitleBarFrameLayout) LayoutInflater.from(context).inflate(R.layout.h5_new_title_layout, parent, false);
        content.getContentBgView().setColor(context.getResources().getColor(R.color.colorPrimary));
        mainTitleView = (TextView) content.findViewById(R.id.mainTitle);
        subTitleView = (TextView) content.findViewById(R.id.subTitle);
        btnBack = content.findViewById(R.id.back);
        btnBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                invokePageBackEvent();
            }
        });
        optionContainer = content.findViewById(R.id.options);
        btHome = content.findViewById(R.id.home);
        int statusBarHeight = H5StatusBarUtils.getStatusBarHeight(context);
        content.setPadding(0, statusBarHeight, 0, 0);
        btHome.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                invokeHomeClickEvent();
            }
        });
        options1 = content.findViewById(R.id.options1);
        options1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                invokeOptionClickEvent(1, false);
            }
        });
    }
    @Override
    public int getBackgroundColor() {
        return content.getContentBgView().getColor();
    }
    @Override
    public void setBackgroundAlphaValue(int i) {
        content.getContentBgView().setAlpha(i);
    }
    @Override
    public void setBackgroundColor(int i) {
        if ((i & 0xffffff) == 0xffffff) {
            mainTitleView.setTextColor(Color.BLACK);
        } else {
            mainTitleView.setTextColor(Color.WHITE);
        }
        content.getContentBgView().setColor(i);
        notifyTitleBarChanged();
    }
    @Override
    public String getTitle() {
        return mainTitleView.getText().toString();
    }
    @Override
    public void setTitle(String s) {
        mainTitleView.setText(s);
    }
```

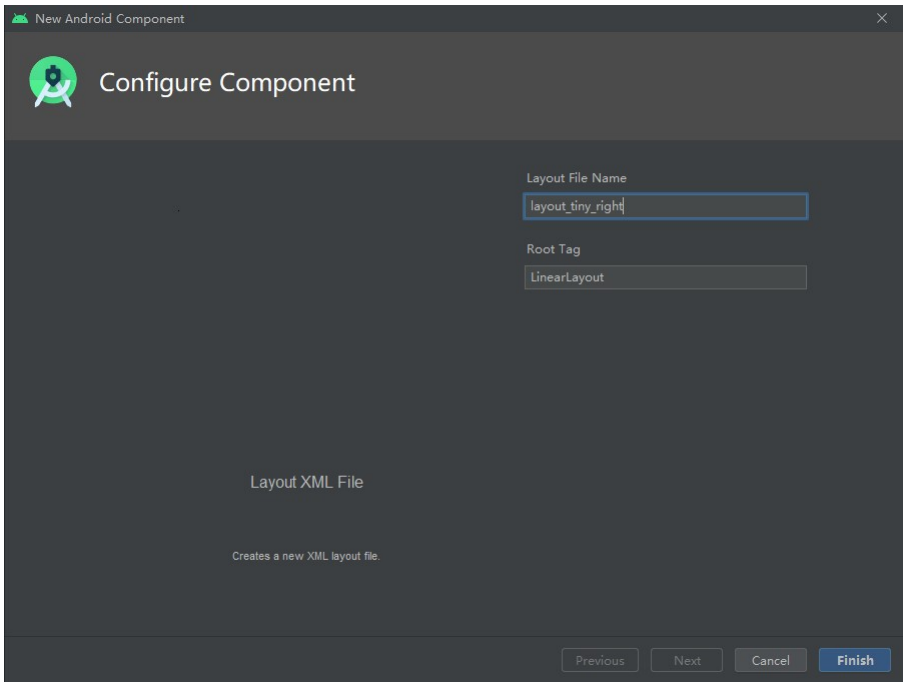
```
    @Override
    public void setSubTitle(String s) {
        subTitleView.setText(s);
    }
    @Override
    public void setTitleImage(Bitmap bitmap) {
    }
    @Override
    public TextView getMainTitleView() {
        return mainTitleView;
    }
    @Override
    public TextView getSubTitleView() {
        return subTitleView;
    }
    @Override
    public void resetTitle() {
        content.getContentBgView().setColor(context.getResources().getColor(R.color.colorPrimary));
    }
    @Override
    public void showCloseButton(boolean b) {
    }
    @Override
    public View getContentView() {
        return content;
    }
    @Override
    public void showBackButton(boolean b) {
        btnBack.setVisibility(b ? View.VISIBLE : View.GONE);
    }
    @Override
    public void showBackHome(boolean b) {
        btHome.setVisibility(b ? View.VISIBLE : View.GONE);
    }
    @Override
    public void showOptionMenu(boolean b) {
        optionContainer.setVisibility(b ? View.VISIBLE : View.GONE);
        options1.setVisibility(b ? View.VISIBLE : View.GONE);
    }
    @Override
    public View getOptionMenuContainer(int i) {
        if (i == 1) {
            return options1;
        }
        return optionContainer;
    }
    @Override
    public void setOptionMenu(boolean reset, boolean override, boolean isTinyApp, List<MenuData> menus) {
        for (int i = 0; i < 2 && i < menus.size(); i++) {
            MenuData menuData = menus.get(i);
            if (isTinyApp) {
                String iconUrl = menuData.getIcon();
                if (!TextUtils.isEmpty(iconUrl)) {
                    H5ImageUtil.loadImage(iconUrl, new H5ImageListener() {
                        @Override
                        public void onImage(Bitmap bitmap) {
                            ((ImageView)options1.findViewById(R.id.o1image)).setImageBitmap(bitmap);
                        }
                    });
                }
            }
        }
    }
    @Override
    public void showTitleLoading(boolean b) {
    }
    @Override
    public View getPopAnchor() {
        return optionContainer;
    }
}
```

Set the OptionMenu of the mini program

1. Right click **res** > **layout** folder, select **New** > **XML** > **Layout XML File**, and press **Enter** key.



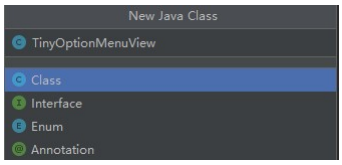
2. Enter **Layout File Name**, and click **Finish**.



3. Add the following code in `layout_tiny_right.xml` folder to set the layout of OptionMenu area.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_gravity="center_vertical">
    <LinearLayout
        android:id="@+id/option_bg"
        android:background="#9fffffff"
        android:layout_width="wrap_content"
        android:layout_height="42dp"
        android:layout_gravity="center_vertical"
        android:gravity="center_vertical">
        <ImageView
            android:id="@+id/more"
            android:layout_width="26dp"
            android:layout_height="26dp"
            android:src="@drawable/icon_more"/>
        <ImageView
            android:id="@+id/close"
            android:layout_width="26dp"
            android:layout_height="26dp"
            android:layout_marginLeft="12dp"
            android:src="@drawable/icon_miniprogram_close"/>
    </LinearLayout>
</FrameLayout>
```

4. Create `TinyOptionMenuView` class.



5. Add the following code in `TinyOptionMenuView` class to customize OptionMenu area.

```
public class TinyOptionMenuView extends AbsTinyOptionMenuView {

    private View container;

    private ImageView ivMore;

    private View ivClose;

    private Context context;

    private View bgView;

    public TinyOptionMenuView(Context context) {
        this.context = context;
        ViewGroup parent = null;
        if (context instanceof Activity) {
            parent = (ViewGroup) ((Activity) context).findViewById(android.R.id.content);
        }
        container = LayoutInflater.from(context).inflate(R.layout.layout_tiny_right, parent, false);
        ivClose = container.findViewById(R.id.close);
        ivMore = (ImageView) container.findViewById(R.id.more);
        bgView = container.findViewById(R.id.option_bg);
    }

    @Override
    public View getView() {
        return container;
    }

    @Override
    public void setOptionMenuOnClickListener(View.OnClickListener onClickListener) {
        ivMore.setOnClickListener(onClickListener);
    }

    @Override
    public void setCloseButtonOnClickListener(View.OnClickListener onClickListener) {
        ivClose.setOnClickListener(onClickListener);
    }

    @Override
    public void setCloseButtonOnLongClickListener(View.OnLongClickListener onLongClickListener) {
        ivClose.setOnLongClickListener(onLongClickListener);
    }

    @Override
    public void onStateChanged(TinyAppActionState state) {
        if (state == null) {
            ivMore.setImageDrawable(context.getResources().getDrawable(R.drawable.icon_more));
        } else if (state.getAction().equals(TinyAppActionState.ACTION_LOCATION)) {
            ivMore.setImageDrawable(context.getResources().getDrawable(R.drawable.icon_miniprogram_location));
        }
    }

    @Override
    protected void onTitleChange(final H5TitleView title) {
        super.onTitleChange(title);
        int color = title.getBackgroundColor();
        if ((color & 0xffffff) == 0xffffff) {
            bgView.setBackgroundColor(Color.RED);
        } else {
            bgView.setBackgroundColor(Color.GREEN);
        }
    }

    @Override
    public void setH5Page(H5Page h5Page) {
        super.setH5Page(h5Page);
        // title becomes available from here.
        if (getTitleBar().getBackgroundColor() == -1) {
            bgView.setBackgroundColor(Color.RED);
        }
    }

    @Override
    public void hideOptionMenu() {

    }
}
```

6. Add the following code in `IInitCallback` initialization callback of `MyApplication` class to customize title bar and upper-right configuration bar.

```
// Customize title bar
                MPNebula.setCustomViewProvider(new H5ViewProvider() {
                    @Override
                    public H5TitleView createTitleView(Context context) {
                        // Return custom title
                        return new TinyNavigationBar(context);
                    }

                    @Override
                    public H5NavMenuView createNavMenu() {
                        return null;
                    }

                    @Override
                    public H5PullHeaderView createPullHeaderView(Context context, ViewGroup viewGroup) {
                        return null;
                    }

                    @Override
                    public H5WebContentView createWebContentView(Context context) {
                        return null;
                    }
                });
                // Custom upper-right configuration bar
                H5Utils.setProvider(TinyOptionMenuViewProvider.class.getName(), new TinyOptionMenuViewProvider() {
                    @Override
                    public AbsTinyOptionMenuView createView(Context context) {
                        return new TinyOptionMenuView(context);
                    }
                });
```

```
import com.mpaas.tinyappcommonres.TinyAppCenterPresetProvider;

public class MyApplication extends Application {
    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS 初始化回调设置
        QuinoxlessFramework.setup( application: this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // 初始化小程序公共资源包
                H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                // 自定义标题栏
                MPNebula.setCustomViewProvider(new H5ViewProvider() {
                    @Override
                    public H5TitleView createTitleView(Context context) {
                        // 返回自定义 title
                        return new TinyNavigationBar(context);
                    }

                    @Override
                    public H5NavMenuView createNavMenu() { return null; }

                    @Override
                    public H5PullHeaderView createPullHeaderView(Context context, ViewGroup viewGroup) {
                        return null;
                    }

                    @Override
                    public H5WebContentView createWebContentView(Context context) { return null; }
                });
                // 自定义小程序右上角配置栏
                H5Utils.setProvider(TinyOptionMenuViewProvider.class.getName(), new TinyOptionMenuViewProvider() {
                    @Override
                    public AbsTinyOptionMenuView createView(Context context) {
                        return new TinyOptionMenuView(context);
                    }
                });


            }
        });
    }
```

Run the mini program to view the navigation bar and OptionMenu after setting

1. Tap

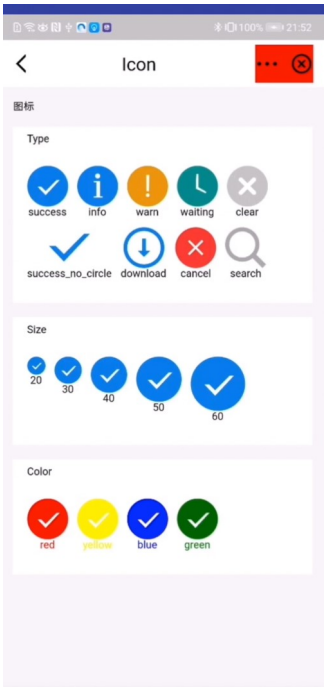
   

   to run the application on real device.
2. Tap **Hello World!** to start up the mini program. The loading page is shown as follows, you can see the customized configuration bar in the upper-right.



3. Tap the **Icon** under **Basic component**, then you can see the the layout of the custom navigation bar in **Icon** page.





1.3. Integrate Mini Program into iOS

1.3.1. Quick start

Note: Mini Programs are only supported in 10.1.60 and above baselines.

Prerequisites

You have accessed the project to mPaaS. For more information, see the following content:

- Access based on existing projects and using CocoaPods

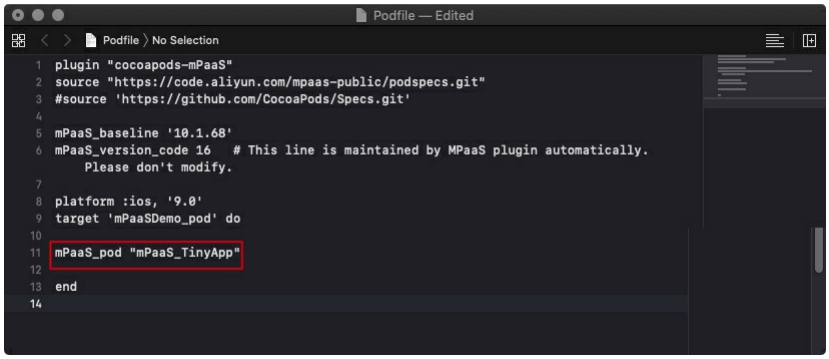
Add the SDK

According to your access method, select the corresponding method to add the SDK.

- **Use CocoaPods-mPaaS plugin**.

  This method is suitable for **Access based on existing projects and using CocoaPods**.

  i. In the Podfile, use `mPaaS_pod "mPaaS_TinyApp"` command to add Mini Program component dependency.



  ii. Execute `pod install` in the command line to complete the access.

> **Note**
>
> If you encounter problems during accessing the Mini Program, please search the group number 31591197 with DingTalk to join the Mini program question answering group for consultation.

Use the SDK

This section introduces the use of the SDK in combination with the official Demo of the Mini Program.

The process of using the Mini Program is mainly divided into the following three steps:

1. Initialization configuration
2. Release the Mini Program

3. Start the Mini Program

1. Initialization configuration

When configuring the project, you need to:

- Initialize the container
- Configure the Mini Program

If your App life cycle is not hosted by mPaaS framework, you also need to perform non-framework host configuration (if your baseline version is ≥ 10.1.68.25, we recommend that you use non-framework host configuration for 10.1.68.25 and later versions).

1.1 Initialize the container

Container initialization operations include starting the container, customizing the container, and updating the mini program package.

1.1.1 Start the container

- In order to use the Nebula container, you need to call the SDK interface after the program is started to initialize the container. The initialization must be done in `-(void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` of `DTFrameworkInterface`.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Initialize the container
    [MPNebulaAdapterInterface initNebula];
}
```

- If you need to use functions such as **preset mini program packages**, **custom JSAPI** and **Plugin**, please replace `initNebula` in the code above with the `initNebulaWith` interface in the code below, and pass in the corresponding parameters to initialize the container.
  - `presetApplistPath`: The package information path of a custom preset mini program package.
  - `appPackagePath`: The custom preset mini program package path.
  - `pluginsJsapisPath`: The storage path of custom JSAPI and Plugin files.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Initialize the container
    NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"MPCustomPresetApps.bundle/h5_json.json"]
ofType:nil];
    NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"MPCustomPresetApps.bundle"] ofType:nil];
    NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"Poseidon-UserDefine-Extra-Config.plist"]
ofType:nil];
    [MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath customPresetAppPackagePath:appPackagePath customPluginsJs
apisPath:pluginsJsapisPath];
}
```

  Note: `initNebula` and `initNebulaWithCustomPresetApplistPath` are two parallel methods, do not call them at the same time.

- Configure the Mini Program package request interval: mPaaS supports the configuration of the Mini Program package request interval, which can be configured globally or individually.
  - **Global configuration**: You can set the update frequency of mini program through the following code when initializing the container.

```
[MPNebulaAdapterInterface shareInstance].nebulaUpdateReqRate = 7200;
```

    `7200` is the value for setting the global update interval, 7200 is the default value, which represents the interval time, in seconds. You can modify this value to set your global mini program package request interval, the range is 0 ~ 86400 seconds (namely 0 ~ 24 hours, 0 means no request interval limitation).
  - **Individual configuration**: To configure only for the current Mini Program package. In the console, you can go to **Add mini program package** > **Extended information** and enter { “asyncReqRate” :” 1800” } to set the request interval. For details, see the **Extended information** in Create a Mini Program package.

1.1.2 Customize the container

If necessary, you can customize the container configuration by setting the property value of `MPNebulaAdapterInterface`. The configuration must be set in `-(void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` of `DTFrameworkInterface`, otherwise it will be overwritten by the container default configuration.

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Customize the container
    [MPNebulaAdapterInterface shareInstance].nebulaVeiwControllerClass = [MPH5WebViewController class];
    [MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = NO;
    [MPNebulaAdapterInterface shareInstance].nebulaUserAgent = @"mPaaS/Portal";
}
```

The meanings of the attributes are as follows:

Name	Meaning	Remarks
nebulaVeiwControllerClass	Base class for HTML5 pages	The default value is H5WebViewController. If you need to specify the base class of all HTML5 pages, you can directly set this interface. **Note**: The base class must inherit from H5WebViewController.

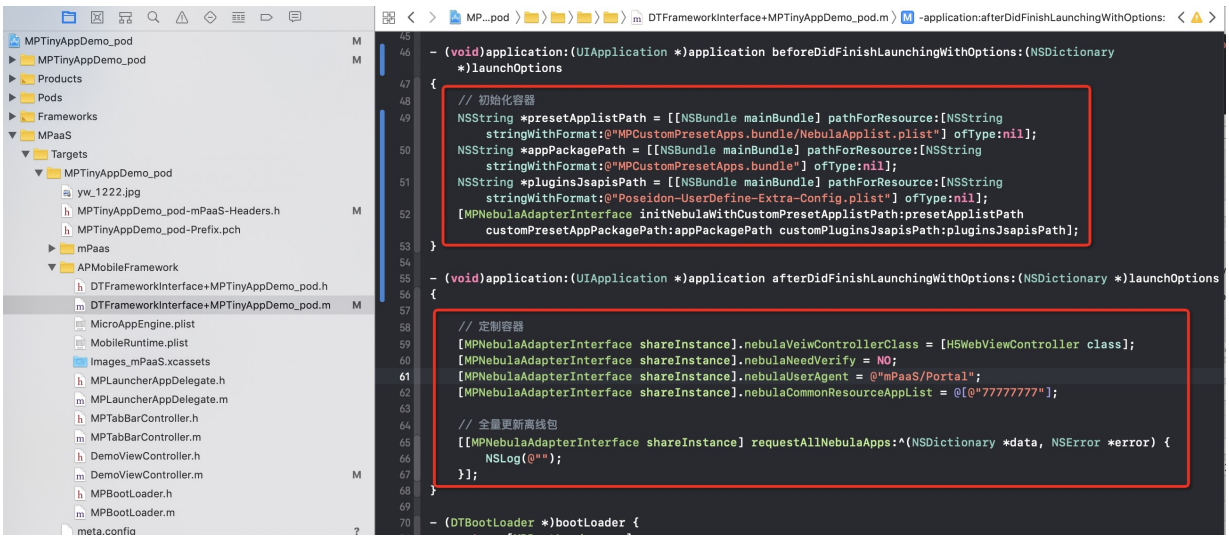
nebulaWebViewClass	Set the base class of WebView	When the baseline version is greater than 10.1.60, the default value is H5WKWebView. The customized WebView must inherit from H5WKWebView. When the baseline version is equal to 10.1.60, customization is not supported.
nebulaUseWKArbitrary	Set whether to use WKWebView to load the mini program package page	When the baseline version is greater than 10.1.60, the default value is `YES`. When the baseline version is equal to 10.1.60, the default value is `NO`.
nebulaUserAgent	Set the UserAgent of the application	The set UserAgent will be added as a suffix to the default UA of the container.
nebulaNeedVerify	Whether to verify or not, the default value is YES	If the private key file is not uploaded when configuring the mini program package, this value must be set to NO, otherwise the loading of mini program package will fail.
nebulaPublicKeyPath	Public key for mini program package verification	The public key corresponding to the private key uploaded when configuring the mini program package.
nebulaCommonResourceAppList	The appId list of the public resource package	-
errorHtmlPath	The HTML error page path displayed when HTML5 page fails to load	`MPNebulaAdapter.bundle/error.html` is read by default.
configDelegate	Set a custom switch delegate	Provides the ability to globally modify the default switch value of the container.

1.1.3 Update mini program package

After the startup is complete, request all mini program package information, and check whether there is an update package on the server. In order not to affect the application launch speed, it is recommended to call after `(void)application:(UIApplication \*)application afterDidFinishLaunchingWithOptions:(NSDictionary \*)launchOptions`.

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Customize the container
    [MPNebulaAdapterInterface shareInstance].nebulaVeiwControllerClass = [MPH5WebViewController class];
    [MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = NO;
    [MPNebulaAdapterInterface shareInstance].nebulaUserAgent = @"mPaaS/Portal";
    [MPNebulaAdapterInterface shareInstance].nebulaCommonResourceAppList = @[@"77777777"];
    // Update all mini program packages
    [[MPNebulaAdapterInterface shareInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
        NSLog(@"");
    }];
}
```

After initialization, the result is as follows:

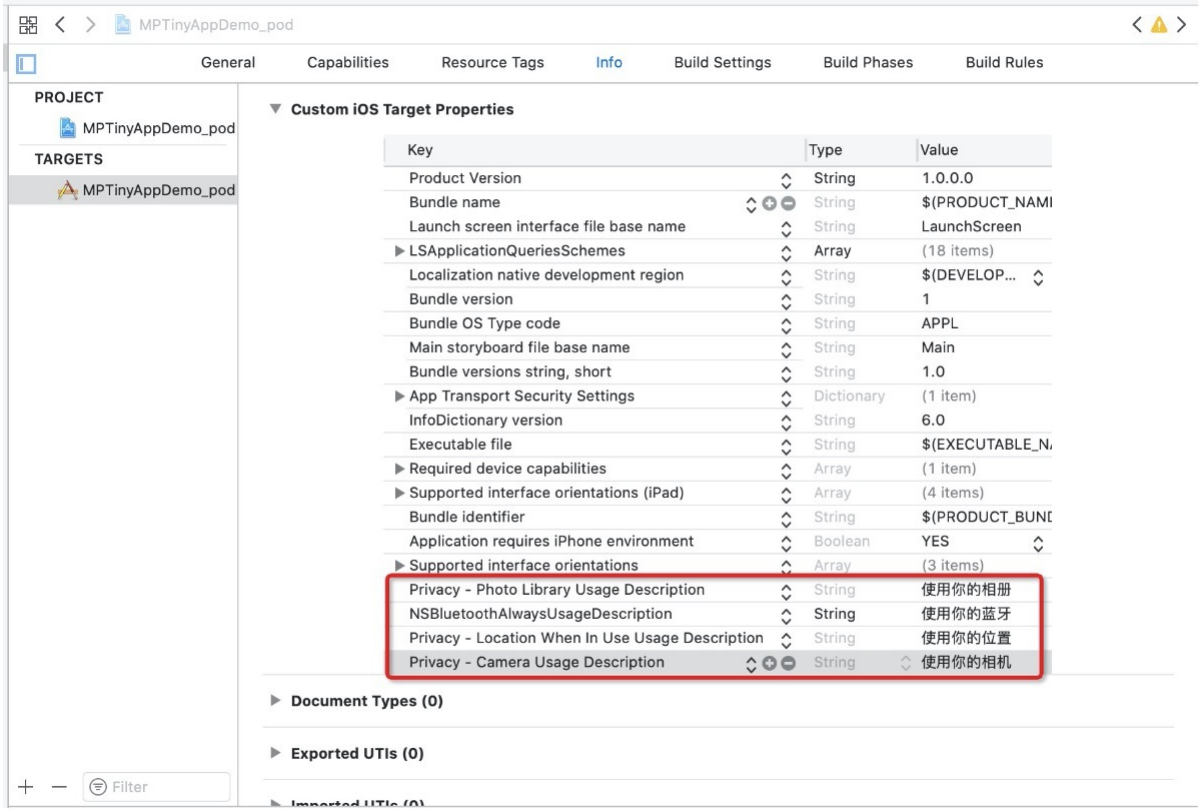


1.2 Configure the Mini Program

1.2.1 Configure permissions

Configure the following App permissions in `info.plist`:

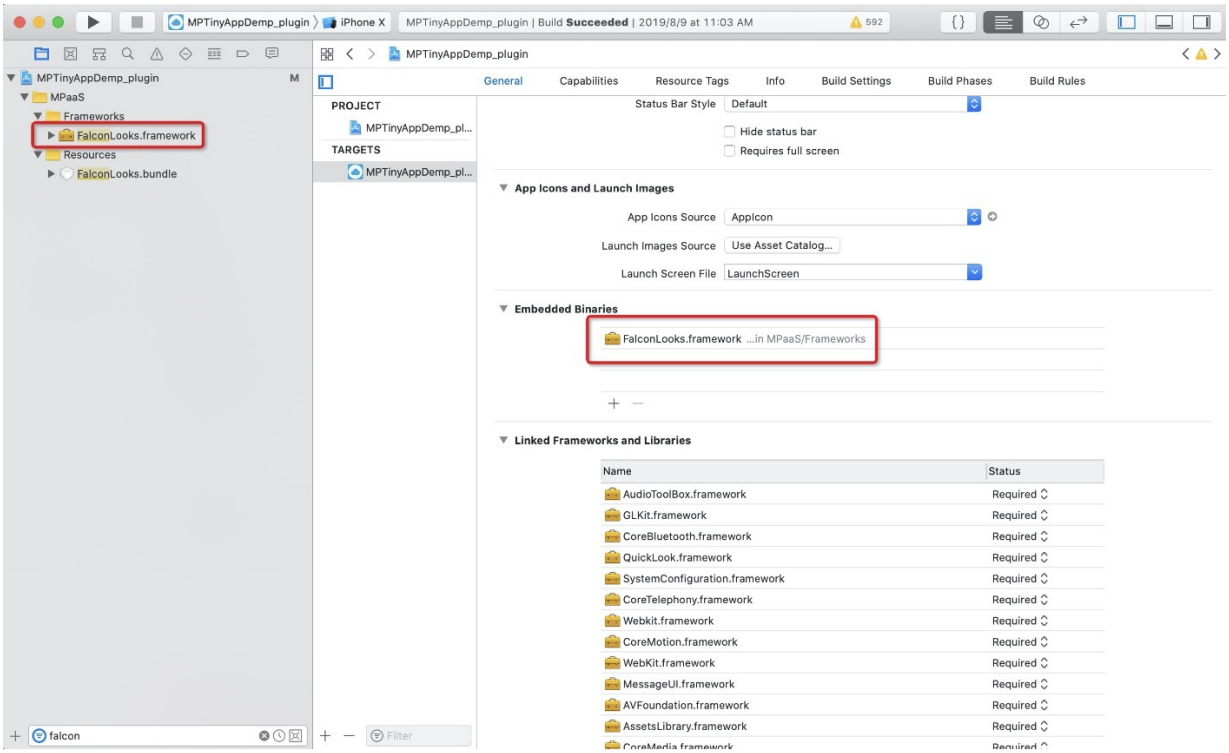
- `NSBluetoothAlwaysUsageDescription`: Bluetooth permission (new permission in iOS 13).
- `NSCameraUsageDescription`: Camera permission.
- `NSPhotoLibraryUsageDescription`: Album permission.
- `NSLocationWhenInUseUsageDescription`: Location permission.



1.2.2 Configure dynamic library

In **TARGETS** of the current project, add the **FalconLooks** library in **General** > **Embedded Binaries**.

Note:The configuration of the dynamic library has been cancelled in the 10.1.68.15 (included) and above baselines, and no configuration is required.



1.3 Non-framework host configuration

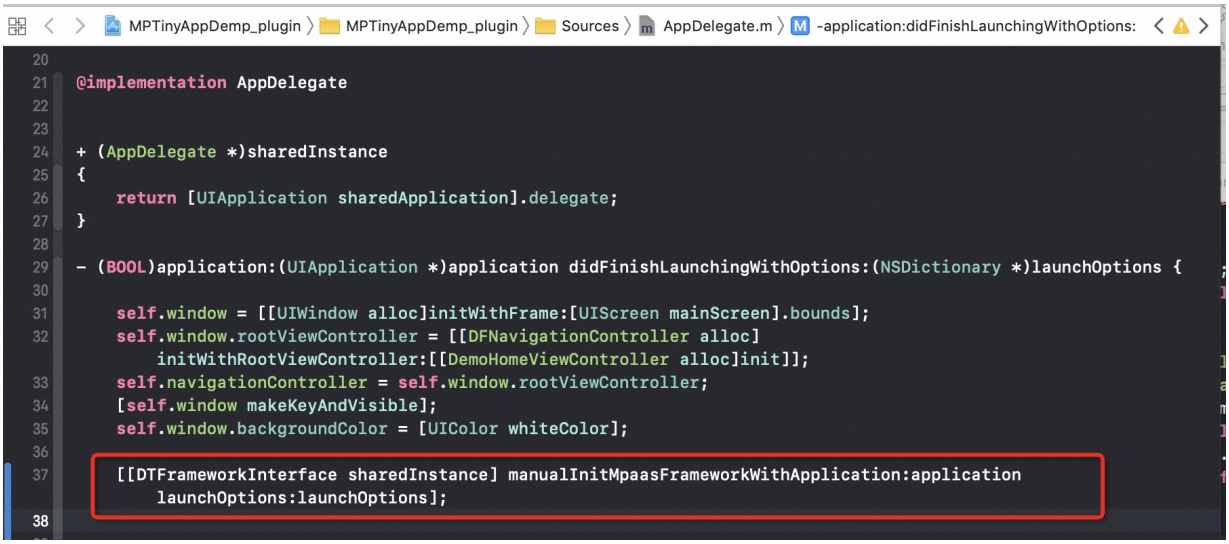
If the life cycle of your App is not hosted by the mPaaS framework, but instead is designated as a delegate defined by yourself, then you need additional configuration for non-framework host.

> **Note**
>
> If your baseline version is ≥ 10.1.68.25, we recommend that you use non-framework host configuration for 10.1.68.25 and later versions.



1.3.1 Start mPaaS framework

Call `[[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:application launchOptions:launchOptions];` in the `didFinishLaunchingWithOptions` method of the current application to start the mPaaS framework.



Note: The framework startup must be called after `window` and `navigationController` of the current application are initialized, otherwise it will not take effect.

1.3.2 Create an application launcher

Create a subclass of `DTBootLoader`, override the `createWindow` and `createNavigationController` methods, and return the `window` and `navigationControlle` of the current application.

- Set `window`: The `keyWindow` of the current application.
- Set `navigationController`: The `navigationController` where the Mini Program is loaded, which must inherit from `DFNavigationController`.
  - If the `rootviewcontroller` of the current application `keyWindow` is a `navigationController`, just set it to this class;
  - If the `rootviewcontroller` of the current application `keyWindow` is a `tabBarViewController`, take the `navigationController` of the tab where the Mini Program is loaded.



MPTinyAppDemp_plugin 〉 MPTinyAppDemp_plugin 〉 Sources 〉 Tinyapp 〉 MPBootLoaderImpl.m 〉 No Selection

```m
//
//  MPBootLoaderImpl.m
//  Portal
//
//  Created by yemingyu on 2019/6/24.
//  Copyright © 2019 Alibaba. All rights reserved.
//

#import "MPBootLoaderImpl.h"
#import "AppDelegate.h"

@implementation MPBootLoaderImpl

- (UINavigationController *)createNavigationController
{
    return [AppDelegate sharedInstance].navigationController;
}

- (UIWindow *)createWindow
{
    return [AppDelegate sharedInstance].window;
}

@end

```

Override the `setupNavigationController` method in the category of `DTBootPhase` to specify the `navigationController` loaded by the Mini Program.

FrameDemo 〉 MPaaS 〉 Targets 〉 FrameDemo 〉 APMobileFramework 〉 DTBootPhase+Category.h 〉 No Selection

```h
//
//  DTBootPhase+Category.h
//  CSMPaaS
//
//  Created by account on 2019/12/26.
//  Copyright © 2019 xzb. All rights reserved.
//

#import <APMobileFramework/APMobileFramework.h>

NS_ASSUME_NONNULL_BEGIN

@interface DTBootPhase (Category)

@end

NS_ASSUME_NONNULL_END
```

FrameDemo 〉 MPaaS 〉 Targets 〉 FrameDemo 〉 APMob...ework 〉 DTBootPhase+Category.m 〉 DTBootPhase(Category)

```m
//
//  DTBootPhase+Category.m
//  CSMPaaS
//
//  Created by account on 2019/12/26.
//  Copyright © 2019 xzb. All rights reserved.
//

#import "DTBootPhase+Category.h"
#import <NebulaSDK/NBContext.h>
#pragma clang diagnostic push
#pragma clang diagnostic ignored "-Wobjc-protocol-method-implementation"

@implementation DTBootPhase (Category)

+ (DTBootPhase *)setupNavigationController {

    return [DTBootPhase phaseWithName:@"setupNavigationController" block:^{

        UINavigationController *navControl = [[[DTFrameworkInterface sharedInstance] bootLoader] createNavigationController];
        DTContextGet().navigationController = navControl;
    }];

}

@end
```

1.3.3 Specify application launcher

Rewrite the method in the category of `DTFrameworkInterface`, specify the current application's own `bootloader`, and hide the default `window` and `launcher` applications of the mPaaS framework.

MPTinyAppDemp_plugin 〉 ... 〉 T...s 〉 ... 〉 A...k 〉 DTFrameworkInterface+MPTinyAppDemp_plugin.m 〉 -bootLoader

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // 全量更新
    [[MPNebulaAdapterInterface shareInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {

    }];
}

- (DTBootLoader *)bootLoader {
    static MPBootLoaderImpl *_bootLoader;
    static dispatch_once_t onceToken;
    dispatch_once(&onceToken, ^{
        _bootLoader = [[MPBootLoaderImpl alloc] init];
    });
    return _bootLoader;
}

- (BOOL)shouldWindowMakeVisable {
    return NO;
}

- (BOOL)shouldShowLauncher {
    return NO;
}

@end

#pragma clang diagnostic pop
```

1.4 Non-framework host configuration (10.1.68.25 and later versions)

This section introduces an easier way to initialize the mPaaS framework for non-framework host applications.

- You only need to call the following methods after the application window and navigationController are created. There is no need to create bootloader, hide the frame window, etc.

H5SizeOrigin 〉 H5SizeOrigin 〉 AppDelegate.m 〉 No Selection

```
//
//  H5SizeOrigin
//
//  Created by yangwei on 2021/1/7.
//  Copyright © 2021 yangwei. All rights reserved.
//

#import "AppDelegate.h"
#import "MPTabBarController.h"

@interface AppDelegate ()

@end

@implementation AppDelegate

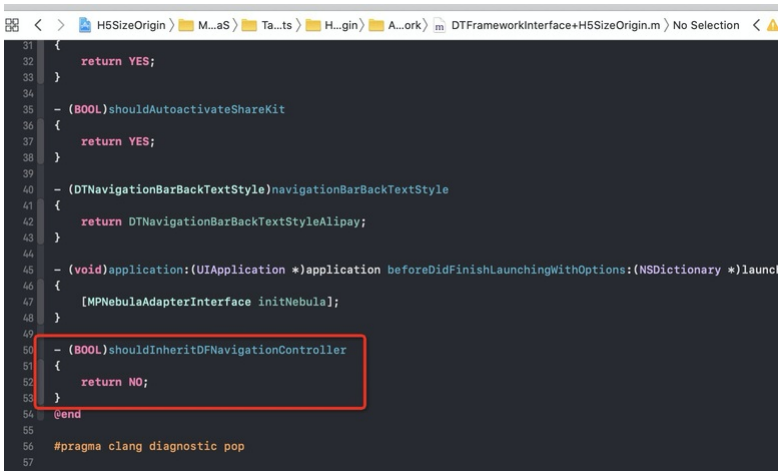

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.
    UIWindow *window = [[UIWindow alloc] initWithFrame:[[UIScreen mainScreen] bounds]];
    self.window = window;
    MPTabBarController *tabBarController = [[MPTabBarController alloc] init];
    [window setRootViewController:tabBarController];
    [window makeKeyAndVisible];
    UINavigationController *navigationController = tabBarController.selectedViewController;

    //启动 mPaaS 框架
//    [[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:[UIApplication sharedApplication]
    [[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:application
        launchOptions:launchOptions window:window navigationController:navigationController];

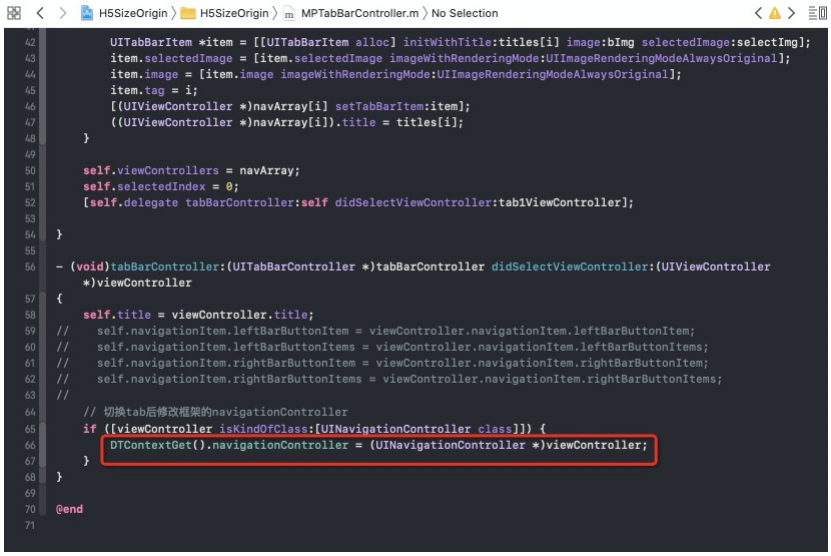
    return YES;
}


@end
```

- Not inheriting DFNavigationController is supported.



- If the application has multiple navigation bars, and different offline packages need to be opened in different navigation bars, the navigation bar of the container needs to be reset after the navigation bar is switched.



2. Release the Mini Program

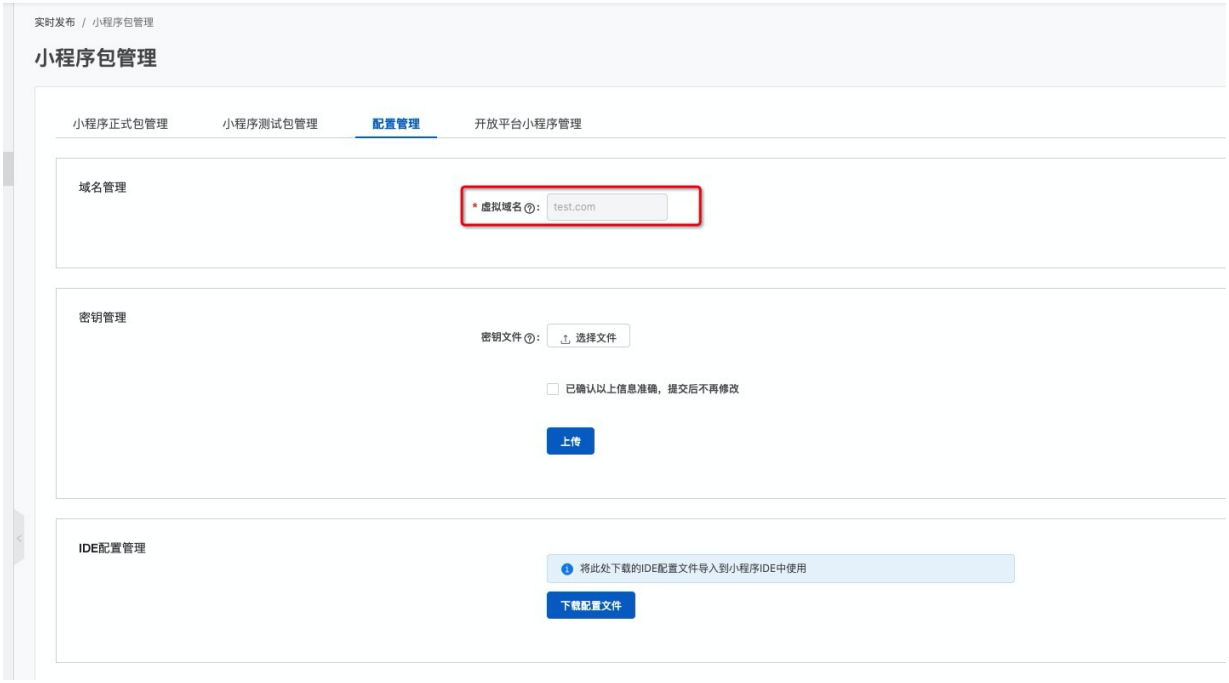
Before starting the Mini Program, you need to release it through the mPaaS console.

2.1 Enter the Mini Program background

Log in to the mPaaS console, enter the target application, and go to the Mini Program > Release Mini Program page from the left navigation bar.

2.2 Configure virtual domain name

If you are using it for the first time, please configure the virtual domain name in **Mini Program** > **Release Mini Program** > **Configuration management**. The virtual domain name can be any domain name. It is recommended to use your enterprise domain name, such as test.com.



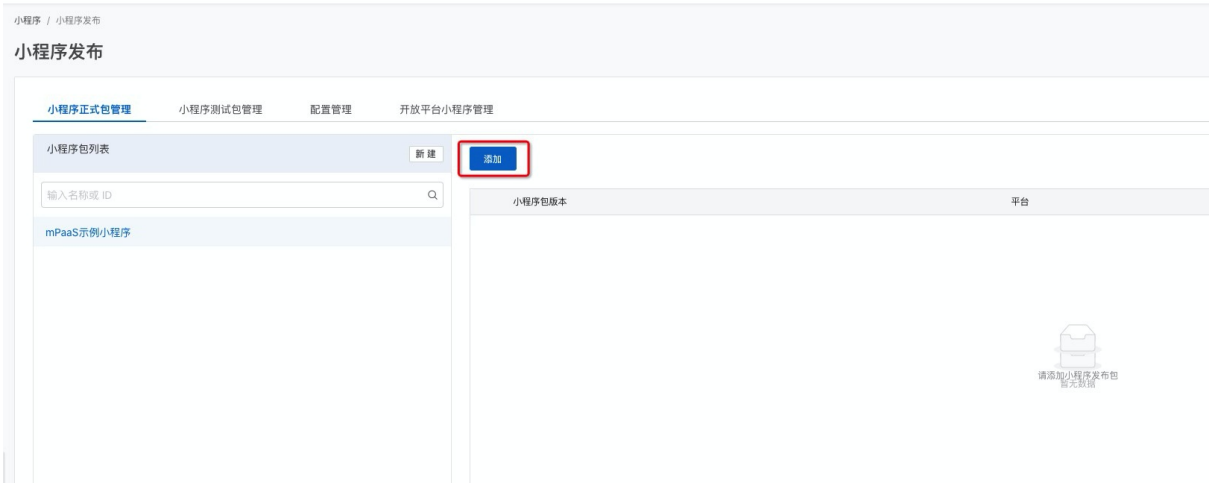
2.3 Create Mini program

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. On the Mini Program package list page, click **New**.
3. In the **New Mini Program** window, enter the ID and the name of the Mini Program, and click **OK**. Among them, the Mini Program ID is a 16-digit number, such as 2018080616290001.

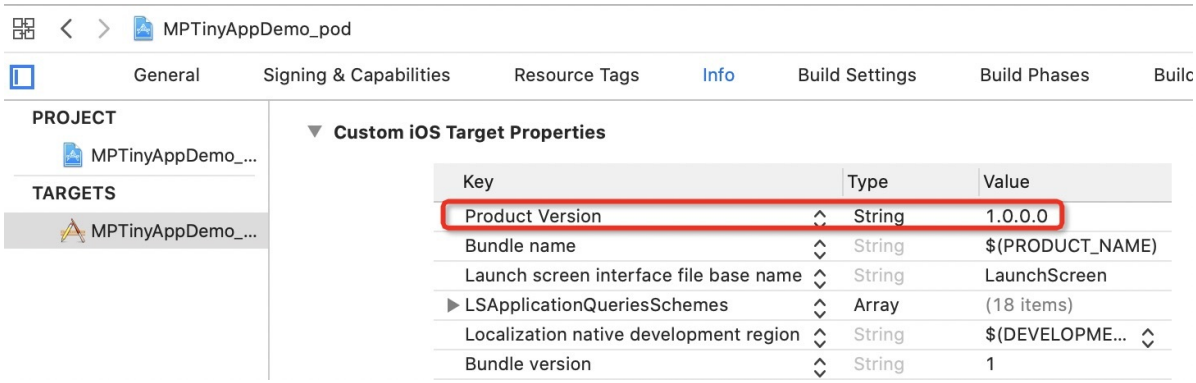


4. Under the Mini Program list, find the new created Mini Program and click **Add**.



5. In the **Basic information** column, complete the following configuration:
   - **Version**: Enter the version number of the Mini Program package, such as 1.0.0.0.
   - **Client range**: Select the minimum and maximum version of the iOS client of the App where the Mini Program is located. The client App within this range can start the corresponding Mini Program, otherwise it cannot be started. The minimum version can be 0.0.0, and the latest version can be left blank, which means that all versions of the client can start this Mini Program.

     Note: The version number here refers to the version number of the current client App, please refer to the **Product Version** field in the `Info.plist` of the project.



   - **Icon**: Click **Upload file** to upload the icon of the Mini Program package. The icon must be uploaded when creating a Mini Program for the first time. Example icons are as follows:



   - File: Upload the Mini Program package resource file, the file format is .zip. We have prepared an mPaaS sample Mini Program for you (click here to download), you can upload it directly.



6. In the configuration information area, complete the following configuration:

- **Main entry URL**: Required, the homepage of the Mini Program package, such as `/index.html#page/tabBar/component/index`.
- Keep other configurations as default.



7. Check **The above information has been confirmed to be accurate and will not be modified after submission**.
8. Click **Submit**.

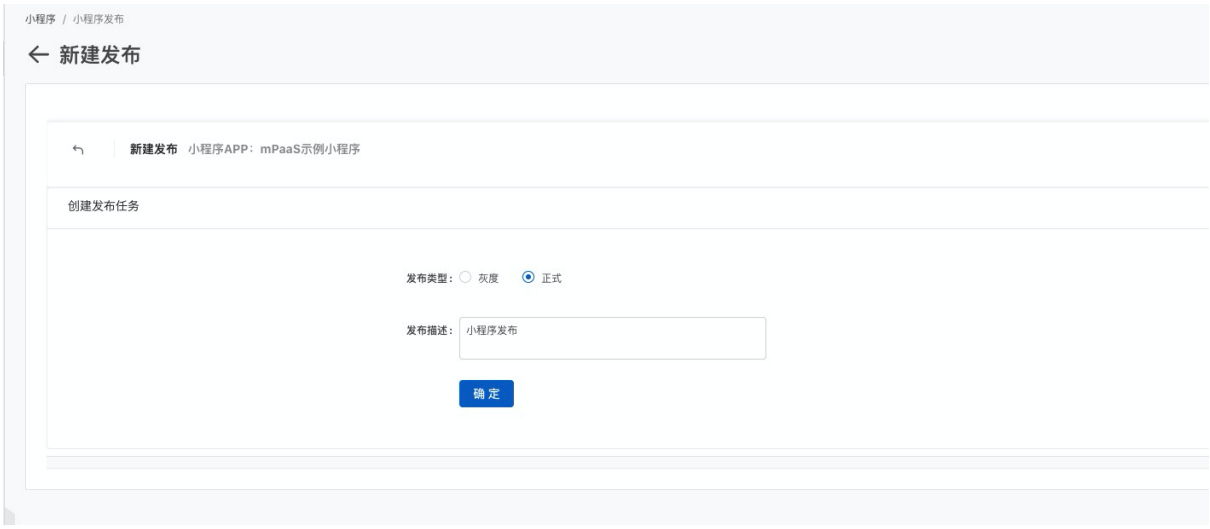
2.4 Release Mini Program

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In the opened Mini Program package list page, select the Mini Program package and version you want to release, and click **Create release**.



3. In the **Create release task** area, complete the following configuration:
   - **Release type**: Select **Official** release type.
   - **Release description**: Optional.
4. Click OK to complete the release creation.



3. Start the Mini Program

After completing the above steps, when entering the corresponding page, call the `startTinyAppWithId` interface method provided by the framework to load the Mini Program.

```
[MPNebulaAdapterInterface startTinyAppWithId:appId params:nil];
```

If you need to pass parameters when starting the Mini Program, you can set it through the `param` parameter. `param` contains two fields: `page` and `query`:

- `page`: Used to specify the path to open a specific page.
- `query`: Used to pass in custom parameters. Multiple key-value pairs are spliced with `&`.

```
NSDictionary *param = @{@"page":@"pages/card/index", @"query":@"own=1&sign=1&code=2452473"};
[MPNebulaAdapterInterface startTinyAppWithId:appId params:dic];
```

1.3.2. Advanced guide

1.3.2.1. Preview and debug iOS Mini Program on real device

Access real-device preview and debug for iOS Mini Program

Note: It is only supported in mPaaS 10.1.60 and above.
Follow the steps below to access the preview and debugging functions:

1. Obtain the QR code content string according to the IDE's QR code (for example, by scanning the code).
2. Call the Mini Program preview debugging interface.
   - Pass in QR code content string:

     ```
     [MPNebulaAdapterInterface startDebugTinyAppWithUrl:qrCode];
     ```

   - Or interface with custom parameters:

     ```
     [MPNebulaAdapterInterface startDebugTinyAppWithUrl:qrCode params:nil];
     ```

     If you need to pass parameters when starting the Mini Program, you can set it through the param parameter. `param` contains two fields: `page` and `query`:

     - `page`: Used to specify the path to open a specific page.
     - `query`: Used to pass in custom parameters. Multiple key-value pairs are spliced with `&`.

       ```
       NSDictionary *param = @{@"page":@"pages/card/index", @"query":@"own=1&sign=1&code=2452473"};
       [MPNebulaAdapterInterface startTinyAppWithId:appId params:dic];
       ```

Configure whitelist

When using the real-device preview and debugging function, the client needs to configure the unique user ID in the `category` of `MPaaSInterface`. According to the actual situation of the application, returns the unique ID of the App in the userId method, such as user name, mobile phone number, email, etc. The value to be entered in the **Whitelist configuration** plugin of the Mini Program IDE must be consistent with the userId configured here.

```
#import <mPaas/MPaaSInterface.h>
@implementation MPaaSInterface (MPTinyAppDemo_pod)

- (NSString *)userId
{
    return @"mPaaS";
}

@end
```

Keep iOS Mini Program alive

Keeping the Mini Program alive means that after starting the Mini Program in the App, when you exit the Mini Program but do not exit the App, the Mini Program will continue to be active for a period of time. When you start the Mini Program again, it will return to the state it was last exited. The keep-alive time is 60s.

Use

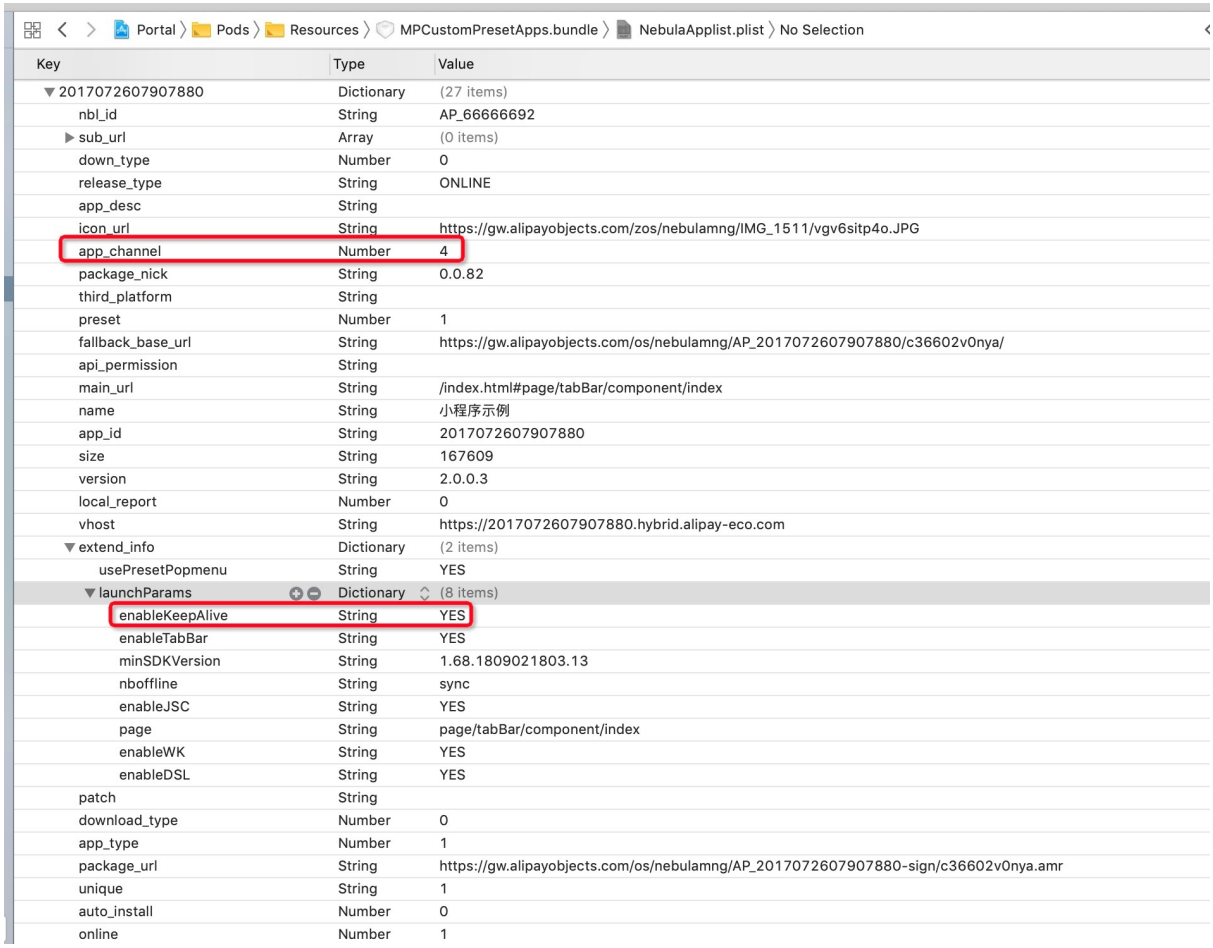
There is a concept of **scene** in the Mini Program, which refers to the path the user enters the Mini Program, and the scene value is the value used to describe the path. Whether the Mini Program can be kept alive depends mainly on whether the scene when the Mini Program is re-opened is the same as the scene when the Mini Program is exited, and whether the time interval between reopening the Mini Program exceeds the keep-alive time.

For example: Scanning and searching are two different scenarios. Assuming that the scene value of the Mini Program opened by scanning a code is A, after exiting, if the Mini Program is re-opened by scanning the code again, the keep-alive takes effect. However, if the Mini Program is opened through search, Assuming that the search scene value is B, the previous cache of the Mini Program will be cleared, and the keep-alive will not take effect.

- In addition to enabling keep-alive when configuring the Mini Program package in the mPaaS console, when calling the starting Mini Program function, you also need to pass in `chInfo`, which is the **scene value**, so that the keep-alive will take effect. The code sample is as follows:

```
[MPNebulaAdapterInterface startTinyAppWithId:item[0] params:@{@"chInfo" : @"MPPortal_home"}]
```

- Please check whether the following configuration is correct for the preset package:



Precautions

- When the account changes, you need to inform the Mini Program container to release the keep-alive Mini Program of the previous account, and you need to call the following code:

```
NSDictionary *userInfo = @{@"login_notifaction_changeAccount" : @(YES)
                           };
    [[NSNotificationCenter defaultCenter] postNotificationName:@"APLoginControllerDidFinishNotification" object:nil userInfo:userInfo];
```

  Note: If the container is not informed, it may happen that when the Mini Program is opened again, due to the existence of keep-alive, it will return to the state when the previous account exited the Mini Program.

- After accessing the Account Link, when the login is authorized by Alipay, the above notification will be automatically sent out due to the account change, and the current cached Mini Program will be cleared, which will cause the keep-alive become invalid.
- Use keep-alive with caution. Once there is a problem with the Mini Program, due to the existence of the keep-alive, users may have to restart the App to use the Mini Program normally.

1.3.2.2. Customize navigation bar of iOS Mini Program

Starting from the 10.1.60 baseline, the iOS Mini Program supports customization of the navigation bar. You can customize the title, background, back button, settings and close buttons on the right side of the navigation bar. This section gives you a detailed introduction to how to customize the navigation bar of the iOS Mini Program.

Starting from the 10.1.60 baseline, the iOS Mini Program supports customization of the navigation bar. You can customize the title, background, back button, settings and close buttons on the right side of the navigation bar. This section gives you a detailed introduction to how to customize the navigation bar of the iOS Mini Program.

Customize navigation bar background and title

Globally customize the navigation bar background and title

If you want to customize the default navigation bar background and title of all pages of the Mini Program globally, you need to modify the `window` configuration in `app.json` .

- Hide the navigation bar: You need to customize the JSAPI implementation.
- Transparent navigation bar: `"transparentTitle": "always"` .
- Navigation bar gradient: `"transparentTitle": "auto"` .
- Navigation bar color: `"titleBarColor": "#f00"` .
- Navigation bar title text: `"defaultTitle": "Alert"` .
- Navigation bar title color: Modify the `titleView` style of the current page, in the `super` of the `viewWillAppear` method of the HTML5 base class.

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    ...
    BOOL isTinyApp = [NBUtils isTinyAppWithSession:self.psdSession];
    if (isTinyApp) {
        id<NBNavigationTitleViewProtocol> titleView = self.navigationItem.titleView;
        [[titleView mainTitleLabel] setFont:[UIFont systemFontOfSize:16]];
        [[titleView mainTitleLabel] setTextColor:[UIColor redColor]];
    }
 }
```

- Navigation bar title image: `"titleImage": "https://pic.alipayobjects.com/e/201212/1ntOVeWwtg.png"` .
- Navigation bar title position: Please refer to the following code.

```
- (NSDictionary *)nebulaCustomConfig
{
    NSString* leftConfig = @"{\"enable\":true,\"appIdBlacklist\":[],\"appIdWhitelist\":[\".*\"]}";
    return @{@"h5_tinyAppTitleViewAlignLeftConfig" : leftConfig };
}
```

Customize the navigation bar background and title of for a page

If you want to customize the navigation bar background and title of a certain page in the Mini Program, you need to configure it in the `.json` of the page.

- Hide the navigation bar: You need to customize the JSAPI implementation.
- Transparent navigation bar: `"transparentTitle": "always"` .
- Navigation bar gradient: `"transparentTitle": "auto"` .
- Navigation bar color: `"titleBarColor": "#f00"` .
- Navigation bar title text: `"defaultTitle": "Alert"` .
- Navigation bar title color: You need to customize JSAPI, and modify the style of `titleView` of the current page in JSAPI.

```
- (void)handler:(NSDictionary *)data context:(PSDContext *)context callback:(PSD JSAPI ResponseCallbackBlock)callback
{
    [super handler:data context:context callback:callback];

     // You can pass font, color, etc. through data
    id<NBNavigationTitleViewProtocol> titleView = context.currentViewController.navigationItem.titleView;
    [[titleView mainTitleLabel] setFont:[UIFont systemFontOfSize:16]];
    [[titleView mainTitleLabel] setTextColor:[UIColor redColor]];
}
```

- Navigation bar title image: `"titleImage": "https://pic.alipayobjects.com/e/201212/1ntOVeWwtg.png"` .

Dynamically modify the navigation bar background and title of the current page

If you want to dynamically modify the navigation bar background and title of the current page, you need to call `my.setNavigationBar` for configuration.

- Hide the navigation bar: You need to customize the JSAPI implementation.
- Transparent navigation bar: Not supported.
- Navigation bar gradient: Not supported.
- Navigation bar color: `"backgroundColor": "#f00"` .
- Navigation bar title text: `"title": "new title"` .
- Navigation bar title color: You need to customize JSAPI, and modify the style of `titleView` of the current page in JSAPI.

```
- (void)handler:(NSDictionary *)data context:(PSDContext *)context callback:(PSD JSAPI ResponseCallbackBlock)callback
{
    [super handler:data context:context callback:callback];

     // You can pass font, color, etc. through data
    id<NBNavigationTitleViewProtocol> titleView = context.currentViewController.navigationItem.titleView;
    [[titleView mainTitleLabel] setFont:[UIFont systemFontOfSize:16]];
    [[titleView mainTitleLabel] setTextColor:[UIColor redColor]];
}
```

- Navigation bar title image: `"image": "https://pic.alipayobjects.com/e/201212/1ntOVeWwtg.png"`.

Customize the navigation bar back button

If you want to modify the style of the back button globally, you need to modify the style of the `leftBarButtonItem` of the current page in the `super` of the `viewWillAppear` method of the HTML5 base class. The editable styles include the following, you can refer to the code block below for more information.

- Modify the back arrow and text color
- Modify the back arrow style and text content
- Hide back arrow
- Hide back text

```
// Modify the style of the return button on the left.
AUBarButtonItem *backItem = self.navigationItem.leftBarButtonItem;
if ([backItem isKindOfClass:[AUBarButtonItem class]]) {
    // Based on the default back button, modify the back arrow and copy color.
    backItem.backButtonColor = [UIColor greenColor];
    backItem.titleColor = [UIColor colorFromHexString:@"#00ff00"];

    // Modify the return arrow style and text content.
    // backItem.backButtonTitle = @"Return";
    // backItem.backButtonImage = [UIImage imageNamed:@"APCommonUI.bundle/add"];

    // Hide the return arrow.
    // backItem.hideBackButtonImage = YES;

    // Hide the return text: The text is set to transparent, and the click area of the return button is reserved.
    // backItem.titleColor = [UIColor clearColor];
}
```

Set and close buttons on the right side of the navigation bar

Globally modify the image and color of the buttons on the right

If you want to modify the image and color of the buttons on the right, you need to import the header file `#import <TinyappService/TASUtils.h>` and configure as follows.

- Modify the color of the close button: `[TASUtils sharedInstance].customItemColor = [UIColor redColor]`.
- Modify the image of the close button: `[TASUtils sharedInstance].customCloseImage = [UIImage imageNamed:@"xx"]`.
- Show the share button: `[TASUtils sharedInstance].shoulShowSettingMenu = YES`.
- Modify the image of the share button: `[TASUtils sharedInstance].customSettingImage = [UIImage imageNamed:@"xx"]`.
- Modify the color of the share button: `[TASUtils sharedInstance].customItemColor = [UIColor redColor]`.

Globally modify the style of the buttons on the right

If you want to modify the right button style globally, you need to override the `rightBarButtonItem` of the current page in the `viewwillAppear` of the HTML5 base class.

```
- (void)viewWillAppear:(BOOL)animated
{
    [super viewWillAppear:animated];
    ...
    BOOL isTinyApp = [NBUtils isTinyAppWithSession:self.psdSession];
    if (isTinyApp) {
        self.navigationItem.rightBarButtonItem = [[UIBarButtonItem alloc] initWithTitle:@"Close" style:UIBarButtonItemStylePlain target:self action:@
selector(onClickClose)];
    }
 }

- (void)onClickClose
{
    [TASUtils exitTinyApplication:self.appId];
}
```

1.3.2.3. Custom bi-directional channel for iOS Mini Program

If the existing Mini Program API or events cannot meet your development requirements, you can also extend it yourself.

Mini Program calls native custom API

1. The client customizes the API and registers.

   Refer to Custom JSAPI to register your custom API.

2. Mini Program call.

```
my.call('tinyToNative', {
param1: 'p1aaa',
param2: 'p2bbb'
}, (result) => {
console.log(result);
my.showToast({
 type: 'none',
 content: result.message,
 duration: 3000,
});
})
```

Native projects send custom events to Mini Program

1. Mini Program registers the event.

```
my.on('nativeToTiny', (res) => {
my.showToast({
 type: 'none',
 content: JSON.stringify(res),
 duration: 3000,
 success: () => {

 },
 fail: () => {

 },
 complete: () => {

 }
});
})
```

2. The client sends the event.

   Get the `viewController` where the current Mini Program page is located, and call the `callHandler` method to send the event.

```
 [self callHandler:@"nativeToTiny" data:@{@"key":@"value"} responseCallback:^(id responseData) {

 }];
```

Parameter description::

Parameter	Description
handlerName	The name of the event monitored by the Mini Program.
data	The parameters passed by the client to the Mini Program.
callback	The call back processing block after the Mini Program executes the event.

Unregister custom event

If you no longer need custom events, please refer to Unregister custom events.

1.3.2.4. Customize startup loading page of iOS Mini Program

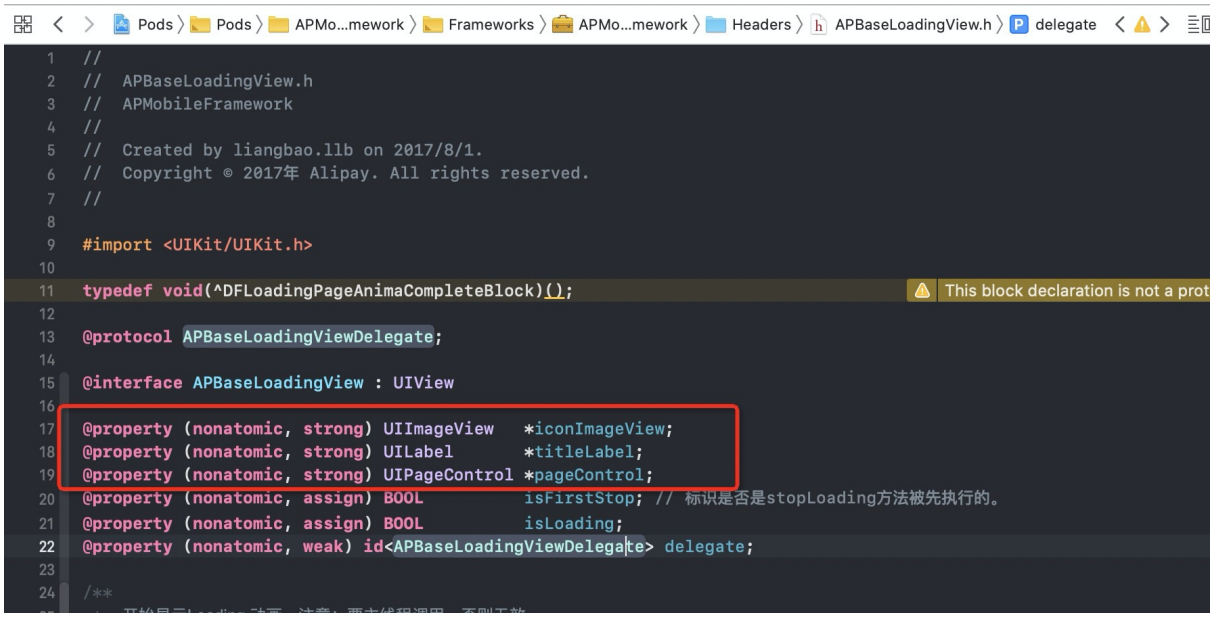
When the Mini Program is started, if the Mini Program has not been downloaded to the device, the Mini Program container will launch the loading page (as shown in the image below) to prompt the user to wait. After the Mini Program is installed on the device, the loading page will be closed and the page will jump to the Mini Program.



Implement a custom loading page

For iOS Mini Program, mPaaS supports developers to customize the content of the loading page. You can configure it according to the following steps:

1. Inherit the subclass of `APBaseLoadingView` and customize the View subclass of the loading page. You can modify the style of the page view in the subclass.





The code sample is as follows:

```
@interface MPBaseLoadingView : APBaseLoadingView

@end

@implementation MPBaseLoadingView

- (instancetype)init
{
    self = [super init];
    if (self) {
        self.backgroundColor = [UIColor grayColor];
        self.titleLabel.backgroundColor = [UIColor redColor];
        self.titleLabel.font = [UIFont boldSystemFontOfSize:8];

        self.iconImageView.backgroundColor = [UIColor blueColor];

        self.pageControl.backgroundColor = [UIColor orangeColor];

    }

    return self;
}

- (void)layoutSubviews
{
    [super layoutSubviews];
    // Adjust the position of the view

    CGSize size  = self.bounds.size;
    CGRect frame = CGRectMake((size.width - 80)/2, 0, 80, 80);
    self.iconImageView.frame = frame;

    frame = CGRectMake(15, CGRectGetMaxY(self.iconImageView.frame) + 6, size.width - 30, 22);
    self.titleLabel.frame = frame;

    frame = CGRectMake((size.width-40)/2, CGRectGetMaxY(self.titleLabel.frame) + 21, 40, 20);
    self.pageControl.frame = frame;
}

@end
```

2. In the category of the `DTFrameworkInterface` class, override the `baseloadViewClass` method to return the custom load page View class name.

```
- (NSString *)baseloadViewClass
{
    return @"MPBaseLoadingView";
}
```

1.3.2.5. Customize error page for iOS mini program

When loading the mini program, if it fails to load the page or the website cannot be opened, an error page similar to the following will appear:



This article introduces how to customize the error page in the above figure.

Procedure

Customizing the error page falls into the following 2 steps:

1. Listen to the `kEvent_Navigation_Error` method in HTML5 base class.

   Introduce `- (void)handleEvent:(PSDEvent *)event` method via `MPH5WebViewController () <PSDPluginProtocol>` interface:

```
- (void)handleEvent:(PSDEvent *)event
{
    [super handleEvent:event];

        if ([kEvent_Navigation_Error isEqualToString:event.eventType]) {
        [self handleContentViewDidFailLoad:(id)event];
    }
}
```

   `handleContentViewDidFailLoad` method is as follows:

```
- (void)handleContentViewDidFailLoad:(PSDNavigationEvent *)event
{
    PSDNavigationEvent *naviEvent = (PSDNavigationEvent *)event;
    NSError *error = naviEvent.error;
    [MPH5ErrorHelper handlErrorWithWebView:(WKWebView *)self.psdContentView error:error];
}
```

2. Set `error` page and HTML5 base class in `afterDidFinishLaunchingWithOptions` method.

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary
    *)launchOptions
{
    #ifdef DEBUG
        NBLogConfigurationGet().enableConsoleLog = YES;
    #endif

    // 设置基类为MPH5WebViewController
    [MPNebulaAdapterInterface shareInstance].nebulaVeiwControllerClass = [MPH5WebViewController class];
        //设置H5容器基类
    [MPNebulaAdapterInterface shareInstance].nebulaWebViewClass = [MPH5WKWebView class];
    [MPNebulaAdapterInterface shareInstance].nebulaUserAgent = @"mPaaS/Portal";//设置H5容器UserAgent
    [MPNebulaAdapterInterface shareInstance].nebulaUseWKArbitrary = YES; //开启 WKWebview
    [MPNebulaAdapterInterface shareInstance].nebulaCommonResourceAppList = @[@"77777777"];// 设置全局资源包
    [MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = NO; // 关闭离线包验签，正式版本请开启验签

    //设置error页面
    NSString *filePath = [[NSBundle mainBundle] pathForResource:@"myerror" ofType:@"html"];
    [MPNebulaAdapterInterface shareInstance].errorHtmlPath = filePath;

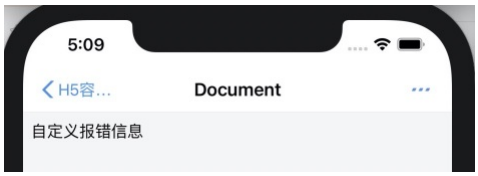
    // 更新离线包
    [[MPNebulaAdapterInterface shareInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
        NSLog(@"[mpaas] nebula rpc data :%@", data);
    }];
}
```

In which, `errorHtmlPath` is the HTML error page path displayed when it fails to load the HTML5 page, and reads `MPNebulaAdapter.bundle/error.html` by default.

The code of `myerror` is as follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Custom error message
</body>
</html>
```

The result is as follows:



1.3.2.6. iOS Mini Program supports custom View

The Mini Program supports the custom view function since mPaaS 10.1.68.36.

The Mini Program supports the custom view function since mPaaS 10.1.68.36.

Procedure

1. Inherit the NBComponent interface.

```
@interface CustomTestView : NBComponent
```

2. Rewrite the following method to return the View created in `init`.

```
- (id)initWithConfig:(NSDictionary *)config messageDelegate:(id<NBComponentMessageDelegate>)messageDelegate {
    self = [super initWithConfig:config messageDelegate:messageDelegate];
    if (self) {
        self.contentSpaceView = [[UIView alloc] init];
        self.contentSpaceView.backgroundColor = [UIColor orangeColor];
        self.contentSpaceView.frame = CGRectMake(0, 0, 100, 100);
        UITapGestureRecognizer *tap = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(postMessage)];
        [self.contentSpaceView addGestureRecognizer:tap];
    }
    return self;
}
```

```
//Return the View created in `init`.
- (UIView *)contentView {
    return self.contentSpaceView;
}
```

3. Receive messages from the Mini Program.

```
- (void)componentReceiveMessage:(NSString *)message data:(NSDictionary *)data callback:(NBComponentCallback)callback {
    if ([message isEqualToString:@"setColor"]) {
        callback(@{@"success":@"1"});
    }else if ([message isEqualToString:@"startAnimation"]) {
        [self.nbComponentMessageDelegate sendCustomEventMessage:@"nbcomponent.mpaasComponent.customEvent" component:self data:@{@"sth":@"start"} c
allback:^(NSDictionary * _Nonnull data) {

        }];
    }
}
```

4. Send a message to the Mini Progrm.

```
[self.nbComponentMessageDelegate sendCustomEventMessage:@"nbcomponent.mpaasComponent.customEvent" component:self data:@{
            @"element":@"elementId",
            @"eventName":@"onXxx",
            @"data":{}
        } callback:^(NSDictionary * _Nonnull data) {

        }];
```

The parameters descriptions are as follows:

Parameter	Description
element	ID in the tag.
eventName	The corresponding event, starts with on.
data	Custom event parameter.

5. Register a custom View.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    [[PSDService sharedInstance] registerComponentWithName:@"componentName" clsName:@"className"];

}
```

6. The Mini Program calls the custom View.

```
<mpaas-component
  id="mpaas-map"
  type="custom_map"
  style="{{ width: 200, height: 200 }}"
/>
```

The label `mpaas-component` is a fixed value, please do not modify it. Other parameters are described as follows:

Parameter	Description
id	The ID of the instance of custom View, duplicate ID should not occur in one mini program.
type	The type of the custom View must be consistent with the custom View parameter `componentName` registered by the client. Adding a prefix is recommended.
style	Set the width and height.

7. The description for custom parameters of Mini Program is as follows:

```
<mpaas-component
  id="mpaas-map"
  type="custom_map"
  style="{{ width: 200, height: 200 }}"
  color="#FFFF00FF"
  ...
/>
```

Note

- Color is a custom rendering parameter, and it can also be named arbitrarily.
- Id, type, and style are the default fields, these fields should not be used as the names of custom rendering parameters for custom View.
- Custom rendering parameters cannot start with on, and the type cannot be func.

8. The client receives custom parameters.

```
- (void)componentDataWillChangeWithData:(NSDictionary *)data {

}

- (void)componentDataDidChangeWithData:(NSDictionary *)data {

}
```

Other component internal methods

```
// self.nbComponentMessageDelegate method
@protocol NBComponentMessageDelegate <NSObject>
@required
/**
 * The component actively sends a message to the page (Native->Page)
 *
 * @param message Message name.
 * @param component The component to send the message.
 * @param data Content of the message.
 * @param callback The callback after the page has processed the message.
 *
 * @return void
 */
- (void)sendMessage:(NSString *)message
          component:(id<NBComponentProtocol>)component
               data:(NSDictionary *)data
           callback:(NBComponentCallback)callback;
@optional
/**
 * Components can directly execute JS in the execution environment.
 *
 * @param javaScriptString JS to be executed.
 * @param completionHandler Executes the callback function.
 *
 * @return void
 */
- (void)evaluateJavaScript:(NSString *)javaScriptString completionHandler:(void (^ _Nullable)(_Nullable id, NSError * _Nullable error))completionHandler;
/**
 * The component actively sends a message to the page (Native->Page).
 *
 * @param message Message name (message is not processed internally).
 * @param component The component to send the message.
 * @param data Content of message.
 * @param callback The callback after the page has processed the message.
 *
 * @return void
 */
- (void)sendCustomEventMessage:(NSString *)message
                     component:(id<NBComponentProtocol>)component
                          data:(NSDictionary *)data
                      callback:(NBComponentCallback)callback;
@end
@protocol NBComponentLifeCycleProtocol <NSObject>
- (void)componentWillAppear;
- (void)componentDidAppear;
/**
 * The component will be destroyed.
 *
 * @return void
 */
- (void)componentWillDestory;
/**
 * After the component is destroyed.
 *
 * @return void
 */
- (void)componentDidDestory;
- (void)componentWillResume;
- (void)componentDidResume;
- (void)componentWillPause;
- (void)componentDidPause;
//fullscreen
/**
 The component is about to enter the fullscreen callback.
 */
- (void)componentWillEnterFullScreen;
/**
 Callback when component enters fullscreen.
 */
- (void)componentWillExitFullScreen;
/**
 The component is about to exit the fullscreen callback.
 */
- (void)componentDidEnterFullScreen;
```

```
/**
 Component exits the callback of fullscreen.
 */
- (void)componentDidExitFullScreen;

//visiblity
/**
 The component is about to exit the fullscreen callback.
 */
- (void)componentDidHidden;
/**
 Component exits the callback of fullscreen.
 */
- (void)componentDidVisiblity;
@end
@protocol NBComponentDataProtocol <NSObject>
/**
 * Component data will be updated.
 *
 * @param data Data content.
 *
 * @return void
 */
- (void)componentDataWillChangeWithData:(NSDictionary *)data;
/**
 * The data of the component has been updated, it is generally necessary to update the interface or perform other operations of the component.
 *
 * @param data Data content.
 *
 * @return void
 */
- (void)componentDataDidChangeWithData:(NSDictionary *)data;
@end
@protocol NBComponentFullScreenProtocol <NSObject>
/**
 Whether it is in fullscreen mode.

 @return Whether it is in fullscreen mode.
 */
- (BOOL)isFullScreen;
/**
 @return If enter fullscreen mode is needed.
 */
- (BOOL)shouldEnterFullScreen;
/**
 Set whether the ContentView needs to be fullscreen, the business can switch to fullscreen mode by changing it.
 @param fullScreen Whether fullscreen is required.
 @param shouldRotate Whether you need to rotate the screen.
 */
- (void)setContentViewFullScreen:(BOOL)fullScreen shouldRotate:(BOOL)shouldRotate;
@end
@protocol NBComponentVisibilityProtocol <NSObject>
/**
 State of visibilityState.
 @return State of visibilityState.
 */
- (NBComponentVisibilityState)visibilityState;
/**
 Set the state of VisibilityState
 @param state VisibilityState
 @return set successfully or not
 */
- (BOOL)setVisibilityState:(NBComponentVisibilityState)state;
/**
 Business rewrites this method to return the result of monitoring visibility changes or not, the default is NO.
 @return if monitor the changing of visibility is needed.
 */
- (BOOL)shouldObServerVisibilityStateChange;
@end
```

1.3.2.7. API permission extended configuration for iOS Mini Program

Some special APIs of Mini Program, such as Location, Camera, Album, etc., usually prompt the user for authorization, and the API can be executed after the user allows it.

Some special APIs of Mini Program, such as Location, Camera, Album, etc., usually prompt the user for authorization, and the API can be executed after the user allows it.



The Mini Program container allows the following extensions for API calls:

- Customize the prompt text, and the access party can control the text and display style.
- Allow the access party to read and write permissions.

Note: This extended configuration is only available when the Mini Program permission control is enabled in the background.

Permission configuration

The Mini Program already has a default configuration key and corresponding API, see the following table for details:

Permission	key	API
Camera	camera	scan, chooseImage, chooseVideo
Album	album	saveImage, saveVideosToPhotosAlbum
Location	location	getLocation, getCurrentLocation
Microphone	audioRecord	startAudioRecord, stopAudioRecord, cancelAudioRecord

You can get the permission dictionary that the current Mini Program has requested:

NebulaSDKPlugins 〉 Sources 〉 NebulaPlugins 〉 Auth 〉 TAAuthorizeStorageManager.h 〉 TAAuthorizeStorageManager

```
@end

@interface TAAuthorizeStorageManager : NSObject


@property(nonatomic, weak) id<MPNebulaAdapterInterfaceAuthorizeAlert> authorizeAlertDelegate; // 授权弹框del

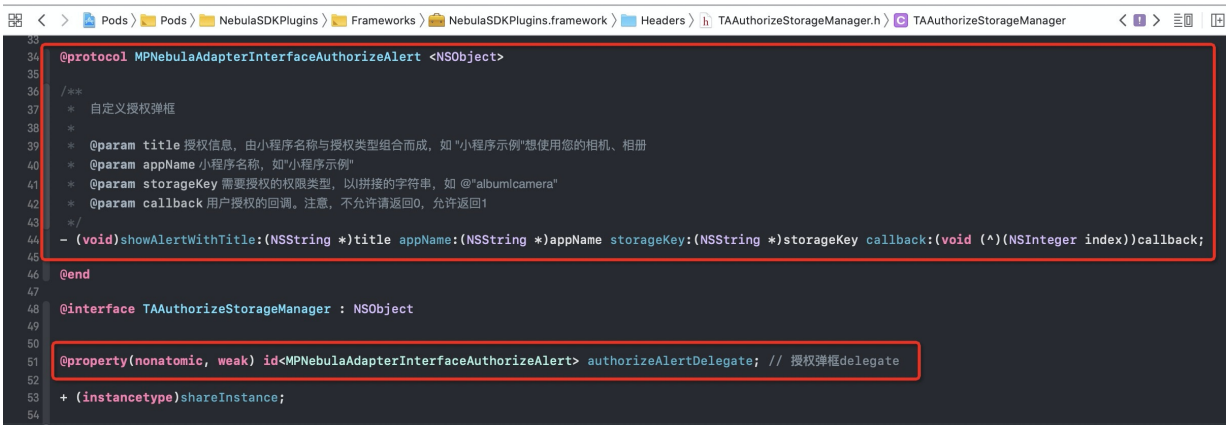
+ (instancetype)shareInstance;

/**
 *  获取appid对应小程序已经请求过的权限字典。
 *
 *  @return 权限状态字典
 */
- (NSMutableDictionary *)authStatusDic4AppId:(NSString *)appid;

/**
```

Custom display

mPaaS supports the display of custom permission popups, you can set it through the interface in the image below.



The implementation steps are as follows:

1. After the container is initialized, specify the delegate of the custom permission popup.

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
   ...

    // Mini Program API Permission Control
   [TAAuthorizeStorageManager shareInstance].authorizeAlertDelegate = self;

   ...
}
```

2. Implement custom popup method.

```
#pragma mark Mini Program API Permission Control
- (void)showAlertWithTitle:(NSString *)title appName:(NSString *)appName storageKey:(NSString *)storageKey callback:(void (^)(NSInteger index))cal
lback
{
    if ([title length] > 0) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:title
                                                        message:nil
                                                       delegate:self
                                              cancelButtonTitle:@"Cancel"
                                              otherButtonTitles:@"OK", nil];

        self.callback = callback;
        [alert show];
    }

}

- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (self.callback) {
        if (buttonIndex == alertView.cancelButtonIndex) {
          // The user allows authorization, the callback parameter is 0
          self.callback(0);
        }else{
          // The user allows authorization, the callback parameter is 1
          self.callback(1);
        }
    }

}
```

1.3.2.8. Preset an iOS mini program in the client

The traditional Mini Program technology can be easily affected by the network environment. Therefore, you may fail to pull the Mini Program package when the network condition is poor. However, you can avoid this problem by presetting the Mini Program. This topic introduces the principle and implementation process of presetting a Mini Program.

About presetting a Mini Program

Presetting a Mini Program is to pack the static resources of the Mini Program such as rendering, logic, and configuration into a compressed package. Then, the client downloads the Mini Program package to the local in advance and loads the resources locally. Presetting the Mini Program can minimize the impact of the network environment on the mPaaS Mini Program page. The benefits of using the preset package are as follows.

- **Improve the user experience**

  By embedding the static resources of the page into the App through the preset package and releasing the resources with the App, when you open the App for the first time, you can directly use the resources without relying on the network environment to download them.

- **Implement dynamic update**

  When a new version or emergency release is about to be launched for a Mini Program, you can conduct iterative development in the mini program IDE and release the new version in the mPaaS console. The Mini Program SDK integrated in the client will then automatically update the Mini Program to the latest version. This release

method does not require App Store review, allowing users to receive updates in a timely manner.

Prerequisites

- You have accessed the mini program component. For details about accessing the mini program component, see Get Started with Mini Programs.

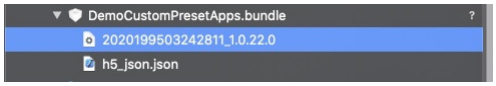
Procedure

1. Preset the mini program package.
   i. In the mPaaS console, release the Mini Program package and download the AMR and configuration files.



   ii. Create an independent bundle, such as `DemoCustomPresetApps.bundle`, and add the AMR offline package and the `h5_json.json` file downloaded from the release platform to this bundle.

> **Note** Currently, the release platform only supports downloading the `h5_json.json` configuration file of a single offline package. To preset multiple mini program package, the data in different `h5_json.json` files need to be manually merged into one configuration file.



   iii. When initializing the mini program, set the offline path of the preset mini program offline package in the `initNebulaWithCustomPresetApplistPath` interface to the bundle created in the previous step.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
// Initialize rpc
[MPRpcInterface initRpc];
// Initialize the container
//    [MPNebulaAdapterInterface initNebula];
// Custom JSAPI path and preset mini program package information
NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle/h5_json.json"] ofTy
pe:nil];
NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPlugins.bundle/Poseidon-UserDefine-Ex
tra-Config.plist"] ofType:nil];
[MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath customPresetAppPackagePath:appPackagePath customPluginsJsapisPa
th:pluginsJsapisPath];
}
```

2. Start the Mini Program.

   Similar to the non-preset mini programs, when entering the corresponding page, the interface method provided by the Nebula container is called to load the mini program.

```
[MPNebulaAdapterInterface startTinyAppWithId:@"2020121720201217" params:nil];
```

3. Update the Mini Program.

   By default, every time you open the App, the Mini Program SDK attempts to check if a newer version is available. The check interval is restricted to 30 minutes by default to minimize the stress on the server. To instantly check for the latest available version, call the following code to request an update. Generally, you can call the code after the App starts or the user logs in.

```
-(void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions{
 // Fully update the local mini program package information
 [[MPNebulaAdapterInterface shareInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
     NSLog(@"[mpaas] nebula rpc data :%@", data);
 }];
}
```

4. Check the security signature.

   The Mini Program has a signature verification mechanism to prevent malicious programs from tampering with the mini program package downloaded to the device. To enable this mechanism, call the mini program API to configure verification parameters.

   Note:

   - Be sure to call the `MPNebulaAdapterInterface` API before opening the mini program package for the first time. Otherwise, public key initialization will fail. For details about public and private keys, see Configure the mini program Package > Manage keys.
   - Enable signature verification.

```
[MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = YES;
```

5. Delete the local Mini Program.

   Nebula provides the API for deleting local App information. After local App information is deleted, opening the App again requests the server to download and update local Mini Program information again.

```
/**
*  @brief Delete local app information (including package information, amr, and installation directory)
*
*  @date 2019-02-28
*
*  @return
*/
-(void)clearAllAppInfo:(NSString *)appId;
// Method to use
[[NBServiceGet() appCenter] clearAllAppInfo:@"2020199503242811"];
```

1.3.2.9. Add extended information for iOS Mini Program

When releasing the mini program in the mPaaS console, you need to add a mini program package and complete configuration for the package. The extension information in the configuration must start with `launchParams`, otherwise the iOS client cannot obtain it.

Obtain extension information

```
    NAMApp *app = [NAMServiceGet() findApp:appId version:@"1.0.0.2"];
    NSString *extend_info = app.extend_info;
```

1.3.2.10. Pass startup parameters to iOS Mini Program

This topic introduces the steps to pass parameters to the default receiving page (`pages/index/index`) of the Mini Program by passing `name` and `pwd` parameters as an example.

Prerequisites

You have accessed the mini program component by referring to Quick start.

Procedure

1. On the client side, add parameters of redirecting page during startup. The following shows an example:

```
 NSString *pwd = [@"123&*!@#$%^*" stringByAddingPercentEncodingWithAllowedCharacters:[[NSCharacterSet characterSetWithCharactersInString:@"?!@#$^&%
*+,:;='\"`<>()[]{}/\\| "] invertedSet]];

 NSString *queryvalue = [NSString stringWithFormat:@"name=mpaas&pwd=%@",pwd];
 NSDictionary * dic = @{@"query":queryvalue};

 [MPNebulaAdapterInterface startTinyAppWithId:@"1234567891234567" params:dic];
```

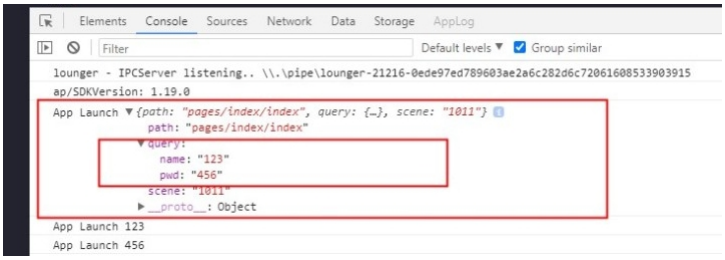
   When the URL starts to pass parameters, the parameter passing field is `query`. When the Mini Program obtains the parameters, it parses the `query` field.

   Description of startApp parameter:

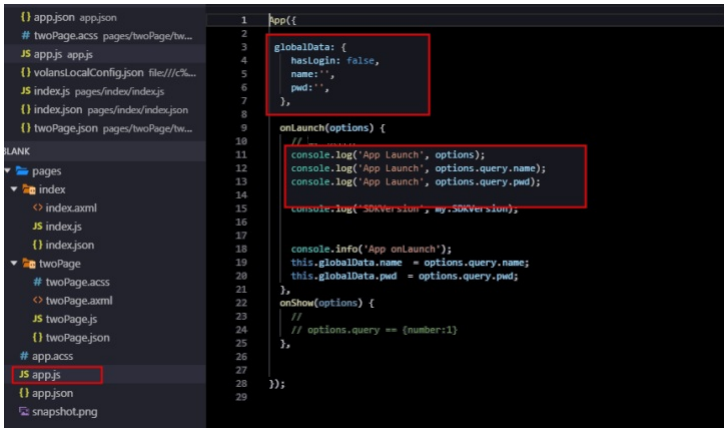
   - `appId`: The Mini Program ID, which can be viewed in the mPaaS console.
   - `params`: The Mini Program parameters. Pass custom parameters in the format of `@{@"query":@"key=value&key=value"};`. Separate the parameters with ampersand (&).

     > **Important**
     >
     > - The Mini Program framework decodes the `value` of the key-value pair for each custom input parameter. If special character `&` is inclduded in `value`, you need to encode the parameters by calling the following method.
     >
     >   `NSString *pwd = [@"123&*!@#$%^*" stringByAddingPercentEncodingWithAllowedCharacters:[[NSCharacterSet characterSetWithCharactersInString:@"?!@#$^&%*+,:;='\"`<>()[]{}/\\| "] invertedSet]];`
     >
     >   Do not encode the parameters if there is no special character.
     > - The Mini Program framework does not process the `key` of the key-value pair for custom input parameters. Therefore, make sure not to set special characters for the `key`, otherwise the Mini Program framework may fail to recognize the custom parameters.

2. The Mini Program obtains the parameters from the `options` parameter of the `onLaunch/onShow(options)` method.



   The `app.js` storage script obtains the parameters passed from the client to the Mini Program and stores them to the global variable `globalData`. The parameters are obtained directly from `globalData` when used. For example, after parameters such as `token` and `user_id` in the request header are passed from Native and stored in `globalData`, the Mini Program directly obtains the values from `globalData`.



Jump to the page specified by the applet through the startup parameters

When the applet has multiple pages, the business needs to display the specified page when opening the applet. This can be achieved by the following code:

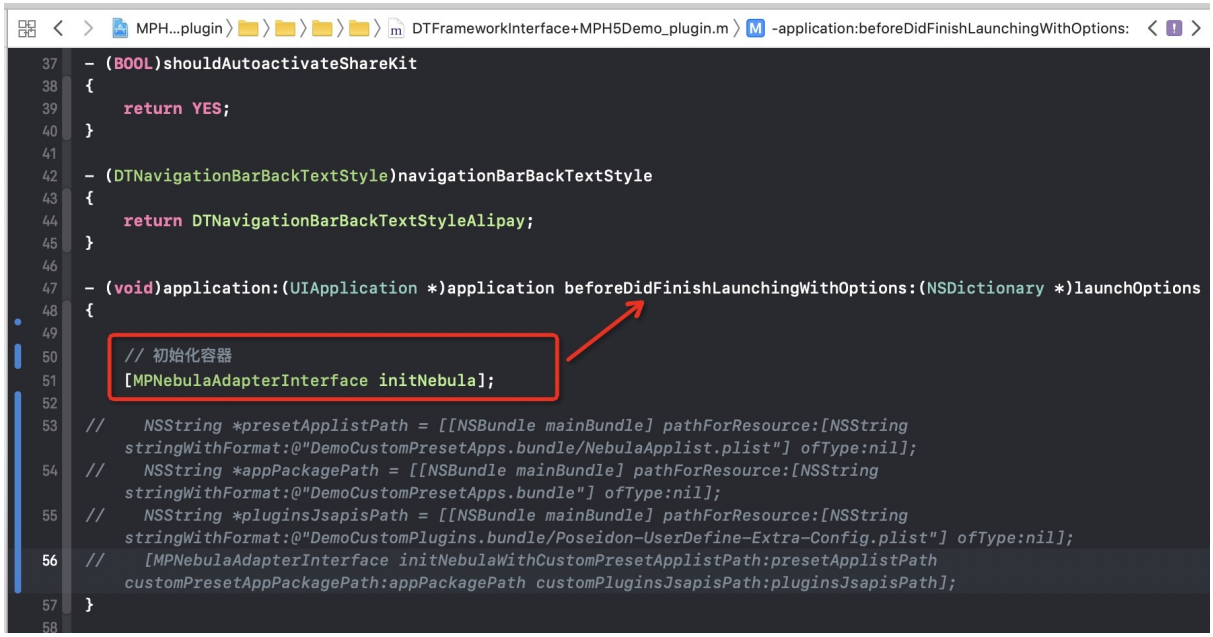
```
NSDictionary * dict = @{
    @"page" : @"/pages/demo3/index"
};
[MPNebulaAdapterInterface startTinyAppWithId:@"1234567891234567" params:dict];
```

1.3.3. Mini Program upgrade instructions

Important: Starting on June 28, 2020, mPaaS stopped maintaining the 10.1.32 baseline. Please use 10.1.68 or 10.1.60 series baseline.

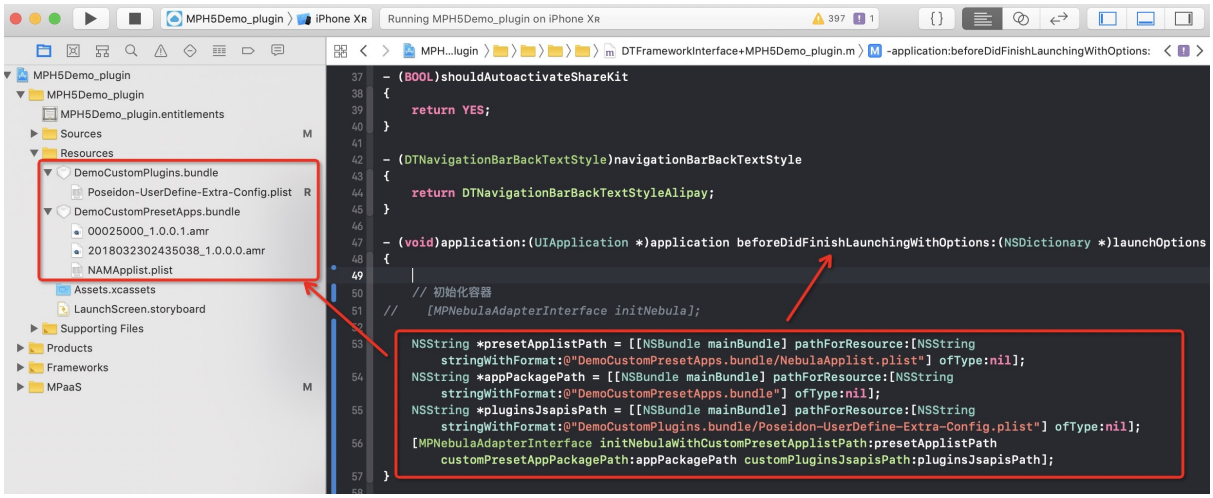
Initialization configuration

- Timing of initialization: It needs to be called before the frame is loaded, and it must be called in `- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` of `DTFrameworkInterface`.



- If the existing project baseline is 10.1.32, you need to modify the custom JSAPI path, preset mini program package and package information path:

`initNebulaWithCustomPresetApplistPath` must be called in `- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions` of `DTFrameworkInterface`.



1.4. Develop mini program

1.4.1. Quick start

Developing a Mini Program usually includes the following steps:

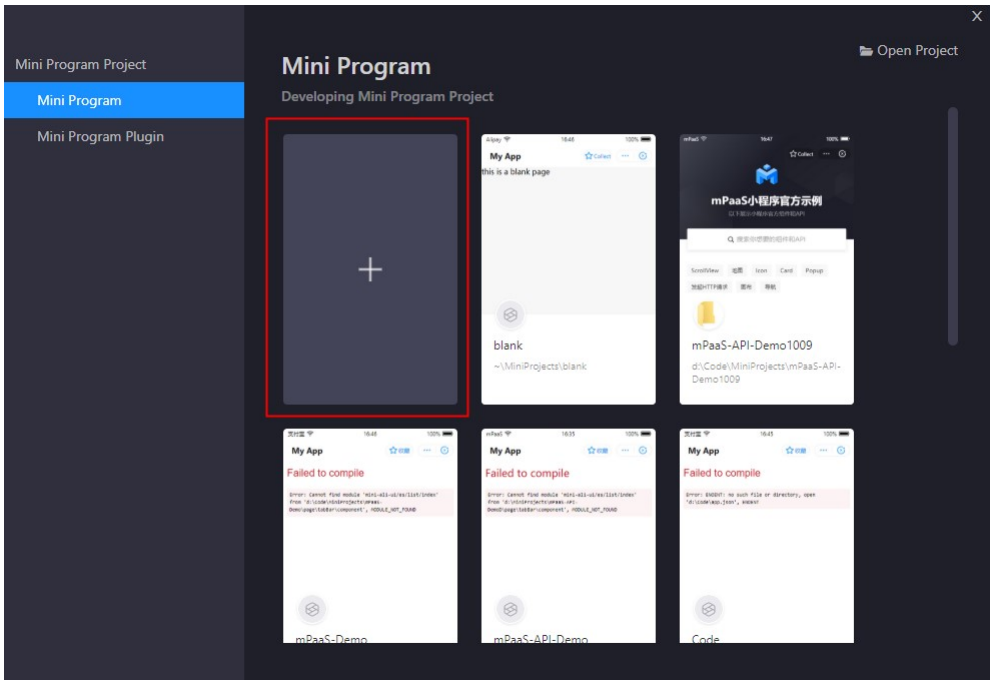
1. Download IDE
2. Create a Mini Program
3. Download configuration file
4. Create login environment and login to IDE
5. Select the associated Mini Program
6. Edit code
7. Upload Mini Program
8. Release Mini Program

Download IDE

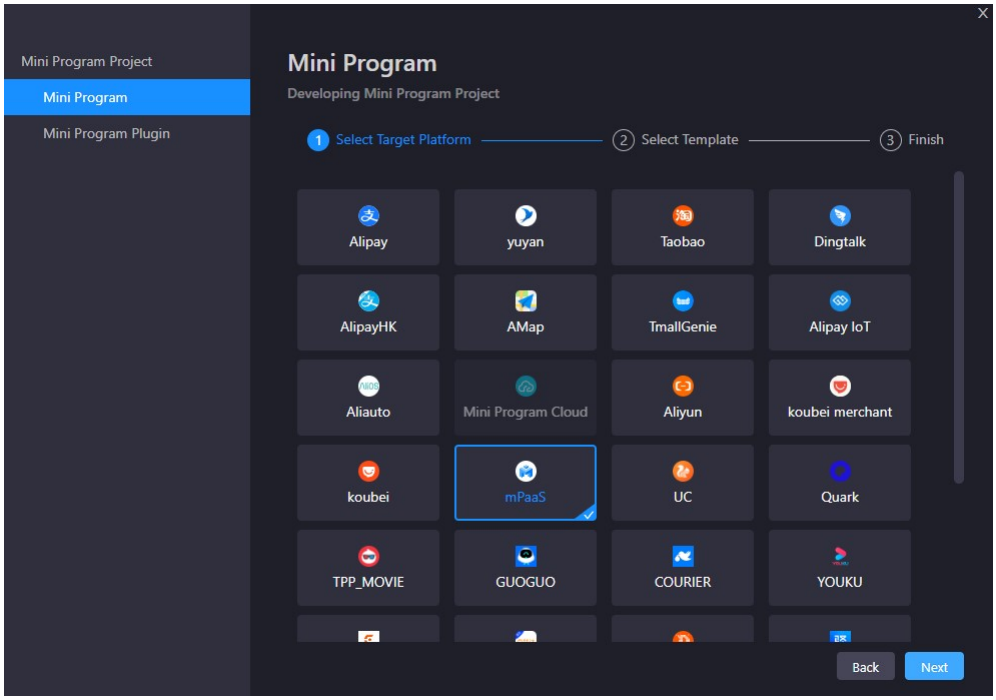
Go to the Mini Program download center and download the required version of the development tool (IDE).

Create a Mini Program

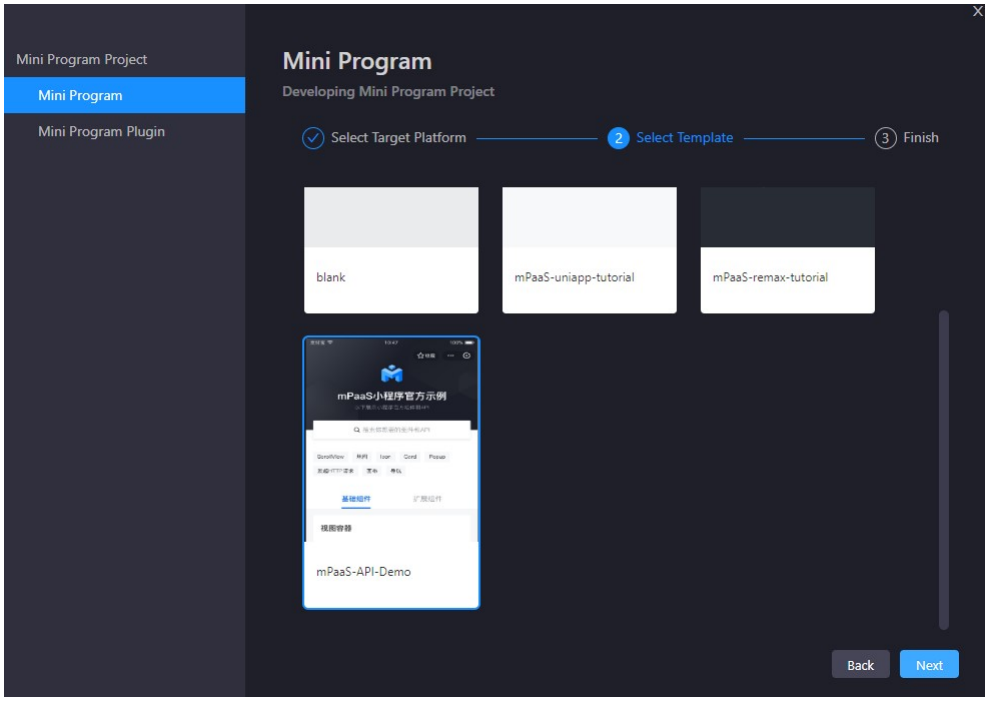
1. After downloading and installing the Mini Program IDE, open the IDE, choose **Mini Program** on the left side menu, and then click the plus icon on the right to add a new project.



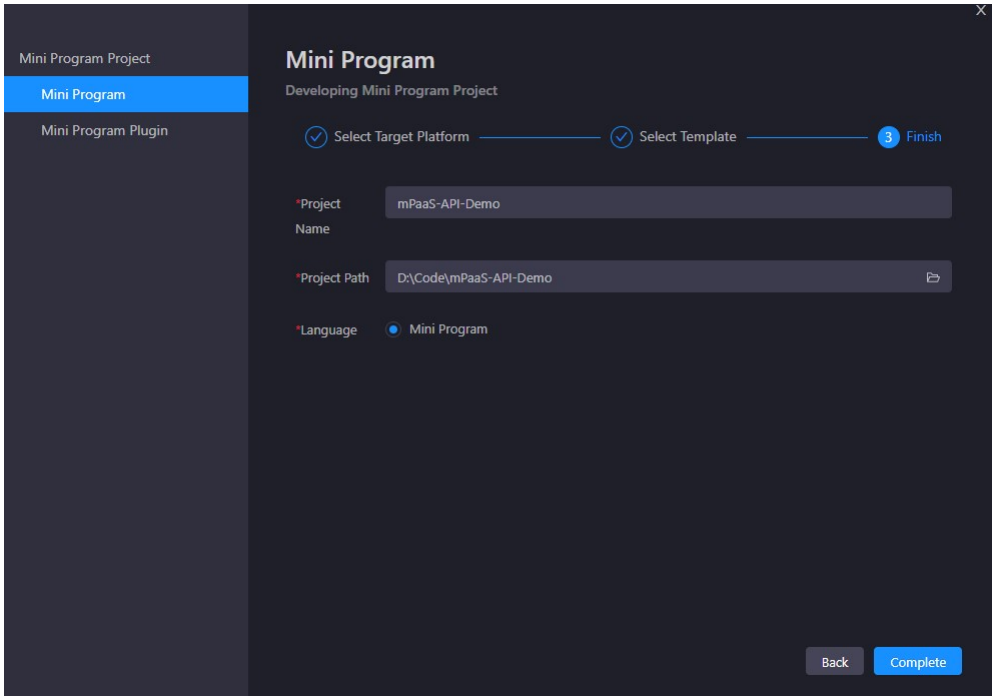
2. In the **Select Target Platform** step, choose **mPaaS** and click **Next**.



3. In the **Select Template** step, choose the mPaaS sample template and click **Next**.



4. Enter the project name, specify the project path, and click **Complete** to complete the Mini Program project creation.



Download configuration file

Every time you create a new environment, you need to upload the IDE configuration file of the corresponding Mini Program downloaded from the console.

1. Go to mPaaS console > **Mini Program** > **Release Mini Program** > **Manage configuration**, enter the configuration file downloading page, click **Download configuration file** in **IDE configuration management**, and download the Mini Program IDE configuration file.
   Note: The IDE configuration file is different from the configuration file of the mPaaS application.
2. After clicking **Download configuration file**, the **Download configuration file** window will pop up. You need to enter a password in the **Dynamic password**, which is the login password used when logging in to the IDE later.

   Note: The default name of the downloaded configuration file is `config.json`.

Create login environment and login to IDE

Create login environment

1. In the Mini Program IDE, click **Login** in the upper right.
2. If the login environment has not been created, the Create environment window will pop up; if the login environment has been created, the login window will pop up.
   - If you are **creating a login environment for the first time**, enter the environment name in the current window and upload the **Mini Program IDE configuration file** (`config.json` file) downloaded from the mPaaS console.
   - If you have **created a login environment**, click the environment selection menu at the top of the window, and select **+ Create environment** at the bottom of the menu. Enter the environment name and upload the **Mini Program IDE configuration file** (`config.json` file) downloaded from the mPaaS console.
3. Click **OK** to create a new login environment.

Log in to the IDE

After successfully creating a login environment, in the login window, enter the account and password to log in.

- The account is the username for logging in to the Alibaba Cloud console.
- The password is the dynamic password set when downloading the IDE configuration file.

Select the associated Mini Program

After logging in to the IDE, at the top left of the interface, click **Select the associated Mini Program**, and select the Mini Program created in the console from the drop-down menu.

Edit code

After selecting the associated Mini Program, you can start editing the Mini Program code.

Upload Mini Program

After editing the code, click **Upload** in the upper right of the IDE interface to upload the Mini Program to the mPaaS console.

Release Mini Program

Go to the mPaaS console > **Mini Program** > **Release Mini Program** to release. For details, see Release Mini Program package.

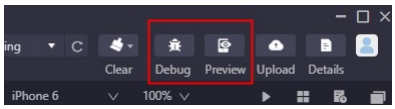
1.4.2. Advanced guide

1.4.2.1. Real-device preview and debugging

Mini program IDE supports real-device preview and debugging, you can preview the actual result of the current code or debug on the mobile phone client.

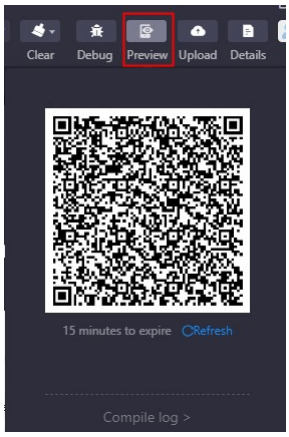
Procedure

1. Click **Preview** or **Debug** in the upper-right corner of the IDE.



   - The IDE will generate a `.zip` package of the current code and upload it to MDS.
   - MDS automatically create a release task, and then generate and return a QR code to IDE.
     Note: In the process of generating the QR code, it is possible that the generation fails because the whitelist is not set, and the QR code cannot be generated. This problem can be solved by setting a whitelist.

2. Use your mobile phone client to scan the QR code displayed in IDE.



   - After scanning the QR code, the MDS will release the Mini program package.
   - The QR code is valid for 5 minutes, and a refresh button is displayed after timeout.

3. When the mobile phone receives the Mini program package, you can start preview or debugging.

Access real-device preview and debugging

See the following documents to learn how to access real-device preview and debugging for Android and iOS Mini programs.

- Android
- iOS

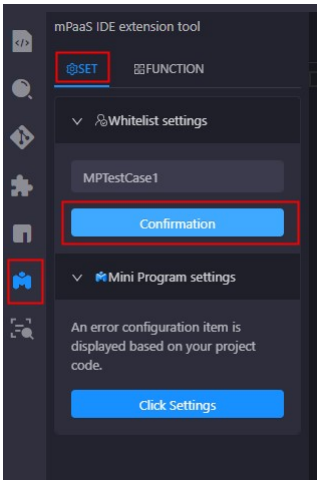
1.4.2.2. Extensions

Extension toolbox

The extended configuration of the mPaaS Mini Program is implemented in the IDE extension toolbox. Click the toolbox icon (



) on the left side of the interface to open the IDE extension toolbox.



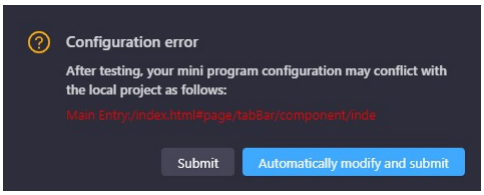
Settings

Whitelist settings

Click **Settings** > **Whitelist Settings** in the toolbox, enter the whitelist and confirm. This whitelist corresponds to the userId logged in to the App client. Only when the whitelist is entered correctly can the Mini Program package for preview and debugging be obtained from the corresponding user.

Mini Program settings

Click **Settings** > **Mini Program settings** in the toolbox to open the settings page of the Mini Program. The system will prompt for error options in the configuration items according to the configurations in your Mini Program code.

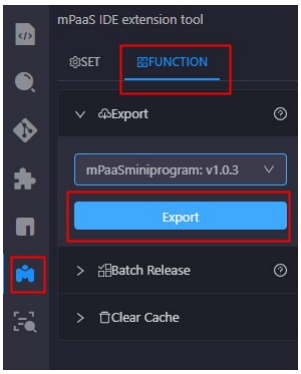


Features

Export

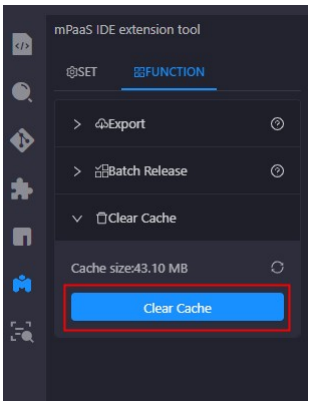
Click **Function** > **Export** in the toolbox, select the Mini Program version and click **Export**.

This function is to obtain the latest version of the official package of the Mini Program and download it to the local.



Clear cache

Click **Function** > **Clear Cache** in the toolbox to clear the cache files generated by the mPaaS Mini Program.

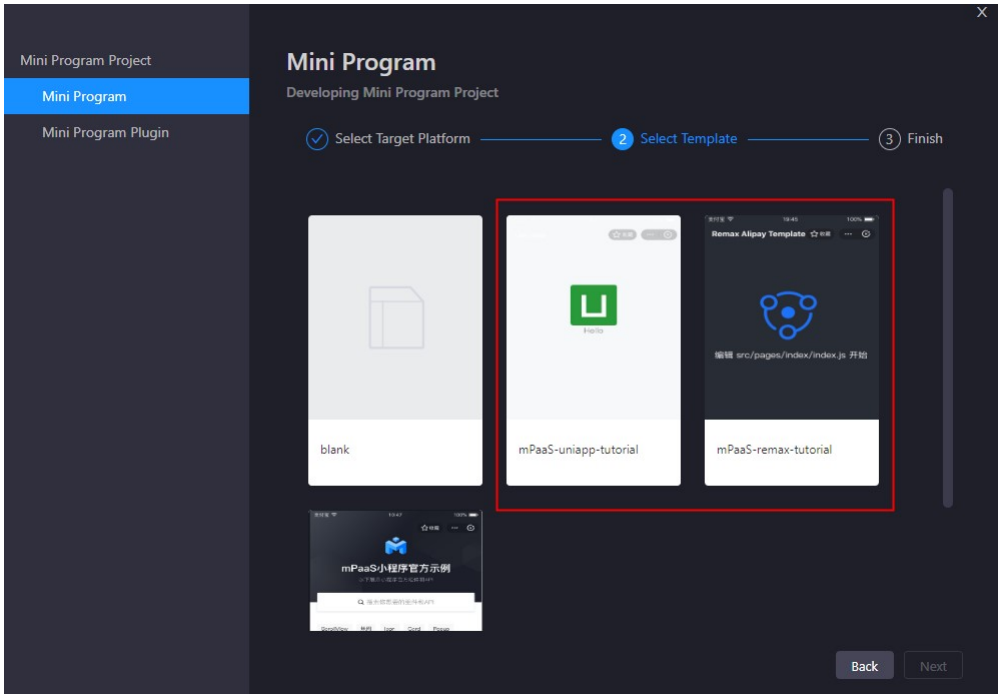


Multi-terminal development

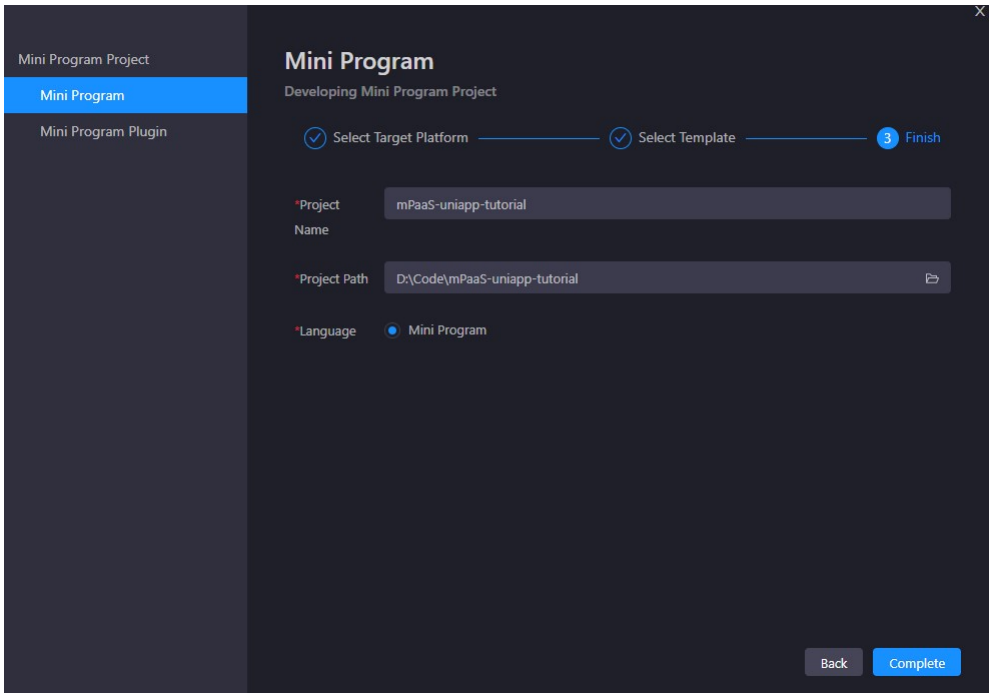
The Mini Program developed in the mPaaS Mini Program IDE can not only be put into the App developed using the mPaaS framework, but also can be used to evoke the WeChat Mini Program IDE for joint debugging through the mPaaS Mini Program IDE, or quickly build into an HTML5 application and realize real-device preview.

WeChat Mini Program

1. Open the Mini Program IDE, in the template selection, use the **multi-terminal development template (uni-app)** or **multi-terminal development template (remax)** to create a multi-terminal development Mini Program project. If you have created a multi-terminal development Mini Program project, you can also directly open it.



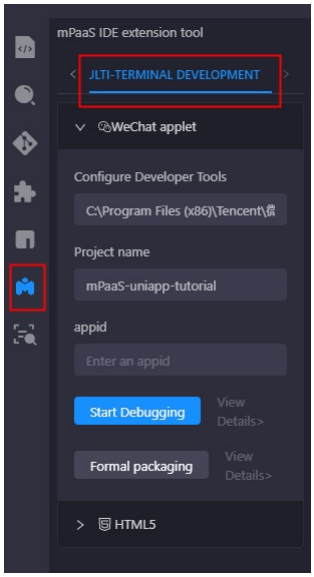
2. Enter the project name and project path, then click **Complete**.



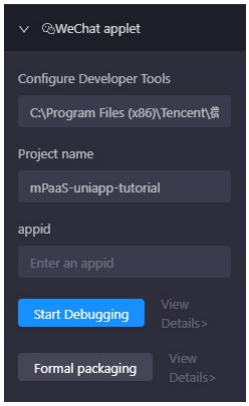
3. Download and install the WeChat Mini Program IDE. Log in and open the service port.
4. Click the mPaaS toolbox (



) on the left side of the IDE interface, and select the multi-terminal development tab on the right side.



5. Select **WeChat Mini Program**. In the configuration items below, you need to enter the following items:
   - **Configure Developer Tools**: The path of WeChat developer tools installed on this device.
   - **Project name**: The name of the Mini Program project displayed in the WeChat developer tool.
   - **appid**: The ID of the project created in the WeChat developer tool to be associated.



6. Click **Start Debugging**, then WeChat Mini Program developer tool will be aroused, and the real-time preview effect of the current Mini Program will be displayed in the developer tool.



7. Real-device preview.

   Click **Preview** in the upper right of the WeChat Mini Program developer Tool interface, and use the mobile WeChat client to scan the generated QR code to preview the Mini Program in WeChat on the real device.

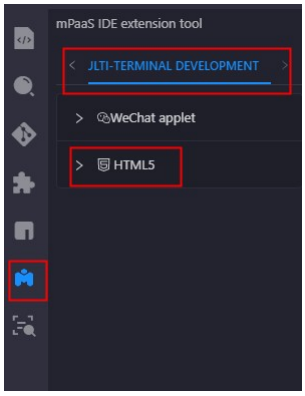
HTML5

1. Start the Mini Program IDE and select a project that you want to develop multi-terminal.

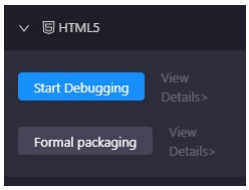
2. Click the mPaaS toolbox (



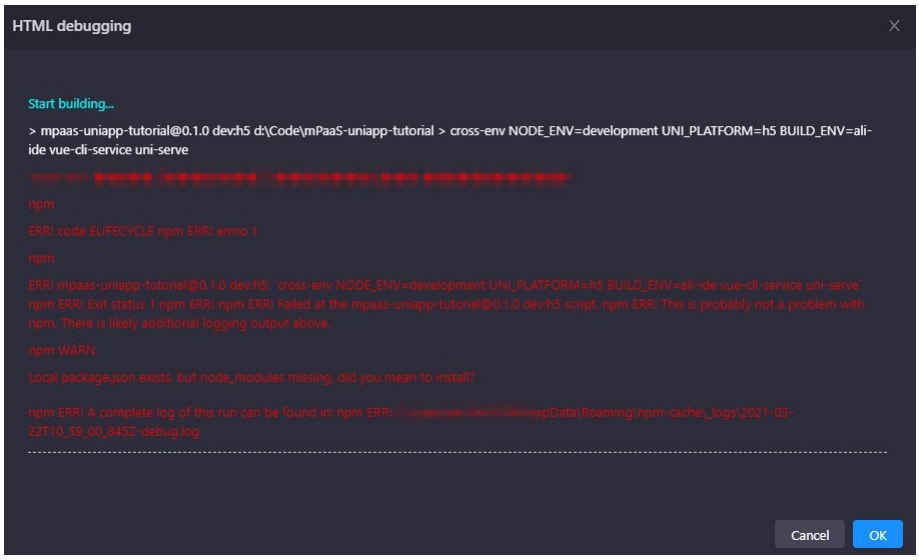
) on the left side of the IDE interface, and select the **Multi-terminal development** tab on the right side.



3. Select **HTML5** to expand, no additional configuration is required below.



4. Click **Start debugging**, the IDE will pop up the **HTML debugging** window and start building, and a QR code for real device scanning will be generated.



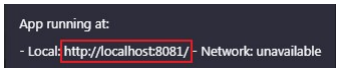
5. Preview.

   You can choose to use your mobile phone for real-device preview, or use a computer browser to preview.

   - **Real-device preview**: Open an App that supports HTML5 on the mobile phone and scan the generated QR code to preview the Mini Program in the mobile App. At this time, you can also choose to open it in the browser and preview the Mini Program through the mobile browser.

     Important: When using a mobile phone for HTML5 preview, make sure that the mobile phone used for previewing is in the same network as the computer.

   - **Computer browser preview**: You can copy the URL at Local (as shown in the image below) and paste it into the browser of your computer to preview the Mini Program.

1.4.2.3. Unregister custom event

In the process of developing mPaaS Mini Program, if the existing Mini Program APIs or events cannot meet the development needs, you can also extend them by yourself; when these custom APIs or events are not needed, you can also unregister them.

Mini Program calls native custom API

The original operation steps are as follows:

1. Customize the APIs and registers them in client.

See Custom JSAPI to register your custom API.

2. Mini program calls the API.

```
const call = my.call('tinyToNative', {
  param1: 'p1aaa',
  param2: 'p2bbb'
}, (result) => {
  console.log(result);
  my.showToast({
    type: 'none',
    content: result.message,
    duration: 3000,
  });
})
```

The method to unregister is as follows:

```
//Unregister
call.remove();
call = undefined;
```

Native App sends custom events to mini program

The original operation steps are as follows:

1. Register the event in mini program:

```
const on = my.on('www',()=>{
      my.alert({
        title: '1212',
        content: '123',
        buttonText: '123123',
        success: () => {
        },
        fail: () => {
        },
        complete: () => {
        }
      });
})
```

2. Client sends events.

   Get the `viewController` where the current mini program page is located, and call the `callHandler` method to send events.

```
[self callHandler:@"nativeToTiny" data:@{@"key":@"value"} responseCallback:^(id responseData) {
}];
```

The method to unregister is as follows:

```
on.remove();
on = undefined;
```

Parameter description:

Parameter	Description
handlerName	The name of the event monitored by the mini program.
data	The parameter passed by the client to the mini program.
callback	After the execution in mini program, call back processes the block.

1.5. OpenAPI in the console

1.5.1. Overview and Preparation

Mini Program developing operations from creating a Mini Program to releasing one can be implemented by calling openAPI, which enables the interaction between the user server side and the mPaaS server side.

Overview

Mini Program developing operations from creating a Mini Program to releasing one can be implemented by calling openAPI, which enables the interaction between the user server side and the mPaaS server side.

Rate limit description

To prevent OpenAPI from being called too frequently and thus affecting the operation of the App, a rate limiting mechanism is adopted for calling OpenAPI. The details are as follows:

- mcube rate limiting runs on a single server, and the limiting dimensions are appId+workspaceId.
- Currently mcube provides two servers for accepting OpenAPI requests, which are then forwarded by load balancer.
- On a single server, the limit for Mini Program uploading interface is 10 times every minute, that is, uploading interface can be called once every 6 seconds; the limit for other interfaces is 600 times every minute, that is, the interfaces can be called once every 0.1 second.

Preparation

Before using OpenAPI, you need to obtain AccessKey, App ID and Workspace ID, configure Maven dependencies and configure file uploading.

Obtain AccessKey

AccessKey includes **AccessKey ID** and **AccessKey Secret**. Click here for obtaining method.

- **AccessKey ID**: used to identify users.
- **AccessKey Secret**: used for user authentication. MUST be kept safe.

Obtain App ID and Workspace ID

1. Log in to mPaaS console, and enter the App.
2. In **Overview** page, click **Code configurations** (choose Android or iOS based on your needs)> **Download configuration file** > **Download now**. You can view App ID and Workspace ID in the **Code configurations** panel.

Configure Maven dependencies

Before using OpenAPI, you need to complete the following Maven dependency configurations.

```
<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-mpaas</artifactId>
  <version>1.1.1</version>
</dependency>

<dependency>
  <groupId>com.aliyun</groupId>
  <artifactId>aliyun-java-sdk-core</artifactId>
  <optional>true</optional>
  <version>[4.3.2,5.0.0)</version>
</dependency>
```

Code sample

The following shows the code sample of using Client in Maven:

```
        private static final String REGION_ID = "cn-hangzhou"; //Region ID, the default is cn-hangzhou
        private static final String ACCESS_KEY_ID = "******************"; //AccessKey ID of RAM account
        private static final String ACCESS_SECRET = "******************"; //AccessKey Secret of RAM account
        private static final String PRODUCT = "mpaas"; //Product name
        private static final String END_POINT = "mpaas.cn-hangzhou.aliyuncs.com"; //The endpoint that is called
        DefaultProfile.addEndpoint(REGION_ID, PRODUCT, END_POINT);
        DefaultProfile profile = DefaultProfile.getProfile(REGION_ID, ACCESS_KEY_ID, ACCESS_SECRET);
        IAcsClient iAcsClient = new DefaultAcsClient(profile);
        QueryMcubeVhostRequest queryMcubeVhostRequest = new QueryMcubeVhostRequest();
        queryMcubeVhostRequest.setAppId(APP_ID);
        queryMcubeVhostRequest.setWorkspaceId(WORKSPACE_ID);
        queryMcubeVhostRequest.setTenantId(TENANT_ID);
        QueryMcubeVhostResponse acsResponse = null;
        try {
            acsResponse = iAcsClient.getAcsResponse(queryMcubeVhostRequest);
            System.out.println(acsResponse.getResultCode());
            System.out.println(acsResponse.getQueryVhostResult());
        } catch (ClientException e) {
            e.printStackTrace();
        }
```

Configure file uploading

Since file streaming is not allowed in all APIs, to upload a file, you need to upload it to OSS first by calling the upload tool class, and then send the returned OSS address as a parameter to the specified API.

You can download the file upload tool class OssPostObject.java.zip.

Code sample

The following shows the code sample of file uploading:

```
        GetMcubeFileTokenRequest getMcubeFileTokenRequest = new GetMcubeFileTokenRequest();
        getMcubeFileTokenRequest.setAppId(APP_ID);
        getMcubeFileTokenRequest.setOnexFlag(true);
        getMcubeFileTokenRequest.setTenantId(TENANT_ID);
        getMcubeFileTokenRequest.setWorkspaceId(WORKSPACE_ID);
        GetMcubeFileTokenResponse acsResponse = iAcsClient.getAcsResponse(getMcubeFileTokenRequest);
        System.out.println(JSON.toJSONString(acsResponse));

        GetMcubeFileTokenResponse.GetFileTokenResult.FileToken fileToken = acsResponse.getGetFileTokenResult().getFileToken();
        OssPostObject ossPostObject = new OssPostObject();
        ossPostObject.setKey(fileToken.getDir());
        ossPostObject.setHost(fileToken.getHost());
        ossPostObject.setOssAccessId(fileToken.getAccessid());
        ossPostObject.setPolicy(fileToken.getPolicy());
        ossPostObject.setSignature(fileToken.getSignature());
        ossPostObject.setFilePath("your/local/file/path");
        String s = ossPostObject.postObject();
```

Refer to Obtain upload file token for descriptions about GetMcubeFileTokenRequest.

1.5.2. Interface description

Obtain upload file token

Request - GetMcubeFileTokenRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
onexFlag	Boolean	The fixed value is true.

Response- GetMcubeFileTokenResponse

```
{
    "getFileTokenResult":{
        "fileToken":{
            "accessid":"LTAI7z7XPfKU****",
            "dir":"mds/tempFileForOnex/ONEXE9B092D/test/PUQYHL/8b574cb7-3596-403f-a0e9-208660fc2081/",
            "expire":"1584327372",
            "host":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com",
            "policy":"QwM2YtYTBlOS0yMDg2NjBmYzIwODEvIl1dfQ==",
            "signature":"kisfP5YhbPtmES8+w="
        },
        "resultMsg":"",
        "success":true
    },
    "requestId":"8BAA3288-662E-422C-9960-2EEBFC08369F",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
getFileTokenResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Query virtual domain

Request - QueryMcubeVhostRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.

Response - QueryMcubeVhostResponse

```
{
    "queryVhostResult":{
        "data":"test.com",
        "resultMsg":"",
        "success":true
    },
    "requestId":"637D5BE0-0111-4C53-BCEE-473CFFA0DBAD",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
queryVhostResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
data	String	The virtual domain name queried.
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Create virtual domain

Request - CreateMcubeVhostRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
vhost	Boolean	Virtual domain name.

Response - CreateMcubeVhostResponse


```
{
    "createVhostResult":{
        "data":"success",
        "resultMsg":"",
        "success":true
    },
    "requestId":"F9C681F2-6377-488D-865B-1144E0CE69D2",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.

Response name	Type	Description
createVhostResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Query if key file exists

Request - ExistMcubeRsaKeyRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.

Response - ExistMcubeRsaKeyResponse

```
{
    "checkRsaKeyResult":{
        "data":"fail",
        "resultMsg":"",
        "success":false
    },
    "requestId":"8F76783A-8070-4182-895D-14E5D66F8BA3",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
checkRsaKeyResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
data	String	Query result: fail indicates the key does not exist, and success indicates the key exists.
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Upload key file

Request - UploadMcubeRsaKeyRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
onexFlag	Boolean	The fixed value is true.
fileUrl	String	The save address of the key file in OSS.

Response - UploadMcubeRsaKeyResponse

```
{
    "requestId":"519E35CF-CC60-4890-8C8E-89A98CEA6BB0",
    "resultCode":"OK",
    "uploadRsaResult":{
        "data":"processed successfully",
        "resultMsg":"",
        "success":true
    }
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
uploadRsaResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Obtain Mini Program list

Request - ListMcubeMiniAppsRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.

Response - ListMcubeMiniAppsResponse

```
{
    "listMiniResult":{
        "miniProgramList":[
            {
                "appCode":"ONEXE9B092D052019-test",
                "gmtCreate":"2019-12-05T21:30:23+08:00",
                "gmtModified":"2019-12-05T21:30:23+08:00",
                "h5Id":"1111111111111111",
                "h5Name":"11111111111111111"
            },
            {
                "appCode":"ONEXE9B092D052019-test",
                "gmtCreate":"2020-03-13T20:04:53+08:00",
                "gmtModified":"2020-03-13T20:04:53+08:00",
                "h5Id":"2222222222222222",
                "h5Name":"test1"
            }
        ],
        "resultMsg":"",
        "success":true
    },
    "requestId":"BE9BF836-72E5-4ACE-A48D-389BA27F8D95",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
listMiniResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
appCode	String	appId+" - "+workspaceId
gmtCreate	Date	Time of creation.
gmtModified	Date	Time of update.
h5Id	String	ID of Mini Program App.
h5Name	String	Name of Mini Program App.
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Create Mini Program

Request - CreateMcubeMiniAppRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
h5Id	String	ID of HTML5 App. 16 digits.

Parameter	Type	Description
h5Name	String	Name of HTML5 App.

Response - CreateMcubeMiniAppResponse

```
{
    "createMiniResult":{
        "data":"processed successfully",
        "resultMsg":"",
        "success":true
    },
    "requestId":"8A593C1D-9688-4409-BB01-8DB8AD897DD4",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
createMiniResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Delete Mini Program

Request - DeleteMcubeMiniAppRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
h5Id	String	ID of Mini Program App.

Response - DeleteMcubeMiniAppResponse

```
{
    "deleteMiniResult":{
        "data":"processed successfully",
        "resultMsg":"",
        "success":true
    },
    "requestId":"3DA95CA4-2579-4A2E-9A44-0A4215AEE431",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.

Response name	Type	Description
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
deleteMiniResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Upload Mini Program resource package

Request - UploadMcubeMiniPackageRequest

Name	Type	Required	Description
appId	String	-	App ID.
workspaceId	String	-	Workspace ID.
tenantId	String	-	Tenant ID.
h5Id	String	Yes	ID of HTML5 App.
h5Name	String	Yes	Name of HTML5 App.
h5Version	String	Yes	Version of Mini Program package. Must be unique in a Mini Program App.
mainUrl	String	Yes	Main URL of Mini Program.
vhost	String	Yes	Virtual domain name of the Mini Program App. The format is Mini Program ID + the virtual domain name previously set.
extendInfo	String	No	Extended field in json format.
autoInstall	Long	Yes	Download time: • 0-Wi-Fi only (Without Wi-Fi connection, download starts only when users use the App) • 1-All networks (uses cellular data. Do not choose this mode unless in special situations)
resourceType	Long	Yes	Resource type. The fixed value is 4.
installType	Long	Yes	Install time: • 0-Not preload (install only when the user enters the Mini Program page) • 1-Preload (automatically install after downloading the Mini Program)
platform	String	Yes	Platform. Includes all, Android and iOS.

Name	Type	Required	Description
clientVersionMin	String	Yes	Minimum client version. Minimum version is required when platform is chosen. The format is `aaa;bbb`. `aaa` indicates iOS client version, and `bbb` indicates Android client version. The semicolon `;` cannot be omitted. If Android is chosen as the platform, then the value is `;bbb`.
clientVersionMax	String	No	Maximum client version. Can be empty. If the value of platform is all, then this value must appear in pairs. That is, iOS and Android version must both be included, or neither of them is included.
resourceFileUrl	String	Yes	Address of Mini Program resource package.
iconFileUrl	String	Yes	OSS address of the Mini Program icon.
enableTabBar	Long	Yes	Display bottom navigation bar: • 0-No • 1-Yes
enableOptionMenu	String	Yes	Display function menu in the upper right corner: • 0-No • 1-Yes
enableKeepAlive	String	Yes	Whether to keep alive in the background.
packageType	Long	No	Mini Program package type: • 1-Function package • 2-Plugin package • 3-Real device test package • 4-Real device preview package If left empty, the value uses 1 as the default value. Currently, the packages are categorized into two categories, official package and test package. 1 and 2 are official packages, and 3 and 4 are test packages.
userId	String	No	If packageType is not 1 or 2, ID of the uploading user is required for uploading and releasing resource package.
uuid	String	No	When packageType is 3 (real device test package), the release and remote debugging of real-device test package can be realized by specifying the value of uuid.

Response - UploadMcubeMiniPackageResponse

packageType ≠ 3 or 4

When packageType is not 3 or 4, the response to a resource package uploading request is as follows:

```
{
    "requestId":"768E2C47-130B-4947-A0BD-2DE81C9090BE",
    "resultCode":"OK",
    "uploadMiniPackageResult":{
        "resultMsg":"",
        "returnPackageResult":{
            "packageId":"3209"
        },
        "success":true
    }
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
uploadMiniPackageResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
packageId	String	Primary key ID of created resource package.

packageType = 3 or 4

When packageType is not 3 or 4, a release task is created automatically, and the response to a resource package uploading request is as follows:

```
{
    "requestId":"768E2C47-130B-4947-A0BD-2DE81C9090BE",
    "resultCode":"OK",
    "uploadMiniPackageResult":{
        "resultMsg":"",
        "returnPackageResult":{
            "packageId":"3209",
              "debugUrl": "mpaas://platformapi/startapp?appId=20001101&token=ide_qr&scheme=mpaas%3A%2F%2Fplatformapi%2Fstartapp%3FappId%3D99702121500
00000&nbsv%3D1.0.1.7&nbsource%3Ddebug&nbprefer%3DYES&nbsn%3DDEBUG&nbof",
            "userId":"user_id"
        },
        "success":true
    }
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
uploadMiniPackageResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
debugUrl	String	Schema for code scanning.
userId	String	ID of the uploading user.

Obtain Mini Program resource package list

Request - ListMcubeMiniPackagesRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
h5Id	String	ID of Mini Program App.
packageTypes	String	Package type, separate with comma (,).

Response - ListMcubeMiniPackagesResponse


```
{
    "listMiniPackageResult":{
        "miniPackageList":[
            {
                "appCode":"ONEXE9B092D052019-test",
                "autoInstall":0,
                "clientVersionMax":"",
                "clientVersionMin":"1.1;1.1",
                "downloadUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.1_all/nebula/1111111111111111_1.1.1.1.amr",
                "extendInfo":"",
                "extraData":"{"enableKeepAlive":"1","enableOptionMenu":"1","enableTabBar":"0","iconUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.1_all/icon/1111111111111111_1.1.1.1_2.jpg","resourceType":"4"}",
                "fallbackBaseUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.1_all/nebula/fallback/;https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.1_all/nebula/fallback/",
                "gmtCreate":"2019-12-05T21:31:03+08:00",
                "gmtModified":"2019-12-17T14:35:50+08:00",
                "h5Id":"1111111111111111",
                "h5Name":"11111111111111111",
                "h5Version":"1.1.1.1",
                "id":3127,
                "installType":0,
                "mainUrl":"",
                "memo":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.1_all/nebula/nebula_json/h5_json.json",
                "packageType":1,
                "platform":"all",
                "publishPeriod":5,
                "resourceType":4,
                "status":1
            }
        ],
        "resultMsg":"",
        "success":true
    },
    "requestId":"96151643-8F75-49B9-A5AA-6F57C6B647BD",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
listMiniPackageResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Name	Type	Description
appCode	String	appId+ "-" +workspaceId
autoInstall	Integer	Download time: • 0-Wi-Fi only (Without Wi-Fi connection, download starts only when users use the App) • 1-All networks (uses cellular data. Do not choose this mode unless in special situations)
clientVersionMax	String	Maximum client version. Can be empty. If the value of platform is all, then this value must appear in pairs. That is, iOS and Android version must both be included, or neither of them is included.
clientVersionMin	String	Minimum client version. Minimum version is required when platform is chosen. The format is `aaa;bbb`. `aaa` indicates iOS client version, and `bbb` indicates Android client version. The semicolon `;` cannot be omitted. If Android is chosen as the platform, then the value is `;bbb`.
creator	String	Creator, currently not used.
debugUrl	String	Has no meaning in current response.
downloadUrl	String	Download address of Mini Program arm file.
extendInfo	String	Extended information sent when uploading.
extraData	String	Extended parameters, include the configurations and icon address set when uploading.
fallbackBaseUrl	String	Mini Program fallback address. Separate with semicolon (`;`). The address before `;` is intranet address, and the address after `;` is internet address.
fileSize	String	File size.
gmtCreate	Date	Time of creation.
gmtModified	Date	Time of update.
h5Id	String	ID of Mini Program App.
h5Name	String	Name of Mini Program App.
h5Version	String	Version of current Mini Program package.
id	Long	Primary key.
installType	Integer	Install time: • 0-Not preload (install only when the user enters the Mini Program page) • 1-Preload (automatically install after downloading the Mini Program)
lazyLoad	Integer	Lazy loading. Currently the value is 0.
mainUrl	String	Main URL of Mini Program package.
md5	String	md 5 of Mini Program package file.

Name	Type	Description
memo	String	Download address of Mini Program package `h5.json` file.
metaId	Long	Has no meaning.
modifier	String	Modifier. Currently not used.
platform	String	Platform. Includes all, Android and iOS.
publishPeriod	Integer	Release status: • 1-Internal gray release • 2-External gray release • 3-Official release • 4-Rollback release • 5-Release task ends
releaseVersion	String	Release version.
resourceType	Integer	Resource type. The fixed value is 4.
status	Integer	Status.
packageType	Integer	Mini Program package type: • 1-Function package • 2-Plugin package • 3-Real device test package • 4-Real device preview package If left empty, the value uses 1 as the default value. Currently, the packages are categorized into two categories, official package and test package. 1 and 2 are official packages, and 3 and 4 are test packages.

Obtain Mini Program resource package info by ID

Request - QueryMcubeMiniPackageRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
id	String	Primary key ID of the resource package.

Response - QueryMcubeMiniPackageResponse

```
{
    "queryMiniPackageResult":{
        "miniPackageInfo":{
            "appCode":"ONEXE9B092D052019-test",
            "autoInstall":0,
            "clientVersionMax":"10;10",
            "clientVersionMin":"1;1",
            "downloadUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.2_all/nebula/1111111111111111_1.1.1.2.amr",
            "extendInfo":"{"key":"value"}",
            "extraData":"{"enableKeepAlive":"1","enableOptionMenu":"1","enableTabBar":"1","iconUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.2_all/icon/1111111111111111_1.1.1.2_2.jpg","resourceType":"4"}",
            "fallbackBaseUrl":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.2_all/nebula/fallback/;https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.2_all/nebula/fallback/",
            "gmtCreate":"2020-03-16T10:55:00+08:00",
            "gmtModified":"2020-03-16T11:22:10+08:00",
            "h5Id":"1111111111111111",
            "h5Name":"11111111111111111",
            "h5Version":"1.1.1.2",
            "id":3209,
            "installType":1,
            "mainUrl":"index",
            "memo":"https://mcube-test.oss-cn-hangzhou.aliyuncs.com/ONEXE9B092D052019-test/1111111111111111/1.1.1.2_all/nebula/nebula_json/h5_json.json",
            "packageType":1,
            "platform":"all",
            "publishPeriod":3,
            "resourceType":4,
            "status":1
        },
        "resultMsg":"",
        "success":true
    },
    "requestId":"1DAFB8E6-F812-48CF-B7F7-1F5FEF57908F",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
queryMiniPackageResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
appCode	String	appId+ "-" +workspaceId
autoInstall	Integer	Download time: • 0-Wi-Fi only (Without Wi-Fi connection, download starts only when users use the App) • 1-All networks (uses cellular data. Do not choose this mode unless in special situations)
clientVersionMax	String	Maximum client version. Can be empty. If the value of platform is all, then this value must appear in pairs. That is, iOS and Android version must both be included, or neither of them is included.

Name	Type	Description
clientVersionMin	String	Minimum client version. Minimum version is required when platform is chosen. The format is `aaa;bbb`. `aaa` indicates iOS client version, and `bbb` indicates Android client version. The semicolon `;` cannot be omitted. If Android is chosen as the platform, then the value is `;bbb`.
creator	String	Creator, currently not used.
debugUrl	String	Has no meaning in current response.
downloadUrl	String	Download address of Mini Program arm file.
extendInfo	String	Extended information sent when uploading.
extraData	String	Extended parameters, include the configurations and icon address set when uploading.
fallbackBaseUrl	String	Mini Program fallback address. Separate with semicolon (`;`). The address before `;` is intranet address, and the address after `;` is internet address.
fileSize	String	File size.
gmtCreate	Date	Time of creation.
gmtModified	Date	Time of update.
h5Id	String	ID of Mini Program App.
h5Name	String	Name of Mini Program App.
h5Version	String	Version of current Mini Program package.
id	Long	Primary key.
installType	Integer	Install time: • 0-Not preload (install only when the user enters the Mini Program page) • 1-Preload (automatically install after downloading the Mini Program)
lazyLoad	Integer	Lazy loading. Currently the value is 0.
mainUrl	String	Main URL of Mini Program package.
md5	String	md 5 of Mini Program package file.
memo	String	Download address of Mini Program package `h5.json` file.
metaId	Long	Has no meaning.
modifier	String	Modifier. Currently not used.
platform	String	Platform. Includes all, Android and iOS.

Name	Type	Description
publishPeriod	Integer	Release status: • 1-Internal gray release • 2-External gray release • 3-Official release • 4-Rollback release • 5-Release task ends
releaseVersion	String	Release version.
resourceType	Integer	Resource type. The fixed value is 4.
status	Integer	Status.
packageType	Integer	Mini Program package type: • 1-Function package • 2-Plugin package • 3-Real device test package • 4-Real device preview package If left empty, the value uses 1 as the default value. Currently, the packages are categorized into two categories, official package and test package. 1 and 2 are official packages, and 3 and 4 are test packages.

Create Mini Program release task

Request - CreateMcubeMiniTaskRequest

Name	Type	Required	Description
appId	String	-	App ID.
workspaceId	String	-	App ID.
tenantId	String	-	Tenant ID.
publishType	Integer	Yes	Release type: • 2-Gray release • 3-Official release
publishMode	Integer	Yes	Release mode: • 1-Whitelist • 2-Time window If publishType is 3, this field should be 0.
memo	String	No	Release note.
id	Long	Yes	Only 0 is allowed. The ID indicates creation, and cannot be modified.
greyEndtimeData	String	No. Required when publishMode is 2.	End time of time window grey release. Format: `YYYY-MM-dd HH:mm:ss`. The time must be greater than current time and less than 7 days from current time.
greyNum	Integer	No. Required when publishMode is 2.	Number of users in time window grey release.

Name	Type	Required	Description
whitelistIds	String	No. Required when publishMode is 1.	Primary key ID of whitelist. Separate with comma `,`.
packageId	Long	Yes	Primary key ID of the resource package.
greyConfigInfo	String	No	Advanced rule, json string. See the table below for meanings.

Advanced rule

Example:

```
{"ruleElement":"city","operation":1,"value":"Shanghai,Beijing,Tianjin"},{"ruleElement":"mobileModel","operation":2,"value":"REDMI NOTE 3,VIVO X5M"},{
"ruleElement":"osVersion","operation":3,"value2":"9.2.1","value1":"9.2.1","value":"9.2.1-9.2.1"}
```

Description:

Name	Type	Description
ruleElement	String	Rule type: • city-City • mobileModel-Mobile phone model • netType-Network • osVersion-Device OS version
value	String	Rule value. Separate with comma (`,`). When operation is 3 or 4, the value is in `aa-bb` format, in which `aa` is the smaller value, and `bb` is the greater value.
operation	Integer	Operation: • 1-Include • 2-Exclude • 3-Within range • 4-Out of range If ruleElement is city, mobileModel and netType, operation value can only be: • 1-Include • 2-Exclude If ruleElement is osVersion, the value of operation can be one of the four value.

Response - CreateMcubeMiniTaskResponse

```
{
    "createMiniTaskResult":{
        "miniTaskId":"5244",
        "resultMsg":"",
        "success":true
    },
    "requestId":"53103033-5018-4090-9FAC-1E1B556DA14F",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
createMiniTaskResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
miniTaskId	String	ID of created task.

Obtain Mini Program release task list

Request - ListMcubeMiniTasksRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
id	String	ID of mini program package corresponding to the release task.

Response - ListMcubeMiniTasksResponse


```
{
    "listMiniTaskResult":{
        "miniTaskList":[
            {
                "appCode":"10EE034211053-default",
                "bizType":"nebula",
                "bundles":[ ],
                "creator":"",
                "gmtCreate":"2019-04-24 17:43:54",
                "gmtModified":"2019-04-24 17:43:54",
                "gmtModifiedStr":"2019-04-24 17:43:54",
                "greyConfigInfo":"{"operator":"and","subRules":[{"operator":"contains","left":["Shanghai","Beijing","Tianjin"],"right":"city","defaul
tResult":false},{"operator":"excludes","left":["REDMI NOTE 3","VIVO X5M"],"right":"mobileModel","defaultResult":false},{"operator":"vLimitIn","exclus
ive":false,"left":{"lower":"9.2.1","upper":"9.2.1"},"right":"osVersion","defaultResult":false}],"defaultResult":false}",
                "greyEndtime":null,
                "greyEndtimeData":"",
                "greyNum":0,
                "id":212,
                "memo":"h5 package release",
                "modifier":"",
                "packageId":572,
                "percent":0,
                "platform":"iOS",
                "productId":"10EE034211053-default-1234567812345678",
                "productVersion":"1.1.1.4",
                "publishMode":1,
                "publishType":2,
                "releaseVersion":"20190424103618",
                "resIds":"572",
                "status":1,
                "syncResult":"",
                "taskName":"1234",
                "taskStatus":1,
                "taskType":0,
                "taskVersion":1556099034158,
                "upgradeNoticeNum":0,
                "upgradeProgress":"",
                "whitelistIds":"931"
            }
        ],
        "resultMsg":"",
        "success":true
    },
    "requestId":"4A0C156E-0B8F-4007-9B3C-4EE267723361",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
listMiniTaskResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
appCode	String	appId+" - "+workspaceId
bizType	String	For Mini Program package, the value is nebula.
bundles	Array	Currently not used.
creator	String	Currently not used.
gmtCreate	Date	Time of creation.
gmtModified	Date	Time of update.
gmtModifiedStr	String	Update time string.
greyConfigInfo	String	Advanced rule string, different from that when uploading. See greyConfigInfo explanation for details.
greyEndtime	Date	End time of time window grey release.
greyEndtimeData	String	End time string of time window grey release.
greyNum	Integer	Number of users in time window grey release.
id	Long	Primary key ID of current release task.
memo	String	Release note.
modifier	String	Release note.
packageId	Long	ID of mini program package corresponding to current task.
percent	Integer	Grey percent. Currently the value is 0.
platform	String	Platform of current task. Includes all, Android and iOS.
productId	String	Product ID. The format is `appId + workspaceId + h5id` .
productVersion	String	ID of mini program resource package.

Name	Type	Description
publishMode	Integer	Release mode: • 0-Default • 1-Whitelist • 2-Time window
publishType	Integer	Release type: • 2-Gray release • 3-Official release
releaseVersion	String	Internal release version.
resIds	String	ID of corresponding Mini Program resource package.
status	Integer	Status: • 0-Invalid • 1-Valid
syncResult	String	Currently not used.
taskName	String	Task name, same as the Mini Program App name.
taskStatus	Integer	Task status: • 0-To be released • 1-Release in progress • 2-Finished • 3-Paused
taskType	Integer	Task type: • 0-Ordinary task. • 1-Rollback task.
taskVersion	Long	Task version, uses the time of task creation.
upgradeNoticeNum	Integer	Currently not used.
upgradeProgress	String	Currently not used.
whitelistIds	String	Primary key ID of whitelist. Separate with comma (`,`).

greyConfigInfo explanation

Name	Type	Description
operator	String	Rule relationship. `and` means all the rules in subRules must be met at the same time.
defaultResult	boolean	The default returned result.
subRules	List	Rule list.
operator	String	Rule name: • contains-Include • excludes-Exclude • vLimitIn-Within range • vLimitOut-Out of range

Name	Type	Description
left	List<String>/Object	• When operator value is `contains` or `excludes`, the value is a list of elements, and each element represents a rule value. • When operator value is `vLimitIn` or `vLimitOut`, it is an object, and lower represents the smaller value, and upper represents the greater value.
right	String	Rule type name.
defaultResult	Boolean	Default result.

Query release task by ID

Request - QueryMcubeMiniTaskRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
id	String	Primary key ID of the release task.

Response - QueryMcubeMiniTaskResponse


```
{
    "queryMiniTaskResult":{
        "miniTaskInfo":{
            "appCode":"ONEXE9B092D052019-test",
            "gmtCreate":"2020-03-16T11:22:10+08:00",
            "gmtModified":"2020-03-16T11:22:10+08:00",
            "greyConfigInfo":"",
            "greyEndtimeData":"",
            "greyNum":0,
            "id":5244,
            "memo":"memo",
            "packageId":3209,
            "platform":"all",
            "productVersion":"1.1.1.2",
            "publishMode":0,
            "publishType":3,
            "status":"1",
            "taskStatus":1,
            "whitelistIds":""
        },
        "resultMsg":"",
        "success":true
    },
    "requestId":"C02CC6F5-9CDB-42C0-8967-9F728A48C7F6",
    "resultCode":"OK"
}
```


Response description

The response is same as that of obtain Mini Program release task list.

Change task status

Request - ChangeMcubeMiniTaskStatusRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.

Name	Type	Description
tenantId	String	Tenant ID.
bizType	String	The value is nebula.
packageId	Long	ID of corresponding Mini Program resource package.
taskId	Long	Current release task ID.
taskStatus	Long	Status to change into: • 0-To be released. • 1-Release in progress. • 2-Finished. • 3-Paused.

Response - ChangeMcubeMiniTaskStatusResponse

```
{
    "changeMiniTaskStatusResult":{
        "data":"processed successfully",
        "resultMsg":"",
        "success":true
    },
    "requestId":"8F2A9BC1-3FDF-4578-A164-B305D6FB59B0",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
changeMiniTaskStatusResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Create whitelist

Request - CreateMcubeWhitelistRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
whiteListName	String	Whitelist name. 2 to 10 characters.

Name	Type	Description
whitelistType	String	Whitelist type: • normal-Ordinary whitelist • regular-Regular expression whitelist

Response - CreateMcubeWhitelistResponse

```
{
    "createWhitelistResult":{
        "resultMsg":"",
        "success":true,
        "whitelistId":"2733"
    },
    "requestId":"72224B8A-351D-4FFD-8F0E-C1D21F0C22C8",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
createWhitelistResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
whitelistId	String	ID of created whitelist.

Obtain whitelists

Request - ListMcubeWhitelistRequest

Parameter	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.

Response - ListMcubeWhitelistResponse

```
{
    "listWhitelistResult":{
        "resultMsg":"",
        "success":true,
        "whitelists":[
            {
                "appCode":"ONEXE9B092D052019-test",
                "gmtCreate":"2020-03-16T12:38:14+08:00",
                "gmtModified":"2020-03-16T12:38:14+08:00",
                "id":2733,
                "whiteListCount":0,
                "whiteListName":"testWhite",
                "whitelistType":"normal"
            },
            {
                "appCode":"ONEXE9B092D052019-test",
                "gmtCreate":"2019-12-05T21:46:15+08:00",
                "gmtModified":"2019-12-05T21:46:15+08:00",
                "id":2701,
                "whiteListCount":0,
                "whiteListName":"23",
                "whitelistType":"normal"
            }
        ]
    },
    "requestId":"C33D3132-5E33-4744-B146-9BDF36D99A5E",
    "resultCode":"OK"
}
```


Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
listWhitelistResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
appCode	String	appId+" - "+workspaceId
gmtCreate	Date	Time of creation.
gmtModified	Date	Time of update.
id	Long	Primary key ID of whitelist.
whiteListCount	Integer	User ID count in a whitelist.
whiteListName	String	Whitelist name.
whitelistType	String	Whitelist type: • normal-Ordinary whitelist • regular-Regular expression whitelist

Add whitelist content

Request - UpdateMcubeWhitelistRequest

Name	Type	Required	Description
appId	String	-	App ID.
workspaceId	String	-	Workspace ID.
tenantId	String	-	Tenant ID.
keyIds	String	No. Only one of keyIds and keyIdsFile is required.	Whitelist user. Separate multiple user IDs with comma (,).
ossUrl	String	No. Only one of keyIds and keyIdsFile is required.	Whitelist file address. One user ID is allowed in one line in the file, and the file size should be less than 5 MB.
onexFlag	Boolean	Yes	The fixed value is true.
id	String	Yes	Whitelist ID.

Response - UpdateMcubeWhitelistResponse

```
{
    "addWhitelistResult":{
        "addWhitelistInfo":{
            "failNum":0,
            "failUserIds":"",
            "successNum":2
        },
        "resultMsg":"",
        "success":true
    },
    "requestId":"A0462D16-824B-40E1-A000-66E0D2376B5B",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
addWhitelistResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
failNum	Integer	The number of whitelists failed to add.
failUserIds	String	The content of whitelists failed to add.
successNum	Integer	The number of whitelists successfully uploaded.

Delete whitelist

Request - DeleteMcubeWhitelistRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
id	Integer	Primary key ID of whitelist.

Response - DeleteMcubeWhitelistResponse

```
{
    "deleteWhitelistResult":{
        "data":"1",
        "resultMsg":"",
        "success":true
    },
    "requestId":"26B80D2D-D62C-428D-8124-D0DC4968350B",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
deleteWhitelistResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
data	String	The number of whitelists deleted.
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.

Whitelist creation used for Mini Program preview and debug in IDE

Request - CreateMcubeWhitelistForIdeRequest

Name	Type	Description
appId	String	App ID.
workspaceId	String	Workspace ID.
tenantId	String	Tenant ID.
userId	Integer	ID of user that needs to be used.
whitelistValue	String	Whitelist content.

Response - CreateMcubeWhitelistForIdeResponse

```
{
    "createWhitelistForIdeResult":{
        "resultMsg":"",
        "success":true,
        "whitelistId":"2734"
    },
    "requestId":"E59980F5-3AF1-4972-8830-F0A4848C4CA4",
    "resultCode":"OK"
}
```

Response description

Response name	Type	Description
requestId	String	Request ID.
resultCode	String	Normally, the code returned is OK. Other values indicate that the API request is abnormal.
createWhitelistForIdeResult	Object	The objects returned. See the table below for meanings.

The objects returned include the following fields:

Name	Type	Description
resultMsg	String	Returned value after query failure.
success	Boolean	Whether the query is successful.
whitelistId	String	ID of created whitelist.

1.6. Mini program base library

The Mini program capability requires the surrpot from client:

Relationship between the base library and the client

The Mini program capability requires the surrpot from client:

- The new functions of each version of base library run on specific versions.
- Some new functions of the high version base library are not compatible with low version clients.

See Compatible with base library for the method. You can check the version number of the current base library via `my.SDKVersion`.

Compatible with base library

Currently, the Mini program components and API capabilities are gradually improved and enriched, but clients in the old versions do not support these new capabilities, so developers are advised to complete the corresponding compatibility processing.

You can implement compatibility judgment through the interface `my.canIUse(String)`. For details, see Interface description.

Example of compatibility

Compatibility processing of new APIs

For new APIs, you can determine whether the current base library supports the API by the following codes.

```
if (my.getLocation) {
    my.getLocation();
} else {
// If you want users to experience your Mini program in the new version of client, you can use the following reminder.
  my.alert({
    title: 'Reminder',
    content: 'The current version is too low to use this feature. Please upgrade to the latest version.'
  });
}
```

Compatibility processing of API new parameters

For new API parameters, you can determine whether they are supported by the following method, and write your subsequent processing methods:

```
if (my.canIUse('getLocation.object.type')) {
    // Write your subsequent processing methods here
} else {
    console.log('This parameter is not supported in the current version')
}
```

Compatibility processing of API new return values

For new API return values, you can determine whether they are supported by the following method, and write your subsequent processing methods:

```
if (my.canIUse('getSystemInfo.return.storage')) {
    // Write your subsequent processing methods here
} else {
    console.log('The return value is not supported in the current version')
}
```

Compatibility processing of component's new properties

New properties of component cannot be implemented on earlier clients, but no error will be reported either. To downgrade the properties, see the following code:

```
Page({
    data: {
        canIUse: my.canIUse('button.open-type.share')
    }
})
```

Base library version

Base library version	Corresponding baseline version	
1.9.0	10.1.32	Download
1.14.1	10.1.60	Download

Description of base library integration

iOS

Base library integration is not needed in iOS.

Android

Due to the iteration of the baseline version, different baseline versions need to adopt different ways of integrating the base library. Before integrating the base library, please confirm your baseline version.

10.1.68.7 and later baseline versions

Set the Provider instance at startup. The code sample is as follows:

```
H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
```

Note: If the H5 public resource package is used in the client, you need to inherit the TinyAppCenterPresetProvider class and merge the related code of the public resource package into the instance of the inherited class.

10.1.60 series, 10.1.68.6 and earlier baseline versions

1. Implment `H5AppCenterPresetProvider` interface class. The code sample is as follows:

```
package com.mpaas.demo.nebula;

import com.alipay.mobile.nebula.appcenter.H5PresetInfo;
import com.alipay.mobile.nebula.appcenter.H5PresetPkg;
import com.alipay.mobile.nebula.provider.H5AppCenterPresetProvider;

import java.io.File;
import java.io.InputStream;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class H5AppCenterPresetProviderImpl implements H5AppCenterPresetProvider {
    private static final String TAG = "H5AppCenterPresetProviderImpl";

    // Set the Mini-program-specific resource bundle. This ID is fixed, do no set to other values.
    private static final String TINY_COMMON_APP = "66666692";

    // The asset directory of preset packages
    private final static String NEBULA_APPS_PRE_INSTALL = "nebulaPreset" + File.separator;

    // Collection of preset packages
    private static final Map<String, H5PresetInfo> NEBULA_LOCAL_PACKAGE_APP_IDS = new HashMap();

    static {

        H5PresetInfo h5PresetInfo2 = new H5PresetInfo();
        // File name of build-in directory
        h5PresetInfo2.appId = TINY_COMMON_APP;
        h5PresetInfo2.version = "1.0.0.0";
        h5PresetInfo2.downloadUrl = "";

        NEBULA_LOCAL_PACKAGE_APP_IDS.put(TINY_COMMON_APP, h5PresetInfo2);

    }

    @Override
    public Set<String> getCommonResourceAppList() {
        Set<String> appIdList = new HashSet<String>();
        appIdList.add(getTinyCommonApp());
        return appIdList;
    }

    @Override
    public H5PresetPkg getH5PresetPkg() {
        H5PresetPkg h5PresetPkg = new H5PresetPkg();
        h5PresetPkg.setPreSetInfo(NEBULA_LOCAL_PACKAGE_APP_IDS);
        h5PresetPkg.setPresetPath(NEBULA_APPS_PRE_INSTALL);
        return h5PresetPkg;
    }

    /**
     * Set the ID of the resource bundle that can be downgraded
     */
    @Override
    public Set<String> getEnableDegradeApp() {
        return null;
    }

    @Override
    public String getTinyCommonApp() {
        return TINY_COMMON_APP;
    }

    @Override
    public InputStream getPresetAppInfo() {
        return null;
    }

    @Override
    public InputStream getPresetAppInfoObject() {
        return null;
    }
}
```

2. Download the corresponding Mini Program base library according to the version integrated in the client, copy the base library to the `asset` catalog specified in the previous step and rename it. Based on the code sample in the previous step, the base library path of the Mini Program is `assets/nebulaPreset/66666692`.
3. Set the Provider instance at startup. The code sample is as follows:

```
H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(), new H5AppCenterPresetProviderImpl());
```

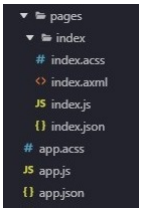
Note :If the H5 public resource package is used in the client, please merge the relevant code of the public resource package into the instance class of the `H5AppCenterPresetProvider`.

1.7. Frameworks

1.7.1. Overview

File structure

The MINI program can be divided into two layers, `app` and `page` layer. `app` is used to describe the entire program, while `page` is used to describe each page.



- `app` consists of the following three files, which must be placed in the root directory of the project.:

File	Required	Purpose
app.acss	No	MINI program global style sheet
app.js	Yes	MINI program logic
app.json	Yes	MINI program global settings

- `page` consists the following four types of files:

File type	Required	Purpose
acss	No	Page style sheet
axml	Yes	Page structure
js	Yes	Page logic
json	No	Page configuration

> **Note** For the convenience of developers, these four files must have the same path and file name.

All the codes written by the developer will eventually be packaged as a JavaScript script, which runs when the MINI program starts, and is destroyed when the MINI program ends.

Logical structure

The core of the MINI program is a responsive data binding system, which logically can be divided into view layer and logical layer. These two layers are always in sync, that is, if the data is modified at the logical layer, the view layer will display the updates accordingly.

For example:

```
<!-- View layer -->
<view> Hello {{name}}! </view>
<button onTap="changeName"> Click me! </button>
```

```
// Logical layer
var initialData = {
  name: 'taobao',
};

// Register a Page.
Page({
  data: initialData,
  changeName(e) {
    // sent data change to view
    this.setData({
      name: 'mPaaS',
    });
  },
});
```

In the above code, the framework automatically binds `name` in the logical layer data to the `name` of the view layer, so that `Hello taobao!` will be displayed when the page opens, and its logic is as follows:

1. When the user taps the button, the view layer sends a `changeName` event to the logical layer;
2. The logic layer finds the corresponding event handler function;
3. The logical layer executes the `setData` operation, changing `name` from `taobao` to `alipay`. Since the data is already bound to the view layer, the view layer will change to `Hello alipay!` automatically.

Note: Since the framework is not running in the browser, some of JavaScript's capabilities applicable to the Web are not available here, such as `document`, `window`.

JavaScript can use the es2015 modular syntax to organize the code of logical layer:

```
import util from './util'; // Load relative path
import absolute from '/absolute'; // Load files in the project root directory
```

Third-party NPM module

The MINI program supports importing third-party NPM modules. Firstly, execute the following command in the root directory of the MINI program to install the module:

```
$ npm install lodash --save
```

Once the third-party NPM module is installed, it can be used directly in the logical layer:

```
import lodash from 'lodash'; // Load third-party npm module
```

Note: Since the third-party module code in `node_modules` does not pass through the converter, the code under `node_modules` should be converted to es5 format before referencing in order to ensure compatibility on each terminal. It is recommended to use the import/export of es2015 as the module format. In addition, browser-related Web capabilities are also unavailable.

1.7.2. App

`App` represents top-level applications, which manages all pages and global data, and provides lifecycle methods. It is also a constructor that generates an `App` instance. A MINI program is an `App` instance.

Introduction

The top level of each MINI program normally contains three files:

- `app.acss` : Application style (optional)
- `app.js` : Application logic
- `app.json` : Application configuration

Here's a simple `app.json` :

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

In the above configuration, the MINI program contains two pages, and the default title of the application window is `Demo` .

`App` provides four events, allowing you to set the hook method:

- `onLaunch` : MINI program starts
- `onShow` : MINI program switches to the foreground
- `onHide` : MINI program switches to the background
- `onError` : MINI program error

`app.js` code sample is as follows:

```
App({
  onLaunch(options) {
    // MINI program initializes
  },
  onShow(options) {
    // MINI program displays
  },
  onHide() {
    // MINI program hides
  },
  onError(msg) {
    console.log(msg)
  },
  globalData: {
    foo: true,
  }
})
```

App()

`App()` Accept a `object` as a parameter to configure the life cycle and other information of the MINI program.

Parameter Description:

Property	Type	Description	Trigger condition
onLaunch	Function	Monitor the MINI program initialization	Triggered when the MINI program initialization is completed (only trigger once globally)

Property	Type	Description	Trigger condition
onShow	Function	Monitor the MINI program display	Triggered when the MINI program starts or switches to foreground from background
onHide	Function	Monitor the MINI program hiding	Triggered when the MINI program switches to background from foreground
onError	Function	Monitor the MINI program error	Triggered when a JS error occurred in the MINI program

Definition of foreground and background : When you tasp the close button in the upper left corner, or presses the Home button to leave the mPaaS client, the MINI program is not directly destroyed, but enters into the background. When you enters the mPaaS client again or opens the MINI program again, it will switch back to foreground from background.

Only when the MINI program enters into the background for a certain period of time, or the system resource usage is too high, will it be actually destroyed.

Parameters of onLaunch/onShow method

Property	Type	Description
query	Object	Query of the current MINI program
path	String	Page address of the current MINI program

- The parameter passing method for Native startup is:

```
// 启动小程序demo
Bundle param = new Bundle();
String queryParam = "param1=value1&param2=value2&param3=value3";
param.putString("query", queryParam);
LauncherApplicationAgent.getInstance().getMicroApplicationContext()
        .startApp( s: null,  s1: "2018080616290001", param);
```

- The parameter passing method for URL startup is: `query` is parsed from the `query` field of startup parameter, while `path` is parsed from the `page` field of startup parameter. For example, in the following URL:

```
alipays://platformapi/startapp?appId=1999&query=number%3D1&page=x%2Fy%2Fz
```

  - The `query` parameter is parsed as follows:

```
number%3D1 === encodeURIComponent('number=1')
```

  - The `path` parameter is parsed as follows:

```
x%2Fy%2Fz === encodeURIComponent('x/y/z')
```

`page` defaults to the home page when it is omitted

So, when you launches the MINI program for the first time, this parameter can be obtained from the `onLaunch` method. Alternatively, when the MINI program is reopened from the background by using schema, this parameter can also be obtained from the `onShow` method.

```
App({
  onLaunch(options) {
    // Opened for the first time
    // options.query == {number:1}
  },
  onShow(options) {
    // Reopened from the background by scheme
    // options.query == {number:1}
  },
})
```

getApp()

We provide a global `getApp()` function to get the MINI program instance, which is typically used in each subpage to get the top-level application.

```
var app = getApp()
console.log(app.globalData) // Get globalData
```

> **Note**
>
> Important: App() must be called in app.js, and cannot be called multiple times. Do not call getApp() in a function defined in App(). You can use this to get the app instance. Do not call getCurrentPages() in onLaunch as page has not been generated at this time. After getting the instance with getApp(), do not call the life-cycle function privately.

Global data can be set in `App()` , and each subpage can get a global application instance through the global function `getApp()` . For example:

```
// app.js
App({
  globalData: 1
})
```

```
// a.js

// localValue is only valid in a.js
var localValue = 'a'

// Generate the app instance
var app = getApp()

// Get global data and change it
app.globalData++
```

```
// b.js

// localValue is only valid in b.js
var localValue = 'b'

// If a.js runs first, globalData will return 2
console.log(getApp().globalData)
```

In the above code, `a.js` and `b.js` both declare the variable `localValue`, but they do not affect each other, because the variables and functions declared by each script are only valid in the file.

app.json

`app.json` is used for global configuration, which determines the path of the page file and window presentation, sets up network timeout, multiple tabs, and so on.

The following is a simple `app.json` that contains some configuration items.

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

The configuration of `app.json` are as follows.

File	Type	Required	Description
pages	String Array	Yes	Set page path
window	Object	No	Set the window presentation of default page
tabBar	Object	No	Set the presentation of bottom tabs

pages

The property of `pages` is an array, each item of which is a string that specifies the page of the MINI program. Each item represents the path information of the corresponding page. The first item of the array represents the home page of the MINI program. To add and reduce pages in the MINI program, you must modify the `pages` array.

The page path does not need to have a `js` suffix, and the framework will automatically load the `.json`, `.js`, `.axml`, and `.acss` files with the same name.

For example, if the development directory is:

```
pages/
pages/index/index.axml
pages/index/index.js
pages/index/index.acss
pages/logs/logs.axml
pages/logs/logs.js
app.js
app.json
app.acss
```

`app.json` must be coded as follows:

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs"
  ]
}
```

window

`window` is used to set the common status bar, navigation bar, title, and window background color of MINI programs.

Its sub-properties include `titleBarColor` , `defaultTitle` , `pullRefresh` , and `allowsBounceVertical` .

File	Type	Required	Description
titleBarColor	Decimal	No	Navigation bar background color
defaultTitle	String	No	Page title
pullRefresh	Boolean	No	Whether to allow pull-to-refresh. It defaults to false.
allowsBounceVertical	String(YES/NO)	No	Whether the page supports vertical drag and drop that exceeds the actual contents. It defaults to YES

This is an example.

```
{
  "window":{
    "defaultTitle": "Alipay interface feature demo"
  }
}
```

tabBar

If your MINI program is a multi-tab application (there are multiple tabs at the bottom of the window to switch pages), you can configure the presentation of the tab bar and the pages displayed when tapping the tabs through the `tabBar` .

Note:

- Any page reached by page jump (`my.navigateTo`) or page redirect (`my.redirectTo`) will not display the bottom tab bar even if it is a page defined in the tabBar configuration.
- The first page of `tabBar` must be the home page.

tabBar configuration:

File	Type	Required	Description
textColor	HexColor	No	Text color
selectedColor	HexColor	No	Color of selected text
backgroundColor	HexColor	No	Background color
items	Array	Yes	Configuration of each tab

Configuration of each item:

File	Type	Required	Description
pagePath	String	Yes	Set page path
name	String	Yes	Name
icon	String	No	Path of normal icon
activeIcon	String	No	Path of highlighted icon

The recommended icon size is 60*60 px. A non-proportional stretch/scale is performed on any imported images.

For example:

```
{
  "tabBar": {
    "textColor": "#dddddd",
    "selectedColor": "#49a9ee",
    "backgroundColor": "#ffffff",
    "items": [
      {
        "pagePath": "pages/index/index",
        "name": "Home Page"
      },
      {
        "pagePath": "pages/logs/logs",
        "name": "Log"
      }
    ]
  }
}
```

Startup parameters

You can bring `page` and `query` parameters when opening a mini program from native code. `Page` is used to specify the path to open a specific page, and `query` is used to bring in parameters.

- iOS code sample

```
NSDictionary *param = @{@"page":@"pages/card/index", @"query":@"own=1&sign=1&code=2452473"};
MPNebulaAdapterInterface startTinyAppWithId:@"1234567891234568" params:param];
```

- Android code sample

```
Bundle param = new Bundle();
param.putString("page", "pages/card/index");
param.putString("query", "own=1&sign=1&code=2452473");
MPNebula.startApp("1234567891234568",param);
```

1.7.3. Page

`Page` represents a page of application, and is used for page presentation and interaction. Each page corresponds to a subdirectory, which means that there are as many subdirectories as pages. It is also a constructor that is used to generate page instances.

Page initialization

- When a page is initialized, you must provide data for rendering the page for the first time:

```
<view>{{title}}</view>
<view>{{array[0].user}}</view>
```

```
Page({
data: {
  title: 'Alipay',
  array: [{user: 'li'}, {user: 'zhao'}]
}
})
```

- When you define an interaction behavior, you must specify the response function in the page script:

```
<view onTap="handleTap">click me</view>
```

The above template defines that the `handleTap` method will be called when the user taps:

```
Page({
handleTap() {
  console.log('yo! view tap!')
}
})
```

- When you re-render a page, you must call `this.setData` method in the page script.

```
<view>{{text}}</view>
<button onTap="changeText"> Change normal data </button>
```

The above code specifies that the `changeText` method will be called when the user taps the button.

```
Page({
data: {
  text: 'init data',
},
changeText() {
  this.setData({
    text: 'changed data'
  })
},
})
```

In the above code, calling the `this.setData` method in the `changeText` method will result in the page re-rendering.

Page()

`Page()` accepts a `object` as a parameter that specifies the page's initial data, life-cycle function, event handler, and so on.

```
//index.js
Page({
  data: {
    title: "Alipay"
  },
  onLoad(query) {
    // Page loading
  },
  onReady() {
    // Page loading completed
  },
  onShow() {
    // Page display
  },
  onHide() {
    // Page hiding
  },
  onUnload() {
    // Page closed
  },
  onTitleClick() {
    // Title tapped
  },
  onPullDownRefresh() {
    // Page pulled down
  },
  onReachBottom() {
    // Page pulled to the bottom
  },
  onShareAppMessage() {
   // Return to custom sharing information
  },
  viewTap() {
    // Event handling
    this.setData({
      text: 'Set data for updat.'
    })
  },
  go() {
    // A jump with parameters, reading xx from the query of onLoad function of page/index
    my.navigateTo('/page/index?xx=1')
  },
  customData: {
    hi: 'alipay'
  }
})
```

In the above code, the parameter objects of the `Page()` method are described as follows:

Attribute	Type	Description
data	Object or Function	Initial data, or the function that returns initialized data
onTitleClick	Function	Triggered after title is tapped
onOptionMenuClick	Function	Supported by basic library 1.3.0+; triggered by tapping the navigation bar icon out of the grid, and can be determined by `my.canIUse('page.onOptionMenuClick')`
onPageScroll	Function({scrollTop})	Triggered when the page scrolls
onLoad	Function(query: Object)	Triggered when the page loads
onReady	Function	Triggered when the first rendering of the page is completed
onShow	Function	Triggered when the page is displayed
onHide	Function	Triggered when the page is hidden
onUnload	Function	Triggered when the page is unloaded
onPullDownRefresh	Function	Triggered when the page is pulled down
onReachBottom	Function	Triggered when the page reaches the bottom
onShareAppMessage	Function	Triggered when the user taps the Share button in the top right corner
Other	Any	The developers can add any functions or properties to the `object` parameter, which can be accessed through `this` in the page function.

> **Note**
>
> Note: In case data is an object, changing it in the page affects different instances of the page.

Lifecycle method

- onLoad: Used on page loadding. It is only called once per page, and the query parameter is the `query` object passed in `my.navigateTo` and `my.redirectTo`.
- onShow: Used on page display. It is called every time a page is displayed.
- onReady: Used when a page is rendered for the first time. It is only called once per page, indicating that a page is ready to interact with the view layer. If you need to configure interface, such as using `my.setNavigationBar`, do it after `onReady`.
- onHide: Used on page hiding. It is called when you use `my.navigateTo` to switch to other pages or use `tab` to switch bottom tabs.
- onUnload: Used on page unloading. It is called when you use `my.redirectTo` or `my.navigateBack` to switch to other pages.

Event handler

- onPullDownRefresh: Pull-to-refresh. To monitor the user's pull-to-refresh event, you must enable `pullRefresh` in the `window` option of `app.json`. When the data is refreshed, `my.stopPullDownRefresh` can stop pull-to-refresh on the current page.
- onShareAppMessage: User sharing, see Share for more information.

Page.prototype.setData()

`setData` function is used to send data from the logical layer to the view layer, and change the value of the corresponding `this.data` at the meantime.

> **Note**
>
> Note: Modifying this.data directly cannot change the page state, and will cause data inconsistency. Do not to set too much data at once.

`setData` accepts an object as a parameter. The object's key name `key` can be specified flexibly in the form of a data path, such as `array[2].message` and `a.b.c.d`, and does not need to be predefined in `this.data`.

Sample code

```
<view>{{text}}</view>
<button onTap="changeTitle"> Change normal data </button>
<view>{{array[0].text}}</view>
<button onTap="changeArray"> Change Array data </button>
<view>{{object.text}}</view>
<button onTap="changePlanetColor"> Change Object data </button>
<view>{{newField.text}}</view>
<button onTap="addNewKey"> Add new data </button>
```

```
Page({
  data: {
    text: 'test',
    array: [{text: 'a'}],
    object: {
      text: 'blue'
    }
  },
  changeTitle() {
    // Incorrect! Do not modify the content in data directly
    // this.data.text = 'changed data'

    // Correct
    this.setData({
      text: 'ha'
    })
  },
  changeArray() {
    // Use data path to modify data directly
    this.setData({
      'array[0].text':'b'
    })
  },
  changePlanetColor(){
    this.setData({
      'object.text': 'red'
    });
  },
  addNewKey() {
    this.setData({
      'newField.text': 'c'
    })
  }
})
```

getCurrentPages()

`getCurrentPages()` function is used to get the instance of the current page stack, which is given in the form of an array and in the order of the stack. The first element is the first page and the last element is the current page. The following code can be used to detect if the current page stack has a 5-level page depth.

```
if(getCurrentPages().length === 5) {
  my.redirectTo('/xx');
} else {
  my.navigateTo('/xx');
}
```

> **Note**
>
> Note: Don't try to modify the page stack, otherwise it will lead to routing and page status errors.

The framework maintains all current pages in the form of stack. When a route switch occurs, the page stack behaves as follows:

Routing method	Page stack behavior
Initialization	A new page is pushed on to the stack
Open a new page	A new page is pushed on to the stack
Page redirection	The current page is popped off the stack while a new page is pushed on to the stack
Page return	The current page is popped off the stack
Tab switch	All the pages are popped off the stack, leaving only the new Tab page

page.json

The window behavior of each page can be configured with the `[page name].json` file.

The page configuration is much simpler than the `app.json` global configuration, and only `window` related configuration items can be set, so there is no need to write the `window` key. Note that the page configuration will override the configuration items in the `window` attribute of `app.json`.

In addition, it also supports configuring navigation icons with `optionMenu`, on which user taps, `onOptionMenuClick` will be triggered.

File	Type	Required	Description
optionMenu	Object	No	Supported by basic library 1.3.0+; can be used to set the navigation bar icons. Currently, it supports setting the icon attribute with a value of an icon URL (starting with https/http) or a base64 string. The recommended icon size is 30*30 px.

For example:

```
{
  "optionMenu": {
    "icon": "https://img.alicdn.com/tps/i3/T1OjaVFl4dXXa.JOZB-114-114.png"
  }
}
```

Page style

The root element in each page is `page`, which can be used to set the height or background color.

```
page {
  background-color: #fff;
}
```

1.7.4. View layer

Introduction

The view file has a suffix of `axml` and defines the label structure of the page.

Here are some examples to show the capabilities of `axml`.

- **Data binding**:

```
<view> {{message}} </view>
```

```
// page.js
Page({
data: {
  message: 'Hello alipay!'
}
})
```

- **List rendering**:

```
<view a:for="{{items}}"> {{item}} </view>
```

```
// page.js
Page({
data: {
  items: [1, 2, 3, 4, 5, 6, 7]
}
})
```

- **Conditional rendering**:

```
<view a:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view a:elif="{{view == 'APP'}}"> APP </view>
<view a:else="{{view == 'alipay'}}"> alipay </view>
```

```
// page.js
Page({
data: {
  view: 'alipay'
}
})
```

- **Template**:

```
  <template name="staffName">
    <view>
      FirstName: {{firstName}}, LastName: {{lastName}}
    </view>
  </template>

  <template is="staffName" data="{{...staffA}}"></template>
  <template is="staffName" data="{{...staffB}}"></template>
  <template is="staffName" data="{{...staffC}}"></template>
```

```
// page.js
// Hats off to the Wechat MINI engineers.
Page({
data: {
  staffA: {firstName: 'san', lastName: 'zhang'},
  staffB: {firstName: 'si', lastName: 'li'},
  staffC: {firstName: 'wu', lastName: 'wang'},
},
})
```

- **Event**:

```
<view onTap="add"> {{count}} </view>
```

```
Page({
data: {
  count: 1
},
add(e) {
  this.setData({
    count: this.data.count + 1
  })
}
})
```

Data binding

All dynamic data in `axml` come from `data` in the corresponding Page.

Simple binding

The data binding uses the Mustache syntax (double braces) to enclose the variables, which can apply to various scenes.

- Applies to content, for example:

```
<view> {{ message }} </view>
```

```
Page({
data: {
  message: 'Hello alipay!'
}
})
```

- Applies to component properties (must be enclosed in double quotes), for example:

```
<view id="item-{{id}}"> </view>
```

```
Page({
data: {
  id: 0
}
})
```

- Applies to control properties (must be enclosed in double quotes), for example:

```
<view a:if="{{condition}}"> </view>
```

```
Page({
data: {
  condition: true
}
})
```

- Applies to keywords (must be enclosed in double quotes), for example:

```
<checkbox checked="{{false}}"> </checkbox>
```

  - true: A boolean true, representing a true value.
  - false: A Boolean false, representing a false value.

> **Note**
> Important: Do not write checked="false" directly, the result of which is a string that represents true after being converted to a Boolean data value.

The following simple operations can be performed within `{{}}`:

- Ternary operation:

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

- Arithmetic operation:

```
<view> {{a + b}} + {{c}} + d </view>
```

```
Page({
data: {
  a: 1,
  b: 2,
  c: 3
}
})
```

The content in the View is `3 + 3 + d`.

- Logical judgment:

```
<view a:if="{{length > 5}}"> </view>
```

- String operation:

```
<view>{{"hello" + name}}</view>
```

```
Page({
data:{
  name: 'alipay'
}
})
```

- Data path operation:

```
<view>{{object.key}} {{array[0]}}</view>
```

```
Page({
data: {
  object: {
    key: 'Hello '
  },
  array: ['alipay']
}
})
```

It can also be combined directly in Mustache to form a new array or object.

- Array:

```
<view a:for="{{[zero, 1, 2, 3, 4]}}"> {{item}} </view>
```

```
Page({
data: {
  zero: 0
}
})
```

Finally combined into an array [0, 1, 2, 3, 4].

- Object:

```
<template is="objectCombine" data="{{foo: a, bar: b}}"></template>
```

```
Page({
data: {
  a: 1,
  b: 2
}
})
```

The final combined object is {foo: 1, bar: 2}.

You can also expand an object with the extended operator `...`.

```
<template is="objectCombine" data="{{...obj1, ...obj2, e: 5}}"></template>
```

```
Page({
data: {
  obj1: {
    a: 1,
    b: 2
  },
  obj2: {
    c: 3,
    d: 4
  }
}
})
```

The final combined object is {a: 1, b: 2, c: 3, d: 4, e: 5}.

If the object's key and value are the same, it can also be expressed indirectly.

```
<template is="objectCombine" data="{{foo, bar}}"></template>
```

```
Page({
data: {
  foo: 'my-foo',
  bar: 'my-bar'
}
})
```

The final combined object is {foo: 'my-foo', bar:'my-bar'}.

> **Note**
>
> Note: The above operations can be combined at will, but if there are cases where the variable names are the same, the latter variable will overwrite the former one.

```
<template is="objectCombine" data="{{...obj1, ...obj2, a, c: 6}}"></template>
```

```
Page({
data: {
  obj1: {
    a: 1,
    b: 2
  },
  obj2: {
    b: 3,
    c: 4
  },
  a: 5
}
})
```

The final combined object is {a: 5, b: 3, c: 6}.

Conditional rendering

a:if

In the framework, you can use `a:if="{{condition}}"` to determine if the code block needs to be rendered.

```
<view a:if="{{condition}}"> True </view>
```

You can also use `a:elif` and `a:else` to add an `else` block.

```
<view a:if="{{length > 5}}"> 1 </view>
<view a:elif="{{length > 2}}"> 2 </view>
<view a:else> 3 </view>
```

block a:if

Since `a:if` is a control attribute, you need to add it to a label. To determine multiple component labels at the same time, you can enclose multiple components with a single `<block/>` label and use `a:if` to control the attribute on it.

```
<block a:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

> **Note**
>
> Note: <block/> is not a component, but only an enclosure element that doesn't need any rendering in the page and only accepts control properties.

List rendering

a:for

By binding an array on the component with the `a:for` attribute, you can repeatedly render the component using the data of each item in the array.

The subscripted variable name of the current item in the default array defaults to `index`. The variable name of the current item in the array defaults to `item`.

```
<view a:for="{{array}}">
  {{index}}: {{item.message}}
</view>
```

```
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar'
    }]
  }
})
```

Use `a:for-item` to specify the variable name of the current element of the array.

Use `a:for-index` to specify the variable name of the current subscript of the array.

```
<view a:for="{{array}}" a:for-index="idx" a:for-item="itemName">
  {{idx}}: {{itemName.message}}
</view>
```

`a:for` can also be nested. Below is an example of Multiplication Table:

```
<view a:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" a:for-item="i">
  <view a:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" a:for-item="j">
    <view a:if="{{i <= j}}">
      {{i}} * {{j}} = {{i * j}}
    </view>
  </view>
</view>
```

block a:for

Similar to `block a:if`, you can also use `a:for` on the `<block/>` label to render a building block containing multiple nodes.

```
<block a:for="{{[1, 2, 3]}}">
  <view> {{index}}: </view>
  <view> {{item}} </view>
</block>
```

a:key

If the position of an item in the list changes dynamically or a new item is added to the list, and you want the item in the list to maintain its own feature and state (such as the input of `<input/>` and the selected state of `<switch/>`), you need to use `a:key` to specify a unique identifier for the item in the list.

`a:key` has a value that is provided in two forms:

- A string, representing an attribute of `item` in `array` in the `for` loop. The value of this attribute must be a unique string or number in the list and cannot be changed dynamically.
- A reserved keyword `*this`, representing the `item` itself in the `for` loop. It indicates that `item` itself must be a unique string or number. For example, when the data change triggers the rendering layer to re-render, the component with `key` is calibrated, and the framework ensures that they are reordered rather than recreated to guarantee that the component maintains its state and improves the efficiency of list rendering.

If you know that the list is static or you don't care its order, you can choose to ignore it.

The code sample is as follows:

```
<view class="container">
  <view a:for="{{list}}" a:key="*this">
    <view onTap="bringToFront" data-value="{{item}}">
    {{item}}: click to bring to front
    </view>
  </view>
</view>
```

```
Page({
  data:{
    list:['1', '2', '3', '4'],
  },
  bringToFront(e) {
    const { value } = e.target.dataset;
    const list = this.data.list.concat();
    const index = list.indexOf(value);
    if (index !== -1) {
      list.splice(index, 1);
      list.unshift(value);
      this.setData({ list });
    }
  }
});
```

key

`key` is a more common version of `a:key`, in which you can populate with any expressions or strings.

The code sample is as follows:

```
<view class="container">
  <view a:for="{{list}}" key="{{item}}">
    <view onTap="bringToFront" data-value="{{item}}">
    {{item}}: click to bring to front
    </view>
  </view>
</view>
```

```
Page({
  data:{
    list:['1', '2', '3', '4'],
  },
  bringToFront(e) {
    const { value } = e.target.dataset;
    const list = this.data.list.concat();
    const index = list.indexOf(value);
    if (index !== -1) {
      list.splice(index, 1);
      list.unshift(value);
      this.setData({ list });
    }
  }
});
```

In addition, you can use `key` to prevent the reuse of components. For example, if you allow users to enter different types of data:

```
<input a:if="{{name}}" placeholder="Enter your username">
<input a:else placeholder="Enter your email address">
```

Then, when you enter name and switch to email, the current input value will be retained. If you don't want to retain it, you can add `key`:

```
<input key="name" a:if="{{name}}" placeholder="Enter your username">
<input key="email" a:else placeholder="Enter your email address">
```

Reference

`axml` provides two file reference methods, `import` and `include`.

import

`import` can be used to load a defined `template`.

For example, a `template` named `item` is defined in `item.axml`.

```
<!-- item.axml -->
<template name="item">
  <text>{{text}}</text>
</template>
```

By referencing `item.axml` in `index.axml`, you can use the `item` template.

```
<import src="./item.axml"/>
<template is="item" data="{{text: 'forbar'}}"/>
```

`import` has a concept of scope, which only `import` the `template` defined in the target file. For example, if B is imported to C and A is imported to B, then you can use the `template` defined by `B` in `C`, and use the `template` defined by `A` in `B`, but you can't use the `template` defined by `A` in `C`.

```
<!-- A.axml -->
<template name="A">
  <text> A template </text>
</template>
```

```
<!-- B.axml -->
<import src="./a.axml"/>
<template name="B">
  <text> B template </text>
</template>
```

```
<!-- C.axml -->
<import src="./b.axml"/>
<template is="A"/>  <!-- Error! Can not use tempalte when not import A. -->
<template is="B"/>
```

Note that a `template` can only have one child node instead of multiple ones, for example:

- Allowed:

```
<template name="x">
<view />
</template>
```

- Restricted:

```
<template name="x">
<view />
<view />
</template>
```

include

`include` can introduce the entire code of target file (except `<template/>`), which is equivalent to copying to `include`.

The code sample is as follows:

```
<!-- index.axml -->
<include src="./header.axml"/>
<view> body </view>
<include src="./footer.axml"/>
```

```
<!-- header.axml -->
<view> header </view>
```

```
<!-- footer.axml -->
<view> footer </view>
```

Template

`axml` provides a template, in which you can define code snippets and call them in different places.

Define a template

Use the `name` attribute as the name of the template and then define the code snippets within `<template/>`.

```
<!--
  index: int
  msg: string
  time: string
-->
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```

Use a template

Use the `is` attribute to declare the template you want to use and then pass in the `data` required by the template, for example:

```
<template is="msgItem" data="{{...item}}"/>
```

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2016-09-15'
    }
  }
})
```

`is` attribute can use the `Mustache` syntax to dynamically determine which template needs to be rendered.

```
<template name="odd">
  <view> odd </view>
</template>
<template name="even">
  <view> even </view>
</template>

<block a:for="{{[1, 2, 3, 4, 5]}}">
    <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
</block>
```

> **Note**
>
> Note: The template has its own scope and can only use data passed by data, however, it can handle functions by using the logic bound to page through onXX.

It is recommended that you use a template to introduce template snippets, because the template will specify its own scope and only use the data passed by `data`, which will be optimized by the MINI program. If the data of the template is not changed, the snippet UI will not be re-rendered.

The import path supports loading third-party modules from the `node_modules` directory, for example: `page.axml`:

```
<import src="./a.axml"/> <!-- Relative path -->
<import src="/a.axml"/> <!-- Project absolute path -->
<import src="third-party/x.axml"/> <!-- Third party npm package path -->
```

1.7.5. Event

What is an event?

- The event is the way how the view layer is communicated with the logical layer.
- The event can feed the user's behavior back to the logical layer for processing.
- The event can be bound to components, and when the trigger condition is satisfied, the corresponding event handler in the logical layer is executed.
- The event objects can carry additional information such as id, dataset, and touches.

How to use

Events are divided into **Bubbling events** and **Non-bubbling events**:

- **Bubbling event**: When an event on a component is triggered, the event is passed to the parent node.
- **Non-bubbling event**: When an event on a component is triggered, the event is not passed to the parent node.

The event binding is written in the same way as the component's properties, in the form of key and value.

- key begins with on or catch, and then followed by the type of event, such as onTap and catchTap.
- The value is a string that needs to define a function with the same name in the corresponding Page. Otherwise, an error will be reported when the event is triggered.

The on event binding does not prevent the propagation of bubbling event, while the catch event binding will prevent it.

```
<view id="outter" onTap="handleTap1">
  view1
  <view id="middle" catchTap="handleTap2">
    view2
    <view id="inner" onTap="handleTap3">
      view3
    </view>
  </view>
</view>
```

In the above code, tapping view3 firstly triggers `handleTap3` and then `handleTap2` (because the tap event propagats to view2, which prevents the propagation of tap event, and the event will not propagate to the parent node). Tapping view2 triggers `handleTap2` while tapping view1 triggers `handleTap1`.

Bubbling event list:

Type	Triggering condition
touchStart	Touch action starts
touchMove	Moves after touching
touchEnd	Touch action ends
touchcancel	The touch action is interrupted such as by incoming call and pop-up dialog
tap	Leaves immediately after touching
longTap	Touches for more than 300 ms before leaving

No bubbling for other events:

- Bind an event handler to the component.

  Take `onTap` for example, when the user taps the component, the event handler will be found in the corresponding `Page`.

```
<view id="tapTest" data-hi="Alipay" onTap="tapName">
<view id="tapTestInner" data-hi="AlipayInner">
  Click me!
</view>
</view>
```

- Write the event handler in the corresponding `Page` definition, in which the parameter should be event:

```
Page({
tapName(event) {
  console.log(event)
}
})
```

You will see the following information in the log:

```
{
"type": "tap",
"timeStamp": 13245456,
"target": {
  "id": "tapTestInner",
  "dataset": {
    "hi": "Alipay"
  },
  "targetDataset": {
    "hi": "AlipayInner"
  }
},
"currentTarget": {
  "id": "tapTest",
  "dataset": {
    "hi": "Alipay"
  }
}
}
```

Event object

When a component triggers an event, the handler bound to the event by the logic layer receives an event object.

- `BaseEvent`: List of basic event object properties.

Attribute	Type	Description
type	String	Event type
timeStamp	Integer	Timestamp when the event was generated
target	Object	A set of attribute values of the component that triggered the event

- `CustomEvent`: List of custom event object properties (inherited from `BaseEvent`).

Attribute	Type	Description
detail	Object	Additional information

- `TouchEvent`: List of touch event object properties (inherited from `BaseEvent`).

Attribute	Type	Description
touches	Array	An array of touch points that are currently on the screen
changedTouches	Array	An array of touch points that are currently changed

- `Type`: Type of event.
- `timeStamp`: Number of milliseconds since page opening to event triggering.
- `target`: Source component that triggered the event.

Attribute	Type	Description
id	String	ID of the event source component
tagName	String	Type of the current component
dataset	Object	A set of custom properties starting with `data-` on the component that the event is bound to
targetDataset	Object	A set of custom properties starting with `data-` on the component that actually triggered the event

dataset

Data can be defined in the component and will be passed to the logic layer through events.

Writing format: Starting with `data-` , multiple words are concatenated by a hyphen (-), words are in lowercase letter (the uppercase letters will automatically be converted to lowercase), such as `data-element-type` . In `event.target.dataset` , the hyphenated string will be converted into camel case `elementType` .

Code sample:

```
<view data-alpha-beta="1" data-alphaBeta="2" onTap="bindViewTap"> DataSet Test </view>
```

```
Page({
  bindViewTap:function(event){
    event.target.dataset.alphaBeta === 1 // Hyphens are onverted into camel case
    event.target.dataset.alphabeta === 2 // The uppercase letters are converted to lowercase
  }
})
```

touches

`touches` is an array, in which each element is a `Touch` object (the `touches` carried in `canvas` touch event is an array of `CanvasTouch`), and represents the touch points that are currently on the screen.

- `Touch` Object

Attribute	Type	Description
identifier	Number	Touch point identifier
pageX, pageY	Number	The distance from the upper left corner of the document. The top left corner is the origin, the horizontal direction is the X axis, and the vertical direction is the Y axis.
clientX, clientY	Number	The distance from the upper left corner of the displayable area in the page (navigation bar excluded). The horizontal direction is the X axis, and the vertical direction is the Y axis.

- `CanvasTouch` Object

Attribute	Type	Description
identifier	Number	Touch point identifier
x, y	Number	The distance from the upper left corner of Canvas. The upper left corner of Canvas is the origin, the horizontal direction is the X axis, and the vertical direction is the Y axis.

- `changedTouches` : `changedTouches` has the same data format as `touches` . It indicates the touch points that are changed, such as from non-existence to existence (`touchstart`), position change (`touchmove`), and from existence to non-existence (`touchend` and `touchcancel`).
- `detail` : The data carried in the custom event. For example, the submission event of the form component carries the user's input, and the error event of the media carries the error message. For details, see definition of each event in Component Definition.

1.7.6. Style

acss (AntFinancial Style Sheet) is a set of style language that describes the component style of `axml` pages and determines how the components of `axml` should be displayed.

In order to adapt to all front-end developers, the acss has most of the features of CSS. At the same time, we also extended CSS to make it more suitable for developing MINI programs.

Compared with CSS, the extended features of acss are:

- **rpx**: rpx (responsive pixel) can adapt to the width of the screen. The specified screen width is 750 rpx. On iPhone 6 screen, the width is 375 px and the number of physical pixels is 750, it means that 750 rpx = 375 px = 750 physical pixels, so 1 rpx = 0.5 px = 1 physical pixel.

Device	Convert rpx to px (screen width/750)	Convert px to rpx (750/screen width)
iPhone5	1 rpx = 0.42 px	1 px = 2.34 rpx
iPhone6	1 rpx = 0.5 px	1 px = 2 rpx
iPhone6 Plus	1 rpx = 0.552 px	1 px = 1.81 rpx

- **Style import**: Use the `@import` statement to import external stylesheet. `@import` should be followed by the relative path of the external stylesheet and end with a semicolon (;).

  Code sample:

```
/** button.acss **/
.sm-button {
padding:5px;
}
```

```
/** app.acss **/
@import "./button.acss";
.md-button {
padding:15px;
}
```

Path importing supports loading third-party modules from the `node_modules` directory, such as `page.acss`:

```
@import "./button.acss"; /*Relative path*/
@import "/button.acss"; /*Project absolute path*/
@import "third-party/button.acss"; /*Third-party npm package path*/
```

- **Inline style**: `style` and `class` properties are supported in the component to control the style.
  - `style` attribute: The static styles are all written to `class`. `style` receives dynamic styles, which are parsed at runtime. Do not write static styles into `style` to avoid affecting rendering speed.

    ```
    <view style="color:{{color}};" />
    ```

  - `class` attribute: Used to specify style rules. The attribute value is a set of class selector names (style class names) in the style rule. The style class names are without dots (.), and are separated with spaces.

    ```
    <view class="my-awesome-view" />
    ```

- **Selector** : Keep consistent with CSS3.

  > **Note**
  > Important: Class selectors starting with .a- or .am- are reserved for system components, do not use them. The attribute selector is not supported.

- **Global styles and local styles**: The styles defined in `app.acss` are global styles that apply to each page. The styles defined in Page's `acss` file are local styles that only apply to the corresponding page and override the same selector in `app.acss`.
- **Page container style**: The page container style (such as the page background color) can be set via the page element selector:

```
page {
background-color: red;
}
```

1.7.7. Global configuration for mini program

1.7.7.1. Introducing global configuration for Mini Program

`App()` is used to obtain the top-layer application that manages all the pages, global data, and lifecycle callbacks. `App()` is also a constructor that can be used to generate an app instance.

A Mini Program is an app instance. Generally, the top layer of each Mini Program consists of three files.

- `app.json` : contains the application configurations.
- `app.js` : contains the application logic.
- `app.acss` : contains the application style. This file is optional.

Sample code

- The following code snippet shows a simple `app.json` file:

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/logs"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

```
As shown in the code, the Mini Program contains two pages: "index" and "logs". The default title of the window is "Demo".
```

- The following code snippet shows a simple `app.js` file:

```
App({
  onLaunch(options) {
    // Open for the first time.
  },
  onShow(options) {
    // The Mini Program is started or re-opened from the background.
  },
  onHide() {
    // The Mini Program is switched from the foreground to the background.
  },
  onError(msg) {
    // A JavaScript error occurs or an API call fails in the Mini Program.
    console.log(msg);
  },
  globalData: {
    // The method is called to obtain global data.
    name: 'mPaaS',
  },
});
```

1.7.7.2. app-json global configuration

`The app.json` file contains the global configurations of a Mini Program, such as the page paths, window display, network timeout, and tabBar configurations.

The following code snippet shows a sample file:

```
{
  "pages": [
    "pages/index/index",
    "pages/logs/index"
  ],
  "window": {
    "defaultTitle": "Demo"
  }
}
```

The following table describes all the configuration items in the app.json file.

Attribute	Type	Mandatory	Description
pages	Array	Yes	The paths of the pages in the Mini Program.
window	Object	No	The window display of the default page.
tabBar	Object	No	The configurations of the bottom tabBar.

pages

The configuration item `pages` in the `app.json` file is an array of strings that specify pages in the Mini Program. To add a page to or remove a page from the Mini Program, modify the configuration item `pages`.

Each string in `pages` represents the path of a page in the Mini Program. The first string corresponds to the homepage of the Mini Program.

You do not need to add an affix to each page path. The mPaaS framework automatically loads the `.json`, `.js`, `.axml`, and `.acss` files with the same names as specified in the page paths. For example, if your developed file structure is as shown in the following code snippet:

```
├── pages
|   ├──index
|   |    ├── index.json
|   |    ├── index.js
|   |    ├── index.axml
|   |    └── index.acss
|   ├──logs
|   |    ├── logs.json
|   |    ├── logs.js
|   |    └── logs.axml
├── app.json
├── app.js
└── app.acss
```

you must write the following code to the `app.json` file:

```
{
  "pages":[
    "pages/index/index",
    "pages/logs/logs"
  ]
}
```

window

The configuration item `window` is used to configure the user interface of the Mini Program, for example, to configure the status bar and navigation bar and set a title and a window background color.

Attribute	Type	Whether Mandatory	Description
defaultTitle	String	No	The default title of the window
pullRefresh	String	No	A string that determines whether to allow pull-to-refresh. Default value: “NO”. Note that for the pull-to-refresh to take effect, the value of `allowsBounceVertical` must be “YES”.
allowsBounceVertical	String	No	A string that determines whether bouncing occurs when vertical scrolling reaches the end of the content. Valid values: `YES` and NO. Default value: `YES`.
transparentTitle	String	No	A string that determines the transparency effect of the navigation bar. Valid values: `always`, `auto`, and `none`. If you set the string to “always”, the navigation bar is always transparent. If you set the string to “auto”, the transparency effect of the navigation bar changes based on slide operations. If you set the string to “none”, the navigation bar is not transparent. Default value: `none`.
titlePenetrate	String	No	A string that determines whether to allow the penetration of the navigation bar. Valid values: `YES` and `NO`. Default value: `NO`
showTitleLoading	String	No	A string that determines whether to display the loading prompt during the startup of the Mini Program. Valid values: `YES` and `NO`. Default value: `NO`.
titleImage	String	No	The image path of the navigation bar.
titleBarColor	HexColor	No	The background color of the navigation bar.
backgroundColor	HexColor	No	The background color to display when the page is slid down.
backgroundImageColor	HexColor	No	The background color of the image to display when the page is slid down.
backgroundImageUrl	String	No	The URL of the background image to display when the page is slid down.
gestureBack	String	No	A string that determines whether to allow a gesture operation to trigger a return event. This event is supported only for iOS. Valid values: `YES` and `NO`. Default value: `NO`
enableScrollBar	Boolean	No	A string that determines whether to display the scroll bar of WebView. This event is supported only for Android. Valid values: `YES` and `NO`. Default value: `YES`

Code sample:

```
{
  "window":{
    "defaultTitle": "API Demo on the Client"
  }
}
```

tabBar

For a Mini Program that has multiple pages, you can configure `tabBar` and the corresponding pages. tabBar is used to switch among the pages and displayed at the bottom of the user interface on the client.

Notes:

- If a page appears after the `my.navigateTo` or `my.redirectTo` operation is performed, tabBar does not appear at the bottom even if the page is defined in tabBar.
- The default page of `tabBar` must be the homepage of the Mini Program.

The following table describes the configuration items of tabBar.

Attribute	Type	Mandatory	Description
textColor	HexColor	No	The text color of a tab icon when the icon is not selected.
selectedColor	HexColor	No	The text color of a tab icon when the icon is selected.
backgroundColor	HexColor	No	The background color
items	Array	Yes	The configurations of each tab in tabBar

The following table describes the detailed configurations of each tab.

Attribute	Type	Mandatory	Description
pagePath	String	Yes	The paths of the pages in the Mini Program.
name	String	Yes	Name.
icon	String	No	The URL of the tab icon when the corresponding page is not displayed.
activeIcon	String	No	The URL of the highlighted tab icon when the corresponding page is displayed.

We recommend that you choose an image of 60 × 60 pixels. Images of other sizes will automatically be adjusted to this size in a non-proportional manner.

The following code snippet shows sample configurations of tabBar:

```
{
  "tabBar": {
    "textColor": "#dddddd",
    "selectedColor": "#49a9ee",
    "backgroundColor": "#ffffff",
    "items": [
      {
        "pagePath": "pages/index/index",
        "name": "Homepage"
      },
      {
        "pagePath": "pages/logs/logs",
        "name": "Logs"
      }
    ]
  }
}
```

1.7.7.3. app acss global style

`app.acss` is used to configure a global style that applies to all the pages of current Mini Program. For more information about **acss**, see ACSS syntax reference.

1.7.7.4. app.js registering Mini Program

App(object: Object)

- `App()` is used to register a Mini Program. `App()` accepts an Object attribute to configure the lifecycle of the Mini Program.
- `You can call App()` only in the `app.js` file and only for once.

Descriptions of object attributes

Attribute	Type	Description	Trigger condition
onLaunch	Function	Lifecycle callback: listens to the initialization of the Mini Program	Triggered when the initialization of the Mini Program completes. Triggered only for once during the lifecycle of the Mini Program.
onShow	Function	Lifecycle callback: listens to the display of the Mini Program	Triggered when the Mini Program is launched or triggered when the display is switched to the foreground from the background.
onHide	Function	Lifecycle callback: listens to hide operations of the Mini Program	Triggered when the Mini Program is switched to the background from the foreground.
onError	Function	Listens to errors in the Mini Program	Triggered when a JavaScript error occurs in the Mini Program.
onShareAppMessage	Function	Configures global sharing	

Definitions of the foreground and background of a Mini Program:

- When a user taps the close button in the upper-right corner to close a Mini Program or presses the Home button on the device to leave Alipay, the Mini Program is not terminated but runs in the background.
- When the user re-opens Alipay or the Mini Program, the Mini Program will run from the background to the foreground.
- A Mini Program is terminated only after it runs in the background for a specific period of time or it occupies excessive system resources.

onLaunch(object: Object) and onShow(object: Object)

The following table describes the object attributes.

Attribute	Type	Description
query	Object	The current query object of the Mini Program. This object is resolved from the "query" field for Mini Program startup.
path	String	The current page address of the Mini Program. This page address is resolved from the "page" field for Mini Program startup. If the "page" field is not specified for Mini Program startup, the path corresponds to the homepage.
referrerInfo	Object	The source information.

For example, the schema URL for starting a Mini Program is shown as follows:

```
alipays://platformapi/startapp?appId=1999&query=number%3D1&page=x%2Fy%2Fz
```

The following code snippet shows the resolution results of the "query" and "path" attributes:

```
query = decodeURIComponent('number%3D1');
// number=1
path = decodeURIComponent('x%2Fy%2Fz');
// x/y/z
```

- When a Mini Program is started for the first time, you can use the `onLaunch` method to obtain the values of the `query` and `path` attributes.
- When the Mini Program is opened by using a schema URL, you can use the `onShow` method to obtain the values of the `query` and `path` attributes.

```
App({
  onLaunch(options) {
    // Open for the first time.
    console.log(options.query);
    // {number:1}
    console.log(options.path);
    // x/y/z
  },
  onShow(options) {
    // Re-opened by schema in the background.
    console.log(options.query);
    // {number:1}
    console.log(options.path);
    // x/y/z
  },
});
```

The following table describes the sub attributes of the `referrerInfo` attribute.

Attribute	Type	Description	Compatibility
appId	String	From the source Mini Program.	
sourceServiceId	String	The source plug-in, which is visible in running mode.	1.11.0
extraData	Object	The data that is transferred from the source Mini Program.	

Notes:

- Do not call a method that manages pages, such as `redirectTo` or `navigateTo`, in the `onShow` event.
- Do not call the `getCurrentPages()` method in the `onLaunch` event, because the Page objects have not been generated.

onHide()

The `onHide()` event is triggered when the Mini Program is switched to the background from the foreground.

Code sample:

```
App({
  onHide() {
    // This event is triggered when the Mini Program is switched to the background.
    console.log('app hide');
  },
});
```

onError(error: String)

This event is triggered when a JavaScript error occurs or an API call fails in the Mini Program.

```
App({
  onError(error) {
    // This event is triggered when an execution error occurs in the Mini Program.
    console.log(error);
  },
});
```

onShareAppMessage(object: Object)

This event is triggered to configure global sharing. If `page.onShareAppMessage` is not configured for a page, global sharing configurations apply when the page is being shared. For more information, see Sharing.

globalData

You can use `globalData` to configure global data in `App()`.

Code sample:

```
// app.js
App({
  globalData: 1
});
```

1.7.7.5. getApp method

The mPaaS framework provides the global method `getApp()` to obtain the current Mini Program instance. The method is generally used to obtain a top-layer application from a page.

```
var app = getApp();
console.log(app.globalData); // Obtain global data.
```

When you use the `getApp()` method, be aware that:

- You cannot call the `getApp()` method in the `App()` function. Use `this` to obtain the current Mini Program instance.
- After you obtain the Mini Program instance by calling the `getApp()` method, do not call a lifecycle callback function without authorization.
- You need to make a difference between global variables and local page variables. For example:

```
// a.js

  // localValue is only available in a.js.
  var localValue = 'a';
  // Obtain the app instance.
  var app = getApp();
  // Obtain global data and modify the data.
  app.globalData++;
```

```
// b.js

  // localValue is only available in b.js.
  var localValue = 'b';
  // If a.js runs first, the obtained global data is 2.
  console.log(getApp().globalData);
```

The variable `localValue` is declared in both the `a.js` file and the `b.js` file. The two variables do not affect each other, because a local variable in a file is available only in the file.

1.7.8. Mini Program page

1.7.8.1. Page introduction

Page represents a page of the app and is responsible for page presentation and interaction. Each page corresponds to one subdirectory. The number of pages is equal to the number of subdirectories. The Page component is also a constructor function used to generate a page instance.

In general, each Mini Program page consists of the following files:

- `[pageName].js`: page logic
- `[pageName].axml`: page structure
- `[pageName].acss`: (Optional) page format
- `[pageName].json`: (Optional) page configurations

Provide the data specified in the following code during initialization of a page.

```
Page({
  data: {
    title: 'mPaaS',
    array: [{user: 'li'}, {user: 'zhao'}],
  },
});
```

Render the content on the page based on the preceding data.

```
<view>{{title}}</view>
<view>{{array[0].user}}</view>
```

Specify a response function when you define an interactive behavior.

```
<view onTap="handleTap">click me</view>
```

The preceding code indicates that the `handleTap` method is invoked when a user touches a button on the page.

```
Page({
  handleTap() {
    console.log('yo! view tap!');
  },
});
```

During re-rendering on the page, invoke the `this.setData` method in the script of the page.

```
<view>{{text}}</view>
<button onTap="changeText"> Change normal data </button>
```

The preceding code indicates that the `changeText` method is invoked when a user touches a button on the page.

```
Page({
  data: {
    text: 'init data',
  },
  changeText() {
    this.setData({
      text: 'changed data',
    });
  },
});
```

In the preceding code, invoke the `this.setData` method in the `changeText` method causes the page to be re-rendered.

1.7.8.2. Page configuration

You can use `.json` file in `/pages` directory to configure the window representation of the current page. Page configuration is much easier than global configuration for `app.json`, you can only set related configuration items of `window`, but do not need to write the `window` key. Page configuration items are prior to global configuration items.

The following are supported at the same time:

- You can use `optionMenu` to configure navigation icons, click to trigger `onOptionMenuClick`.

  Note: `optionMenu` configuration will be deprecated, we recommend you to set the navigation icon by using `my.setOptionMenu`.
- You can use `titlePenetrate` to set the click-through of the navigation bar.

The following table describes all the configuration items in the app.json file.

File	Type	Required	Description
optionMenu	Object	No	Base library 1.3.0 supports this configuration item, you can set additional icons for the navigation bar. Currently, you can set the value of the attribute icon to icon URL (starts with https/http) or base64 string, the recommended size is 30*30 pixels.
titlePenetrate	BOOL	No	Client 10.1.52+ supports this configuration item, you can set click through of the navigation bar.

The following code snippet shows a basic example:

```
{
  "optionMenu": {
    "icon": "https://img.alicdn.com/tps/i3/T1OjaVFl4dXXa.JOZB-114-114.png"
  },
  "titlePenetrate": true
}
```

1.7.8.3. Page structure

You can use `.axml` file in `/pages` directory to define the structure of the current page.

The file content follows AXML syntax, which is very similar to HTML, while with some differences. For more information, see AXML.

1.7.8.4. Page style

You can use `.acss` file in `/pages` directory to define page style.

The root element on each page is `page`, use the following methods to set the page height or background color as you need:

```
page {
  background-color: #fff;
}
```

For more information about **acss**, see ACSS syntax reference.

1.7.8.5. Page registration

Page(object: Object)

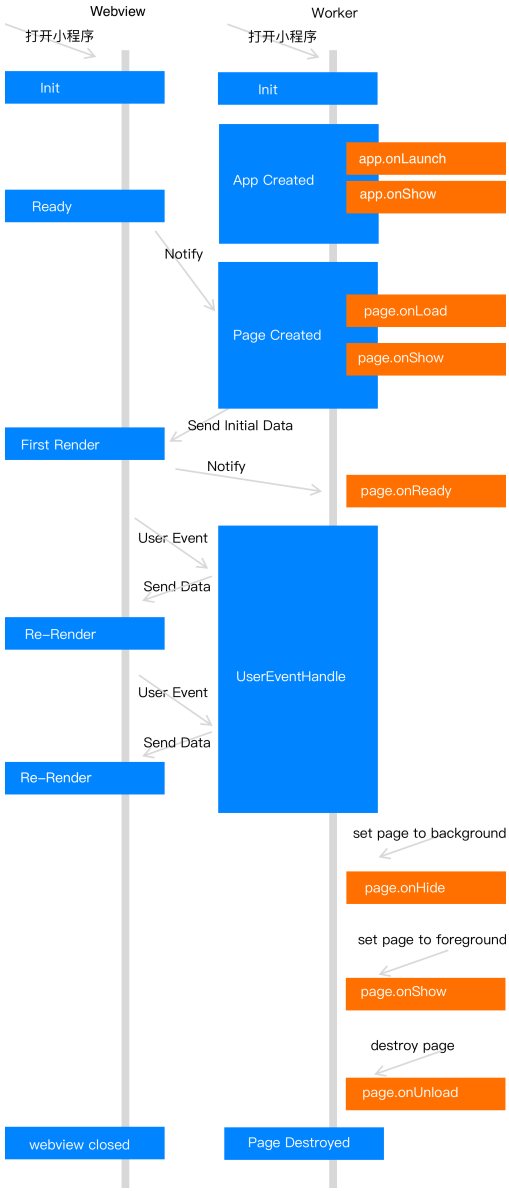
Define `Page()` in `.js` file of `/pages` directory, which is used for registering a Mini Program page and accept an object as an attribute, to specify information such as initial data, life cycle callback, and event processing of the page.

The following code snippet shows a basic page code:

```
// pages/index/index.js
Page({
  data: {
    title: "Alipay",
  },
  onLoad(query) {
    // Page loading
  },
  onShow() {
    // Page display
  },
  onReady() {
    // Page loading completed
  },
  onHide() {
    // Page hiding
  },
  onUnload() {
    // Page is closed
  },
  onTitleClick() {
    // Title is clicked
  },
  onPullDownRefresh() {
    // Page is pulled down
  },
  onReachBottom() {
    // Page reaches the bottom
  },
  onShareAppMessage() {
   // Return custom shared message
  },
  // Event object
  events: {
    onBack() {
      console.log('onBack');
    },
  },
  // Custom event handlers
  viewTap() {
    this.setData({
      text: 'Set data for update.',
    });
  },
  // Custom event handlers
  go() {
    // Page jumps with a parameter. Get type by using the query method of onLoad function from the URL: page/ui/index.
    my.navigateTo({url:'/page/ui/index?type=mini'});
  },
  // Custom data object
  customData: {
    name: 'alipay',
  },
});
```

Page life cycle

The following figure shows the life cycle of Page object.



Mini Program controls management mainly based on Webview and Worker. Webview and Worker run at the same time.

1. After the application service thread has started, it will run `app.onLaunch` and `app.onShow` to complete creation of the app. Then it will run `page.onLoad` and `page.onShow` to complete creation of a Page. At this moment, this thread will wait for the view thread to notify it of the completion of initialization.
2. The view thread notifies the application service thread after completing initialization. The application service thread then sends the initial data to the view thread for rendering. At this moment, the view thread completes rendering of the data for the first time.
3. After the first rendering is complete, the view thread will enter the ready state and notify the application service thread, which will then call the function `page.onReady` and enter the active state.
4. After the application service thread has entered the active state, it will notify the view thread to render the page every time the data is modified.
   - When the page is switched to the background, the application service thread will call the function `page.onHide` and enter the standby state.
   - When the page returns to the foreground, the application service thread will call the function `page.onShow` and enter the active state.
   - After the function for returning to the previous page or redirecting the page is called, the application service thread will call the function `page.onUnload` to destroy the page.

Descriptions of object attributes

Attribute	Type	Description	The minimum version
data	Object | Function	The initial data or the function for returning the initialization data.	
events	Object	The object of the event handling function.	1.13.7
onLoad	Function(query: Object)	Triggered when the page is loading.	

Attribute	Type	Description	The minimum version
onShow	Function	Triggered when the page is being displaying.	
onReady	Function	Triggered when the initial rendering of the page is complete.	
onHide	Function	Triggered when the page is being hidden.	
onUnload	Function	Triggered when the page is being unloaded.	
onShareAppMessage	Function(options: Object)	Triggered when the user shares the page by tapping the upper-right corner.	
onTitleClick	Function	Triggered when the title is tapped.	
onOptionMenuClick	Function	Triggered when the options icon in the navigation bar is tapped.	1.3.0
onPopMenuClick	Function		1.3.0
onPullDownRefresh	Function({from: `manual` | `code`})	Triggered when the page is pulled downward.	
onPullIntercept	Function	Triggered when pulling downward is stopped.	1.11.0
onTabItemTap	Function	Triggered when a `tabItem` is tapped.	1.11.0
onPageScroll	Function({scrollTop})	Triggered when the page is scrolled.	
onReachBottom	Function	Triggered when the bottom of the page is reached by user pulling upward.	
Others	Any	A developer can add to `object` any function or attribute, which can be accessed using `this` within a function of the page.	

The object that stores data of a page - data

You can specify the initial data of a page by setting `data`. When `data` is an object, it is shared by all pages. This means when a user opens page A, returns to the previous page and then opens page A again, data of the previous page will be displayed instead of the initial data. In this case, the problem can be resolved using the following two methods:

- Set `data` to be constant data.

```
Page({
 data: { arr:[] },
 doIt() {
   this.setData({arr: [...this.data.arr, 1]});
 },
});
```

Note: Do not directly modify `this.data`. Doing this will not change the status of the page and will cause inconsistent data.

For example, the following code is wrong:

```
Page({
 data: { arr:[] },
 doIt() {
   this.data.arr.push(1); // Never write code like this!
   this.setData({arr: this.data.arr});
 }
});
```

- Set `data` to be unique data of the page (not recommended):

```
Page({
 data() { return { arr:[] }; },
 doIt() {
   this.setData({arr: [1, 2, 3]});
 },
});
```

Life cycle functions

onLoad(query: Object)

Triggered when the page is loaded. It is called only once by each page. `query` is the query object passed within `my.navigateTo` and `my.redirectTo`.

Attribute	Type	Description	The minimum version
query	Object	The parameter for opening the path of the current page.	

onShow()

Triggered when the page is displayed or switched to the foreground.

onReady()

Triggered when the initial rendering of the page is complete. It is called only once by each page. When it is called, it means the page is ready and can interact with the view layer.

Setting of the page, for example, `my.setNavigationBar`, must be done after `onReady`.

onHide()

Triggered when the page is hidden or switched to the background. For example, it is triggered when `my.navigateTo` is called to go to another page or the user switches to another tab at the bottom of the page.

onUnload()

Triggered when the page is unloaded. For example, it is triggered when a different page is opened by `my.redirectTo` or `my.navigateBack`.

Event handlers for a page

onShareAppMessage(options: Object)

Triggered when the share button in the general menu in the upper-right corner or a share button inside a page is tapped. For more details, see Share.

onTitleClick()

Triggered when the title is tapped.

onOptionMenuClick()

Triggered when the menu in the upper-right corner is tapped.

onPopMenuClick()

Triggered when the general menu in the upper-right corner is tapped.

onPullDownRefresh({from: manual | code})

Triggered when the page is pulled to refresh. This method can be used only when `pullRefresh` is enabled in the `window` attribute in the `app.json` file. For more information, see Set global configurations in app.json. After data is refreshed, the `my.stopPullDownRefresh` operation can be called to stop pull-to-refresh on the current page.

onPullIntercept()

Triggered when the pull-down operation is stopped.

onTabItemTap(object: Object)

Triggered when a `tabItem` is tapped.

Attribute	Type	Description	The minimum version
from	String	The source of the tap operation.	
pagePath	String	The path to the page where the tapped tabItem is located.	
text	String	The text on the button that corresponds to the tapped tabItem.	
index	Number	The serial number of the tapped tabItem. The serial numbers start from 0.	

onPageScroll({scrollTop})

Triggered when the page is scrolled. `The scrollTop` attribute indicates the distance that the page is scrolled.

onReachBottom()

Triggered when the page is pulled up to the top.

events

Note: For brevity of the code, the platform provides a new event processing object named events. Existing events are equivalent to the events function exposed in the page instance. Events objects are supported since base library **1.13.7**.

Event	Type	Description	The minimum version
onBack	Function	Triggered when the system returns to the current page.	1.13.7
onKeyboardHeight	Function	Triggered when the height of the keypad changes.	1.13.7
onOptionMenuClick	Function	Triggered when the menu in the upper-right corner is tapped.	1.13.7

Event	Type	Description	The minimum version
onPopMenuClick	Function	Triggered when the general menu in the upper-right corner is tapped.	1.13.7
onPullIntercept	Function	Triggered when the pull-down operation is stopped.	1.13.7
onPullDownRefresh	Function({from: `manual` | `code`})	Triggered when the page is pulled downward.	1.13.7
onTitleClick	Function	Triggered when the title is tapped.	1.13.7
onTabItemTap	Function	Triggered after a `tabItem` is tapped and the tab is switched to the destination tab.	1.13.7
beforeTabItemTap	Function	Triggered when a `tabItem` is tapped but before the tab is switched to the destination tab.	1.13.7

Code sample:

```
Page({
  data: {
    text: 'This is page data.'
  },
  onLoad(){
    // Sets custom menus.
    my.setCustomPopMenu({
      menus:[
        {name: 'Menu 1', menuIconUrl: 'https://menu1'},
        {name: 'Menu 2', menuIconUrl: 'https://menu2'},
      ],
    })
  },
  events:{
    onBack(){
      // Triggered when the system returns to the current page.
    },
    onKeyboardHeight(e){
      // Triggered when the height of the keypad changes.
      console.log('Keypad height:', e.height)
    },
    onOptionMenuClick(){
      // Triggered when the menu in the upper-right corner is tapped.
    },
    onPopMenuClick(e){
      // Triggered when a custom menu in the general menu in the upper-right corner is tapped.
      console.log('Index of the custom menu tapped by the user', e.index)
      console.log('Name of the custom menu tapped by the user', e.name)
      console.log('menuIconUrl of the custom menu tapped by the user', e.menuIconUrl)
    },
    onPullIntercept(){
      // Triggered when the pull-down operation is stopped.
    },
    onPullDownRefresh(e){
      // Triggered when the page is pulled down. When e.from is set to code, the event is triggered by startPullDownRefresh. When e.from is set to ma
nual, the event is triggered by the pull-down operation performed by the user.
      console.log('Trigger type of pull-to-refresh', e.from)
      my.stopPullDownRefresh()
    },
    onTitleClick(){
      // Triggered when the title is tapped.
    },
    onTabItemTap(e){
      // The e.from event is triggered after a tabItem is tapped and the tab is switched to the destination tab. When e.from is set to user, the even
t is triggered by a tap operation performed by the user. When e.from is set to api, the event is triggered by the switchTab operation called by the u
ser.
      console.log('Trigger type of the tab change)', e.from)
      console.log('Path to the page that corresponds to the tapped tab', e.pagePath)
      console.log('Text on the tapped tab', e.text)
      console.log('Index of the tapped tab', e.index)
    },
    beforeTabItemTap(){
      // Triggered when a tabItem is tapped but before the tab is switched to the destination tab.
    },
  }
})
```

Page.prototype.setData(data: Object, callback: Function)

`The setData` method sends data from the logical layer to the view layer and changes the value of the `this.data` object.

The parameters are described as follows:

Event	Type	Description	The minimum version
data	Object	The data that needs to be changed.	
callback	Function	A callback function. The callback function is executed after page rendering is complete.	The compatibility issue needs to be handled by using the `my.canIUse('page.setData.callback')` operation. For more information, see Introduction of the base library of the Mini Program component.

The value of `Object` is expressed in the `key: value` format. Set the value of `key` in the `this.data` object to the value of `value`. In particular, the `key` can be exported as a data path, such as `array[2].message` and `a.b.c.d`. You do not need to predefine keys in the `this.data` object.

Be aware of the following when you use the setData method:

- Do not directly modify the `this.data` object. Otherwise, the status of the page cannot be changed, and data inconsistency will occur.
- Only data that can be converted to the JSON format is supported.
- Do not perform the operation on an excessive volume of data.
- Do not set one or multiple attribute values of the data parameter to undefined. Otherwise, the attribute is ignored, which leads to some potential problems.

Code sample:

```
<view>{{text}}</view>
<button onTap="changeTitle"> Change normal data </button>
<view>{{array[0].text}}</view>
<button onTap="changeArray"> Change Array data </button>
<view>{{object.text}}</view>
<button onTap="changePlanetColor"> Change Object data </button>
<view>{{newField.text}}</view>
<button onTap="addNewKey"> Add new data </button>
<view>hello: {{name}}</view>
<button onTap="changeName"> Chane name </button>
```

```
Page({
  data: {
    text: 'test',
    array: [{text: 'a'}],
    object: {
      text: 'blue',
    },
    name: 'taobao',
  },
  changeTitle() {
    // Wrong setting. Do not directly modify the data indicated by the data parameter.
    // this.data.text = 'changed data'

    // Right setting.
    this.setData({
      text: 'ha',
    });
  },
  changeArray() {
    // You can modify data by using the path to the data.
    this.setData({
      'array[0].text': 'b',
    });
  },
  changePlanetColor(){
    this.setData({
      'object.text': 'red',
    });
  },
  addNewKey() {
    this.setData({
      'newField.text': 'c',
    });
  },
  changeName() {
    this.setData({
      name: 'alipay',
    }, () => { // Accepts transferring the callback function.
      console.log(this); // This parameter indicates the current page instance.
      this.setData({ name: this.data.name + ', ' + 'welcome!'});
    });
  },
});
```

Page.prototype.$spliceData(data: Object, callback: Function)

Note: `$spliceData` is supported since base library 1.7.2. The compatibility issue can be handled by calling the **my.canIUse('page.\$spliceData')** operation. For more information, see Introduction of the base library of the Mini Program component.

`The spliceData` method is also used to send data from the logical layer to the view layer. Different from the `setData` method, this data has higher performance in processing long lists.

The parameters are described as follows:

Event	Type	Description	The minimum version
data	Object	The data that needs to be changed.	
callback	Function	A callback function. The callback function is executed after page rendering is complete.	

Object is expressed in the `key: value` format. Set the value of `key` in the `this.data` object to the value of `value`. The following content describes the parameters of the object:

- `key` parameter can be a data path, such as `array[2].message` and `a.b.c.d`. You do not have to predefine this parameter in the `this.data` object.
- `value` parameter is an array in the format [start, deleteCount, ...items]). In the array, the first element is the start position of the operation, the second element is the number of deleted elements, and the remaining elements are inserted data. This parameter corresponds to the `splice` method in the array of `es5`.

Code sample:

```
<!-- pages/index/index.axml -->
<view class="spliceData">
  <view a:for="{{a.b}}" key="{{item}}" style="border:1px solid red">
    {{item}}
  </view>
</view>
```

```
// pages/index/index.js
Page({
  data: {
    a: {
      b: [1,2,3,4],
    },
  },
  onLoad(){
    this.$spliceData({ 'a.b': [1, 0, 5, 6] });
  },
});
```

Output on the page:

```
1
5
6
2
3
4
```

Page.prototype.$batchedUpdates(callback: Function)

Batch update the specified data.

Note: The `$spliceData` method is supported since base library 1.14.0. The compatibility issue can be handled by calling the **my.canIUse('page.\$batchedUpdates')** operation. For more information, see Introduction of the base library of the Mini Program component.

The parameters are described as follows:

Event	Type	Description	The minimum version
callback	Function	A callback function. The data involved in the callback function will be batch updated.	

Code sample:

```
// pages/index/index.js
Page({
  data: {
    counter: 0,
  },
  plus() {
    setTimeout(() => {
      this.$batchedUpdates(() => {
        this.setData({
          counter: this.data.counter + 1,
        });
        this.setData({
          counter: this.data.counter + 1,
        });
      });
    }, 200);
  },
});
```

```
<!-- pages/index/index.axml -->
<view>{{counter}}</view>
<button onTap="plus">+2</button>
```

In the code above:

- The value of `counter` on the page increases by 2 each time a button is tapped on the page.
- The `setData` method is placed in `this.$batchedUpdates`. This ensures that the data is transmitted only once, regardless of how many times the `setData` method is invoked.

1.7.8.6. getcurrentpages method

`Use getCurrentPages()` method to obtain instances of the current page stack and return the page array stack. The first element is the homepage, the last element is the current page.

The framework maintains all the current pages in the form of stacks. The following table shows the relationship between routing switch and page stack:

Routing mode	Page stack performance
Initialize	Push a new page
Open a new page	Push a new page
Redirect a page	Pop the current page, push a new page
Return to a page	Pop the current page
Switch Tab	Pop all pages, only leave the new Tab page

You can use the following code snippet to detect whether the current page stack has a five-level page depth.

```
if(getCurrentPages().length === 5) {
  my.redirectTo('/pages/logs/logs');
} else {
  my.navigateTo('/pages/index/index');
}
```

Note: Do not try to change the page stack, otherwise, it will cause routing and page status errors.

1.7.9. AXML

1.7.9.1. AXML introduction

AXML is a set of tag language based on the Mini Program framework to describe the structure of Mini Program pages. AXML syntax can be divided into five parts:

AXML is a set of tag language based on the Mini Program framework to describe the structure of Mini Program pages. AXML syntax can be divided into five parts:

- Data binding
- Conditional rendering
- List rendering
- Template
- Referencing

AXML code example:

```
<!-- pages/index/index.axml -->
<view a:for="{{items}}"> {{item}} </view>
<view a:if="{{view == 'WEBVIEW'}}"> WEBVIEW </view>
<view a:elif="{{view == 'APP'}}"> APP </view>
<view a:else> alipay </view>
<view onTap="add"> {{count}} </view>
```

Example of the corresponding `.js` file:

```
// pages/index/index.js
Page({
  data: {
    items: [1, 2, 3, 4, 5, 6, 7],
    view: 'alipay',
    count: 1,
  },
  add(e) {
    this.setData({
      count: this.data.count + 1,
    });
  },
});
```

1.7.9.2. data binding

The dynamic data in AXML is bound to the `data` content on the corresponding `Page`.

Simple binding

Data binding wraps variables with two pairs of braces (`{{}}`) based on Mustache and can be applied in various syntax scenarios.

The content of the bubble window

```
<view> {{ message }} </view>
```

```
Page({
  data: {
    message: 'Hello alipay!',
  },
});
```

Component attribute

Component attributes must be enclosed in double quotation marks (`""`).

```
<view id="item-{{id}}"> </view>
```

```
Page({
  data: {
    id: 0,
  },
});
```

Controlled attribute

Controlled attributes must be enclosed in double quotation marks (`""`).

```
<view a:if="{{condition}}"> </view>
```

```
Page({
  data: {
    condition: true,
  },
});
```

Keywords

Keywords need to be enclosed in double quotation marks (`""`).

- true: boolean-type true, represents a true value.
- false: boolean-type false, represents a false value.

```
<checkbox checked="{{false}}"> </checkbox>
```

Note: Do not directly write `checked="false"` , the computing result is a string, and represents a true value when converted to a boolean type.

Operation

Wrap easy operations with two pairs of braces (`{{}}`).The following operation methods are supported.

Ternary operation

```
<view hidden="{{flag ? true : false}}"> Hidden </view>
```

Arithmetic operation

```
<view> {{a + b}} + {{c}} + d </view>
```

```
Page({
  data: {
    a: 1,
    b: 2,
    c: 3,
  },
});
```

The page output is `3 + 3 + d` .

Logical judgment

```
<view a:if="{{length > 5}}"> </view>
```

String operation

```
<view>{{"hello" + name}}</view>
```

```
Page({
  data:{
    name: 'alipay',
  },
});
```

Data path operation

```
<view>{{object.key}} {{array[0]}}</view>
```

```
Page({
  data: {
    object: {
      key: 'Hello ',
    },
    array: ['alipay'],
  },
});
```

Combination

You can directly implement combinations in Mustache syntax to build new objects or arrays.

Array

```
<view a:for="{{[zero, 1, 2, 3, 4]}}"> {{item}} </view>
```

```
Page({
  data: {
    zero: 0,
  },
});
```

The end result is the array `[0, 1, 2, 3, 4]`.

Object

```
<template is="objectCombine" data="{{foo: a, bar: b}}"></template>
```

```
Page({
  data: {
    a: 1,
    b: 2,
  },
});
```

The end result is the object `{foo: 1, bar: 2}`.

You can also use destructuring operator `...` to extend an object:

```
<template is="objectCombine" data="{{...obj1, ...obj2, e: 5}}"></template>
```

```
Page({
  data: {
    obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      c: 3,
      d: 4,
    },
  },
});
```

The end result is the object `{a: 1, b: 2, c: 3, d: 4, e: 5}`.

If the key and value of the object are identical, this can be indirectly expressed.

```
<template is="objectCombine" data="{{foo, bar}}"></template>
```

```
Page({
  data: {
    foo: 'my-foo',
    bar: 'my-bar',
  },
});
```

The end result is the object `{foo: 'my-foo', bar:'my-bar'}`.

The preceding methods can be freely combined. However, when variable names are identical, the latter variable will overwrite the former variable, for example:

```
<template is="objectCombine" data="{{...obj1, ...obj2, a, c: 6}}"></template>
```

```
Page({
  data: {
    obj1: {
      a: 1,
      b: 2,
    },
    obj2: {
      b: 3,
      c: 4,
    },
    a: 5,
  },
});
```

The end result is the object `{a: 5, b: 3, c: 6}`.

1.7.9.3. Conditional rendering

a:if

Use `a:if="{{condition}}"` to determine whether the code block needs to be rendered in the framework.

```
<view a:if="{{condition}}"> True </view>
```

You can also use `a:elif` and `a:else` to add a **else** block.

```
<view a:if="{{length > 5}}"> 1 </view>
<view a:elif="{{length > 2}}"> 2 </view>
<view a:else> 3 </view>
```

block a:if

`a:if` is a controlled attribute, therefore, you need to use this attribute in tags. If you want to determine multiple component tags at once, you can wrap multiple components `<block/>` tags and use `a:if` to control attributes.

```
<block a:if="{{true}}">
  <view> view1 </view>
  <view> view2 </view>
</block>
```

Note: `<block/>` is not a component but just a wrapper element. This element does not do rendering on the page and only accepts controlled attributes.

Compare a:if with hidden

- Templates in `a:if` may contain data bindings. Therefore, when the condition value of `a:if` is switched, the framework has local rendering procedures for ensuring the condition block is destroyed or re-rendered during switching. In addition, `a:if` does not trigger rendering action when the initial rendering condition is false. Local rendering starts only when the condition value switches to true for the first time.
- `hidden` controls display and hiding, and components are rendered all the time.

In general, `a:if` has higher switching consumption while `hidden` has higher initial rendering consumption. Therefore, in scenarios that require frequent switching, `hidden` is better. If the condition value is not frequently switched at runtime, `a:if` is better.

1.7.9.4. List rendering

a:for

Use `a:for` attribute on a component to bind an array and use data of each item in the array to re-render this component.

The subscript variable name of the current item in the array is `index` by default, the variable name of the current item in the array is `item` by default.

```
<view a:for="{{array}}">
  {{index}}: {{item.message}}
</view>
```

```
Page({
  data: {
    array: [{
      message: 'foo',
    }, {
      message: 'bar',
    }],
  },
});
```

Use `a:for-item` to specify the variable name of the current element in the array. Use `a:for-index` to specify the variable name of the current subscript in the array.

```
<view a:for="{{array}}" a:for-index="idx" a:for-item="itemName">
  {{idx}}: {{itemName.message}}
</view>
```

`a:for` supports embedding.The following is the embedded sample code for Chinese multiplication table.

```
<view a:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" a:for-item="i">
  <view a:for="{{[1, 2, 3, 4, 5, 6, 7, 8, 9]}}" a:for-item="j">
    <view a:if="{{i <= j}}">
      {{i}} * {{j}} = {{i * j}}
    </view>
  </view>
</view>
```

block a:for

Similar to `block a:if` , you can use `a:for` on the `<block/>` tag to render a structural block that contains multiple nodes.

```
<block a:for="{{[1, 2, 3]}}">
  <view> {{index}}: </view>
  <view> {{item}} </view>
</block>
```

a:key

If positions of list items dynamically change or a new item is added to the list, and you want to retain features and statuses of list items (for example, the input content in `<input/>` and the selected status of `<switch/>`), you need to use `a:key` to specify a unique identifier for each list item.

The value of `a:key` are provided in two forms:

- String: represents an attribute of the list item, the attribute value needs to be a unique string or number in the list, such as ID, and cannot be dynamically changed.
- The reserved word `*this` represents the list item itself and is a unique string or number. For example, when data change triggers a re-render, components with `key` will be corrected, the framework will ensure that the list items are reordered rather than recreated. This ensures components retain their statuses and improves the efficiency of list rendering.

Notes:

- If `a:key` is not provided, an error will occur.
- If you know that the list is static, or you do not need to pay attention to the list order, then `a:key` can be ignored.

Code sample:

```
<view class="container">
  <view a:for="{{list}}" a:key="*this">
    <view onTap="bringToFront" data-value="{{item}}">
    {{item}}: click to bring to front
    </view>
  </view>
</view>
```

```
Page({
  data:{
    list:['1', '2', '3', '4'],
  },
  bringToFront(e) {
    const { value } = e.target.dataset;
    const list = this.data.list.concat();
    const index = list.indexOf(value);
    if (index !== -1) {
      list.splice(index, 1);
      list.unshift(value);
      this.setData({ list });
    }
  },
});
```

key

`key` is a more common style of writing compared with `a:key`, which can be filled with arbitrary expressions or strings.

Note: `key` cannot be set on a block.

Code sample:

```
<view class="container">
  <view a:for="{{list}}" key="{{item}}">
    <view onTap="bringToFront" data-value="{{item}}">
    {{item}}: click to bring to front
    </view>
  </view>
</view>
```

```
Page({
  data:{
    list:['1', '2', '3', '4'],
  },
  bringToFront(e) {
    const { value } = e.target.dataset;
    const list = this.data.list.concat();
    const index = list.indexOf(value);
    if (index !== -1) {
      list.splice(index, 1);
      list.unshift(value);
      this.setData({ list });
    }
  },
});
```

At the same time, you can use `key` to avoid using a component repeatedly, for example, you can allow different types of data:

```
<input a:if="{{name}}" placeholder="Enter your name" />
<input a:else placeholder="Enter your email address" />
```

When you input the name value and switch to email, the current input value will be retained. If you do not want to retain the input value, you can add `key` .

```
<input key="name" a:if="{{name}}" placeholder="Enter your name" />
<input key="email" a:else placeholder="Enter your email address" />
```

1.7.9.5. Template

AXML provides `templates` , you can define code snippets in a template, and call them in different places.

Note: we recommend you to use the `template` method to introduce the template fragment because `template` specifies its scope and only uses the data passed by `data` . If the `data` of the `template` does not change, the fragment UI will not be re-rendered.

Define a template

Use the name attribute to specify the template name, and define a code snippet in `<template/>` .

```
<!--
  index: int
  msg: string
  time: string
-->
<template name="msgItem">
  <view>
    <text> {{index}}: {{msg}} </text>
    <text> Time: {{time}} </text>
  </view>
</template>
```

Use a template

Use the `is` attribute to declare the required template and pass the required `data` .

```
<template is="msgItem" data="{{...item}}"/>
```

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-19',
    },
  },
});
```

Use the `is` attribute to dynamically determine the specific template to be rendered based on Mustache syntax.

```
<template name="odd">
  <view> odd </view>
</template>
<template name="even">
  <view> even </view>
</template>

<block a:for="{{[1, 2, 3, 4, 5]}}">
    <template is="{{item % 2 == 0 ? 'even' : 'odd'}}"/>
</block>
```

Template scope

A template has its scope and can only use the data passed by `data` . In addition, you can also bind page logic to process functions by calling `onXx` events. As shown in the following code snippet:

```
<!-- templ.axml -->
<template name="msgItem">
    <view>
        <view>
            <text> {{index}}: {{msg}} </text>
            <text> Time: {{time}} </text>
        </view>
        <button onTap="onClickButton">onTap</button>
    </view>
</template>
```

```
<!-- index.axml -->
<import src="./templ.axml"/>
<template is="msgItem" data="{{...item}}"/>
```

```
Page({
  data: {
    item: {
      index: 0,
      msg: 'this is a template',
      time: '2019-04-22'
    }
  },
  onClickButton(e) {
    console.log('button clicked', e)
  },
});
```

1.7.9.6. Reference

AXML provides two reference methods: `import` and `include`.

import

`import` can load a defined template.

For example: a template named `item` is defined in `item.axml`.

```
<!-- item.axml -->
<template name="item">
  <text>{{text}}</text>
</template>
```

You can use the `item` template by importing `item.axml` in `index.axml`.

```
<import src="./item.axml"/>
<template is="item" data="{{text: 'forbar'}}"/>
```

The scope of `import` is limited. It only imports the template that is defined in the target file, and the template being imported in the target file will not be imported.

For example, C imports B, B imports A, then the template defined in B can be used in C, and the template defined in A can be used in B. However, C cannot use the template defined in A.

```
<!-- a.axml -->
<template name="A">
  <text> A template </text>
</template>
```

```
<!-- b.axml -->
<import src="./a.axml"/>
<template name="B">
  <text> B template </text>
</template>
```

```
<!-- c.axml -->
<import src="./b.axml"/>
<template is="A"/>  <! -- Note: You cannot use import A -->
<template is="B"/>
```

A template can only have one child node, for example:

Correct example:

```
<template name="x">
  <view />
</template>
```

Error example:

```
<template name="x">
  <view />
  <view />
</template>
```

include

`include` can introduce the entire code of the target file excluding `<template/>`, which is equivalent to copying the code to the location of `include`.

Code sample:

```
<!-- index.axml -->
<include src="./header.axml"/>
<view> body </view>
<include src="./footer.axml"/>
```

```
<!-- header.axml -->
<view> header </view>
```

```
<!-- footer.axml -->
<view> footer </view>
```

Introduction path

The template introduction path supports relative path and absolute path. You can also load third-party modules from the `node_modules` directory.

```
<import src="./a.axml"/> <! -- Relative path -->
<import src="/a.axml"/> <! -- Project absolute path -->
<import src="third-party/x.axml"/> <! -- Third-party npm package path -->
```

1.7.10. SJS syntax reference

1.7.10.1. SJS introduction

SJS (safe/subset javascript) is a set of custom scripting language for Mini Program and can be used in AXML to build page structure.

SJS (safe/subset javascript) is a set of custom scripting language for Mini Program and can be used in AXML to build page structure.

SJS is a subset of JavaScript language but different from JavaScript. Do not equate SJS with JavaScript, because their syntax are different.

Usage

Define SJS in `.sjs` file:

```
// pages/index/index.sjs
const message = 'hello alipay';
const getMsg = x => x;
export default {
  message,
  getMsg,
};
```

```
// pages/index/index.js
Page({
  data: {
    msg: 'hello taobao',
  },
});
```

```
<!-- pages/index/index.axml -->
<import-sjs name="m1" from="./index.sjs"/>
<view>{{m1.message}}</view>
<view>{{m1.getMsg(msg)}}</view>
```

Output on the page:

```
hello alipay
hello taobao
```

Note:

- SJS can only be defined in `.sjs` files. Use `<import-sjs>` tag in AXML to import SJS.
- SJS can call functions defined in other `.sjs` files.
- SJS is a subset of JavaScript language, do not equate SJS with JavaScript.
- The SJS runtime environment is isolated from other JavaScript codes. Therefore, SJS cannot call functions defined in other JavaScript files nor APIs provided by Mini Program.
- SJS functions cannot be used as component event callbacks.
- SJS does not depend on the base library version and can run in Mini Program of all versions.

import-sjs tag

Attribute	Type	Mandatory	Description
name	String	Yes	The module name of the current `<import-sjs>` tag.
from	String	Yes	The relative path of the imported .sjs file.

Note:

- The name attribute specifies the module name of the current `<import-sjs>` tag. You are recommended to set a unique name value in an individual AXML file. If module names are identical, they are overwritten in sequence (the latter overwrites the former). The `<import-sjs>` module names in different AXML files do not overwrite each other.
- The value of the name attribute can be a string to represent the default module name, or a `{x}` to represent the export of the named module.

Code sample:

```
// pages/index/index.js
Page({
  data: {
    msg: 'hello alipay',
  },
});
```

```
// pages/index/index.sjs
function bar(prefix) {
  return prefix;
}
export default {
  foo: 'foo',
  bar: bar,
};
```

```
// pages/index/namedExport.sjs
export const x = 3;
export const y = 4;
```

```
<!-- pages/index/index.axml -->
<import-sjs from="./index.sjs" name="test"></import-sjs>
<! -- You can also use one closing tag.
<import-sjs from="./index.sjs" name="test" />
-->

<! -- Call the bar function in the test module, and the argument is foo in the test module.
<view> {{test.bar(test.foo)}} </view>
<! -- Call the bar function in the test module, and the argument is msg in the page/index/index.js
.
<view> {{test.bar(msg)}} </view>

<! -- Support named export -->
<import-sjs from="./namedExport.sjs" name="{x, y: z}" />
<view>{{x}}</view>
<view>{{z}}</view>
```

Output on the page:

```
foo
hello alipay
3
4
```

Note:

- Be sure to use `.sjs` file suffix when you import an `.sjs` file.
- If an `.sjs` module is defined but not imported, the module will not be parsed or run.

1.7.10.2. Variable

All variables in SJS are references to values.

All variables in SJS are references to values.

Syntax rule

- Supports var (variable) hoisting as in JavaScript.
- Supports `const` and `let` as in JavaScript.
- Undeclared variables are directly assigned and used, and defined as global variables.

- If a variable is declared but not assigned, its default value is `undefined`.

```
var num = 1;
var str = "hello alipay";
var undef; // undef === undefined
const n = 2;
let s = 'string';
globalVar = 3;
```

Variable name

Naming conventions

The naming of variables must conform to the following two rules:

- The first character must be a letter (a-z, A-Z) or an underline (_).
- Characters other than the first character can be letters (a-z, A-Z), underlines (_), or numbers (0-9).

Reserved identifiers

Consistent with JavaScript syntax rules, the following identifiers cannot be used as variable names:

```
arguments
break
case
continue
default
delete
do
else
false
for
function
if
Infinity
NaN
null
require
return
switch
this
true
typeof
undefined
var
void
while
```

1.7.10.3. Comments

You can write comments on your SJS code the same way as you do on JavaScript code, as shown in the following code snippet:

You can write comments on your SJS code the same way as you do on JavaScript code, as shown in the following code snippet:

```
// page.sjs
// Method 1: This is a single-line comment.
/*
Method 2: This is a multiple-line comment.
All the content in between is part of the comment.
*/
let h = 'hello';
const w = ' alipay';
```

1.7.10.4. Operator

Arithmetic operator

```
var a = 10, b = 20;
// Addition operation
console.log(30 === a + b);
// Substraction operation
console.log(-10 === a - b);
// Multiplication operation
console.log(200 === a * b);
// Division operation
console.log(0.5 === a / b);
// Complementation operation
console.log(10 === a % b);
```

The addition `+<` /> operator can be used for string concatenation.

```
var a = 'hello', b = ' alipay';
// String concatenation
console.log('hello alipay' === a + b);
```

Comparison operator

```
var a = 10, b = 20;

// Less than
console.log(true === (a < b));
// More than
console.log(false === (a > b));
// Less than or equal to
console.log(true === (a <= b));
// More than or equal to
console.log(false === (a >= b));
// Equal sign
console.log(false === (a == b));
// Not equal sign
console.log(true === (a != b));
// Equivalent sign
console.log(false === (a === b));
// Not equivalent sign
console.log(true === (a !== b));
```

Binary logical operator

```
var a = 10, b = 20;
// AND
console.log(20 === (a && b));
// OR
console.log(10 === (a || b));
// NOT, negation operation
console.log(false === !a);
```

Bitwise operator

```
var a = 10, b = 20;

// Left shift operation
console.log(80 === (a << 3));
// Unsigned right shift operation
console.log(2 === (a >> 2));
// Signed right shift operation
console.log(2 === (a >>> 2));
// AND operation
console.log(2 === (a & 3));
// XOR operation
console.log(9 === (a ^ 3));
// OR operation
console.log(11 === (a | 3));
```

Assignment operator

```
var a = 10;
a = 10; a *= 10;
console.log(100 === a);
a = 10; a /= 5;
console.log(2 === a);
a = 10; a %= 7;
console.log(3 === a);
a = 10; a += 5;
console.log(15 === a);
a = 10; a -= 11;
console.log(-1 === a);
a = 10; a <<= 10;
console.log(10240 === a);
a = 10; a >>= 2;
console.log(2 === a);
a = 10; a >>>= 2;
console.log(2 === a);
a = 10; a &= 3;
console.log(2 === a);
a = 10; a ^= 3;
console.log(9 === a);
a = 10; a |= 3;
console.log(11 === a);
```

Unary operator

```
var a = 10, b = 20;
// Auto increment operation
console.log(10 === a++);
console.log(12 === ++a);
// Auto decrement operation
console.log(12 === a--);
console.log(10 === --a);
// Positive value operation
console.log(10 === +a);
// Negative value operation
console.log(0-10 === -a);
// Not operation
console.log(-11 === ~a);
// Negation operation
console.log(false === !a);
// Delete operation
console.log(true === delete a.fake);
// Void operation
console.log(undefined === void a);
// Typeof operation
console.log("number" === typeof a);
```

Ternary operator

```
var a = 10, b = 20;
// Conditional operator
console.log(20 === (a >= 10 ? a + 10 : b + 10));
```

Comma operator

```
var a = 10, b = 20;
// Comma operator
console.log(20 === (a, b));
```

Operator precedence

SJS operators have the same precedence as Javascript operators.

1.7.10.5. Statement

if statement

In `.js` files, you can use `if` statements in the following formats:

- `if (expression) statement` : when `expression` is truthy, execute `statement` .
- `if (expression) statement1 else statement2` : when `expression` is truthy, execute `statement1` . Otherwise, execute `statement2` .
- `if ... else if ... else statementN` With this format of if statement, a statement among `statement1` ~ `statementN` can be selected to execute.

Syntax example:

```
// if ...
if (expression) statement;

if (expression)
  statement;

if (expression) {
  code block;
}


// if ... else
if (expression) statement;
else statement;

if (expression)
  statement;
else
  statement;

if (expression) {
  code block;
} else {
  code block;
}

// if ... else if ... else ...
if (expression) {
  code block;
} else if (expression) {
  code block;
} else if (expression) {
  code block;
} else {
  code block;
}
```

switch statement

- The `default` branch can be omitted.
- Only `variable`, `number`, and `string` can follow the `case` keyword.

Syntax example:

```
switch (expression) {
  case variable:
    statement;
  case number:
    statement;
    break;
  case string:
    statement;
  default:
    statement;
}
```

Code sample:

```
var exp = 10;

switch ( exp ) {
case "10":
  console.log("string 10");
  break;
case 10:
  console.log("number 10");
  break;
case exp:
  console.log("var exp");
  break;
default:
  console.log("default");
}
```

Output:

```
number 10
```

for statement

Supports the use of `break` and `continue` keywords.

Syntax example:

```
for (statement; statement; statement)
  statement;

for (statement; statement; statement) {
  code block;
}
```

Code sample:

```
for (var i = 0; i < 3; ++i) {
  console.log(i);
  if( i >= 1) break;
}
```

Output:

```
0
1
```

while statement

- When `expression` is true, loop through the `statement` or `code block`.
- Supports the use of `break` and `continue` keywords.

Syntax example:

```
while (expression)
  statement;

while (expression){
  code block;
}

do {
  code block;
} while (expression)
```

1.7.10.6. Data type

SJS supports the following data types:

SJS supports the following data types:

- **String**: String
- **Boolean**: Boolean
- **Number**: Number
- **Object**: Object
- **Function**: Function
- **Array**: Array
- **Date**: Date
- **Regexp**: Regular expression

Obtain the data type

SJS provides `constructor` and `typeof` for you to obtain the data type.

constructor

```
const number = 10;
console.log(number.constructor); // "Number"
const string = "str";
console.log(string.constructor); // "String"
const boolean = true;
console.log(boolean.constructor); // "Boolean"
const object = {};
console.log(object.constructor); // "Object"
const func = function(){};
console.log(func.constructor); // "Function"
const array = [];
console.log(array.constructor); // "Array"
const date = getDate();
console.log(date.constructor); // "Date"
const regexp = getRegExp();
console.log(regexp.constructor); // "RegExp"
```

typeof

```
const num = 100;
const bool = false;
const obj = {};
const func = function(){};
const array = [];
const date = getDate();
const regexp = getRegExp();
console.log(typeof num); // 'number'
console.log(typeof bool); // 'boolean'
console.log(typeof obj); // 'object'
console.log(typeof func); // 'function'
console.log(typeof array); // 'object'
console.log(typeof date); // 'object'
console.log(typeof regexp); // 'object'
console.log(typeof undefined); // 'undefined'
console.log(typeof null); // 'object'
```

String

Syntax

```
'hello alipay';
"hello taobao";
```

ES6 syntax

```
// A string template
const a = 'hello';
const str = `${a} alipay`;
```

Attribute

- `constructor`: returns `"String"`.
- `length`

> **Note** For definitions of other attributes than `constructor`, see the ES6 specification.

Method

- toString
- valueOf
- charAt
- charCodeAt
- concat
- indexOf
- lastIndexOf
- localeCompare
- match
- replace
- search
- slice
- split
- substring
- toLowerCase
- toLocaleLowerCase
- toUpperCase
- toLocaleUpperCase
- trim

> **Note** For information about how to use these global attributes, see the ES6 specification.

Number

Syntax

```
const num = 10;
const PI = 3.141592653589793;
```

Attribute

`constructor`: returns `"Number"`.

Method

- toString
- toLocaleString

- valueOf
- toFixed
- toExponential
- toPrecision

> **Note** For information about how to use these global attributes, see the ES6 specification.

Boolean

A boolean value is either `true` or `false`.

Syntax

```
const a = true;
```

Attribute

- `constructor`: returns `"Boolean"`.

Method

- toString
- valueOf

> **Note** For information about how to use these global attributes, see the ES6 specification.

Object

Syntax

```
var o = {}; // Generate an empty object.
// Generate an object that is not empty.
o = {
  'str': "str", // The key of the object can be a string.
  constVar: 2, // The key of the object can also be an identifier that complies with variable definition rules.
  val: {}, // The value of the object can be of any data type.
};
// Read the attributes of the object.
console.log(1 === o['string']);
console.log(2 === o.constVar);
// Write the attributes of the object.
o['string']++;
o['string'] += 10;
o.constVar++;
o.constVar += 10;
// Read the attributes of the object.
console.log(12 === o['string']);
console.log(13 === o.constVar);
```

ES6 syntax:

```
// ES6 syntax is supported.
let a = 2;
o = {
  a, // The attributes of the object.
  b() {}, // The method of the object.
};
const { a, b, c: d, e = 'default'} = {a: 1, b: 2, c: 3}; // Destructure an object and assign default values.
const {a, ...other} = {a: 1, b: 2, c: 3}; // Destructure an object and assign values.
const f = {...others}; // Destructure an object.
```

Attribute

`constructor`: returns `"Object"`.

```
console.log("Object" === {a:2,b:"5"}.constructor);
```

Method

toString: returns the string `"[object Object]"`.

Function

Syntax

```
// Method 1: Declare a function
function a (x) {
  return x;
}
// Method 2: Specify an expression for the function
var b = function (x) {
  return x;
};
// Method 3: Create an arrow function
const double = x => x * 2;
function f(x = 2){} // Default function parameters.
function g({name: n = 'xiaoming', ...other} = {}) {} // Destructure function parameters and assign values.
function h([a, b] = []) {} // Destructure function parameters and assign values.
// Create an anonymous function and a closure
var c = function (x) {
  return function () { return x;}
};
var d = c(25);
console.log(25 === d());
```

You can use the `arguments` keyword in a function.

```
var a = function(){
    console.log(2 === arguments.length);
    console.log(1 === arguments[0]);
    console.log(2 === arguments[1]);
};
a(1,2);
```

Output:

```
true
true
true
```

Attribute

- `constructor`: returns `"Function"`.
- `length`: returns the number of formal parameters in a function.

Method

toString: returns a string in the `"[function Function]"` format.

Example

```
var f = function (a,b) { }
console.log("Function" === f.constructor);
console.log("[function Function]" === f.toString());
console.log(2 === f.length);
```

Output:

```
true
true
true
```

Array

Syntax

```
var a = []; // An empty array.
a = [5,"5",{},function(){}]; // An array that is not empty. The elements in the array can be of any data type.
const [b, , c, d = 5] = [1,2,3]; // Destructure an array and assign default values.
const [e, ...other] = [1,2,3]; // Destructure an array and assign values.
const f = [...other]; // Destructure an array.
```

Attribute

- `constructor`: returns `"Array"`.
- `length`

> **Note** For definitions of other attributes than `constructor`, see the ES6 specification.

Method

- toString
- concat
- join
- pop
- push

- reverse
- shift
- slice
- sort
- splice
- unshift
- indexOf
- lastIndexOf
- every
- some
- forEach
- map
- filter
- reduce
- reduceRight

> **Note** For information about how to use these global attributes, see the ES6 specification.

Date

Syntax

To generate a date object, use `getDate` function. The function returns a date object whose value is the current time.

```
getDate()
getDate(milliseconds)
getDate(datestring)
getDate(year, month[, date[, hours[, minutes[, seconds[, milliseconds]]]]])
```

Field

- `milliseconds` : returns the number of milliseconds elapsed since January 1, 1970 00:00:00 UTC.
- `datestring` : returns a string in the `month day, year hours:minutes:seconds` format.

Attribute

`constructor` : returns `"Date"` .

Method

- toString
- toDateString
- toTimeString
- toLocaleString
- toLocaleDateString
- toLocaleTimeString
- valueOf
- getTime
- getFullYear
- getUTCFullYear
- getMonth
- getUTCMonth
- getDate
- getUTCDate
- getDay
- getUTCDay
- getHours
- getUTCHours
- getMinutes
- getUTCMinutes
- getSeconds
- getUTCSeconds
- getMilliseconds
- getUTCMilliseconds
- getTimezoneOffset
- setTime
- setMilliseconds
- setUTCMilliseconds
- setSeconds
- setUTCSeconds
- setMinutes

- setUTCMinutes
- setHours
- setUTCHours
- setDate
- setUTCDate
- setMonth
- setUTCMonth
- setFullYear
- setUTCFullYear
- toUTCString
- toISOString
- toJSON

> **Note** For information about how to use these global attributes, see the ES6 specification.

Example

```
let date = getDate(); // Return a date object whose value is the current time.
date = getDate(1500000000000);
// Fri Jul 14 2017 10:40:00 GMT+0800 (China Standard Time)
date = getDate('2016-6-29');
// Fri June 29 2016 00:00:00 GMT+0800 (China Standard Time)
date = getDate(2017, 6, 14, 10, 40, 0, 0);
// Fri Jul 14 2017 10:40:00 GMT+0800 (China Standard Time)
```

Regexp

Syntax

Call `getRegExp` function to generate a regexp object.

```
getRegExp(pattern[, flags])
```

Field

- pattern: content of regular expression.
- flags: modifying signs. The flags can only contain the following characters:
  - `g` : global
  - `i` : ignoreCase
  - `m` : multiline

Attribute

- constructor: returns the string `"RegExp"` .
- global
- ignoreCase
- lastIndex
- multiline
- source

> **Note** For definitions of other attributes than `constructor` , see the ES6 specification.

Method

- exec
- test
- toString

> **Note** For information about how to use these global attributes, see the ES6 specification.

Example

```
var reg = getRegExp("name", "img");
console.log("name" === reg.source);
console.log(true === reg.global);
console.log(true === reg.ignoreCase);
console.log(true === reg.multiline);
```

1.7.10.7. Basic library

Global

Note: SJS does not support most of the global attributes and methods of JavaScript.

Attribute

- Infinity
- NaN
- undefined

Note: For information about how to use these global attributes, see the ES5 specification.

Method

- decodeURI
- decodeURIComponent
- encodeURI
- encodeURIComponent
- isNaN
- isFinite
- parseFloat
- parseInt

Note: For information about how to use these global attributes, see the ES5 specification.

console

`The console.log` method is used to output relevant information in the console. This method can accept multiple parameters and connect the parameter results in the output.

Date

Method

- now
- parse
- UTC

Note: For information about how to use these global attributes, see the ES5 specification.

Number

Attribute

- MAX_VALUE
- MIN_VALUE
- NEGATIVE_INFINITY
- POSITIVE_INFINITY

Note: For information about how to use these global attributes, see the ES5 specification.

JSON

Method

- `stringify(object)`: converts an object into a JSON string and returns the string.
- `parse(string)`: converts a JSON string into an object and returns the object.

Example

```
console.log(undefined === JSON.stringify());
console.log(undefined === JSON.stringify(undefined));
console.log("null"===JSON.stringify(null));
console.log("222"===JSON.stringify(222));
console.log('"222"'===JSON.stringify("222"));
console.log("true"===JSON.stringify(true));
console.log(undefined===JSON.stringify(function(){}));
console.log(undefined===JSON.parse(JSON.stringify()));
console.log(undefined===JSON.parse(JSON.stringify(undefined)));
console.log(null===JSON.parse(JSON.stringify(null)));
console.log(222===JSON.parse(JSON.stringify(222)));
console.log("222"===JSON.parse(JSON.stringify("222")));
console.log(true===JSON.parse(JSON.stringify(true)));
console.log(undefined===JSON.parse(JSON.stringify(function(){})));
```

Math

Attribute

- E
- LN10
- LN2
- LOG2E
- LOG10E
- PI
- SQRT1_2
- SQRT2

Note: For information about how to use these global attributes, see the ES5 specification.

Method

- abs
- acos
- asin
- atan
- atan2
- ceil
- cos
- exp
- floor
- log
- max
- min
- pow
- random
- round
- sin
- sqrt
- tan

Note: For information about how to use these global attributes, see the ES5 specification.

1.7.10.8. esnext

SJS supports part of ES6 grammar.

SJS supports part of ES6 grammar.

let & const

```
function test(){
  let a = 5;
  if (true) {
    let b = 6;
  }
  console.log(a); // 5
  console.log(b); // Reference error: b is not defined
}
```

Arrow function

```
const a = [1,2,3];
const double = x => x * 2; // Arrow function
console.log(a.map(double));

var bob = {
  _name: "Bob",
  _friends: [],
  printFriends() {
    this._friends.forEach(f =>
      console.log(this._name + " knows " + f));
  }
};
console.log(bob.printFriends());
```

Enhanced object literal

```
var handler = 1;
var obj = {
  handler, // Object attribute
  toString() { // Object method
    return "string";
  },
};
```

Note: The `super` keyword is not supported and cannot be used in the object method.

Template string

```
const h = 'hello';
const msg = `${h} alipay`;
```

Destructuring

```
// Array destructuring assignment
var [a, ,b] = [1,2,3];
a === 1;
b === 3;
// Object destructuring assignment
var { op: a, lhs: { op: b }, rhs: c }
       = getASTNode();
// Shorthand for object destructuring assignment
var {op, lhs, rhs} = getASTNode();
// Destructured parameters
function g({name: x}) {
  console.log(x);
}
g({name: 5});
// Default values in destructuring assignment
var [a = 1] = [];
a === 1;
// Function parameters: destructured parameters + default values
function r({x, y, w = 10, h = 10}) {
  return x + y + w + h;
}
r({x:1, y:2}) === 23;
```

Default + Rest + Spread

```
// Default parameters
function f(x, y=12) {
  // If no value is passed to y, or if the passed value is undefined, then the value of y is 12.
  return x + y;
}
f(3) == 15;

function f(x, ...y) {
  // y is an array
  return x * y.length;
}
f(3, "hello", true) == 6;
function f(x, y, z) {
  return x + y + z;
}
f(...[ 1,2,3]) == 6; // Array destructuring
const [a, ...b] = [1,2,3]; // Array destructuring assignment, b = [2, 3]
const {c, ...other} = {c: 1, d: 2, e: 3}; // Object destructuring assignment, other = {d: 2, e: 3}
const d = {...other}; // Object destructuring
```

1.7.11. ACSS syntax reference

ACSS is a set of style language used to describe the component style of AXML and determine the display effect of AXML components.

ACSS is a set of style language used to describe the component style of AXML and determine the display effect of AXML components.

For the convenience of all developers, ACSS is completely consistent with the CSS rules, and is 100% available. At the same time, CSS has been expanded for developing Mini programs.

rpx

rpx (responsive pixel) can be adaptive based on the screen width, and the screen width is 750rpx. Take Apple iPhone 6 as an example, the screen width is 375px, and there are 750 physical pixels, then 750rpx = 375px = 750 physical pixels, and 1rpx = 0.5px = 1 physical pixel.

Device	rpx converted to px(screen width / 750)	px converted to rpx(750 / screen width)
iPhone 5	1rpx = 0.42px	1px = 2.34rpx
iPhone 6	1rpx = 0.5px	1px = 2rpx
iPhone 6 Plus	1rpx = 0.552px	1px = 1.81rpx

Import style

Use `@import` statement to import external style table. The relative path of the external style table is added after `@import`, and `;` indicates the end.

Code sample:

```
/** button.acss **/
.sm-button {
  padding: 5px;
}
```

```
/** app.acss **/
@import "./button.acss";
.md-button {
  padding: 15px;
}
```

It is supported to load third-party module from `node_modules` directory, such as `page.acss` :

```
@import "./button.acss"; /*Relative path*/
@import "/button.acss"; /*Project absolute path*/
@import "third-party/page.acss"; /*Third-party npm package path*/
```

Inline style

Component supports `style` and `class` attribute for style control.

style attribute

Used to receive dynamic style, which is parsed in runtime.

```
<view style="color:{{color}};" />
```

class attribute

Used to receive static style. Attribute value is the cluster of class selector names (style class names) in style rules. `.` doesn't need to be added to the style class name. Multiple class attributes are separated with spaces.

```
<view class="my-awesome-view" />
```

Static styles are written into `class` . Do not write static styles in `style` to avoid affecting rendering speed.

Selector

Consistent with CSS3.

Note:

- Class selectors start with `.a-` and `.am-` are occupied by system components, not usable.
- Attribute selectors not supported.

Global style and partial style

- The styles in `app.acss` are global style, which acts on every page.
- The styles defined in `.acss` file under page folders are partial style, which acts on part of the pages and overrides the same selector in `app.acss` .

Local resource references

For local resource references in ACSS files, use absolute paths. Relative path references are not supported. For example:

```
/* Supported */
background-image: url('/images/ant.png');
/* Not supported */
background-image: url('./images/ant.png');
```

1.7.12. Event system

1.7.12.1. Event introduction

- An event is used for communication between the view layer and the logical layer.
- An event can pass user behavior to the logical layer for processing.
- You can bind an event to a component. When the event is triggered, the corresponding event handler at the logical layer is executed.
- An event object can carry extra information, such as the `id` , `dataset` , and `touches` attributes.

Usage

To bind an event handler, for example `onTap` , to a component, define the corresponding `onTap` event handler in the `Page` object in the `.js` file of the page.

```
<view id="tapTest" data-hi="Alipay" onTap="tapName">
  <view id="tapTestInner" data-hi="AlipayInner">
    Click me!
  </view>
</view>
```

In the relevant `Page` object, define the corresponding event handler `tapName` with the event object as the parameter.

```
Page({
  tapName(event) {
    console.log(event);
  },
});
```

The event information generated in the console is shown as follows.

```
{
  "type": "tap",
  "timeStamp": 1550561469952,
  "target": {
    "id": "tapTestInner",
    "dataset": {
      "hi": "Alipay"
    },
    "targetDataset": {
      "hi": "AlipayInner"
    }
  },
  "currentTarget": {
    "id": "tapTest",
    "dataset": {
      "hi": "Alipay"
    }
  }
}
```

When you use a basic component, an extension component, or a custom component, whether an event is available depends on whether the component supports the event. You can find the supported events of each component in the component document.

Event type

Events are classified into the following two types:

- Bubbling event: The name of a bubbling event starts with the prefix `on`. After a bubbling event is triggered in a component, the event will be transferred to the parent node.
- Non-bubbling event: The name of a non-bubbling event starts with the prefix `catch`. After a non-bubbling event is triggered in a component, the event will not be transferred to the parent node.

The syntax of event binding is similar to the syntax of a component attribute and consists of a key and a value.

- A key starts with `on` or `catch`, followed by the event type. For example, the key can be `onTap` or `catchTap`.
- A value is a string that corresponds to the function name defined in the Page object. If the corresponding function name does not exist, an error occurs.

```
<view id="outter" onTap="handleTap1">
  view1
  <view id="middle" catchTap="handleTap2">
    view2
    <view id="inner" onTap="handleTap3">
      view3
    </view>
  </view>
</view>
```

Refer to the preceding code. A tap on view3 triggers handleTap3 and handleTap2 in order, because the tap event bubbles to view2 and view2 stops the tap event from going up to the parent node. A tap on view2 triggers handleTap2. A tap on view1 triggers handleTap1.

The following table describes all the types of bubbling events.

Type	Trigger condition
touchStart	Start the touch action
touchMove	Move after the touch action
touchEnd	End the touch action
touchCancel	The touch action is interrupted, such as by a incoming call reminder or pop-up window
tap	Leave after a short tap on the screen
longTap	Leave after a long tap on the screen that lasts more than 500 ms

1.7.12.2. Event object

When an event is triggered in a component, the event handler that is bound to the event at the logical layer receives an event object.

When an event is triggered in a component, the event handler that is bound to the event at the logical layer receives an event object.

BaseEvent object

The following table describes the attributes of a BaseEvent object.

Attribute	Type	Description
type	String	The type of the event
timeStamp	Integer	The timestamp when the event object is generated
target	Object	The attribute-value set of the component where the event is triggered

type

The type of the event.

timeStamp

The timestamp when the event object is generated.

target

The source component object where the event is triggered. The following table describes the attributes of a source component object.

Attribute	Type	Description
id	String	The ID of the source component
tagName	String	The type of the current component
dataset	Object	The set of custom attributes of the component to which the triggered event is bound. Each custom attribute in the set starts with `data-`.
targetDataset	Object	The set of custom attributes of the component where the event is triggered. Each custom attribute in the set starts with `data-`.

You can use `dataset` to define data in a component. After an event in the component is triggered, the defined data is transferred to the logical layer. Each data item in the dataset starts with `data-`. The words that follow are connected by hyphens `-`. All the letters must be lowercase and uppercase letters are automatically converted to lowercase letters. For example, for a string `data-element-type`, the `event.target.dataset` method converts this string to camel case `elementType`.

Code sample:

```
<view data-alpha-beta="1" data-alphaBeta="2" onTap="bindViewTap"> DataSet Test </view>
```

```
Page({
  bindViewTap:function(event) {
    event.target.dataset.alphaBeta === 1; // - Convert the string to camel case.
    event.target.dataset.alphabeta === 2; // Convert uppercase letters to lowercase letters.
  },
});
```

CustomEvent object

A `CustomEvent` object inherits from a BaseEvent object. The following table describes the attribute of a `CustomEvent` object.

Attribute	Type	Description
detail	Object	The extra information

detail

The data carried by the custom event. When an event is triggered in a form component, the generated event object carries user input information. For example, after the onChange event in the switch component is triggered, you can use `event.detail.value` to obtain the status value that is selected by the user. When an error event is triggered in a media component, the generated event object carries the error information. For more information, see event descriptions in component documents.

TouchEvent object

A TouchEvent object inherits from a BaseEvent object. The following table describes the attributes of a TouchEvent object.

Attribute	Type	Description
touches	Array	The array of the information about the current fixed touch points on the screen
changedTouches	Array	The array of the information about the current changed touch points

The `touches` attribute is an array of Touch objects that represent the current fixed touch points on the screen. Note that for a Canvas TouchEvent object, the `touches` attribute is an array of CanvasTouch objects.

The data format of the “changedTouches” attribute is the same as the data format of the “touches” attribute. The “changedTouches” attribute represents the current changed touch points, such as touchstart, touchmove, touchend, and touchcancel.

Touch object

Attribute	Type	Description
identifier	Number	The identifier of the touch point
pageX, pageY	Number	The coordinates of the touch point, with the upper-left corner of the text as the origin, the horizontal axis as X-axis, and the vertical axis as Y-axis.

Attribute	Type	Description
clientX, clientY	Number	The coordinates of the touch point, with the upper-left corner of the area of the page to display as the origin, the horizontal axis as X-axis, and the vertical axis as Y-axis. Note that the area of the page to display equals to the area of the screen minus the area of the navigation bar.

CanvasTouch object

Attribute	Type	Description
identifier	Number	The identifier of the touch point.
x, y	Number	The coordinates of the touch point, with the upper-left corner of the Canvas as the origin, the horizontal axis as X-axis, and the vertical axis as Y-axis.

1.7.13. Custom components

1.7.13.1. Custom component introduction

The base library **1.7.0** and later versions of the Mini Program support the custom component function. By calling `my.canIUse('component')`, you can check whether the custom component function can be used in the current version.

The custom component function can abstract the functional modules that need to be reused into custom components, so that they can be reused in different pages.

Note: The custom components have undergone major changes in basic library **1.14.0** and later versions.

- Added `onInit` and `deriveDataFromProps` Life cycle function.
- `ref` can be used to Obtain custom component instances.

1.7.13.2. Create custom component

Similar to `Page`, a custom component consists of `axml`, `js`, `json`, and `acss`. To create a custom component, perform the following steps.

1. Declare a component.
2. Use the `Component` function to register the custom component.

The following example shows a basic component.

```
// app.json
{
  "component": true
}
```

```
// /components/customer/index.js
Component({
  mixins: [], // minxin facilitates code reuse.
  data: { x: 1 }, // Internal component data.
  props: { y: 1 }, // Adds default values for the properties passed from the external.
  didMount(){}, // Life cycle function.
  didUpdate(){},
  didUnmount(){},
  methods: { // Custom method.
    handleTap() {
        this.setData({ x: this.data.x + 1}); // setData can be used to change the internal property.
    },
  },
})
```

```
<!-- /components/customer/index.axml -->
<view>
  <view>x: {{x}}</view>
  <button onTap="handleTap">plusOne</button>
  <slot>
    <view>default slot & default value</view>
  </slot>
</view>
```

1.7.13.3. Component configuration

A custom component must be declared in `[component].json`. If the custom component also depends on other components, you also need to declare them.

```
{
  "component": true, // Mandatory. The value of the custom component must be true
  "usingComponents": {
    "other":"../other/index" // Components that the custom component depends on
  }
}
```

Parameter details

Field	Type	Mandatory	Description
component	Boolean	Yes	Declares that the component is a custom component.
usingComponents	Object	No	Declares the path of the components that the custom component depends on. An absolute path starts with `/`, and a relative path starts with `./` or `../`.

1.7.13.4. Component template and style

Similar to pages, custom components can have their own AXML templates and ACSS styles.

Similar to pages, custom components can have their own AXML templates and ACSS styles.

AXML

AXML is a mandatory part of a custom component.

```
<!-- /components/index/index.axml -->
<view onTap="onMyClick" id="c-{{$id}}"/>
```

```
// /components/index/index.js
Component({
  methods: {
      onMyClick(e) {
      console.log(this.is, this.$id);
    },
  },
});
```

Note: Unlike pages, custom events need to be placed in `methods`.

Slot

If the component `js` supports `props`, a custom component can interact with external callers and receives data from external callers. It can also call functions from external callers and notifies external callers of changes in the component.

However, custom components are not flexible enough. In addition to data processing and notification, the slot provided by the Mini Program can be used to assemble the axml structure of a custom component and the axml passed from an external caller. An external caller passes axml to the custom component, and the custom component uses it to assemble the final component axml structure.

Default slot

Sample code:

```
<!-- /components/index/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <view>other</view>
</view>
```

Caller does not pass AXML:

```
// /pages/index/index.json
{
   "usingComponents": {
     "my-component": "/components/index/index"
   }
}
```

```
<!-- /pages/index/index.axml -->
<my-component />
```

Output:

```
default slot & default value
other
```

Caller passes axml:

```
<!-- /pages/index/index.axml -->
<my-component>
  <view>header</view>
  <view>footer</view>
</my-component>
```

Output:

```
header
footer
other
```

If the caller does not pass axml between the component tags `<xx>`, the default slot is rendered. If the caller passes axml between the component tags `<xx>`, axml is used instead of the default slot to assemble the final amxl for render.

Named slot

The default slot can only pass one copy of axml. A complex component needs to render different axmls in different positions, that is, multiple axmls need to be passed. Therefore, the named slot is required. If the named slot is used, the external caller can specify a part of axml in the subtag of the custom component tag and put it into a specific named slot of the custom component. The other part in the sub-tab of the custom component tag that does not have a specified named slot will be placed in the default slot. If the named slot is passed only, the default slot will not be overwritten.

Sample code:

```
<!-- /components/index/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <slot name="header"/>
  <view>body</view>
  <slot name="footer"/>
</view>
```

Passes the named slot only:

```
<!-- /pages/index/index.axml -->
<my-component>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</my-component>
```

Output:

```
default slot & default value
header
body
footer
```

Passes the named slot and the default slot:

```
<!-- /pages/index/index.axml -->
<my-component>
  <view>this is to default slot</view>
  <view slot="header">header</view>
  <view slot="footer">footer</view>
</my-component>
```

Output:

```
this is to default slot
header
body
footer
```

slot-scope

When the named slot is used, the custom component either uses its own axml or axml of an external caller (such as a page).If axml of the custom component is used, the internal component data can be accessed. With the help of props, the data of the external caller can also be accessed.

Sample code:

```
// /components/index/index.js
Component({
  data: {
    x: 1,
  },
  props: {
    y: '',
  },
});
```

```
<!-- /components/index/index.axml -->
<view>component data: {{x}}</view>
<view>page data: {{y}}</view>
```

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

```
<!-- /pages/index/index.axml -->
<my-component y="{{y}}" />
```

Output on the page:

```
component data: 1
page data: 2
```

If the custom component uses axml of an external caller (such as a page), only the external caller can be accessed.

Sample code:

```
<!-- /components/index/index.axml -->
<view>
  <slot>
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

```
<!-- /pages/index/index.axml -->
<my-component>
  <view>page data: {{y}}</view>
</my-component>
```

Output:

```
page data: 2
body
```

The slot scope allows the content of the slot to access the internal component data.

Sample code:

```
// /components/index/index.js
Component({
  data: {
    x: 1,
  },
});
```

```
<!-- /components/index/index.axml -->
<view>
  <slot x="{{x}}">
    <view>default slot & default value</view>
  </slot>
  <view>body</view>
</view>
```

```
// /pages/index/index.js
Page({
  data: { y: 2 },
});
```

```
<!-- /pages/index/index.axml -->
<my-component>
  <view slot-scope="props">
    <view>component data: {{props.x}}</view>
    <view>page data: {{y}}</view>
  </view>
</my-component>
```

Output:

```
component data: 1
page data: 2
body
```

As shown above, the custom component defines the slot property to expose the internal component data. When the page uses the component, the scope slot is used, and the property value is defined using the temporary variable name props. In this way, the internal data of the component can be accessed.

acss

Like pages, custom components can define their own acss styles. acss will be automatically introduced into the page that uses the component. For details, see acss syntax.

1.7.13.5. Component object

Component constructor

Description

Field	Type	Mandatory	Description	Minimum version
data	Object	No	Internal status of the component	
props	Object	No	Sets default values for external data	
onInit	Function	No	Life cycle function of the component, triggered when the component creation starts	1.14.0
deriveDataFromProps	Function	No	Life cycle function of the component, triggered when the component creation starts and before the component is updated	1.14.0
didMount	Function	No	Life cycle function of the component, triggered when the component creation completes	
didUpdate	Function	No	Life cycle function of the component, triggered when the component is updated	
didUnmount	Function	No	Life cycle function of the component, triggered when the component is deleted	
mixins	Array	No	Code reuse mechanism between components	
methods	Object	No	Method of the component, which can be an event response function or any custom method	

Code sample:

```
Component({
  mixins:[{ didMount() {}, }],
  data: {y:2},
  props:{x:1},
  didUpdate(prevProps,prevData){},
  didUnmount(){},
  methods:{
    onMyClick(ev){
      my.alert({});
      this.props.onXX({ ...ev, e2:1});
    },
  },
})
```

Note: `onInit` and `deriveDataFromProps` apply to basic library 1.14.0 and later versions. You can use `my.canIUse('component2')` to implement compatibility.

methods

A custom component can not only render static data, but also respond to user click events, which processes and triggers the custom component to re-render. Any custom method can be defined in methods.

Note: Unlike pages, custom components need to define the event handling function in methods.

```
// /components/counter/index.axml
<view>{{counter}}</view>
<button onTap="plusOne">+1</button>
```

```
// /components/counter/index.js
Component({
  data: { counter: 0 },
  methods: {
    plusOne(e) {
      console.log(e);
      this.setData({ counter: this.data.counter + 1 });
    },
  },
});
```

A page renders a button. The number of the page increases by 1 each time you tap the button.

props

The custom component can accept input from the external. After the component processes the input, it notifies the external that the work is done. These can be implemented using props.

Notes:

```
// /components/counter/index.js
Component({
  data: { counter: 0 },
  // Sets default properties
  props: {
    onCounterPlusOne: (data) => console.log(data),
    extra: 'default extra',
  },
  methods: {
    plusOne(e) {
      console.log(e);
      const counter = this.data.counter + 1;
      this.setData({ counter });
      this.props.onCounterPlusOne(counter); // Events in axml can be responded to only by the method in methods.
    },
  },
});
```

In the above code, props is used to set default properties, which can be obtained using `this.props` in the event handler.

```
// /components/counter/index.axml
<view>{{counter}}</view>
<view>extra: {{extra}}</view>
<button onTap="plusOne">+1</button>
```

```
// /pages/index/index.json
{
   "usingComponents": {
     "my-component": "/components/counter/index"
   }
}
```

External does not pass props

```
// /pages/index/index.axml
<my-component />
```

Output on the page:

```
0
extra: default extra
+1
```

No parameters are passed. Therefore, the page displays the default value set by props in js of the component.

External passes props

Note: When the external uses a custom component, if the passed parameter is a function, the parameter must be prefixed with `on`. Otherwise, it will be processed as a string.

```
// /pages/index/index.js
Page({
  onCounterPlusOne(data) {
    console.log(data);
  },
});
```

```
// /pages/index/index.axml
<my-component extra="external extra" onCounterPlusOne="onCounterPlusOne" />
```

Output on the page:

```
0
extra: external extra
+1
```

A parameter is passed. Therefore, the page displays the extra value passed externally, which is `external extra`.

Component instance properties

Property	Type	Description
data	Object	Internal data of the component
props	Object	Properties passed to the component
is	String	Component path
$page	Object	Page instance to which the component belongs
$id	Number	Component ID, which can be rendered in axml of the component

Code sample:

```
// /components/index/index.js
Component({
  didMount(){
    this.$page.xxCom = this; // This operation mounts the component instance on the page instance to which the component belongs.
    console.log(this.is);
    console.log(this.$page);
    console.log(this.$id);
  }
});
```

```
<! -- /components/index/index.axml The component ID can be rendered in axml of the component.
<view>{{$id}}</view>
```

```
// /pages/index/index.json
{
  "usingComponents": {
    "my-component": "/components/index/index"
  }
}
```

```
// /pages/index/index.js
Page({
  onReady() {
    console.log(this.xxCom); // You can access the components mounted on the current page.
  },
})
```

After the component is rendered on the page, the `didMount` callback is executed. The console output is as follows:

```
/components/index/index
{$viewId: 51, route: "pages/index/index"}
1
```

Component instance method

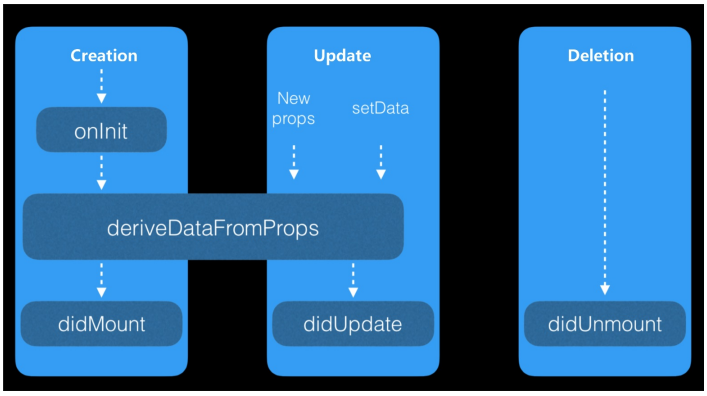
Method	Field	Description	Minimum version
setData	Object	Sets data to trigger view render	-
$spliceData	Object	Sets data to trigger view render	-

The usage is same as that in Page.

1.7.13.6. Component lifecycle

The life cycle function of a component is triggered by the framework at a special time point . The following diagram shows the life cycle of a component.

The life cycle function of a component is triggered by the framework at a special time point . The following diagram shows the life cycle of a component.



Detailed information of the life cycle function is as follows.

Life cycle	Field	Description	Minimum version
onInit	N/A	Triggered when the component creation starts	1.14.0
deriveDataFromProps	nextProps	Triggered when the component creation starts and before the component is updated	1.14.0
didMount	N/A	Triggered when the component creation completes	-
didUpdate	(prevProps,prevData)	Triggered when the component is updated	-
didUnmount	N/A	Triggered when the component is deleted	-

Note: `onInit` and `deriveDataFromProps` apply to basic library 1.14.0 and later versions. You can use `my.canIUse('component2')` for compatibility.

onInit

`onInit` is triggered when a component is created. In `onInit` , you can:

- Access the `this.is` , `this.$id` , and `this.$page` properties.
- Access the `this.data` and `this.props` properties.
- Access the custom properties of the methods component.
- Call `this.setData` and `this.$spliceData` to modify data.

Code sample 1

```
// /components/counter/index.js
Component({
  data: {
    counter: 0,
  },
  onInit() {
    this.setData({
      counter: 1,
      is: this.is,
    });
  },
})
```

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
<view>{{is}}</view>
```

After the component is rendered on the page, the following output is displayed.

```
1
/components/counter/index
```

Code sample 2

```
// /components/counter/index.js
Component({
  onInit() {
    this.xxx = 2;
    this.data = { counter: 0 };
  },
})
```

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
```

After the component is rendered on the page, the following output is displayed.

```
0
```

deriveDataFromProps

`deriveDataFromProps` is triggered when the component is created and updated. In `deriveDataFromProps`, you can:

- Access the `this.is`, `this.$id`, and `this.$page` properties.
- Access the `this.data` and `this.props` properties.
- Access the custom properties of the methods component.
- Call `this.setData` and `this.$spliceData` to modify data.
- Use `nextProps` to obtain the `props` parameter to be modified.

Code sample

```
// /components/counter/index.js
Component({
  data: {
    counter: 5,
  },
  deriveDataFromProps(nextProps) {
    if (this.data.counter < nextProps.pCounter) {
      this.setData({
        counter: nextProps.pCounter,
      });
    }
  },
})
```

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
```

```
// /pages/index/index.js
Page({
  data: {
    counter: 1,
  },
  plus() {
    this.setData({ counter: this.data.counter + 1 })
  },
})
```

```
<!-- /pages/index/index.axml -->
<counter pCounter="{{counter}}" />
<button onTap="plus">+</button>
```

Note: In this example, after you tap the "+" button, the counter on the page remains unchanged until the value of pCounter is greater than 5.

didMount

`didMount` is the callback after the custom component is rendered for the first time. At this time, the page has been rendered, and the server data is usually requested at this time.

Code sample

```
Component({
  data: {},
  didMount() {
    let that = this;
    my.httpRequest({
      url: 'http://httpbin.org/post',
      success: function(res) {
        console.log(res);
        that.setData({name: 'xiaoming'});
      }
    });
  },
});
```

didUpdate

`didUpdate` is the callback after the data of the custom component is updated. It will be called every time the data of the component changes.

Code sample

```
Component({
  data: {},
  didUpdate(prevProps, prevData) {
    console.log(prevProps, this.props, prevData, this.data);
  },
});
```

Notes:

- `didUpdate` is triggered when `this.setData` is called in the component.
- `didUpdate` can also be triggered when `this.setData` is called by an external caller.

didUnmount

`didUnmount` is the callback after the custom component is deleted. It is triggered every time the component instance is deleted from the page.

Code sample

```
Component({
  data: {},
  didUnmount() {
    console.log(this);
  },
});
```

1.7.13.7. mixins

Developers may implement multiple custom components. The Mini Program provides `mixins` to help process the common logic of these components.

The following shows an example.

```
// /minxins/lifecylce.js
export default {
  onInit(){},
  deriveDataFromProps(nextProps){},
  didMount(){},
  didUpdate(prevProps,prevData){},
  didUnmount(){},
};
```

Note: `onInit` and `deriveDataFromProps` apply to basic library 1.14.0 and later versions. You can use `my.canIUse('component2')` for compatibility.

```
// /pages/index/index.js
import lifecylce from './minxins/lifecylce';

const initialState = {
  data: {
    isLogin: false,
  },
};

const defaultProps = {
  props: {
    age: 30,
  },
};

const methods = {
  methods: {
     onTapHandler() {},
  },
}

Component({
  mixins: [
    lifecylce,
    initialState,
    defaultProps,
    methods
  ],
  data: {
    name: 'alipay',
  },
});
```

1.7.13.8. Obtain component instances by ref

In `1.14.0` and later versions, `ref` can be used to obtain custom component instances, and `my.canIUse('component2')` can be used for compatibility.

```
// /pages/index/index.js
Page({
  plus() {
    this.counter.plus();
  },
  // The ref parameter of the saveRef method is the custom component instance. The parameter is passed to saveRef by the framework during operation.
  saveRef(ref) {
    // Stores the custom component instance for future call.
    this.counter = ref;
  },
})
```

```
<!-- /pages/index/index.axml -->
<counter ref="saveRef" />
<button onTap="plus">+</button>
```

Note:

- After `saveRef` is bound using `ref`, the `saveRef` method will be triggered when the component is initialized.
- The `ref` parameter of the `saveRef` method is the custom component instance. The parameter is passed to `saveRef` by the framework.
- `ref` can be used for the parent component to obtain the instance of the child component.

```
// /components/counter/index.js
Component({
  data: {
    counter: 0,
  },
  methods: {
    plus() {
      this.setData({ counter: this.data.counter + 1 })
    },
  },
})
```

```
<!-- /components/counter/index.axml -->
<view>{{counter}}</view>
```

1.7.13.9. Use custom component

The usage of a custom component is similar to that of a basic component.

Note: By default, a custom component does not support events, such as the `onTap` event. To enable a custom component to support events, you must complete relevant configurations. For information about how to enable a custom component to support events, see Component constructor.

Usage

The usage of a custom component is similar to that of a basic component.

1. In the JSON file of a page, specify the custom component that you want to use.

```
// /pages/index/index.json
{
"usingComponents": {
 "customer": "/components/customer/index"
}
}
```

2. In the AXML file of the page, use the specified custom component the same way you use a basic component.

```
<!-- /pages/index/index.axml -->
<view>
<! -- Assign values to the "name" and "age" attributes of the custom component -->
<customer name="tom" age="{{23}}"/>
</view>
```

Note:

- When adopting the custom component, you can use `this.props` to obtain the specified attributes in the custom component. For more information, see props.
- You can use a custom component only in the AXML files of a page and of the component. You cannot use a custom component by `import` or `include`.

Sample of correct usage:

```
<!-- /pages/index/index.axml -->
<my-com />
```

Sample of invalid usage:

```
<!-- /pages/index/index.axml -->
<include src="./template.axml" />

<!-- /pages/index/template.axml -->
<view>
  <my-com />
</view>
```

Introduce custom component

```
// Configure in /pages/index/index.json not in the app.json file.
{
  "usingComponents":{
    "your-custom-component":"mini-antui/es/list/index",
    "your-custom-component2":"/components/card/index",
    "your-custom-component3":"./result/index",
    "your-custom-component4":"../result/index"
  }
}

// The absolute path of the project starts with "/", while the relative path starts with "./" or "../".
```

1.7.13.10. Publish custom component

The mPaaS Mini Program natively supports the introduction of third-party npm modules. Therefore, custom components can be published to npm, which facilitates code reuse and sharing.

The mPaaS Mini Program natively supports the introduction of third-party npm modules. Therefore, custom components can be published to npm, which facilitates code reuse and sharing.

Recommended custom component directory for releasing

The following directory structure is for reference only.

File structure

```
├── src // The file path of a single custom component
│   ├── index.js
│   ├── index.json
│   ├── index.axml
│   ├── index.acss
│   └── demo // A demonstration of the custom component
│       ├── index.js
│       ├── index.json
│       ├── index.axml
│       └── index.acss
├── app.js // A demonstration of the custom component in a Mini Program
├── app.json
└── app.acss
```

JSON file sample

```
// package.json
{
  "name": "your-custom-compnent",
  "version": "1.0.0",
  "description": "your-custom-compnent",
  "repository": {
    "type": "git",
    "url": "your-custom-compnent-repository-url"
  },
  "files": [
    "es"
  ],
  "keywords": [
    "custom-component",
    "mini-program"
  ],
  "devDependencies": {
    "rc-tools": "6.x"
  },
  "scripts": {
    "build": "rc-tools run compile && node scripts/cp.js && node scripts/rm.js",
    "pub": "git push origin && npm run build && npm publish"
  }
}
```

js file sample

```
// scripts/cp.js
const fs = require('fs-extra');
const path = require('path');
// copy file
fs.copySync(path.join(__dirname, '../src'), path.join(__dirname, '../es'), {
  filter(src, des){
    return !src.endsWith('.js');
  }
});
```

```
// scripts/rm.js
const fs = require('fs-extra');
const path = require('path');

// remove unnecessary file
const dirs = fs.readdirSync(path.join(__dirname, '../es'));

dirs.forEach((item) => {
  if (item.includes('app.') || item.includes('DS_Store') || item.includes('demo')) {
    fs.removeSync(path.join(__dirname, '../es/', item));
  } else {
    const moduleDirs = fs.readdirSync(path.join(__dirname, '../es/', item));
    moduleDirs.forEach((item2) => {
      if (item2.includes('demo')) {
        fs.removeSync(path.join(__dirname, '../es/', item, item2));
      }
    });
  }
});

fs.removeSync(path.join(__dirname, '../lib/'));
```

1.7.14. Optimization

The architecture for running Mini Programs is different from the architecture for running traditional HTML5 applications and consists of two parts: WebView and Worker. WebView is responsible for rendering. Worker is responsible for storing data and executing business logic.

How it works

The architecture for running Mini Programs is different from the architecture for running traditional HTML5 applications and consists of two parts: WebView and Worker. WebView is responsible for rendering. Worker is responsible for storing data and executing business logic.

1. The communication between WebView and Worker is asynchronous. This means that when you call the setData method, the data is asynchronously transferred from Worker to WebView instead of being immediately rendered.
2. Data needs to be serialized into strings before being transferred by using `evaluateJavascript`. The size of data affects the transfer performance.

Optimize the first screen

The definition of the first screen may differ. In this case, the first screen refers to the first page that is meaningfully rendered from the perspective of business logic. For example, for a list page, the first screen refers to the content that is rendered from the list for the first time.

Limit the size of the resource package of a Mini Program

When a user opens a Mini Program, Alipay downloads the resource package of the Mini Program from the relevant content delivery network. The size of the resource package affects the startup performance of the Mini Program.

Optimization suggestions:

- Regularly delete useless image resources. By default, all image resources are included in the resource package.
- Limit the sizes of images. We recommend that you do not include large images in the resource package and that you upload large images by using a content delivery network.
- Regularly clear useless code.

Use onLoad to request data in advance.

- When a Mini Program is running, the lifecycle function “onLoad” is triggered on a page. Then, the initial page data is transferred from Worker to WebView for initial rendering.
- After the initial rendering is complete, a notification is sent from WebView to Worker and the lifecycle function “onReady” is triggered.

For specific Mini Programs, a request for data may be included in the “onReady” function, which slows down the rendering of the first screen.

Optimization suggestion: Use onLoad to request data in advance.

Limit the number of nodes on the first screen that can be rendered at a time

Generally, after a business request receives a response, the setData method triggers to re-render the page. The execution process is described as follows:

1. Data is transferred from Worker to WebView.
2. On WebView, a virtual DOM is constructed based on the transferred data, and the virtual DOM is compared against the original DOM from the root node. Then, the rendering starts.

Data needs to be serialized for the communication between Worker and WebView. After the data is transferred to WebView, `evaluateJavascript` is executed. Therefore, if the volume of data transferred at a time is excessively large, the rendering performance of the first screen is affected.

There may be excessive nodes on WebView or the file structure has excessive directories. For example, the list page of a Mini Program may render more than 100 list items at a time. Each list item contains embedded content. However, only less than 10 list items are displayed on the screen. This causes the comparison to take more time. In addition, due to the fact that this is the rendering of the first screen, a lot of DOMs may be constructed, which affects the rendering performance of the first screen.

Optimization suggestions:

- Limit the volume of data when you call the setData method. This helps avoid transferring an excessively long list at a time.
- Do not create excessive nodes for the first screen. The server may request a large volume of data at a time. In this case, do not specify all the data when you call the setData method. You can specify some of the data and wait a specific period of time, such as 400 ms. The specific period of time varies based on your business. Then, you can call the `$spliceData` method to transfer the rest of the requested data.

Optimize the logic of the setData method

A change on a page triggers the setData method. Multiple changes on a page may trigger the setData method at the same time to re-render the page. Each of the following interfaces triggers the re-rendering of the WebView of a page:

- `Page.prototype.setData`: triggers the comparison of the whole page.
- `Page.prototype.$spliceData`: used for a long list to avoid transferring the whole list and triggering the comparison of the whole page every time.
- `Component.prototype.setData`: triggers the comparison from the relevant component node.
- `Component.prototype.$spliceData`: used for a long list to avoid transferring the whole list every time and trigger the comparison from the relevant component node.

Optimization suggestions:

- Do not frequently trigger the setData or `$spliceData` method on a page or in a component. Some pages may contain countdown logic. However, the countdown logic may have an excessively high frequency that causes millisecond-level triggering.
- If you need to frequently trigger re-rendering, do not call the setData or `$spliceData` method on the page. Instead, package the relevant resources into a custom component and call the setData or $spliceData method in the component for re-rendering.
- For the rendering of long list data, use `$spliceData` to add data for multiple times. You do not need to transfer the whole list at one time.
- For a complex page, we recommend that you package the relevant resources into a custom component to avoid calling the setData method on the page.

Optimization example:

We recommend that you set data in a specified path, as shown in the following code snippet:

```
this.setData({
   'array[0]': 1,
   'obj.x':2,
 });
```

Do not use the following code. In the code, `this.data` is copied, but the attributes are modified.

```
const array = this.data.array.concat();
 array[0] = 1;
 const obj={...this.data.obj};
 obj.x=2;
 this.setData({array,obj});
```

Do not directly modify the data that is obtained by using `this. data`, as shown in the following code snippet. This breaches the principle of data integrity.

```
this.data.array[0]=1;
 this.data.obj.x=2;
 this.setData(this.data)
```

Use the `$spliceData` method for long lists, as shown in the following code:

```
this.$spliceData({ 'a.b': [1, 0, 5, 6] })
```

Note: Business logic may be encapsulated into a component. When the user interface of the component needs re-rendering, you only need to call the setData method in the component. However, in some scenarios, the re-rendering of a component is triggered by an event on a page. For example, a page listens to the `onPageScroll` event. When the event is triggered, the component needs to be notified and start re-rendering. The following code snippets show how to implement this process:

```
// /pages/index/index.js
 Page({
     onPageScroll(e) {
         if (this.xxcomponent) {
             this.xxcomponent.setData({
                 scrollTop: e.scrollTop
             })
         }
     }
 })
```

```
// /components/index/index.js
 Component({
     didMount(){
         this.$page.xxcomponent = this;
     }
 })
```

You can bind the component to the page in the didMount() method. When you call the component-level setData method from the page, the call only triggers the re-rendering of the component.

Use the key parameter

You can use the key parameter in `for` to improve performance.

Note: Do not use the key parameter in a block.

Code sample:

```
<view a:for="{{array}}" key="{{item.id}}"></view>
 <block a:for="{{array}}"><view key="{{item.id}}"></view></block>
```

1.8. Basic components

1.8.1. Component overview

The Mini Program framework provides developers with a set of basic components. Developers can combine these basic components for service development.

Basic components

The Mini Program framework provides developers with a set of basic components. Developers can combine these basic components for service development.

Common component attributes

All components have the following attributes:

Property	Type	Description	Remarks
id	String	The unique identifier of the component.	-
class	String	Style-related attributes	-
style	String	The inline style	-
data-*	Any	Custom attributes	When an event is triggered, custom attributes are passed to the event handler.
on* / catch*	EventHandle	Bind the event, which follows the camel case naming convention, such as `onTap`.	For details, see Events.

Component attribute types

Each component provides a set of attribute configurations, and each attribute value has type requirements:

Type	Description	Remarks
Boolean	Boolean	-
Number	Numeric	-
String	String	-
Array	Array	-
Object	Object	-
EventHandle	The event handler	Define the implementation for the event handler name in Page.
any	Any type is allowed.	-

Component data binding

You must pass in the specified attribute type data through `{{}}`. For details, see Bind Data.

Basic component overview

The view container

Name	Function
view	The view container.

swiper	The slider view container.
scroll-view	The scrollable view area.
cover-view	The text view overlaid on native components.
movable-view	The movable view container.
movable-area	`The movable area of <movable-view>` .

Basic content

Name	Function
text	The text of the word.
icon	The icon.
progress	The progress bar.
rich-text	The rich-text component.

Form component

Name	Function
button	The button.
form	The form.
label	It is used to improve the usability of form components.
input	The entry box
textarea	The multi-line entry box.
radio	The single-choice item.
checkbox	The multi-choice selector group.
switch	The single-choice item.
slider	The slide selector.
picker-view	The scroll selector embedded in the page.
picker	The scroll selector that pops up from the bottom.

Navigation

Name	Description
navigator	Page links

Media component

Name	Description
image	The image component.

video	The video component.

Canvas

Name	Function
canvas	The canvas.

Map

Name	Function
map	The map component.

Open components

Name	Function
web-view	The component that hosts HTML5 webpages.

1.8.2. Component FAQ

Interaction exception between keyboard and component

For the components that need to launch **keyboard** (such as input, textarea, etc.), the native keyboard is currently used by default. If there is an exception to the interaction between the keyboard and the component, add the `enableNative="{{false}}"` attribute to the component, such as:

```
<textarea value="{{inputValue}}" enableNative="{{false}}" maxlength="500" onInput="onInput" />
```

Then you can restore the keyboard using WKWebview.

1.8.3. View container

1.8.3.1. View

A view container, equivalent to the div label of Web or the View component of React Native.

A view container, equivalent to the div label of Web or the View component of React Native.

API

Attribute name	Type	Default value	Description
disable-scroll	Boolean	false	Whether or not to prevent page scrolling in the area
hover-class	String		Style class added when tapping
hover-start-time	Number		The period that the button has been held before the tapped status appears, in milliseconds
hover-stay-time	Number		The period that the tapped status lasts after the button is released, in milliseconds
hidden	boolean	false	Whether or not to hide
class	String		Custom style name
style	String		Inline style
animation			Used for animation. See my.createAnimation for more information
onTap	EventHandle		Tap
onTouchStart	EventHandle		Touch action starts
onTouchMove	EventHandle		Moves after touching
onTouchEnd	EventHandle		Touch action ends
onTouchCancel	EventHandle		The touch action is interrupted such as by incoming call and pop-up dialog
onLongTap	EventHandle		Triggered after a long press of 500 ms. When a long press event is triggered, moving will not trigger the screen scrolling

Code sample

```
<view class="post">
  <!-- hidden -->
  <view class="postUser" hidden>
    <view class="postUser__name">Jessie</view>
  </view>
  <!-- hover class -->
  <view class="postBody" hover-class="red">
    <view class="postBody__content">
      Like!
    </view>
    <view class="postBody__date">
      June 1
    </view>
  </view>
</view>
```

1.8.3.2. Swiper

API

Property name	Type	Default value	Description
indicator-dots	Boolean	false	Whether or not to display the indicator
indicator-color	Color	rgba(0, 0, 0, .3)	Indicator color
indicator-active-color	Color	#000	Color of the currently selected indicator
autoplay	Boolean	false	Whether or not to switch automatically
current	Number	0	Index of the current page
duration	Number	500(ms)	Duration of sliding animation
interval	Number	5000(ms)	Interval of automatic switching
circular	Boolean	false	Whether or not to enable cohesive sliding
onChange	Function	No	Triggered when `current` changes, `event.detail = {current: current}`

swiper-item

It can only be placed in the component, and the width and height are automatically set to 100%.

Legend



Code example

```
<swiper
  indicator-dots="{{indicatorDots}}"
  autoplay="{{autoplay}}"
  interval="{{interval}}"
>
  <block a:for="{{background}}">
    <swiper-item>
      <view class="swiper-item bc_{{item}}"></view>
    </swiper-item>
  </block>
</swiper>
<view class="btn-area">
  <button class="btn-area-button" type="default" onTap="changeIndicatorDots">indicator-dots</button>
  <button class="btn-area-button" type="default" onTap="changeAutoplay">autoplay</button>
</view>
<slider onChange="intervalChange" value="{{interval}}" show-value min="2000" max="10000"/>
<view class="section__title">interval</view>
```

```
Page({
  data: {
    background: ['green', 'red', 'yellow'],
    indicatorDots: true,
    autoplay: false,
    interval: 3000,
  },
  changeIndicatorDots(e) {
    this.setData({
      indicatorDots: !this.data.indicatorDots
    })
  },
  changeAutoplay(e) {
    this.setData({
      autoplay: !this.data.autoplay
    })
  },
  intervalChange(e) {
    this.setData({
      interval: e.detail.value
    })
  },
})
```

1.8.3.3. Scroll view

When using vertical scrolling, you need to set a fixed height via ACSS.

API

Attribute name	Type	Default value	Description
class	String		External style name
style	String		Inline style name
scroll-x	Boolean	false	Whether or not to allow horizontal scrolling
scroll-y	Boolean	false	Whether or not to allow vertical scrolling
upper-threshold	Number	50	Trigger distance (in px) from the top/left side
lower-threshold	Number	50	Trigger distance (in px) from the bottom/right side
scroll-top	Number		Position of vertical scroll bar
scroll-left	Number		Position of horizontal scroll bar
scroll-into-view	String		The value is the ID of a child element. Scrolling to the element enables the top of the element to be aligned with the top of the scrolling area
scroll-with-animation	Boolean	false	Whether or not to use animated transition when selecting a position on the scroll bar
onScrollToUpper	EventHandle		Triggered when scrolling to the top/left side
onScrollToLower	EventHandle		Triggered when scrolling to the bottom/right side
onScroll	EventHandle		Triggered when scrolling, `event.detail` = `{scrollLeft, scrollTop, scrollHeight, scrollWidth, deltaX, deltaY}`

When using vertical scrolling, you need to set a fixed height via ACSS.

Code sample

```
<view class="page">
  <view class="page-description">Scroll view area</view>
  <view class="page-section">
    <view class="page-section-title">vertical scroll</view>
    <view class="page-section-demo">
      <scroll-view scroll-y="{{true}}" style="height: 200px;" onScrollToUpper="upper" onScrollToLower="lower" onScroll="scroll" scroll-into-view="{{t
oView}}" scroll-top="{{scrollTop}}">
        <view id="blue" class="scroll-view-item bc_blue"></view>
        <view id="red"  class="scroll-view-item bc_red"></view>
        <view id="yellow" class="scroll-view-item bc_yellow"></view>
        <view id="green" class="scroll-view-item bc_green"></view>
      </scroll-view>
    </view>
    <view class="page-section-btns">
      <view onTap="tap">next</view>
      <view onTap="tapMove">move</view>
      <view onTap="scrollToTop">scrollToTop</view>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">horizontal scroll</view>
    <view class="page-section-demo">
      <scroll-view class="scroll-view_H" scroll-x="{{true}}" style="width: 100%" >
        <view id="blue2" class="scroll-view-item_H bc_blue"></view>
        <view id="red2"  class="scroll-view-item_H bc_red"></view>
        <view id="yellow2" class="scroll-view-item_H bc_yellow"></view>
        <view id="green2" class="scroll-view-item_H bc_green"></view>
      </scroll-view>
    </view>
  </view>
</view>
```

```
const order = ['blue', 'red', 'green', 'yellow'];

Page({
  data: {
    toView: 'red',
    scrollTop: 100,
  },
  upper(e) {
    console.log(e);
  },
  lower(e) {
    console.log(e);
  },
  scroll(e) {
    console.log(e.detail.scrollTop);
  },
  scrollToTop(e) {
    console.log(e);
    this.setData({
      scrollTop: 0,
    });
  },
});
```

Attentions

- `scroll-into-view` has a higher priority than `scroll-top`.
- Since scrolling `scroll-view` prevents page bounce, so `onPullDownRefresh` will not be triggered when scrolling with `scroll-view`.

1.8.3.4. Cover view

cover-view

Text view overlaid on native components. `map` native components can be overlaid. Embedding applies to base library 1.10.0 and later versions of the Mini Program.

Attribute	Type	Default value	Description	Minimum version
onTap	EventHandle	-	Tapping event callback.	1.9.0

cover-image

Image view overlaid on native components. `cover-view` native components can be overlaid. Embedding applies to base library 1.10.0 and later versions of the Mini Program.

Attribute	Type	Default value	Description	Minimum version
src	String	-	Image address. The address format supported is the same as that of image.	1.9.0

Attribute	Type	Default value	Description	Minimum version
onTap	EventHandle	-	Tapping event callback.	1.9.0

Code sample

```
<view class="page">
  <view class="page-description">cover-view</view>
  <view class="page-section">
    <view class="page-section-demo" style="position: relative;">
      <map
        longitude="{{longitude}}"
        latitude="{{latitude}}"
        scale="{{scale}}"
        style="width: 100%; height: 200px;"
        include-points="{{includePoints}}"
      />
      <cover-view class="cover-view">
        <cover-view class="cover-view-item cover-view-item-1"></cover-view>
        <cover-view class="cover-view-item cover-view-item-2"></cover-view>
        <cover-view class="cover-view-item cover-view-item-3"></cover-view>
      </cover-view>
      <cover-image style="" src="/image/ant.png" />
    </view>
  </view>
</view>
```

1.8.3.5. Movable view

It is supported starting from the 1.11.0 basic library. For earlier versions, compatibility handling is required. For details, see About the Mini-program Basic Library.

movable-area

Movable area of `movable-view`.

Note: The `width` and `height` attributes of movable-area must be configured. If not specified, the default value is `10px`.

movable-view

Movable view container, which can be dragged to slide on the page.

Attribute	Type	Default value	Description	Minimum version
direction	String	none	Moving direction of movable-view. The value can be `all`, `vertical`, `horizontal`, or `none`.	-
x	Number	0	Offset in the X-axis direction, which is converted to the `left` property. If the value of X is not in the movable range, it automatically moves to the movable range.	-
y	Number	0	Offset in the Y-axis direction, which is converted to the `top` property. If the value of Y is not in the movable range, it automatically moves to the movable range.	-
disabled	Boolean	false	Whether to disable the button.	-
onTouchStart	EventHandle	-	Start the touch action.	1.11.5
onTouchMove	EventHandle	-	Move after the touch action.	1.11.5
onTouchEnd	EventHandle	-	End the touch action.	1.11.5
onTouchCancel	EventHandle	-	Tapping action interrupted by call reminder or pop-up window.	1.11.5
onChange	EventHandle	-	Event triggered during dragging. `event.detail = {x: x, y: y, source: source}`, where "source" indicates the cause of the movement. The value can be touch (drag).	-
onChangeEnd	EventHandle	-	Event triggered after dragging. `event.detail = {x: x, y: y}`	-

Code sample

```
<movable-area style="width: 100px;height: 100px;background-color: red;margin-left: 100px;">
  <movable-view
    onChange="onMovableViewChange"
    onChangeEnd="onMovableViewChangeEnd"
    direction="vertical"
    x="{{10}}"
    y="{{10}}"
    style="width: 40px;height: 40px;background-color: rgba(0, 0, 0, 0.5);"
  >
    <view onTap="onTapMovableView">movable-view</view>
  </movable-view>
</movable-area>
```

Note

- The width and height attributes of `movable-view` must be configured. If not specified, the default value is 10 px.
- By default, `movable-view` uses absolute positioning (do not modify it). The values of top and left are 0 px.
- If the value of `movable-view` is smaller than that of `movable-area`, the movable area of `movable-view` is within `movable-area`. If the value of `movable-view` is larger than that of `movable-area`, the movable area of `movable-view` must cover `movable-area`. X-axis and Y-axis need to be considered separately.
- `movable-view` must be in the `<movable-area/>` component and must be a direct child node. Otherwise, it cannot move.

1.8.4. Basic elements

1.8.4.1. Text

Only `<text/>` nesting is supported within the text component.

API

Property name	Type	Defaults	Description
selectable	Boolean	false	Whether it is optional
class	String		Style name
style	String		Inline style

Code example

```
<view class="page">
  <view class="text-view">
    <text>{{text}}</text>
  </view>
</view>
```

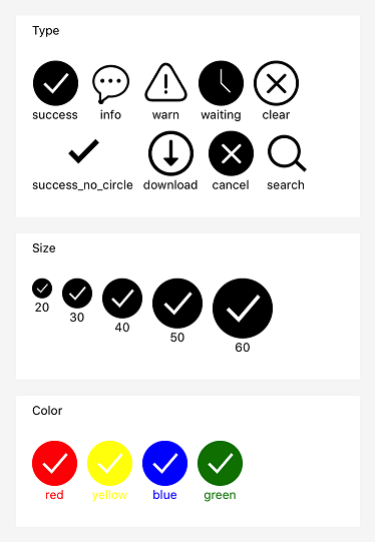
```
Page({
  data: {
    text: `Alipay is a large-scale lifestyle service platform, serving a wide variety of users ranging from middle-aged and elderly to teenagers.
      There are not only self-operated official apps, but also integrated third-party apps, to provide users with rich choices.
      Only if your product is simple enough can it be used by more users. And the more people use your product, the greater your gains.\ n\n:)
    `,
  },
})
```

1.8.4.2. Icon

API

Property name	Type	Defaults	Description
type	String		Icon type, valid values include: info, warn, waiting, cancel, download, search, clear, success, success_no_circle
size	Number	23	Icon size, in px
color	Color		Icon color, same as CSS color

Legend



Code example

```
<block a:for="{{iconType}}">
  <view class="item">
    <icon type="{{item}}" aria-label="{{item}}" size="45"/>
    <text>{{item}}</text>
  </view>
</block>

<block a:for="{{iconSize}}">
  <view class="item">
    <icon type="success" size="{{item}}"/>
    <text>{{item}}</text>
  </view>
</block>

<block a:for="{{iconColor}}">
  <view class="item">
    <icon type="success" size="45" color="{{item}}"/>
    <text style="color:{{item}}">{{item}}</text>
  </view>
</block>
```

```
Page({
  data: {
    iconSize: [20, 30, 40, 50, 60],
    iconColor: [
      'red', 'yellow', 'blue', 'green'
    ],
    iconType: [
      'success',
      'info',
      'warn',
      'waiting',
      'clear',
      'success_no_circle',
      'download',
      'cancel',
      'search',
    ]
  }
})
```

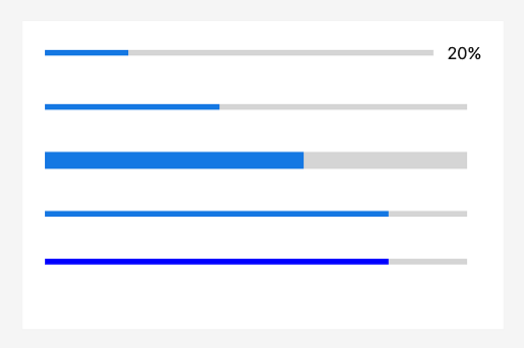
1.8.4.3. Progressbar

API

Property name	Type	Defaults	Description
percent	Float		Percentage (0~100)
show-info	Boolean	false	Show percentage value on the right
stroke-width	Number	6	Line thickness, in px
activeColor	Color	#09BB07	Selected progress bar color

Property name	Type	Defaults	Description
backgroundColor	Color		Unselected progress bar color
active	Boolean	false	Whether to load animation from left to right

Legend



Code example

```
<progress percent="20" show-info/>
<progress percent="40" active/>
<progress percent="60" stroke-width="10"/>
<progress percent="80" active/>
<progress percent="80" color="#10AEFF"/>
```

1.8.4.4. Rich text

The rich-text component. It is supported starting from the 1.11.0 basic library. For earlier versions, compatibility handling is required. For details, see About the Mini-program Basic Library.

Attribute	Type	Default value	Description	Minimum version
nodes	Array	[]	Only the **node list** is supported.	-

The following events are supported by default:

- tap
- touchstart
- touchmove
- touchcancel
- touchend
- longtap

Note: The nodes attribute only supports the Array type. To support the HTML String type, you must convert the HTML String string to a nodes array by using mini-html-parser.

nodes

Currently, two types of nodes, which are distinguished by type, are supported: the element node and the text node.

Element node

Attribute	Type	Description	Required
type	String	The node type, which defaults to node.	No
name	String	The label name, which supports certain trusted HTML nodes.	Yes
attrs	Object	The attribute, which supports certain trusted attributes and follows the Pascal naming convention.	No
children	Array	The list of child nodes, whose structure is the same as that of nodes.	No

Text node

Attribute	Type	Description	Required
type	String	Node type	Yes
text	String	The text of the word.	Yes

Supported HTML nodes and attributes

The class and style attributes are supported, and the ID attribute is not supported.

Node	Attribute
a	-
abbr	-
b	-
blockquote	-
br	-
code	-
col	span, width
colgroup	span, width
dd	-
del	-
div	-
dl	-
dt	-
em	-
fieldset	-
h1	-
h2	-
h3	-
h4	-
h5	-
h6	-
hr	-
i	-
img	alt, src, height, width
ins	-
label	-
legend	-
li	-
ol	start, type
p	-
q	-
span	-
strong	-
sub	-
sup	-
table	width
tbody	-
td	colspan, height, rowspan, width
tfoot	-
th	colspan, height, rowspan, width
thead	-

Node	Attribute
tr	-
ul	-

Code sample

```
<!-- page.axml -->
<rich-text nodes="{{nodes}}" onTap="tap"></rich-text>
```

```
// page.js
Page({
  data: {
    nodes: [{
      name: 'div',
      attrs: {
        class: 'test_div_class',
        style: 'color: green;'
      },
      children: [{
        type: 'text',
        text: 'Hello World! This is a text node.'
      }]
    }]
  },
  tap() {
    console.log('tap')
  }
})
```

Note: Only the following character entities are supported. Other character entities lead to rendering failure of components.

Output	Description	Entity name
	Space	
<	Less-than sign	<
>	Greater-than sign	>
&	Ampersand	&
"	Quotation mark	"
'	Apostrophe	'

1.8.5. Form component

1.8.5.1. button

This topic describes the button component.

Attribute	Type	Default value	Description	Minimum version
size	String	default	Possible values are default and mini.	-
type	String	default	The style of the button, which can be primary, default, or warn.	-
plain	Boolean	false	Whether the button is hollow.	-
disabled	Boolean	false	Whether to disable the button.	-
loading	Boolean	false	Whether the button text is accompanied by the loading icon.	-
hover-class	String	button-hover	The style class to which the button is pressed. `hover-class="none"` indicates no clicked effect.	-
hover-start-time	Number	20	The number of events after which the clicked state appears, in milliseconds.	-

Attribute	Type	Default value	Description	Minimum version
hover-stay-time	Number	70	The retention period of the clicked state after you release the finger, in milliseconds.	-
hover-stop-propagation	Boolean	false	Whether to prevent the ancestor element of the current element from appearing the clicked state.	1.10.0
form-type	String	-	Possible values are submit and reset. This attribute is used for components, and clicking it triggers the submit or reset event.	-
open-type	String	-	Open capabilities	1.1.0
scope	String	-	This attribute is valid when `open-type` is set to `getAuthorize`.	1.11.0
onTap	EventHandle	-	Tap.	-

Note

`button-hover` defaults to `{background-color: rgba(0, 0, 0, 0.1); opacity: 0.7;}`.

open-type possible values

Value	Description	Minimum version
Share	This attribute triggers custom sharing. You can determine its state by running `canIUse('button.open-type.share')`.	1.1.0
getAuthorize	Support Mini Program authorization. You can determine its state by running `canIUse('button.open-type.getAuthorize')`.	1.11.0
contactShare	Share with friends in the address book. You can determine its state by running `canIUse('button.open-type.contactShare')`.	1.11.0

scope possible values

When `open-type` is set to `getAuthorize`, you can set scope to one of the following values:

Value	Description	Minimum version
phoneNumber	Display the authorization interface, and the user can authorize the Mini Program to obtain the user's mobile number.	1.11.0

Illustration



Code sample

```
<view class="page">
  <view class="section">
    <view class="title">Type</view>
    <button type="default">default</button>
    <button type="primary">primary</button>
    <button type="warn">warn</button>
  </view>
  <view class="section" style="background:#ddd;">
    <view class="title">Misc</view>
    <button type="default" plain>plain</button>
    <button type="default" disabled>disabled</button>
    <button type="default" loading={{true}}>loading</button>
    <button type="default" hover-class="red">hover-red</button>
  </view>
  <view class="section">
    <view class="title">Size</view>
    <button type="default" size="mini">mini</button>
  </view>
  <view class="section">
    <view class="title">Type</view>
    <form onSubmit="onSubmit" onReset="onReset">
      <button form-type="submit">submit</button>
      <button form-type="reset">reset</button>
    </form>
  </view>
</view>
```

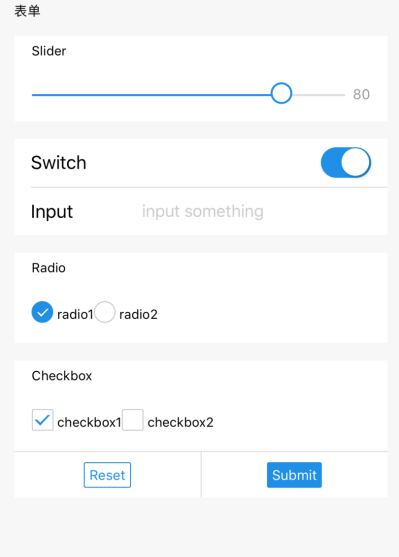
1.8.5.2. form

The form, which is used to submit `<textarea>`, `<switch/>`, `<input/>`, `<checkbox-group/>`, `<slider/>`, `<radio-group/>`, and `<picker/>` components entered by the user.

When you click the `button` component whose `form-type` is `submit` in the `form` component, the `value` in the form component is submitted. In this case, you need to add `name` as `key` in the form component.

Attribute	Type	Default value	Description	Minimum version
onSubmit	EventHandle	-	The data carried by the form component triggers the submit event, where `event.detail = {value : {'name': 'dao14'}, buttonTarget: {'dataset': 'buttonDataset'} }`.	This attribute is supported starting from buttonTarget 1.7.0.
onReset	EventHandle	-	Trigger the reset event when the form is reset.	-

Illustration



Code sample

```
<form onSubmit="formSubmit" onReset="formReset">
  <view class="section section_gap">
    <view class="section__title">switch</view>
    <switch name="switch"/>
  </view>
  <view class="section section_gap">
    <view class="section__title">slider</view>
    <slider name="slider" show-value ></slider>
  </view>

  <view class="section">
    <view class="section__title">input</view>
    <input name="input" placeholder="please input here" />
  </view>
  <view class="section section_gap">
    <view class="section__title">radio</view>
    <radio-group name="radio-group">
      <label><radio value="radio1"/>radio1</label>
      <label><radio value="radio2"/>radio2</label>
    </radio-group>
  </view>
  <view class="section section_gap">
    <view class="section__title">checkbox</view>
    <checkbox-group name="checkbox">
      <label><checkbox value="checkbox1"/>checkbox1</label>
      <label><checkbox value="checkbox2"/>checkbox2</label>
    </checkbox-group>
  </view>
  <view class="btn-area">
    <button formType="submit">Submit</button>
    <button formType="reset">Reset</button>
  </view>
</form>
```

```
Page({
  formSubmit: function(e) {
    console.log('The submit event occurs in form, and the carried data: ', e.detail.value)
  },
  formReset: function() {
    console.log('The reset event occurs in form')
  }
})
```

1.8.5.3. label

You can use `label` to improve the usability of form components. Use the `for` attribute to find the `id` of the corresponding component, or place the component under this label. When you click the label, the corresponding component is focused.

`for` has a higher priority than internal components. When multiple internal components exist, the first component is triggered by default.

Currently, you can bind the following controls: `<checkbox/>`, `<radio/>`, `<input/>`, and `<textarea/>`.

Attribute	Type	Description	Minimum version
for	String	The ID of the bound component.	-

Illustration



Code sample

```
<view class="section">
  <view class="title">Checkbox, which is the checkbox nested in the label.</view>
    <checkbox-group>
      <view>
        <label>
          <checkbox value="aaa" />
          <text>aaa</text>
        </label>
      </view>
      <view>
        <label>
          <checkbox value="bbb" />
          <text>bbb</text>
        </label>
      </view>
    </checkbox-group>
  </view>
</view>

<view class="section">
  <view class="title">Radio, which is associated by for.</view>
    <radio-group>
      <view>
        <radio id="aaa" value="aaa" />
        <label for="aaa">aaa</label>
      </view>
      <view>
        <radio id="bbb" value="bbb" />
        <label for="bbb">bbb</label>
      </view>
    </radio-group>
  </view>
</view>

<view class="section">
  <view class="title">When multiple Checkbox items are clicked, only one of them is selected.</view>
    <label>
      <checkbox>Select me</checkbox>
      <checkbox>Do not select</checkbox>
      <checkbox>Do not select</checkbox>
      <checkbox>Do not select</checkbox>
      <view>
        <text>Click Me</text>
      </view>
    </label>
  </view>
</view>
```

1.8.5.4. input

This topic describes the input component.

Attribute	Type	Default value	Description	Minimum version
value	String	-	Initial content	-

name	String	-	The component name, which is used to submit the form to obtain data.	-
type	String	text	The input type, which can be `text`, `number`, `idcard`, `digit`, `numberpad`, `digitpad`, and `idcardpad`.	`The numberpad`, `digitpad`, and `idcardpad` 1.13.0 types are supported starting from the 10.1.50 client. You can determine the value by running `my.canIUse('input.type.numberpad')`.
password	Boolean	false	Whether the type is password.	-
placeholder	String	-	The placeholder.	-
placeholder-style	String	-	The specified placeholder style.	1.6.0
placeholder-class	String	-	The specified placeholder style class.	1.6.0
disabled	Boolean	false	Whether to disable the button.	-
maxlength	Number	140	The maximum length.	-
focus	Boolean	false	Obtain the focus.	This feature is not supported.
confirm-type	String	done	Set the text of the button in the lower-right corner of the keyboard. Possible values are done (display “done”), go (display “go”), next (display “next”), search (display “search”), and send (display “send”). The displayed text varies slightly depending on different platforms. This attribute is valid only when type=text.	1.7.0
confirm-hold	Boolean	false	Whether to keep the keyboard unfolded when the button in the lower-right corner of the keyboard is clicked.	1.7.0
cursor	Number	-	Specify the cursor position when focused.	-
selection-start	Number	-1	The starting position of the focus cursor corresponding to the selected text when the cursor is obtained. This attribute must be used in conjunction with selection-end.	1.7.0
selection-end	Number	-1	The ending position of the focus cursor corresponding to the selected text when the cursor is obtained. This attribute must be used in conjunction with selection-start.	1.7.0
randomNumber	Boolean	false	Whether to arrange the numeric keypad randomly when type is set to number, digit, or idcard.	1.9.0

controlled	Boolean	false	Whether the component is a controlled component. If the value is true, the content of value is fully controlled by setData.	1.8.0
onInput	EventHandle	-	Trigger the input event when a keyboard entry occurs, where event.detail = {value: value}.	-
onConfirm	EventHandle	-	Trigger when the keyboard clicking action is completed, where event.detail = {value: value}.	-
onFocus	EventHandle	-	Trigger when the focus action occurs, where event.detail = {value: value}.	-
onBlur	EventHandle	-	Trigger when the focus is lost, where event.detail = {value: value}.	-

Illustration



Code sample

```
<input maxlength="10" placeholder="The maximum entry length is 10" />
<input onInput="bindKeyInput" placeholder="The input value will be synced to View"/>
<input type="number" placeholder="This is a numeric entry box" />
<input password type="text" placeholder="This is a password entry box" />
<input type="digit" placeholder="Numeric keypad with decimal point"/>
<input type="idcard" placeholder="The ID number entry keypad" />
```

```
Page({
  data: {
    inputValue: '',
  },
  bindKeyInput(e) {
    this.setData({
      inputValue: e.detail.value,
    });
  },
});
```

Handle interaction exceptions between the iOS keyboard and components

For components that need to start the keyboard, such as input and textarea, the native keyboard is currently used by default. When an interaction exception occurs between the keyboard and a component, you can add the `enableNative="{{false}}"` attribute to the component to restore to the keyboard that uses WKWebview, as shown in the following code. Meanwhile, since the system keyboard is used, the numeric keypad provided by mPaaS cannot be used. Currently, no special adaptation is available for keyboard-related exceptions. To handle such interaction exceptions, use this method.

```
<input placeholder="The ID number entry keypad" enableNative="{{false}}" />
```

1.8.5.5. textarea

This topic describes `textarea`, the multi-line entry box.

Attribute	Type	Default value	Description	Minimum version
name	String	-	The component name, which is used to submit the form to obtain data.	-
value	String	-	Initial content	-
placeholder	String	-	The placeholder.	-
placeholder-style	String	-	The specified placeholder style.	1.6.0
placeholder-class	String	-	The specified placeholder style class.	1.6.0
disabled	Boolean	false	Whether to disable the button.	-
maxlength	Number	140	The maximum length. When the value is 1, the maximum length is not limited.	-
focus	Boolean	false	Obtain the focus.	-
auto-height	Boolean	false	Whether to increase the height automatically.	-
show-count	Boolean	true	Whether to render the word count function.	1.8.0
controlled	Boolean	false	Whether the component is a controlled component. If the value is true, the content of value is fully controlled by setData.	1.8.0
onInput	EventHandle	-	Trigger when a keyboard entry occurs, where event.detail = {value: value}. You can directly return a string to replace the content of the entry box.	-
onFocus	EventHandle	-	Trigger when the entry box is focused, where event.detail = {value: value}.	-
onBlur	EventHandle	-	Trigger when the entry box is not focused, where event.detail = {value: value}.	-
onConfirm	EventHandle	-	Trigger when the clicking action is completed, where event.detail = {value: value}.	-

Illustration



Code sample

```
<view class="section">
  <textarea onBlur="bindTextAreaBlur" auto-height placeholder="Increase the height automatically" />
</view>
<view class="section">
  <textarea placeholder="Focus this only when the button is clicked" focus="{{focus}}" />
  <view class="btn-area">
    <button onTap="bindButtonTap">Make the entry box to be focused.</button>
  </view>
</view>
<view class="section">
  <form onSubmit="bindFormSubmit">
    <textarea placeholder="textarea in the form" name="textarea"/>
    <button form-type="submit"> Submit </button>
  </form>
</view>
```

```
Page({
  data: {
    focus: false,
    inputValue: ''
  },
  bindButtonTap() {
    this.setData({
      focus: true
    })
  },
  bindTextAreaBlur: function(e) {
    console.log(e.detail.value)
  },
  bindFormSubmit: function(e) {
    console.log(e.detail.value.textarea)
  }
})
```

Handle interaction exceptions between the iOS keyboard and components

For components that need to start the keyboard, such as input and textarea, the native keyboard is currently used by default. If interaction exceptions occur between the keyboard and a component, you can add the `enableNative="{{false}}"` attribute to the component to restore to the keyboard that uses WKWebview, as shown in the following code. Meanwhile, since the system keyboard is used, the Native keyboard-related functions provided by mPaaS cannot be used. Currently, no special adaptation is available for keyboard-related exceptions. For interaction exceptions, use this method to handle them.

```
<textarea value="{{inputValue}}" enableNative="{{false}}" maxlength="500" onInput="onInput" />
```

1.8.5.6. radio

This topic describes the single-choice selector, that is, radio.

This topic describes the single-choice selector, that is, radio.

radio-group

The single-choice selector group.

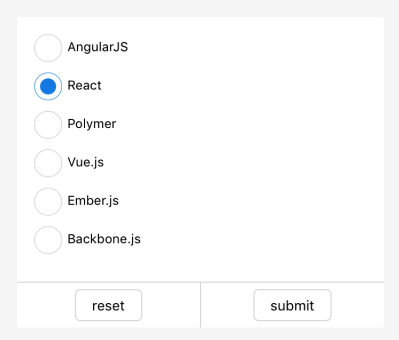
Attribute	Type	Default value	Description	Minimum version
onChange	EventHandle	-	Trigger when the selected item is modified, where `event.detail = {value: The value of the selected radio item}`	-
name	String	-	The component name, which is used to submit the form to obtain data.	-

radio

The single-choice item.

Attribute	Type	Default value	Description	Minimum version
value	String	-	The component value, which is the value carried by the change event when the component is selected.	-
checked	Boolean	false	Whether the item is currently selected.	-
disabled	Boolean	false	Whether to disable the button.	-
color	String	-	The radio color, which can be a color code like blue, or a hexadecimal color code like #0000FF.	1.10.0

Illustration



Code sample

```
<radio-group class="radio-group" onChange="radioChange">
  <label class="radio" a:for="{{items}}">
    <radio value="{{item.name}}" checked="{{item.checked}}"/>{{item.value}}
  </label>
</radio-group>
```

```
Page({
  data: {
    items: [
      {name: 'angular', value: 'AngularJS'},
      {name: 'react', value: 'React', checked: true},
      {name: 'polymer', value: 'Polymer'},
      {name: 'vue', value: 'Vue.js'},
      {name: 'ember', value: 'Ember.js'},
      {name: 'backbone', value: 'Backbone.js'},
    ]
  },
  radioChange: function(e) {
    console.log('You have selected this framework: ', e.detail.value)
  }
})
```

1.8.5.7. checkbox

This topic describes the multi-choice selector, that is, checkbox.

This topic describes the multi-choice selector, that is, checkbox.

checkbox-group

The multi-selector group.

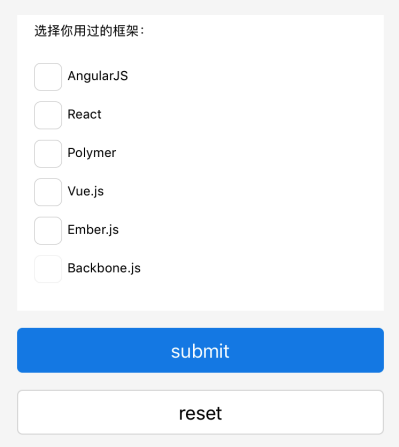
Attribute	Type	Default value	Description	Minimum version
name	String	-	The component name, which is used to submit the form to obtain data.	-
onChange	EventHandle	-	Trigger when the selected item is modified, where `detail = {value: The value of the selected checkbox item}`.	-

checkbox

The multi-choice item.

Attribute	Type	Default value	Description	Minimum version
value	String	-	The component value, which is the value carried by the change event when the component is selected.	-
checked	Boolean	false	Whether the current item is selected. You can use this attribute to set the item as selected by default.	-
disabled	Boolean	false	Whether to disable the button.	-
onChange	EventHandle	-	Trigger when the component is modified, where detail = {value: Whether this checkbox is selected}.	-
color	Color	-	The checkbox color.	1.10.0

Illustration



Sample code

```
// acss
.checkbox {
  display: block;
  margin-bottom: 20rpx;
}

.checkbox-text {
  font-size:34rpx;
  line-height: 1.2;
}
```

```
<checkbox-group onChange="onChange">
  <label class="checkbox" a:for="{{items}}">
    <checkbox value="{{item.name}}" checked="{{item.checked}}" disabled="{{item.disabled}}" />
    <text class="checkbox-text">{{item.value}}</text>
  </label>
</checkbox-group>
```

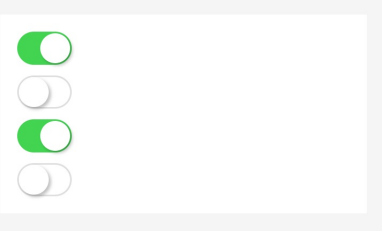
```
Page({
  data: {
    items: [
      {name: 'angular', value: 'AngularJS'},
      {name: 'react', value: 'React', checked: true},
      {name: 'polymer', value: 'Polymer'},
      {name: 'vue', value: 'Vue.js'},
      {name: 'ember', value: 'Ember.js'},
      {name: 'backbone', value: 'Backbone.js', disabled: true},
    ],
  },
  onChange(e) {
    my.alert({
      title: `You have selected the ${e.detail.value} framework`,
    });
  },
});
```

1.8.5.8. switch

This topic describes `switch`, the single-choice item.

Attribute	Type	Default value	Description	Minimum version
name	String	-	The component name, which is used to submit the form to obtain data.	-
checked	Boolean	-	Whether to select.	-
disabled	Boolean	-	Whether to disable the button.	-
color	String	-	The component color.	-
onChange	EventHandle	-	Trigger when checked is modified, where `event.detail={value:checked}`.	-
controlled	Boolean	false	Whether the component is a controlled component. If the value is true, checked is fully controlled by setData.	1.8.0
color	Color	-	The switch color.	1.10.0

Illustration



Code sample

```
<view class="page">
  <view class="switch-list">
    <view class="switch-item">
      <switch checked onChange="switchChange"/>
    </view>
  </view>
</view>
```

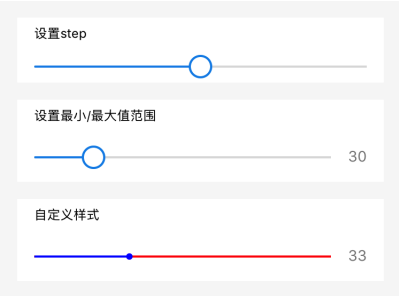
```
Page({
  switchChange (e){
    console.log('The switchChange event, and the value is: ', e.detail.value)
  },
})
```

1.8.5.9. slider

This topic describes the scroll selector, that is, slider.

Attribute	Type	Default value	Description	Minimum version
name	String	-	The component name, which is used to submit the form to obtain data.	-
min	Number	0	The minimum value.	-
max	Number	100	The maximum value.	-
step	Number	1	The step size, whose value must be greater than 0 and can be divisible by (max-min).	-
disabled	Boolean	false	Whether to disable the button.	-
value	Number	0	The current value.	-
show-value	Boolean	false	Whether to display the current value.	-
active-color	String	#108ee9	The selected color.	-
background-color	String	#ddd	The color of the background bar.	-
track-size	Number	4	The height of the track line, in px.	-
handle-size	Number	22	The height of the track line, in px.	-
handle-color	String	#fff	The filling color of the slider.	-
onChange	EventHandle	-	Trigger when a drag is completed, where `event.detail = {value: value}.`	-
onChanging	EventHandle	-	The event triggered during the drag, where `event.detail = {value: value}.`	1.5.0

Illustration



Code sample

```
<view class="section section-gap">
  <text class="section-title">Set step.</text>
  <view class="body-view">
    <slider value="60" onChange="sliderChange" step="5"/>
  </view>
</view>

<view class="section section-gap">
  <text class="section-title">Display the current value.</text>
  <view class="body-view">
    <slider value="50" show-value/>
  </view>
</view>

<view class="section section-gap">
  <text class="section-title">Set the minimum or maximum value.</text>
  <view class="body-view">
    <slider value="100" min="50" max="200" show-value/>
  </view>
</view>

<view class="page-section">
  <view class="page-section-title">The custom style.</view>
  <view class="page-section-demo">
    <slider value="33" onChange="slider4change" min="25" max="50" show-value
    backgroundColor="#FFAA00" activeColor="#00aaee" trackSize="2" handleSize="6" handleColor="blue" />
  </view>
</view>
```

```
Page({
  sliderChange(e)
    console.log('The changed slider value:', e.detail.value)
})
```

1.8.5.10. picker-view

This topic describes `picker-view`, the scroll selector embedded in a page.

Attribute	Type	Default value	Description	Minimum version
value	Number Array	-	A number indicates the corresponding index, which starts from 0, in `picker-view-column`.	-
indicator-style	String	-	The style of the checkbox.	-
indicator-class	String	-	The class name of the checkbox.	1.10
mask-style	String	-	The style of the mask.	1.10
mask-class	String	-	The class name of the mask.	1.10
onChange	EventHandle	-	Trigger when the scroll selection value is changed, where event.detail = {value: value}. Here, the value is an array, which indicates a picker-view-column index in picker-view. The index starts from 0.	-

Note: Only components can be placed, and other nodes will not be displayed. Do not place this component in the hidden or display none node. To hide it, use `a:if` to switch its state.

The recommended usage is as follows:

```
<view a:if="{{xx}}"><picker-view/></view>
```

The incorrect usage is as follows:

```
<view hidden><picker-view/></view>
```

Illustration



Code sample

```
<!-- API-DEMO page/component/picker-view/picker-view.axml -->
<view class="page">
  <view class="page-description">The scroll selector embedded in the page.</view>
  <view class="page-section">
    <view class="page-section-demo">
      <picker-view value="{{value}}" onChange="onChange" class="my-picker">
        <picker-view-column>
          <view>2011</view>
          <view>2012</view>
          <view>2013</view>
          <view>2014</view>
          <view>2015</view>
          <view>2016</view>
          <view>2017</view>
          <view>2018</view>
        </picker-view-column>
        <picker-view-column>
          <view>Spring</view>
          <view>Summer</view>
          <view>Fall</view>
          <view>Winter</view>
        </picker-view-column>
      </picker-view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/component/picker-view/picker-view.js
Page({
  data: {},
  onChange(e) {
    console.log(e.detail.value);
    this.setData({
      value: e.detail.value,
    });
  },
});
```

```
/* API-DEMO page/component/picker-view/picker-view.acss */
.my-picker {
  background: #EFEFF4;
}
```

1.8.5.11. Bottom scroll selector

A scroll selector that pops up from the bottom.

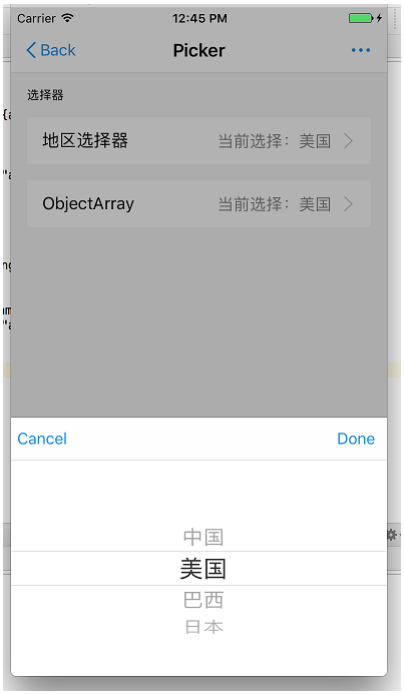
A scroll selector that pops up from the bottom.

API

Property name	Type	Defaults	Description
range	String[] / Object[]	[]	= String[] indicates a list of selectable strings; when it is Object[], you need to specify `range-key` to indicate that it is a selectable field.

Property name	Type	Defaults	Description
range-key	String		When `range` is an Object[], `range-key` is used to specify the value of `key` in the Object as the contents displayed in the selector.
value	Number		Indicates that which one is selected in `range` (the subscript starts from 0).
onChange	EventHandle		`value` changing will trigger it, `event.detail` = `{value: value}`.
disabled	Boolean	false	Whether to disable

Legend



Code example

```
<view class="section">
  <view class="section-title">Region selector</view>
  <picker onChange="bindPickerChange" value="{{index}}" range="{{array}}">
    <view class="picker">
      Current selection: {{array[index]}}
    </view>
  </picker>

  <picker onChange="bindObjPickerChange" value="{{arrIndex}}" range="{{objectArray}}" range-key="name">
    <view class="row">
      <view class="row-title">ObjectArray</view>
      <view class="row-extra">Current selection: {{objectArray[arrIndex].name}}</view>
      <image class="row-arrow" src="/image/arrowright.png" mode="aspectFill" />
    </view>
  </picker>
</view>
```

```
Page({
  data: {
    array: ['China', 'US', 'Brazil', 'Japan'],
    objectArray: [
      {
        id: 0,
        name: 'US',
      },
      {
        id: 1,
        name: 'China',
      },
      {
        id: 2,
        name: 'Brazil',
      },
      {
        id: 3,
        name: 'Japan',
      },
    ],
    arrIndex: 0,
    index: 0
  },
  bindPickerChange(e) {
    console.log('The picker sends a selection change with a value of ', e.detail.value);
    this.setData({
      index: e.detail.value,
    });
  },
  bindObjPickerChange(e) {
    console.log('The picker sends a selection change with a value of ', e.detail.value);
    this.setData({
      arrIndex: e.detail.value,
    });
  },
});
```

1.8.6. Navigator

API

Attribute name	Type	Default value	Description
hover-class	String	none	Class attached when tapped
hover-start-time	Number		The period that the button has been held before the tapped status appears, in milliseconds
hover-stay-time	Number		The period that the tapped status lasts after the button is released, in milliseconds
url	String		In-App jump link
open-type	String	navigate	Jump type

Valid values of `open-type`:

Attribute name	Description
navigate	Corresponds to the feature of `my.navigateTo`
redirect	Corresponds to the feature of `my.redirectTo`
switchTab	Corresponds to the feature of `my.switchTab`
navigateBack	Corresponds to the feature of `my.navigateBack`

Code sample

```
<!-- sample.axml -->
<view class="btn-area">
  <navigator url="/page/navigate/navigate?title=navigate" hover-class="navigator-hover">Jump to a new page</navigator>
  <navigator url="../../redirect/redirect/redirect?title=redirect" open-type="redirect" hover-class="other-navigator-hover">Open in the current page<
/navigator>
  <navigator url="/page/index/index" open-type="switchTab" hover-class="other-navigator-hover">Switch Tab</navigator>
</view>
```

1.8.7. Media component

1.8.7.1. image

This topic describes the image component.

Attribute	Type	Default value	Description	Minimum version
src	String	-	The image address	-
mode	String	scaleToFill	The image mode.	-
class	String	The external style.	-	-
style	String	The inline style	-	-
lazy-load	Boolean	false	Whether to enable lazy load. The lazy load is applicable to the scenarios that you fail to control image presentation or hide through CSS.	1.9.0
onLoad	EventHandle	-	Triggered when the image is fully loaded, where the event object is `event.detail = {height:'The image height in pixels', width:'The image width in pixels'}`	-
onError	EventHandle	-	Triggered when the image failed to be loaded, where the event object is `event.detail = {errMsg: 'something wrong'}`.	-
onTap	EventHandle	-	Triggered when the image is clicked.	-
catchTap	EventHandle	-	Triggered when the image is clicked and prevent event bubbles.	-

Note: The default width and height of the image component are 300px and 225px respectively.

Mode

13 modes are available, 4 of which are scaling modes and 9 are cropping modes.

Scaling modes

Attribute	Description
scaleToFill	Do not keep aspect-ratio scaling, but stretch the width and height of the image to fill up the image element.
aspectFit	Keep aspect-ratio scaling and fully display the long side of the image. In other words, fully display the image.
aspectFill	Keep aspect-ratio scaling and only ensure that the short side of the image can be fully displayed. In other words, the image is complete horizontally or vertically and is truncated in the other direction.
widthFix	Keep the width unchanged, change the height automatically, and keep the original aspect ratio of the image.

Cropping modes

Attribute	Description
top	Do not zoom the image and display only the top area.
bottom	Do not zoom the image and display only the bottom area.
center	Do not zoom the image and display only the central area.
left	Do not zoom the image and display only the left area.

Attribute	Description
right	Do not zoom the image and display only the right area.
top left	Do not zoom the image and display only the upper-left area.
top right	Do not zoom the image and display only the upper-right area.
bottom left	Do not zoom the image and display only the lower-left area.
bottom right	Do not zoom the image and display only the lower-right area.

> Note
> Do not set the image height to auto. To set the image height to auto, set the mode to `widthFix`.

Illustration

Original image



scaleToFill

The image is zoomed without keeping the aspect ratio, so that the image is stretched to fill all pixels.



aspectFit

Keep aspect-ratio scaling and fully display the long side of the image.



aspectFill

Keep aspect-ratio scaling and only ensure that the short side of the image can be fully displayed.



widthFix

The height changes automatically without changing the width, and the aspect ratio of the original image remains unchanged.



top

The image is not zoomed. Only the upper part is displayed.



bottom

The image is not zoomed. Only the lower part is displayed.



center

The image is not zoomed. Only the middle part is displayed.



left

The image is not zoomed. Only the left part is displayed.



right

The image is not zoomed. Only the right part is displayed.



top left

The image is not zoomed. Only the upper left part is displayed.



top right

The image is not zoomed. Only the upper right part is displayed.



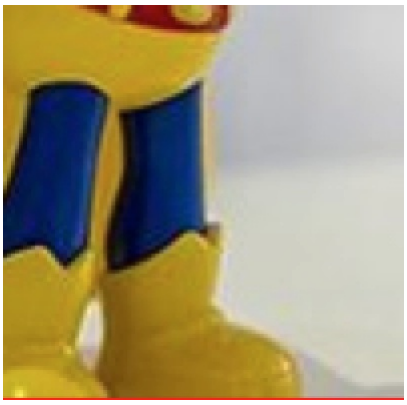
bottom left

The image is not zoomed. Only the lower left part is displayed.



bottom right

The image is not zoomed. Only the lower right part is displayed.



Code sample

```
<view class="section" a:for="{{array}}" a:for-item="item">
  <view class="title">{{item.text}}</view>
  <image style="background-color: #eeeeee; width: 300px; height:300px;" mode="{{item.mode}}" src="{{src}}" onError="imageError" onLoad="imageLoad" />
</view>
```

```
Page({
  data: {
    array: [{
      mode: 'scaleToFill',
      text: 'scaleToFill: The image is zoomed without keeping the aspect ratio, so that the image is stretched to fill all pixels.
    }, {
      mode: 'aspectFit',
      text: 'aspectFit: The image is zoomed with the aspect ratio unchanged, so that the longer edge of the image can be fully displayed.
    }, {
      mode: 'aspectFill',
      text: 'aspectFill: The image is zoomed with the aspect ratio unchanged, so that the shorter edge of the image can be fully displayed.
    }, {
      mode: 'top',
      text: 'top: The image is not zoomed. Only the upper part is displayed.
    }, {
      mode: 'bottom',
      text: 'bottom: The image is not zoomed. Only the lower part is displayed.
    }, {
      mode: 'center',
      text: 'center: The image is not zoomed. Only the middle part is displayed.
    }, {
      mode: 'left',
      text: 'left: The image is not zoomed. Only the left part is displayed.
    }, {
      mode: 'right',
      text: 'right: The image is not zoomed. Only the right part is displayed.
    }, {
      mode: 'top left',
      text: 'top left: The image is not zoomed. Only the upper left part is displayed.
    }, {
      mode: 'top right',
      text: 'top right: The image is not zoomed. Only the upper right part is displayed.
    }, {
      mode: 'bottom left',
      text: 'bottom left: The image is not zoomed. Only the lower left part is displayed.
    }, {
      mode: 'bottom right',
      text: 'bottom right: The image is not zoomed. Only the lower tight part is displayed.
    }],
    src: './2.png'
  },
  imageError: function (e) {
    console.log('Image3 has an error', e.detail.errMsg)
  },
  imageLoad: function (e) {
    console.log('Image is loaded successfully', e);
  }
})
```

1.8.7.2. video

The video component is supported in base library 1.14.1 and later versions.

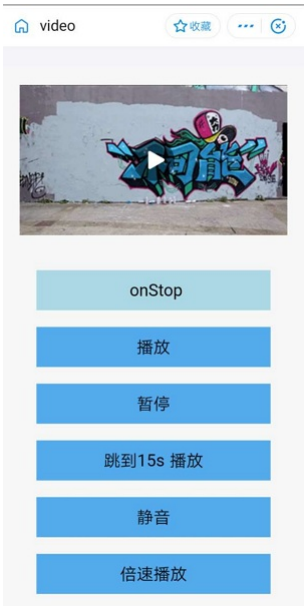
In an Android project, only mPaaS 10.1.58.12 and later versions support the video component. To use this component, add the **Mini Program - Video** component to the project first.

You can use the video component to upload and play videos. Related API: my.createVideoContext.

Notes:

- The video component does not apply to CSS animations.
- If white paddings appear on both sides when you watch a video in the custom portrait display mode:
  - If the aspect ratio of the video is inconsistent with that of the video component, adjust the aspect ratio of the video using object-fit.
  - If the actual aspect ratio of the poster is inconsistent with that of the container, adjust the aspect ratio of the poster using poster-size.

Effect example



Attribute

Attribute	Type	Default Value	Description
Style	String	-	Inline style.
Class	String	-	External style.

src	String	-	Resource address of the video to be played. Youku video encoding is supported.
			The source supports the following protocols:
			`apFilePath:`
			`https://resource/xxx.video`

poster	String	-	URL of the video thumbnail. Images in .jpg, .png and other formats are supported. For example, `https://***.jpg`. If no URL is specified, the URL of the first frame snapshot is used by default.
objectFit	String	contain	Form of the video when the video size is inconsistent with the video container size. The value can be contain or fill.
initial-time	Number	-	Start time for video playback. The value is in seconds.
duration	Number	-	Video duration. The default value is the total duration of the video, in seconds.
controls	Boolean	true	Whether to display the default playback widget, which is the toolbar at the bottom, including the play/pause button, playback progress and time.
autoplay	Boolean	false	Whether to play the video automatically. This function can cause compatibility problems in the Android 10 system. Therefore, you are advised not to enable autoplay.
direction	Number	-	Direction of the video in full screen. If the value is not specified, it is automatically determined based on the aspect ratio. The value can be • 0 (vertical), • 90 (90 degrees counterclockwise), or • -90 (90 degrees clockwise).
loop	Boolean	false	Whether to enable loop playback.
muted	Boolean	false	Whether to play the video in mute mode.

Attribute	Type	Default Value	Description
show-fullscreen-btn	Boolean	true	Whether to display the full screen button.
show-play-btn	Boolean	true	Whether to display the play button in the control bar at the bottom of the video.
show-center-play-btn	Boolean	true	Whether to display the play button in the middle of the video.
show-mute-btn	Boolean	true	Whether to display the mute button in the toolbar.
show-thin-progress-bar	Boolean	false	Whether to display the thin progress bar when the toolbar at the bottom is hidden. (This property is invalid when the value of controls is false.)
enableProgressGesture	Boolean	true	Whether to enable gestures for progress control in full screen mode.
mobilenetHintType	Number	1	Style of mobile network reminders. The value can be: • 0 - no reminders • 1- remind using tips • 2 - block reminders (no reminders for data traffic consumption) • 3 - block reminders (has reminders for data traffic consumption).
onPlay	EventHandle	-	play event, triggered when you play the video.
onPause	EventHandle	-	pause event, triggered when you pause the video.
onEnded	EventHandle	-	ended event, triggered when the video playback ends.
onTimeUpdate	EventHandle	-	Triggered when the playback progress changes. `event.detail = {currentTime: 'current playback time',userPlayDuration:'user watching duration',videoDuration:'total video duration'}`
onLoading	EventHandle	-	Triggered when video buffering occurs.
onError	EventHandle	-	Triggered when a video playback error occurs. For details about error codes, see the Error codes list.
onFullScreenChange	EventHandle	-	Triggered when the full screen mode is enabled or disabled. `event.detail = {fullScreen, direction}`. `direction` can be vertical or horizontal.
onTap	EventHandle	-	Triggered when you tap the video view. `event.detail = {ptInView: {x:0,y:0}}`
onUserAction	EventHandle	-	User operation event. `event.detail = {tag:"mute", value:0}`. "tag" indicates a user operation element, which can be: • play (the play button at the bottom) • centerplay (the play button in the middle) • mute (the mute button) • fullscreen (the fullscreen button) • retry (the retry button) • mobilenetplay (the play button for network reminders)
onStop	EventHandle	-	Video playback ends.
onRenderStart	EventHandle	-	Triggered when the video is loaded and played.
onTap	EventHandle	-	Triggered when you tap the video view. `event.detail = {ptInView: {x:0,y:0}}`

Code sample

```
<view>
  <video id="myVideo"
    src="{{video.src}}"
    controls="{{video.showAllControls}}"
    loop="{{video.isLooping}}"
    muted="{{video.muteWhenPlaying}}"
    show-fullscreen-btn="{{video.showFullScreenButton}}"
    show-play-btn="{{video.showPlayButton}}"
    show-center-play-btn="{{video.showCenterButton}}"
    object-fit="{{video.objectFit}}"
    autoplay="{{video.autoPlay}}"
    direction="{{video.directionWhenFullScreen}}"
    initial-time="{{video.initTime}}"
    mobilenetHintType="{{video.mobilenetHintType}}"
    onPlay="onPlay"
    onPause="onPause"
    onEnded="onEnded"
    onError="onPlayError"
    onTimeUpdate="onTimeUpdate"
   />
</view>
```

```
Page({
  data: {
      status: "inited",
    time: "0",
    video: {
    src: "http://flv.bn.netease.com/tvmrepo/2012/7/C/7/E868IGRC7-mobile.mp4",
    showAllControls: false,
    showPlayButton: false,
    showCenterButton: true,
    showFullScreenButton: true,
    isLooping: false,
    muteWhenPlaying: false,
    initTime: 0,
    objectFit: "contain",
    autoPlay: false,
    directionWhenFullScreen: 90,
    mobilenetHintType: 2,
 },},

    onPlay(e) {
    console.log('onPlay');
},

    onPause(e) {
    console.log('onPause');
},

    onEnded(e) {
    console.log('onEnded');
},

    onPlayError(e) {
    console.log('onPlayError, e=' + JSON.stringify(e));
},

    onTimeUpdate(e) {
    console.log('onTimeUpdate:', e.detail.currentTime);
},
});
```

Error codes

Error Code	Category	Description
1	Thrown during loading or playing	Unknown error
1002	Thrown during loading or playing	Internal player error
1005	Thrown during loading or playing	Network connection failure
1006	Loading exception	Data source error
1007	Loading exception	Player preparation failure
1008	Loading exception	Network error
1009	Loading exception	Video search error, which is a source error
1010	Loading exception	Preparation timeout, or playback failure due to slow network/data source

Error Code	Category	Description
400	Loading exception	UPS information reading timeout
3001	Loading exception	Audio rendering error
3002	Loading exception	Hard-decision decoding error
2004	Thrown during playing	Loading timeout during playing
1023	Thrown during playing	Internal error during playing (internal FFmpeg error)

Video encapsulation formats supported

The iOS and Android systems support the following video encapsulation formats:

- MP4
- MOV
- M4V
- 3GP
- M3U8
- FLV

Encoding modes supported

The iOS and Android systems support the following video encoding modes:

- H.264
- H.265
- AAC

FAQ

- **Q**: After I load and watch a video played in the video component, if I watch it again, does the system retrieve the cache or use the network to reload?

  A: The current caching policies are:
  - If the video is played in a loop, the cache is retrieved when the video is played again. This policy provides the caching capability mainly for scenarios where short videos are looped.
  - If the video is not played in a loop, it is reloaded using the network every time.

- **Q**: When will a video in the cache be cleared?

  A: It will be cleared when the page is destroyed or when the Mini Program is closed.

- **Q**: How does the Mini Program get the video duration?

  A: The Mini Program obtains the video duration from the `onTimeUpdate` method of the video component.

- **Q**: The loop mode is enabled in the video component. I try to play a video for the second time after the video resource is deleted, but the video cannot be played.

  A: Although the video in the cache is retrieved when it is played again in loop mode, the video resource is verified.

1.8.8. Canvas

API

Attribute name	Type	Default value	Description
id	String	-	Unique identifier of a component
style	String	-	Declare the style of the canvas.
class	String	-	Stycle class
width	String	-	Canvas width
height	String	-	Canvas height
disable-scroll	Boolean	false	Disable screen scrolling and pull-to-refresh.
onTap	EventHandle	-	Click
onTouchStart	EventHandle	-	Touch action starts
onTouchMove	EventHandle	-	Moves after touching
onTouchEnd	EventHandle	-	Touch action ends

Attribute name	Type	Default value	Description
onTouchCancel	EventHandle	-	The touch action is interrupted such as by incoming call and pop-up dialog.
onLongTap	EventHandle	-	Triggered after a long press of 500 ms. When a long press event is triggered, moving will not trigger the screen scrolling.

Important:

- Canvas has a width of 300 px and a height of 225 px by default.
- IDs on the same page cannot be duplicate.
- If you require a more detailed display under high DPR, you need to first zoom in canvas with its attribute settings and then zoom out with styles. For example:

```
<!-- getSystemInfoSync().pixelRatio === 2 -->
<canvas width="200" height="200" style="width:100px;height:100px;"/>
```

Legend



Code sample

```
<canvas
  id="canvas"
  class="canvas"
  onTouchStart="log"
  onTouchMove="log"
  onTouchEnd="log"
/>
```

```
Page({
  onReady() {
    this.point = {
      x: Math.random() * 295,
      y: Math.random() * 295,
      dx: Math.random() * 5,
      dy: Math.random() * 5,
      r: Math.round(Math.random() * 255 | 0),
      g: Math.round(Math.random() * 255 | 0),
          b: Math.round(Math.random() * 255 | 0),
    };

    this.interval = setInterval(this.draw, 17);
  },

  draw() {
    var ctx = my.createCanvasContext('canvas');
    ctx.setFillStyle('#FFF');
    ctx.fillRect(0, 0, 305, 305);

    ctx.beginPath();
    ctx.arc(this.point.x, this.point.y, 10, 0, 2 * Math.PI);
    ctx.setFillStyle("rgb(" + this.point.r + ", " + this.point.g + ", " + this.point.b + ")");
    ctx.fill();
    ctx.draw();

    this.point.x += this.point.dx;
    this.point.y += this.point.dy;
    if (this.point.x <= 5 || this.point.x >= 295) {
      this.point.dx = -this.point.dx;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }

    if (this.point.y <= 5 || this.point.y >= 295) {
      this.point.dy = -this.point.dy;
      this.point.r = Math.round(Math.random() * 255 | 0);
      this.point.g = Math.round(Math.random() * 255 | 0);
      this.point.b = Math.round(Math.random() * 255 | 0);
    }
  },
  drawBall() {

  },
  log(e) {
    if (e.touches && e.touches[0]) {
      console.log(e.type, e.touches[0].x, e.touches[0].y);
    } else {
      console.log(e.type);
    }
  },
  onUnload() {
    clearInterval(this.interval)
  }
})
```

1.8.9. Map

API

Attribute name	Type	Default value	Description
style	String		Inline style
class	String		Style name
longitude	Number		Longitude of the center
latitude	Number		Latitude of the center
scale	Number	16	Scale level, ranging from 5~18
markers	Array		Marker
polyline	Array		Polyline

Attribute name	Type	Default value	Description
circles	Array		Circle
controls	Array		Control
polygon	Array		Polygon
include-points	Array		Zoom out the view to include all given coordinates.
show-location	Boolean		Whether or not to display the current location with direction.
onMarkerTap	EventHandle		Triggered when a marker is tapped.
onCalloutTap	EventHandle		Triggered when the bubble of the marker is tapped.
onControlTap	EventHandle		Triggered when a control is tapped.
onRegionChange	EventHandle		Triggered when the view changes, {type: “begin” / “end” , latitude, longitude, scale}.
onTap	EventHandle		Triggered when the map is tapped.

markers

A marker is used to display the location marked on the map.

Attribute name	Description	Type	Required	Remarks
id	Marker ID	Number	No	The tap event callback will return this ID.
latitude	Latitude	Float	Yes	Range -90 ~ 90
longitude	Longitude	Float	Yes	Range -180 ~ 180
title	Name of a marker	String	No	
iconPath	Displayed icon	String	Yes	Image path under the project directory, which can be a relative path, starting with "/" to indicate that it is a relative path of the MINI program root directory.
rotate	Rotation degree	Number	No	Clockwise angle, ranging from 0 ~ 360. It defaults to 0.
alpha	Marker transparency	Number	No	Whether it is transparent or not. It defaults to 1.
width	Marker icon width	Number	No	It defaults to the actual width of the image.
height	Marker icon height	Number	No	It defaults to the actual height of the image.
callout	Bubble on the marker	Object	No	{content}
anchorX	The horizontal position (ratio) of anchor (coordinate) in the marker icon	Double	No	These two values need to appear in pairs, and anchorX represents horizontal (0-1) while anchorY represents vertical (0-1). For example: anchorX:0.5, anchorY:1; represents the middle of the bottom.
anchorY	The vertical position (ratio) of anchor (coordinate) in the marker icon	Double	No	

polygon

Polygon is used to construct a polygonal object.

Attribute name	Description	Type	Required	Remarks
points	Latitude and longitude array	Array	Yes	[{latitude: 0, longitude: 0}]
color	Line color	String	No	Expressed in 8-digit hexadecimal, and the last two digits represent alpha value, such as: #eeeeeeAA.

Attribute name	Description	Type	Required	Remarks
fillColor	Fill color	String	No	Expressed in 8-digit hexadecimal, and the last two digits represent alpha value, such as: #eeeeeeAA.
width	Line width	Number	No	-

polyline

Polyline is used to specify a series of coordinates that connect the first element of the array to the last element.

Attribute name	Description	Type	Required	Remarks
points	Latitude and longitude array	Array	Yes	[{latitude: 0, longitude: 0}]
color	Line color	String	No	Expressed in 8-digit hexadecimal, and the last two digits represent alpha value, such as: #eeeeeeAA.
width	Line width	Number	No	
dottedLine	Whether it is the dotted line	Boolean	No	Defaults to false

circles

Circles is used to display circles on the map.

Attribute name	Description	Type	Required	Remarks
latitude	Latitude	Float	Yes	Range -90 ~ 90
longitude	Longitude	Float	Yes	Range -180 ~ 180
color	Stroke color	String	No	Expressed in 8-digit hexadecimal, and the last two digits represent alpha value, such as: #eeeeeeAA.
fillColor	Fill color	String	No	Expressed in 8-digit hexadecimal, and the last two digits represent alpha value, such as: #eeeeeeAA.
radius	Radius	Number	Yes	-
strokeWidth	Stroke width	Number	No	-

controls

Controls is used to display controls on the map. The controls do not move with the map.

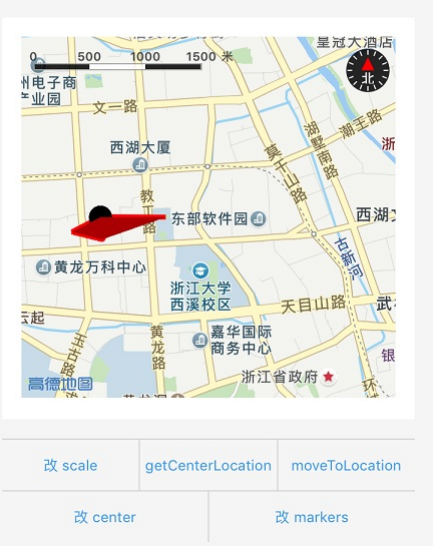
Attribute name	Description	Type	Required	Remarks
id	Control ID	Number	No	The tap event callback will return this ID.
position	The position of the control on the map	Object	Yes	The position relative to the map.
iconPath	Displayed icon	String	Yes	Image path under the project directory, which can be a relative path, starting with “/ “ to indicate that it is a relative path of the MINI program root directory.
clickable	Whether it can be tapped..	Boolean	No	It defaults to false.

position

Attribute name	Description	Type	Required	Remarks
left	How far it is from the left boundary of the map.	Number	No	It defaults to 0, in px.
top	How far it is from the upper boundary of the map.	Number	No	It defaults to 0, in px.
width	Control width	Number	No	It defaults to the image width, in px.
height	Control height	Number	No	It defaults to the image height, in px.

The latitude and longitude of the map component are mandatory. If the latitude and longitude are not provided, then it defaults to the latitude and longitude of Beijing.

Legend



Code sample

```
<view>
  <map id="map" longitude="120.131441" latitude="30.279383" scale="{{scale}}" controls="{{controls}}"
  onControlTap="controltap" markers="{{markers}}"
  onMarkerTap="markertap"
  polyline="{{polyline}}" circles="{{circles}}"
  onRegionChange="regionchange"
  onTap="tap"
  show-location style="width: 100%; height: 300px;"
  include-points="{{includePoints}}"></map>
  <button onTap="changeScale">Change scale</button>
  <button onTap="getCenterLocation">getCenterLocation</button>
  <button onTap="moveToLocation">moveToLocation</button>
  <button onTap="changeCenter">Change center</button>
  <button onTap="changeMarkers">Change markers</button>
</view>
```

```
Page({
  data: {
    scale: 14,
    longitude: 120.131441,
    latitude: 30.279383,
    markers: [{
      iconPath: "/image/green_tri.png",
      id: 10,
      latitude: 30.279383,
      longitude: 120.131441,
      width: 50,
      height: 50
    }],
    includePoints: [{
      latitude: 30.279383,
      longitude: 120.131441,
    }],
    polyline: [{
```

```
    points: [{
      longitude: 120.131441,
      latitude: 30.279383
    }, {
      longitude: 120.128821,
      latitude: 30.278200
    }, {
      longitude: 120.131618,
      latitude: 30.277600
    }, {
      longitude: 120.132520,
      latitude: 30.279393
    }, {
      longitude: 120.137517,
      latitude: 30.279383
    }],
    color: "#FF0000DD",
    width: 5,
    dottedLine: false
  }],
  circles: [{
    latitude: 30.279383,
    longitude: 120.131441,
    color: "#000000AA",
    fillColor: "#000000AA",
    radius: 80,
    strokeWidth: 5,
  }],
  controls: [{
    id: 5,
    iconPath: '../../resources/pic/2.jpg',
    position: {
      left: 0,
      top: 300 - 50,
      width: 50,
      height: 50
    },
    clickable: true
  }]
},

onReady(e) {
  // Get the map context using my.createMapContext
  this.mapCtx = my.createMapContext('map')
},

getCenterLocation() {
 this.mapCtx.getCenterLocation({
    success: (res) => {
      my.alert({
        content: 'longitude:' + res.longitude + '\nlatitude:' + res.latitude + '\nscale:' + res.scale,
      });
      console.log(res.longitude);
      console.log(res.latitude);
      console.log(res.scale);
    },
  });
},

moveToLocation() {
  this.mapCtx.moveToLocation()
},

regionchange(e) {
  console.log('regionchange', e);
  // Note: After zooming in or out the map, reset the scale value of data in the onRegionChange function.
  // If not set, reloading after dragging the map may cause the scale to restore to original size before scaling.
  if (e.type === 'end') {
    this.setData({
      scale: e.scale
    });
  }
},

markertap(e) {
  console.log('marker tap', e);
},

controltap(e) {
  console.log('control tap', e);
},

tap() {
  console.log('tap:');
},

changeScale() {
```

```
    this.setData({
      scale: 8,
    });
  },

  changeCenter() {
    this.setData({
      longitude: 113.324520,
      latitude: 23.199994,
      includePoints: [{
        latitude: 23.199994,
        longitude: 113.324520,
      }],
    });
  },

  changeMarkers() {
    this.setData({
      markers: [{
        iconPath: "/image/green_tri.png",
        id: 10,
        latitude: 21.21229,
        longitude: 113.324520,
        width: 50,
        height: 50
      }],
      includePoints: [{
        latitude: 21.21229,
        longitude: 113.324520,
      }],
    });
  },
})
```

Attentions

- `map` component is a native component created by a client and placed at highest level of hierarchy.
- Do not use `map` component in `scroll-view`.
- CSS animations are not valid for `map` components.
- If you zoomed in or out the map, reset the `scale` value of `data` in the `onRegionChange` function. If not set, reloading after dragging a map area may cause the scale to restore to original size before scaling. For details, see the `regionchange` function section in Code sample.

1.8.10. Open components

1.8.10.1. web-view

The `<web-view />` component bears HTML5 pages and automatically fills the entire Mini Program page.

Attribute	Type	Default value	Description
src	String	N/A	URL of the HTML5 page that `web-view` is about to render.
onMessage	EventHandle	N/A	Message that the web page sends to the Mini Program using postMessage `e.detail = { data }`

Note: The component applies to basic library 1.6.0 and later versions. Operations to resolve the compatibility problems need to be performed in earlier versions. For the operations, see Mini-program Basic Library Description.

Each page can have only one `<web-view />`. Do not render multiple `<web-view />` components. It will automatically fill the entire page and cover other components.

Code sample

```
<!-- axml -->
<! -- web-view that points to Alipay homepage -->
<web-view src="https://ds.alipay.com/" onMessage="test"></web-view>
```

Related APIs

On a `<web-view />` HTML5 page, you can manually introduce `https://appx/web-view.min.js` (this link can be accessed only from the mPaaS client). Related APIs are provided to return to the Mini Program page. The following APIs are supported.

API Type	API Name	Description
Navigation Bar	my.navigateTo	Keeps the current page and redirects to a specified page in the application.
Navigation Bar	my.navigateBack	Closes the current page and returns to a previous page.

API Type	API Name	Description
Navigation Bar	my.switchTab	Redirects to a specified tabBar page and closes all non-tabBar pages.
Navigation Bar	my.reLaunch	Closes all current pages and redirects to a specified page in the application.
Navigation Bar	my.redirectTo	Closes the current page and redirects to a specified page in the application.
Image	my.chooseImage	Takes a picture or select one from the mobile album.
Image	my.previewImage	Previews the image.
Location	my.getLocation	Obtains the current geographic location information of the user.
Location	my.openLocation	Uses the built-in map to view the location.
Interaction feedback	my.alert	Alert box.
Interaction feedback	my.showLoading	Displays the loading reminder.
Interaction feedback	my.hideLoading	Hides the loading reminder.
Cache	my.setStorage	Stores the data in the key specified in the local cache, which overwrites the original data of the key.
Cache	my.getStorage	Obtains the cache data.
Cache	my.removeStorage	Deletes the cache data.
Cache	my.clearStorage	Deletes the local cache data.
Cache	my.getStorageInfo	Asynchronously obtains information about the current storage.
Network status	my.getNetworkType	Obtains the current network status.
Sending message to Mini Program	my.postMessage	Send messages to the Mini Program. The message can be one or multiple sets of custom key and value data in the format of JSON, for example, my.postMessage({name:" test web-view" })
Monitoring message sent from Mini Program	my.onMessage	Monitor messages sent from the Mini Program. web-view component control.
Obtaining current environment	my.getEnv	Obtaining current environment.

Code sample

- `<web-view />` HTML5 page:

```
 <!-- html -->
  <script type="text/javascript" src="https://appx/web-view.min.js"></script>
  // If the HTML5 page needs to be used in a non-mPaaS client at the same time, to avoid the 404 error for the request, refer to the following code writing method.
  // Run the following script in the HTML head.
  <script>
    if (navigator.userAgent.indexOf('AlipayClient') > -1 || navigator.userAgent.indexOf('mPaaSClient') > -1) {
      document.writeln('<script src="https://appx/web-view.min.js"' + '>' + '<' + '/' + 'script>');
    }

  // javascript
  my.navigateTo({url: '../get-user-info/get-user-info'});

  // The web page sends messages to the Mini Program using postMessage.
  my.postMessage({name:"test web-view"});

  // Receives messages from the Mini Program.
  my.onMessage = function(e) {
    console.log(e); //{'sendToWebView': '1'}
  }

  // Determines whether the running is in the Mini Program environment.
  my.getEnv(function(res) {
    console.log(res.miniprogram) // true
  });

  my.startShare();

  </script>
```

- After the message is sent using `my.postMessage`, the Mini Program page executes the method configured by `onMessage`.

```
// The page.js corresponding to the Mini Program page declares the test method.
 // The web-view component in page.axml has onMessage="test" configured.
 // Therefore, after the web page executes my.postMessage, the test method will be executed.
 Page({
   onLoad(e){
     this.webViewContext = my.createWebViewContext('web-view-1');
   },
   test(e){
     my.alert({
       content:JSON.stringify(e.detail),
     });
     this.webViewContext.postMessage({'sendToWebView': '1'});
   },
 )};
```

- `my.getEnv` code sample:

```
// Determines whether the running is in the Mini Program environment.
 my.getEnv(function(res){
     console.log(res.miniprogram); //true
 });
```

You can obtain the URL of the current `<web-view />` when you share it. That is, the `webViewUrl` parameter is returned during the `onShareAppMessage` callback.

```
Page({
   onShareAppMessage(options) {
     console.log(options.webViewUrl)
   }
 });
```

FAQ

How does the HTML5 page pass information to the Mini Program?

The HTML5 page uses the `my.postMessage` interface to transfer data. The code sample is as follows:

```
my.postMessage({key1:"value1",key2:"value2"});
```

How does the Mini Program pass information to the HTML5 page?

`<web-view />` supports two-way communication capabilities. For more details, see Web-view component control.

How to return to the Mini Program in the web-view component?

On a `<web-view />` HTML5 page, you can manually introduce `https://appx/web-view.min.js` (this link can be accessed only from the mPaaS client). Then use my.navigateTo interfaces.

How to upload images on the HTML5 page when the chooseImage interface of the Mini Program is used?

The path of the image obtained can be uploaded after the relevant data is sent to the Mini Program using `my.postMessage()`.

1.9. Component extension

1.9.1. Overview

The extension component library of the Mini Program is an important supplement to the Basic component library. It is an open-source UI component library developed based on Custom component specifications of the Mini Program. The extension component library helps mini-program developers with quick reuse.

Installation

```
$ npm install mini-antui --save
```

Usage

Register the component in the .json file. The following example shows how to register the card component.

```
{
  "usingComponents": {
    "card": "mini-antui/es/card/index",
  }
}
```

Call the component in the .axml file.

```
<card
  thumb="{{thumb}}"
  title="Card Title 2"
  subTitle="Subheading not required 2"
  onClick="onCardClick"
  info="Click the second card"
/>
```

Component update log

For update logs, see changelog.

1.9.2. Navigation layout

1.9.2.1. List

This section describes the list component.

This section describes the list component.

list

Property	Description	Type	Default value
className	Custom class	String	-

slots

slotName	Description
header	Top of the list, which is optional.
footer	Used to render the bottom of the list, which is optional.

list-item

Attribute	Description	Type	Default value
className	Custom class	String	-
thumb	Thumbnail address	String	-
arrow	Whether there are arrows.	Boolean	false
align	Vertical alignment of child elements. The value can be `top`, `middle`, `bottom`.	String	middle
index	Unique index of a list item.	String	-
onClick	The function is called when you tap list-item.	({index, target}) => void	-
last	Whether it is the last item in the list.	Boolean	false
disabled	Tapping forbidden and no hover effect.	Boolean	false
multipleLine	Multiple lines.	Boolean	false
wrap	Whether to break a line. The text will be hidden by default if it is too long.	Boolean	false

slots

slotname	Description
extra	Used to render the list item description on the right, which is optional.
prefix	Used to render the list description on the left, which is optional.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index"
  }
}
```

```
<view>
  <list>
    <view slot="header">
      Head of the list.
    </view>
    <block a:for="{{items}}">
      <list-item
        thumb="{{item.thumb}}"
        arrow="{{item.arrow}}"
        align="{{item.align}}"
        index="{{index}}"
        onClick="onItemClick"
        key="items-{{index}}"
        last="{{index === (items.length - 1)}}"
      >
      {{item.title}}
        <view class="am-list-brief">{{item.brief}}</view>
        <view slot="extra">
          {{item.extra}}
        </view>
      </list-item>
    </block>
    <view slot="footer">
      Tail of the list.
    </view>
  </list>
  <list>
    <view slot="header">
      Head of the list.
    </view>
    <block a:for="{{items2}}">
      <list-item
        thumb="{{item.thumb}}"
        arrow="{{item.arrow}}"
        onClick="onItemClick"
        index="items2-{{index}}"
        key="items2-{{index}}"
        last="{{index === (items2.length - 1)}}"
      >
       {{item.title}}
        <view class="am-list-brief">{{item.brief}}</view>
        <view a:if="{{item.extra}}" slot="extra">
          {{item.extra}}
        </view>
      </list-item>
    </block>
    <view slot="footer">
      Tail of the list.
    </view>
  </list>
</view>
`
```

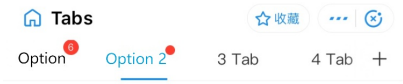
```
Page({
  data: {
    items: [
      {
        title: 'Single-line list',
        extra: 'Detailed information',
      },
    ],
    items2: [
      {
        title: 'Multi-line list',
        arrow: true,
      },
      {
        title: 'Multi-line list',
        arrow: 'up',
      },
      {
        title: 'Multi-line list',
        arrow: 'down',
      },
      {
        title: 'Multi-line list',
        arrow: 'empty',
      },
      {
        title: 'Multi-line list',
      },
    ],
  },
  onItemClick(ev) {
    my.alert({
      content: `Tap line ${ev.index}`,
    });
  },
});
```

1.9.2.2. Tabs

Tabs can be used to switch among different views.

Tabs can be used to switch among different views.

Note: The tabs component does not support the function that the content in the tab adapts to the screen size.



content of Option 2

tabs

Property	Description	Type	Default value	Mandatory
className	Custom class	String	-	false
activeCls	Class of custom active tabBar.	String	-	-
tabs	Tab data, including the option title `title` and badge type `badgeType`. badgeType can be `dot` or `text`. If the value of `badgeType` is not specified, no badge is displayed. The badge text `badgeText` is valid when `badgeType` is set to `text`.	Array<title, badgeType, badgeText>	-	true
activeTab	Index of the current active tabs.	Number	-	true
showPlus	Whether to display the '+' icon.	Boolean	false	false

Property	Description	Type	Default value	Mandatory
onPlusClick	Callback when you tap the '+' icon.	() => {}	-	false
onTabClick	Callback when you tap a tab.	(index: Number) => void	-	false
onChange	Triggered when a tab changes.	(index: Number) => void	-	false
swipeable	Whether you can slide the content to switch the tab.	Boolean	true	false
duration	Duration of the sliding animation when swipeable is true, in ms.	Number	500 (ms)	false
tabBarBackgroundColor	Background color of the tabBar	String	-	false
tabBarActiveTextColor	Text color of the active tabs in the tabBar.	String	-	false
tabBarInactiveTextColor	Text color of the non-active tabs in the tabBar.	String	-	false
tabBarUnderlineColor	Color of the tabBar underline.	String	-	false
tabBarCls	Class of custom tabBar styles.	String	-	false

tab-content

View content

Property	Description	Type	Default value	Mandatory
index	Unique index of a list item.	String	-	-

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "tabs": "mini-antui/es/tabs/index",
    "tab-content": "mini-antui/es/tabs/tab-content/index"
  }
}
```

```
<view>
  <tabs
    tabs="{{tabs}}"
    showPlus="{{true}}"
    onTabClick="handleTabClick"
    onChange="handleTabChange"
    onPlusClick="handlePlusClick"
    activeTab="{{activeTab}}"
  >
    <block a:for="{{tabs}}">
      <tab-content key="{{index}}">
        <view class="tab-content">content of {{item.title}}</view>
      </tab-content>
    </block>
  </tabs>
</view>
```

```
Page({
  data: {
    tabs: [
      {
        title: 'Option',
        badgeType: 'text',
        badgeText: '6',
      },
      {
        title: 'Option 2',
        badgeType: 'dot',
      },
      { title: '3 Tab' },
      { title: '4 Tab' },
      { title: '5 Tab' },
    ],
    activeTab: 2,
  },
  handleTabClick({ index }) {
    this.setData({
      activeTab: index,
    });
  },
  handleTabChange({ index }) {
    this.setData({
      activeTab: index,
    });
  },
  handlePlusClick() {
    my.alert({
      content: 'plus clicked',
    });
  },
});
```

```
.tab-content {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 300px;
}
```

1.9.2.3. Vertical tabs

Vertical tabs (vtabs) is used to switch among different views.

Vertical tabs (vtabs) is used to switch among different views.

vtabs

Property	Description	Type	Default value	Mandatory
activeTab	Index of the current active tabs.	Number	-	true
tabs	Tab data, including the option title `title`, unique anchor point value of the list, and badge type `badgeType`. badgeType can be `dot` or `text`. If the value of `badgeType` is not specified, no badge is displayed. The badge text `badgeText` is valid when `badgeType` is set to `text`.	Array<title, anchor>	-	true
animated	Whether to enable the animation.	Boolean	-	false
swipeable	Whether sliding to switch is valid.	Boolean	-	true
tabBarActiveBgColor	Background color when the tabBar is in active state.	String	-	false
tabBarInactiveBgColor	Background color when the tabBar is in non-active state.	String	-	false
tabBarActiveTextColor	Text color of the active tabs in the tabBar.	String	-	false
tabBarInactiveTextColor	Text color of the non-active tabs in the tabBar.	String	-	false

Property	Description	Type	Default value	Mandatory
tabBarlineColor	Color of the tabBar side line.	String	-	false
onTabClick	Callback when you tap a tab.	(index: Number) => void	-	false
onChange	Triggered when vtab-content changes.	(index: Number) => void	-	false

vtab-content

View content

Property	Description	Type	Default value	Mandatory
anchor	Unique anchor point value of the list.	String	-	true

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "vtabs": "mini-antui/es/vtabs/index",
    "vtab-content": "mini-antui/es/vtabs/vtab-content/index"
  }
}
```

```
<view>
  <vtabs
    tabs="{{tabs}}"
    onTabClick="handleChange"
    onChange="onChange"
    activeTab="{{activeTab}}"
  >
    <block a:for="{{tabs}}">
      <vtab-content anchor="{{item.anchor}}">
        <view style="border: 1px solid #eee; height: 800px; box-sizing: border-box">
          <text>content of {{item.title}}</text>
        </view>
      </vtab-content>
    </block>
  </vtabs>
</view>
```

```
Page({
  data: {
    activeTab: 2,
    tabs: [
      { title: 'Option 2', anchor: 'a', badgeType: 'dot' },
      { title: 'Option', anchor: 'b', badgeType: 'text', badgeText: 'New' },
      { title: 'No more than five characters', anchor: 'c' },
      { title: 'Option 4', anchor: 'd' },
      { title: 'Option 5', anchor: 'e' },
      { title: 'Option 6', anchor: 'f' },
    ],
  },
  handleChange(index) {
    this.setData({
      activeTab: index,
    });
  },
  onChange(index) {
    console.log('onChange', index);
    this.setData({
      activeTab: index,
    });
  },
});
```

1.9.2.4. Card

This section describes the card component.

This section describes the card component.

Property	Description	Type	Default value	Mandatory
thumb	Card thumbnail address	String	-	false
title	Card title	String	-	true

Property	Description	Type	Default value	Mandatory
subTitle	Card subtitle	String	-	false
footer	Text of the footer.	String	-	false
footerImg	Address of the footer image.	String	-	false
onCardClick	Callback when a card is tapped.	(info: Object) => void	-	false
info	Used to send data out when the card is tapped.	String	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "card": "mini-ali-ui/es/card/index"
  }
}
```

```
<card
  thumb="{{thumb}}"
  title="Title"
  subTitle="Subtitle is optional"
  onCardClick="onCardClick"
  footer="Description"
  footerImg="{{footerImg}}"
  info="Tap card"
/>
```

```
Page({
  data: {
    tagData: [
      { date: '2018-05-14', tag: 'Pay mortgage', tagColor: 5 },
      { date: '2018-05-28', tag: 'Housing provident fund', tagColor: 2 },
    ],
  },
  handleSelect() {},
  onMonthChange() {},
});
```

1.9.2.5. Grid

This section describes the grid component.

This section describes the grid component.

Property	Description	Type	Default value	Mandatory
list	Grid data	Array<icon, text>	[]	true
onGridItemClick	Callback when you tap the grid item.	(index: Number) => void	-	false
columnNum	Number of columns that each line displays.	`2` , `3` , `4` , `5`	`3`	false
circular	Whether the corners are rounded.	Boolean	`false`	false
hasLine	Whether there are borders.	Boolean	`true`	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "grid": "mini-antui/es/grid/index"
  }
}
```

```
<grid onGridItemClick="onItemClick" columnNum="{{3}}" list="{{list3}}" />
```

```
Page({
  data: {
    list3: [
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
      {
        icon: 'https://gw.alipayobjects.com/zos/rmsportal/VBqNBOiGYkCjqocXjdUj.png',
        text: 'Title',
        desc: 'Description',
      },
    ],
  },
  onItemClick(ev) {
    my.alert({
      content: ev.detail.index,
    });
  },
});
```

1.9.2.6. Steps

This section describes steps, the progress bar displayed based on steps.

This section describes steps, the progress bar displayed based on steps.

Property	Description	Type	Default value	Mandatory
className	Style of the outermost cover.	String	-	false
activeIndex	Current active step	Number	1	true
failIndex	Current failed step, which is valid only in vertical mode.	Number	0	false
direction	Display direction, which can be `vertical` or `horizontal`.	String	horizontal	false
size	Unified size of icons, in px.	Number	0	false
items	Task details.	Array[{title, description, icon, activeIcon, size}]	[]	true

Detailed description of items properties:

Property	Description	Type	Default value	Mandatory
items.title	Title of step details.	String	-	true
items.description	Description of step details.	String	-	true
items.icon	Icon whose step has not been reached, which is valid only in vertical mode.	String	-	true
items.activeIcon	Icon whose step is reached, which is valid only in vertical mode.	String	-	true
items.size	Size of the icon whose step is reached, in px. This property is valid only in vertical mode.	Number	-	true

Sample code

```
{
  "usingComponents": {
    "steps": "mini-antui/es/steps/index"
  }
}
```

```
<steps
  activeIndex="{{activeIndex}}"
  items="{{items}}"
></steps>
```

```
Page({
  data: {
    activeIndex: 1,
    items: [{
      title: 'Step 1',
      description: 'This is step 1',
    }, {
      title: 'Step 2',
      description: 'This is step 2',
    }, {
      title: 'Step 3',
      description: 'This is step 3',
    }]
  }
});
```

1.9.2.7. Footer

This section describes the page footer component.

This section describes the page footer component.

Property	Description	Type	Default value	Mandatory
copyright	Copyright information	String	-	false
links	Footer link	Array<text, url>	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents":{
    "footer": "mini-antui/es/footer/index"
  }
}
```

```
<view>
  <footer
    copyright="{{copyright}}"
    links="{{links}}" />
</view>
```

```
Page({
 data: {
    copyright: '© 2004-2019 Alipay.com. All rights reserved.',
    links: [
      { text: 'Bottom link', url: '.. /../list/demo/index' },
      { text: 'Bottom link', url: '.. /../card/demo/index' },
    ],
 },
});
```

1.9.2.8. Flex

This section describes CSS flex wrap.

This section describes CSS flex wrap.

flex

Property	Description	Type	Default value	Mandatory
direction	Project positioning direction. The value can be row, row-reverse, column, or column-reverse.	String	row	false
wrap	Line-wrap method of the child element . The value can be nowrap, wrap, or wrap-reverse.	String	nowrap	false
justify	Alignment method on the main axis of the child element. The value can be start, end, center, between, and around.	String	start	false
align	Alignment method on cross axes of the child element. The value can be start, center, end, baseline, and stretch.	String	center	false
alignContent	Alignment method on multiple axes. The value can be start, end, center, between, around, and stretch.	String	stretch	false

flex-item

The style `flex:1` is added for the flex-item component by default to ensure that all items are equally divided in width. The children of the flex container may not be flex-item.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "flex": "mini-antui/es/flex/index",
    "flex-item": "mini-antui/es/flex/flex-item/index"
  }
}
```

```
<view class="flex-container">
  <view class="sub-title">Basic</view>
  <flex>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
  </flex>
  <view style="height: 20px;" />
  <flex>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
  </flex>
  <view style="height: 20px;" />
  <flex>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
    <flex-item><view class="placeholder">Block</view></flex-item>
  </flex>
  <view className="sub-title">Wrap</view>
  <flex wrap="wrap">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <view className="sub-title">Align</view>
  <flex justify="center">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <flex justify="end">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <flex justify="between">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <flex align="start">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline small">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <flex align="end">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline small">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
  <flex align="baseline">
    <view class="placeholder inline">Block</view>
    <view class="placeholder inline small">Block</view>
    <view class="placeholder inline">Block</view>
  </flex>
</view>
```

```
.flex-container {
  padding: 10px;
}

.sub-title {
  color: #888;
  font-size: 14px;
  padding: 30px 0 18px 0;
}

.placeholder {
  background-color: #ebebef;
  color: #bbb;
  text-align: center;
  height: 30px;
  line-height: 30px;
  width: 100%;
}

.placeholder.inline {
  width: 80px;
  margin: 9px 9px 9px 0;
}

.placeholder.small {
  height: 20px;
  line-height: 20px
}
```

```
Page({});
```

1.9.2.9. Pagination

This section describes the pagination component.

This section describes the pagination component.

Property	Description	Type	Default value
Mode	Form of the button. The value can be `text` or `icon`.	String	text
total	Total number of pages.	Number	0
current	Current page number	Number	0
simple	Whether to hide the value.	Boolean	false
disabled	Disabled state.	Boolean	false
prevText	Text for the pagination button to return to the previous page.	String	Previous page
nextText	Text for the pagination button to go to the next page.	String	Next page
btnClass	Form of pagination buttons. Only text buttons are valid.	String	-
onChange	Callback when pagination occurs.	(index: Number) => void	N/A

`prevText` and `nextText` are valid only when mode is set to `text`.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "pagination": "mini-antui/es/pagination/index"
  }
}
```

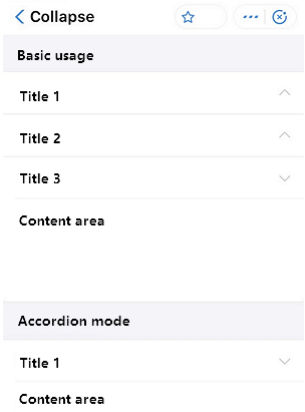
```
<view>
  <view class="demo-title">Basic usage</view>
  <pagination total="{{20}}" current="{{1}}"/>
  <view class="demo-title">Arrow button</view>
  <pagination mode="icon" total="{{20}}" current="{{10}}"/>
  <view class="demo-title">Basic mode</view>
  <pagination simple total="{{20}}" current="{{1}}"/>
  <view class="demo-title">Button disabled</view>
  <pagination total="{{20}}" current="{{1}}" disabled/>
  <view class="demo-title">Custom button text</view>
  <pagination arrow prevText="Previous" nextText="Next" total="{{20}}" current="{{1}}"/>
</view>
```

```
Page({})
```

1.9.2.10. Collapse

This section describes collapse, the collapsible panel.

This section describes collapse, the collapsible panel.



collapse

Property	Description	Type	Default value	Mandatory
activeKey	Key of the current active tab panel.	Array / String	There is no key by default. In accordion mode, the first element is used by default.	false
onChange	Callback when a panel switchover occurs.	(activeKeys: Array): void	-	false
accordion	Accordion mode.	Boolean	false	false
collapseKey	Unique identifier of collapse and the corresponding collapse-item.	String	false	false

collapse-item

Property	Description	Type	Default value	Mandatory
itemKey	Corresponding to activeKey.	String	Unique identifier of the component.	false
header	Content of the panel header.	String	N/A	false
collapseKey	Unique identifier of collapse and the corresponding collapse-item.	String	false	false

If multiple collapse components exist on the page, the collapseKey property of collapse and the corresponding collapse-item are mandatory and must be the same. If only one collapse component exists on the page, collapseKey does not need to be provided.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "collapse": "mini-antui/es/collapse/index",
    "collapse-item": "mini-antui/es/collapse/collapse-item/index"
  }
}
```

```
<view>
  <view class="demo-title">Basic usage</view>
  <collapse
    className="demo-collapse"
    collapseKey="collapse1"
    activeKey="{{['item-11', 'item-13']}}"
    onChange="onChange"
  >
    <collapse-item header="Title 1" itemKey="item-11" collapseKey="collapse1">
      <view class="item-content content1">
        <view>Content area</view>
      </view>
    </collapse-item>
    <collapse-item header="Title 2" itemKey="item-12" collapseKey="collapse1">
      <view class="item-content content2">
        <view>Content area</view>
      </view>
    </collapse-item>
    <collapse-item header="Title 3" itemKey="item-13" collapseKey="collapse1">
      <view class="item-content content3">
        <view>Content area</view>
      </view>
    </collapse-item>
  </collapse>
  <view class="demo-title">Accordion mode</view>
  <collapse
    className="demo-collapse"
    collapseKey="collapse2"
    activeKey="{{['item-21', 'item-23']}}"
    onChange="onChange"
    accordion="{{true}}"
  >
    <collapse-item header="Title 1" itemKey="item-21" collapseKey="collapse2">
      <view class="item-content content1">
        <view>Content area</view>
      </view>
    </collapse-item>
    <collapse-item header="Title 2" itemKey="item-22" collapseKey="collapse2">
      <view class="item-content content2">
        <view>Content area</view>
      </view>
    </collapse-item>
    <collapse-item header="Title3" itemKey="item-23" collapseKey="collapse2">
      <view class="item-content content3">
        <view>Content area</view>
      </view>
    </collapse-item>
  </collapse>
</view>
```

```
.item-content {
  padding: 14px 16px;
  font-size: 17px;
  color: #333;
  line-height: 24px;
}

.content1 {
  height: 200px;
}

.content2 {
  height: 50px;
}

.content3 {
  height: 100px;
}

.demo-title {
  padding: 14px 16px;
  color: #999;
}

.demo-collapse {
  border-bottom: 1px solid #eee;
}
```

```
Page({});
```

1.9.3. Floating layer

1.9.3.1. Popover

This section describes the popover.

This section describes the popover.

popover

Property	Description	Type	Default value	Mandatory
className	Style of the outermost cover.	String	-	false
show	Whether to display the popover.	Boolean	false	true
showMask	Whether to display the mask.	Boolean	true	false
position	Position of the popover. The value can be `top`, `topRight`, `topLeft`, `bottom`, `bottomLeft`, `bottomRight`, `right`, `rightTop`, `rightBottom`, `left`, `leftBottom`, `leftTop`.	String	bottomRight	false

popover-item

Property	Description	Type	Default value	Mandatory
className	Style of the single item.	String	-	false
onItemClick	Single tapping event.	() => void	-	false

Code sample

```
{
  "usingComponents": {
    "popover": "mini-antui/es/popover/index",
    "popover-item": "mini-antui/es/popover/popover-item/index"
  }
}
```

```
<popover
  position="{{position}}"
  show="{{show}}"
  showMask="{{showMask}}"
  onMaskClick="onMaskClick"
>
  <view onTap="onShowPopoverTap">Tap to view.</view>
  <view slot="items">
    <popover-item onItemClick="itemTap1">
      <text>line1</text>
    </popover-item>
    <popover-item>
      <text>line2</text>
    </popover-item>
  </view>
</popover>
```

```
Page({
  data: {
    position: 'bottomRight',
    show: false,
    showMask: true,
  },
  onMaskClick() {
    this.setData({
      show: false,
    });
  },
  onShowPopoverTap() {
    this.setData({
      show: true,
    });
  },
  itemTap1() {
    my.alert({
      content: 'Tap 1',
    });
  },
});
```

1.9.3.2. Filter

This component is used to filter items.

filter

Property	Description	Type	Default value	Mandatory
show	Whether to display.	Boolean	false	false
max	Maximum number of optional items. Only one item can be selected when the value is 1.	Number	10000	false
onChange	Callback when multiple selected items are submitted.	(e: Object) => void	-	false

filter-item

Property	Description	Type	Default value	Mandatory
className	The custom style.	String	-	false
value	Value	String	-	true
id	Custom identifier.	String	-	false
selected	Selected by default.	Boolean	false	false
onChange	Callback when a selected item is submitted.	(e: Object) => void	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "filter": "mini-antui/es/filter/index",
    "filter-item": "mini-antui/es/filter/filter-item/index"
  }
}
```

```
<filter show="{{show}}" max="{{5}}" onChange="handleCallBack">
  <block a:for="{{items}}">
    <filter-item value="{{item.value}}" id="{{item.id}}" selected="{{item.selected}}"/>
  </block>
</filter>
```

```
Page({
  data: {
    show: true,
    items: [
      { id: 1, value: 'clothes', selected: true },
      { id: 1, value: 'cabinet' },
      { id: 1, value: 'clothes hanger' },
      { id: 3, value: 'digital product' },
      { id: 4, value: 'security door' },
      { id: 5, value: 'chair' },
      { id: 7, value: 'display' },
      { id: 6, value: 'the latest digital product' },
      { id: 8, value: 'video game base of a brand' },
    ]
  },
  handleCallBack(data) {
    my.alert({
      content: data
    });
  },
  toggleFilter() {
    this.setData({
      show: !this.data.show,
    });
  }
});
```

1.9.3.3. Modal

This section describes the modal dialog box.

This section describes the modal dialog box.

Property	Description	Type	Default value
className	Custom class.	String	-
show	Whether to display `modal`	Boolean	false
showClose	Whether to render `close`	Boolean	true
closeType	Type of the icon for closure. The value can be 0 - gray icon or 1 - white icon.	String	0
onModalClick	Callback when you tap the `footer` part.	() => void	-
onModalClose	Callback when you tap `close`. The value does not need to be specified when `showClose` is false.	() => void	-
topImage	Top image.	String	-
topImageSize	Rule of the top image, which can be `lg`, `md`, `sm`	String	md
advice	Whether it is an operation pop-up window.	Boolean	false

slots

slotName	Description
header	Modal head, which is optional.
footer	Modal tail, which is optional.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "modal": "mini-antui/es/modal/index"
  }
}
```

```
<view>
  <button onTap="openModal">Enables modal.</button>
  <modal
    show="{{modalOpened}}"
    onModalClick="onModalClick"
    onModalClose="onModalClose"
    topImage="https://gw.alipayobjects.com/zos/rmsportal/yFeFExbGpDxvDYnKHcrs.png"
  >
    <view slot="header">Single line title.</view>
    Explains the current status and provides solutions, preferably no more than two lines.
    <view slot="footer">OK</view>
  </modal>
</view>
```

```
Page({
  data: {
    modalOpened: false,
  },
  openModal() {
    this.setData({
      modalOpened: true,
    });
  },
  onModalClick() {
    this.setData({
      modalOpened: false,
    });
  },
  onModalClose() {
    this.setData({
      modalOpened: false,
    });
  }
});
```

1.9.3.4. Popup

This section describes the pop-up window.

This section describes the pop-up window.

Property	Description	Type	Default value	Mandatory
className	Custom class.	String	-	false
show	Whether to display the menu.	Boolean	false	false
animation	Whether to enable the animation.	Boolean	true	false
mask	Whether to display the mask. If no, tapping the external area will not trigger the onClose event.	Boolean	true	true
position	Pop-up direction of the pop-up window. The vale can be bottom, left, top, or right.	String	bottom	false
disableScroll	Whether to disable page scroll when the mask is displayed.	Boolean	true	false
zIndex	Level of the pop-up window.	Number	0	false

slots

You can define the part to be displayed in the popup component. For details, see the following example.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "popup": "mini-antui/es/popup/index"
  }
}
```

```
<view>
  <view class="btn-container">
    <button onTap="onTopBtnTap">Pops up a window</button>
  </view>
  <popup show="{{showTop}}" position="top" onClose="onPopupClose">
    <view style="height: 200px; background: #fff; display: flex; justify-content: center; align-items: center;">hello world</view>
  </popup>
</view>
```

```
Page({
  data: {
    showTop: false,
  },
  onTopBtnTap() {
    this.setData({
      showTop: true,
    });
  },
  onPopupClose() {
    this.setData({
      showTop: false,
    });
  },
});
```

1.9.4. Result class

1.9.4.1. PageResult

This section describes PageResult, the error page.

This section describes PageResult, the error page.

Property	Description	Type	Default value	Mandatory
type	Type of the error page. This property is optional. The value can be `network` (network error), `busy` (busy service), `error` (abnormal service), `empty` (empty state), or `logoff` (user logout)	String	network	false
local	Whether it is a local error	Boolean	false	false
title	Title of the error message	String	-	false
brief	Brief of the error message	String	-	false

Sample code

```
{
  "defaultTitle": "Error feedback",
  "usingComponents": {
    "page-result": "mini-antui/es/page-result/index"
  }
}
```

```
<page-result
  type="network"
  title = "The network signal strength is poor"
  brief="The page cannot be opened due to poor network signal strength"
/>
<page-result
  type="network"
  title = "The network signal strength is poor"
  brief="The page cannot be opened due to poor network signal strength"
>
  <view class="am-page-result-btns">
    <view onTap="backHome">Back to the homepage</view>
    <view>Button example</view>
  </view>
</page-result>
```

1.9.4.2. Message

This section describes the message.

This section describes the message.

Property	Description	Type	Default value	Mandatory
className	Custom class	String	-	false
type	The messages can be divided into five types, success, fail, info, warn, waiting, and info. The default type is success	String	success	false
title	The title	String	-	true
subTitle	Subtitle	String	-	false
mainButton	Text and relevant availability of the main button	Object<buttonText, disabled>	-	false
subButton	Text and relevant availability of the secondary button	Object<buttonText, disabled>	-	false
onTapMain	Click function of the main button	() => {}	-	false
onTapSub	Click function of the secondary button	() => {}	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "message": "mini-antui/es/message/index"
  }
}
```

```
<view>
  <message
    title="{{title}}"
    subTitle="{{subTitle}}"
    type="success"
    mainButton="{{messageButton.mainButton}}"
    subButton="{{messageButton.subButton}}"
    onTapMain="goBack">
  </message>
</view>
```

```
Page({
  data: {
    title: "Operation succeeded",
    subTitle: "The content details can be folded. No more than two lines is recommended",
    messageButton: {
      mainButton: {
        buttonText: "Main operation"
      },
      subButton: {
        buttonText: "Auxiliary operation"
      }
    }
  },
  goBack() {
    my.navigateBack();
  }
});
```

1.9.5. Reminders

1.9.5.1. Tips

This topic describes tips. Tips can be divided into two types, `tips-dialog` and `tips-plain`.

tips-dialog

Attribute	Description	Type	Default value	Mandatory
className	Custom class	String	-	false
show	Whether to display the component.	Boolean	true	false
type	Style of tips. The value can be `dialog` or `rectangle`, indicating the dialog box or rectangle, respectively.	String	dialog	false
onCloseTap	Callback when you tap the icon to close and when `type` is set to `rectangle`.	() => void	-	false
iconUrl	Displays the URL of the icon.	String	-	false

slots

slotName	Description
content	Used to render the content of the tip.
operation	Used to render the operation area on the right.

tips-plain

Attribute	Description	Type	Default value	Mandatory
className	Custom class	String	-	false
time	Time of automatic close, in ms.	Number	5000 (ms)	false
onClose	Callback and closes the dialog box.	() => void	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "tips-dialog": "mini-antui/es/tips/tips-dialog/index",
    "tips-plain": "mini-antui/es/tips/tips-plain/index"
  }
}
```

tips-dialog

```
<view>
  <tips-dialog
    show="{{showDialog}}"
    className="dialog"
    type="dialog"
  >
    <view class="content" slot="content">
      <view>hello,</view>
      <view>Welcome to use the mini-program extension library mini-antui</view>
    </view>
    <view slot="operation" class="opt-button" onTap="onDialogTap">OK</view>
  </tips-dialog>
  <tips-dialog
    iconUrl="https://gw.alipayobjects.com/zos/rmsportal/AzRAgQXlnNbEwQRvEwiu.png"
    type="rectangle"
    className="rectangle"
    onCloseTap="onCloseTap"
    show="{{showRectangle}}">
    <view class="content" slot="content">
      Add "City Service" to homepage
    </view>
    <view slot="operation" class="add-home" onTap="onRectangleTap">Add now</view>
  </tips-dialog>
</view>
```

```
Page({
  data: {
    showRectangle: true,
    showDialog: true,
  },
  onCloseTap() {
    this.setData({
      showRectangle: false,
    });
  },
  onRectangleTap() {
    my.alert({
      content: 'do something',
    });
  },
  onDialogTap() {
    this.setData({
      showDialog: false,
    });
  },
});
```

```
.rectangle {
  position: fixed;
  bottom: 100px;
}

.dialog {
  position: fixed;
  bottom: 10px;
}

.content {
  font-size: 14px;
  color: #fff;
}

.opt-button {
  width: 51px;
  height: 27px;
  display: flex;
  justify-content: center;
  align-items: center;
  color: #fff;
  font-size: 12px;
  border: #68BAF7 solid 1rpx;
}

.add-home {
  width: 72px;
  height: 27px;
  display: flex;
  justify-content: center;
  align-items: center;
  background-color: #56ADEB;
  color: #fff;
  font-size: 14px;
}
```

tips-plain

```
<tips-plain onClose="onClose" time="{{time}}">{{content}}</tips-plain>
```

```
Page({
  data: {
    content: 'OK',
    time: 2000,
  },
  onClose() {
    my.alert({
      title: '12321'
    });
  }
});
```

1.9.5.2. Notice

This topic describes the notice bar.

This topic describes the notice bar.

Property	Description	Type	Default value
Mode	Type of the notice, which can be: `link` or `closable`.	String	''
action	The notice shows text.	String	''
actionCls	The notice shows the custom class of text.	String	''
show	Whether to display the notice bar.	Boolean	true
onClick	Callback when you tap the button.	() => void	-
enableMarquee	Whether to enable the animation.	Boolean	false
marqueeProps	marquee parameters, where `loop` indicates whether to loop, `leading` indicates a pause before the animation starts, `trailing` indicates an interval at animations when `loop` is true, and `fps` indicates the animation frame rate.	Object<loop, leading, trailing, fps>	{loop: false, leading: 500, trailing: 800, fps: 40 }

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "notice": "mini-antui/es/notice/index"
  }
}
```

```
<view class="demo-title">Notice bar</view>
<view class="demo-item">
  <notice>Due to the upgrade of the national citizenship system, please add a bank card.</notice>
</view>
<view class="demo-item">
  <notice mode="link" onClick="linkClick">Due to the upgrade of the national citizenship system, please add a bank card.</notice>
</view>
<view class="demo-item">
  <notice mode="closable" onClick="closableClick" show="{{closeShow}}">Due to the upgrade of the national citizenship system, please add a bank card.
</notice>
</view>
<view class="demo-item">
  <notice mode="link" action="Check it out" onClick="linkActionClick">Due to the upgrade of the national citizenship system, please add a bank card.<
/notice>
</view>
<view class="demo-item">
  <notice mode="closable" action="Don't remind again" onClick="closableActionClick" show="{{closeActionShow}}">Due to the upgrade of the national cit
izenship system, please add a bank card.</notice>
</view>
```

```
Page({
  data:{
    closeShow:true,
    closeActionShow:true
  },
  linkClick() {
    my.showToast({
      content: 'You have tapped the icon Link NoticeBar',
      duration: 3000
    });
  },
  closableClick() {
    this.setData({
      closeShow:false
    })
    my.showToast({
      content: ''You have tapped the icon close NoticeBar',
      duration: 3000
    });
  },
  linkActionClick() {
    my.showToast({
      content: 'You have tapped the text Link NoticeBar',
      duration: 3000
    });
  },
  closableActionClick() {
    this.setData({
      closeActionShow:false
    })
    my.showToast({
      content: 'You have tapped the text close NoticeBar',
      duration: 3000
    });
  }
})
```

1.9.5.3. Badge

The badge is a red dot, a number, or text used to remind users of what to do or the number of updates.

The badge is a red dot, a number, or text used to remind users of what to do or the number of updates.

Property	Description	Type	Default value	Mandatory
text	Number or text displayed.	String / Number	-	false
dot	Displays a red dot only.	Boolean	-	false
overflowCount	Displays the maximum number allowed. A plus sign (+) is used to indicate the excess part.	Number	99	false

slots

slotName	Description
inner	Used to render the internal area when the badge is used as wrapper. This property is optional.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "badge": "mini-antui/es/badge/index"
  }
}
```

```
<view>
  <list>
    <block a:for="{{items}}">
      <list-item
        arrow="{{true}}"
        index="{{index}}"
        key="items-{{index}}"
        last="{{index === (items.length - 1)}}"
      >
        <view>
          <badge a:if="{{item.isWrap}}" text="{{item.text}}" dot="{{item.dot}}">
            <view slot="inner" style="height: 26px; width: 26px; background-color: #ddd;"></view>
          </badge>
          <text style="margin-left: {{ item.isWrap ? '12px' : '0' }}">{{item.intro}}</text>
        </view>
        <view slot="extra">
          <badge a:if="{{!item.isWrap}}" text="{{item.text}}" dot="{{item.dot}}" overflowCount="{{item.overflowCount}}" />
        </view>
      </list-item>
    </block>
  </list>
</view>
```

```
Page({
  data: {
    items: [
      {
        dot: true,
        text: '',
        isWrap: true,
        intro: 'Dot Badge',
      },
      {
        dot: false,
        text: 1,
        isWrap: true,
        intro: 'Text Badge',
      },
      {
        dot: false,
        text: 99,
        isWrap: false,
        intro: 'Number',
      },
      {
        dot: false,
        text: 100,
        overflowCount: 99,
        isWrap: false,
        intro: 'The number exceeds overflowCount',
      },
      {
        dot: false,
        text: 'new',
        isWrap: false,
        intro: 'Text',
      },
    ],
  },
});
```

1.9.6. Form class

1.9.6.1. InputItem

This section describes InputItem, the text input component.

This section describes InputItem, the text input component.

Property	Description	Type	Default value
className	Custom class.	String	''
labelCls	Class of the custom labels.	String	''
inputCls	Class of the custom inputs.	String	''
last	Whether it is the last line.	Boolean	false
value	Initial content	String	''
name	The component name, which is used to submit the form to obtain data.	String	''

Property	Description	Type	Default value
type	Type of the input, which can be `text`, `number`, `idcard`, or `digit.`	String	text
password	Whether the type is password.	Boolean	false
placeholder	The placeholder.	String	''
placeholderStyle	The specified placeholder style.	String	''
placeholderClass	The specified placeholder style class.	String	''
disabled	Whether to disable the button.	Boolean	false
maxlength	The maximum length.	Number	140
focus	Obtain the focus.	Boolean	false
clear	Whether the clear function is enabled. This property is valid only when the value of disabled is false.	Boolean	false
onInput	The input event is triggered when keyboard input occurs.	(e: Object) => void	-
onConfirm	Triggered when the keyboard is tapped.	(e: Object) => void	-
onFocus	Triggered when the focus action occurs.	(e: Object) => void	-
onBlur	Triggered when the focus is lost.	(e: Object) => void	-
onClear	Triggered when you tap to clear the icon.	() => void	-

slots

slot name	Description
extra	Used to render the description of input-item on the right. This property is optional.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "input-item": "mini-antui/es/input-item/index",
    "picker-item": "mini-antui/es/picker-item/index"
  }
}
```

```
<view>
  <view style="margin-top: 10px;" />
  <list>
    <input-item
      data-field="cardNo"
      clear="{{true}}"
      value="{{cardNo}}"
      className="dadada"
      placeholder="Bank card number"
      focus="{{inputFocus}}"
      onInput="onItemInput"
      onFocus="onItemFocus"
      onBlur="onItemBlur"
      onConfirm="onItemConfirm"
      onClear="onClear"
    >
      Card number
      <view slot="extra" class="extra" onTap="onExtraTap"></view>
    </input-item>
    <picker-item
      data-field="bank"
      placeholder="Select the card-issuing bank"
      value="{{bank}}"
      onPickerTap="onPickerTap"
    >
      Card-issuing bank
    </picker-item>
    <input-item
      data-field="name"
      placeholder="Name"
      type="text"
      value="{{name}}"
      clear="{{true}}"
      onInput="onItemInput"
      onClear="onClear"
    >
      Name
    </input-item>
    <input-item
      data-field="password"
      placeholder="Password"
      password
    >
      Password
    </input-item>
    <input-item
      data-field="remark"
      placeholder="Remarks"
      last="{{true}}"
    />
  </list>
  <view style="margin: 10px;">
    <button type="primary" onTap="onAutoFocus">Focus</button>
  <view>
</view>
```

```
const banks = ['MYbank', 'China Construction Bank', 'Industrial and Commercial Bank of China', 'Shanghai Pudong Development Bank'];

Page({
  data: {
    cardNo: '1234****',
    inputFocus: true,
    bank: '',
    name: '',
  },
  onAutoFocus() {
    this.setData({
      inputFocus: true,
    });
  },
  onExtraTap() {
    my.alert({
      content: 'extra tapped',
    });
  },
  onItemInput(e) {
    this.setData({
      [e.target.dataset.field]: e.detail.value,
    });
  },
  onItemFocus() {
    this.setData({
      inputFocus: false,
    });
  },
  onItemBlur() {},
  onItemConfirm() {},
  onClear(e) {
    this.setData({
      [e.target.dataset.field]: '',
    });
  },
  onPickerTap() {
    my.showActionSheet({
      title: 'Select the card-issuing bank',
      items: banks,
      success: (res) => {
        this.setData({
          bank: banks[res.index],
        });
      },
    });
  },
});
```

```
.extra {
  background-image: url('https://gw.alipayobjects.com/zos/rmsportal/dOfSJfWQvYdvsZiJStvg.svg');
  background-size: contain;
  background-repeat: no-repeat;
  background-position: right center;
  opacity: 0.2;
  height: 20px;
  width: 20px;
  padding-left: 10px;
}
```

1.9.6.2. PickerItem

This section describes PickerItem, the selection input.

This section describes PickerItem, the selection input.

Property	Description	Type	Default value
className	Custom class	String	-
labelCls	Class of the custom labels.	String	-
pickerCls	Class of the custom areas.	String	-
last	Whether it is the last line.	Boolean	false
value	Initial content	String	-
name	The component name, which is used to submit the form to obtain data.	String	-
placeholder	The placeholder.	String	-
onPickerTap	Triggered when you tap pickeritem.	(e: Object) => void	-

slots

slotname	Description
extra	Used to render the description of picker-item on the right. This property is optional.

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "picker-item": "mini-antui/es/picker-item/index",
    "input-item": "mini-antui/es/input-item/index"
  }
}
```

```
<view>
  <list>
    <input-item
      data-field="password"
      placeholder="Password"
      password
    >
      Password
    </input-item>
    <picker-item
      data-field="bank"
      placeholder="Select the card-issuing bank"
      value="{{bank}}"
      onPickerTap="onSelect"
    >
      Card-issuing bank
    </picker-item>
  </list>
</view>
```

```
const banks = ['MYbank', 'China Construction Bank', 'Industrial and Commercial Bank of China', 'Shanghai Pudong Development Bank'];

Page({
  data: {
    bank: '',
  },
  onSelect() {
    my.showActionSheet({
      title: 'Select the card-issuing bank',
      items: banks,
      success: (res) => {
        this.setData({
          bank: banks[res.index],
        });
      },
    });
  },
});
```

1.9.6.3. AmountInput

This section describes AmountInput, an amount input box.

This section describes AmountInput, an amount input box.

Property	Description	Type	Default value	Mandatory
type	Type of the input, which can be digit or number.	String	number	false
title	Title in the upper left.	String	-	false
extra	Description in the lower left.	String	-	false
value	Current value of the input box.	String	-	false
btnText	Text of the button in the lower right.	String	-	false
placeholder	placeholder	String	-	false
focus	Obtains the cursor automatically.	Boolean	false	false

Property	Description	Type	Default value	Mandatory
onInput	Triggered when keyboard input occurs.	(e: Object) => void	-	false
onFocus	Triggered when the focus is obtained.	(e: Object) => void	-	false
onBlur	Triggered when the focus is lost.	(e: Object) => void	-	false
onConfirm	Triggered when the keyboard is tapped.	(e: Object) => void	-	false
onClear	Triggered when the clear icon is tapped.	() => void	-	false
onButtonClick	Triggered when the button in the lower right is tapped.	() => void	-	false
maxLength	Maximum number of characters allowed.	Number	-	false
controlled	Whether the component is a controlled component. If the value is true, the content of value is fully controlled by setData.	Boolean	false	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "amount-input": "mini-antui/es/amount-input/index"
  }
}
```

```
<view>
  <amount-input
    type="digit"
    title="Amount to transfer in"
    extra="You are advised to transfer in RMB 100 or more"
    placeholder="Enter the amount to transfer in"
    value="{{value}}"
    maxLength="5"
    focus="{{true}}"
    btnText="Withdraw all"
    onClear="onInputClear"
    onInput="onInput"
    onConfirm="onInputConfirm" />
</view>
```

```
Page({
  data: {
    value: 200,
  },
  onInputClear() {
    this.setData({
      value: '',
    });
  },
  onInputConfirm() {
    my.alert({
      content: 'confirmed',
    });
  },
  onInput(e) {
    const { value } = e.detail;
    this.setData({
      value,
    });
  },
  onButtonClick() {
    my.alert({
      content: 'button clicked',
    });
  },
  onInputFocus() {},
  onInputBlur() {},
});
```

1.9.6.4. SearchBar

This section describes SearchBar, a search box.

This section describes SearchBar, a search box.

Property	Description	Type	Default value	Mandatory
value	Current value of SearchBar.	String	-	false
placeholder	placeholder	String	-	false
focus	Obtains the cursor automatically.	Boolean	false	false
onInput	Triggered when keyboard input occurs.	(value: String) => void	-	false
onClear	Triggered when the clear icon is tapped.	(val: String) => void	-	false
onFocus	Triggered when the focus is obtained.	() => void	-	false
onBlur	Triggered when the focus is lost.	() => void	-	false
onCancel	Triggered when you tap “cancel”.	() => void	-	false
onSubmit	Triggered when you tap Enter on the keyboard.	(val: String) => void	-	false
disabled	Whether to disable.	Boolean	-	false
maxLength	Maximum number of characters allowed.	Number	-	false
showCancelButton	Whether to always display the cancel button.	Boolean	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "search-bar": "mini-antui/es/search-bar/index"
  }
}
```

```
<view>
  <search-bar
    value="{{value}}"
    placeholder="Search"
    onInput="handleInput"
    onClear="handleClear"
    onFocus="handleFocus"
    onBlur="handleBlur"
    onCancel="handleCancel"
    onSubmit="handleSubmit"
    showCancelButton="{{false}}" />
</view>
```

```
Page({
  data: {
    value: 'Fine food',
  },
  handleInput(value) {
    this.setData({
      value,
    });
  },
  handleClear(value) {
    this.setData({
      value: '',
    });
  },
  handleFocus() {},
  handleBlur() {},
  handleCancel() {
    this.setData({
      value: '',
    });
  },
  handleSubmit(value) {
    my.alert({
      content: value,
    });
  },
});
```

1.9.6.5. AMCheckBox

This section describes AMCheckBox, a checkbox.

This section describes AMCheckBox, a checkbox.

Property	Description	Type	Default value	Mandatory
value	The component value, which is the value carried by the change event when the component is selected.	String	-	false
checked	Whether the current item is selected. You can use this attribute to set the item as selected by default.	Boolean	false	false
disabled	Whether to disable the button.	Boolean	false	false
onChange	Callback function triggered by the change event.	(e: Object) => void	-	false
id	Used in combination with the “for” attribute of the label component.	String	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "am-checkbox": "mini-antui/es/am-checkbox/index"
  }
}
```

```
<list>
  <view slot="header">
    List and checkbox
  </view>
  <block a:for="{{items}}">
    <list-item
      thumb=""
      arrow="{{false}}"
      index="{{index}}"
      key="items-{{index}}"
      last="{{index === (items.length - 1)}}"
    >
      <view slot="prefix" style="display: flex; align-items: center;">
        <am-checkbox id="{{item.id}}" data-name="{{item.value}}" disabled="{{item.disabled}}" checked="{{item.checked}}" onChange="onChange" />
      </view>
      <label for="{{item.id}}">{{item.title}}</label>
    </list-item>
  </block>
</list>
<view style="padding: 16px;">
  <view style="color: #888; font-size: 14px;">
    Protocol
  </view>
  <view style="margin-top: 10px;">
    <label style="display: flex; line-height: 24px;">
      <am-checkbox />
      <text style="text-indent: 8px; color: #888">Agree with [Credit Payment Service Contract]</text>
    </label>
  </view>
</view>
<view style="padding: 16px; background-color: #fff;">
  <form onSubmit="onSubmit" onReset="onReset">
    <view>
      <view style="color: #666; font-size: 14px; margin-bottom: 5px;">Select the frameworks you have used:</view>
      <view>
        <checkbox-group name="libs">
          <label a:for="{{items2}}" style="display: flex; align-items: center; height: 30px;">
            <am-checkbox value="{{item.name}}" checked="{{item.checked}}" disabled="{{item.disabled}}" />
            <text style="color: #888; font-size: 14px; margin-left: 8px;">{{item.value}}</text>
          </label>
        </checkbox-group>
      </view>
      <view style="margin-top: 10px;">
        <button type="primary" size="mini" formType="submit">submit</button>
      </view>
    </view>
  </form>
</view>
```

```
Page({
  data: {
    items: [
      { checked: true, disabled: false, value: 'a', title: 'Checkbox - selected by default', id: 'checkbox1' },
      { checked: false, disabled: false, value: 'b', title: 'Checkbox - unselected by default', id: 'checkbox2' },
      { checked: true, disabled: true, value: 'c', title: 'Checkbox - disabled is selected by default', id: 'checkbox3' },
      { checked: false, disabled: true, value: 'd', title: 'Checkbox - disabled is unselected by default', id: 'checkbox4' },
    ],
    items2: [
      { name: 'react', value: 'React', checked: true },
      { name: 'vue', value: 'Vue.js' },
      { name: 'ember', value: 'Ember.js' },
      { name: 'backbone', value: 'Backbone.js', disabled: true },
    ],
  },
  onSubmit(e) {
    my.alert({
      content: `You have selected ${e.detail.value.libs.join(', ')}`,
    });
  },
  onReset() {},
  onChange(e) { console.log(e); },
});
```

1.9.7. Gesture class

1.9.7.1. Swipe action

This section describes SwipeAction, the slidable cell.

This section describes SwipeAction, the slidable cell.

Property	Description	Type	Default value
className	Custom class	String	-
right	Sliding options, no more than two.	Array[Object{type: `edit` / `delete`, text: string}]	[]
onRightItemClick	The click event of a swiped cell.	({index, detail, extra, done}) => void	({index, detail, extra, done}) => void
restore	Restores the component to its initial state. If there are multiple swipe-action components, when you slide one of them, you need to set the `restore` value of other components to `true`. This avoids the situation where multiple swipe-actions care active on one page at the same time.	Boolean	false
onSwipeStart	Callback when sliding starts.	EventHandle	(e: Object) => void
extra	Affiliated information, obtained in the `onRightItemClick` callback.	String	-

Sample code

```
{
  "defaultTitle": "SwipeAction",
  "usingComponents": {
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index",
    "swipe-action": "mini-antui/es/swipe-action/index"
  }
}
```

```
<view>
  <list>
    <view a:for="{{list}}" key="{{item.content}}">
      <swipe-action
        index="{{index}}"
        restore="{{swipeIndex === null || swipeIndex !== index}}"
        right="{{item.right}}"
        onRightItemClick="onRightItemClick"
        onSwipeStart="onSwipeStart"
        extra="item{{index}}"
      >
        <list-item
          arrow="horizontal"
          index="{{index}}"
          key="items-{{index}}"
          onClick="onItemClick"
          last="{{index === list.length - 1}}"
        >
          {{item.content}}
        </list-item>
      </swipe-action>
    </view>
  </list>
</view>
```

```
Page({
  data: {
    swipeIndex: null,
    list: [
      { right: [{ type: 'delete', text: 'Delete' }], content: 'AAA' },
      { right: [{ type: 'edit', text: 'Remove from collection' }, { type: 'delete', text: 'Delete' }], content: 'BBB' },
      { right: [{ type: 'delete', text: 'Delete' }], content: 'CCC' },
    ],
  },
  onRightItemClick(e) {
    const { type } = e.detail;
    my.confirm({
      title: 'Reminder',
      content: `${e.index}-${e.extra}-${JSON.stringify(e.detail)}`,
      confirmButtonText: 'OK',
      cancelButtonText: 'Cancel'
      success: (result) => {
        const { list } = this.data;
        if (result.confirm) {
          if (type === 'delete') {
            list.splice(this.data.swipeIndex, 1);
            this.setData({
              list: [...list],
            });
          }

          my.showToast({
            content: 'OK => Undo the result of sliding to delete',
          });
          e.done();
        } else {
          my.showToast({
            content: 'Cancel => Keep the result of sliding to delete',
          });
        }
      },
    });
  },
  onItemClick(e) {
    my.alert({
      content: `dada${e.index}`,
    });
  },
  onSwipeStart(e) {
    this.setData({
      swipeIndex: e.index,
    });
  },
});
```

1.9.8. Others

1.9.8.1. Calendar

This section describes the calendar component.

This section describes the calendar component.

Property	Description	Type	Default value	Mandatory
type	Type of the calendar, which can be `single` - single day or `range` - date range.	String	single	false
tagData	Tag data, including `date`, `tag`, `disable`, and `tagColor`. The value of `tagColor` can be #f5a911, #e8541e, #07a89b, #108ee9, or #b5b5b5.	Array<date, tag, tagColor>	-	false
onSelect	Callback when you select a range.	([startDate, endDate]) => void	-	false
onMonthChange	Callback when you tap to change the month. This property has two parameters, currentMonth and prevMonth, indicating the months after and before the change, respectively.	(currentMonth, prevMonth) => void	-	false
onSelectHasDisableDate	The range selected contains invalid dates.	(currentMonth, prevMonth) => void	-	false

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents":{
    "calendar": "mini-antui/es/calendar/index"
  }
}
```

```
<view>
  <calendar
    type="single"
    tagData="{{tagData}}"
    onSelect="handleSelect" />
</view>
```

```
Page({
  data: {
    tagData: [
      { date: '2018-05-14', tag: 'Pay mortgage', tagColor: 5 },
      { date: '2018-05-28', tag: 'Housing provident fund', tagColor: 2 },
    ],
  },
  handleSelect() {},
  onMonthChange() {},
});
```

1.9.8.2. Stepper

The stepper is used to increase or decrease the current value.

The stepper is used to increase or decrease the current value.

Property	Description	Type	Default value	Mandatory
min	The minimum value.	Number	0	true
max	The maximum value.	Number	10000	true
value	Initial value	Number	10	true
step	Number of steps changed each time, which can be a decimal.	Number	1	false
onChange	Callback when a change occurs.	(value: Number) => void	-	false
disabled	Whether the function is disabled.	Boolean	false	false
readOnly	Whether the input is readonly.	Boolean	false	false
showNumber	Whether to display the value. The value is not displayed by default.	Boolean	false	false

Sample code

```
{
  "defaultTitle": "Stepper",
  "usingComponents":{
    "stepper": "mini-antui/es/stepper/index",
    "list": "mini-antui/es/list/index",
    "list-item": "mini-antui/es/list/list-item/index"
  }
}
```

```
<list>
  <list-item disabled="{{true}}">
    Show number value
    <view slot="extra">
      <stepper onChange="callBackFn" step="{{1}}" showNumber readOnly="{{false}}" value="{{value}}" min="{{2}}" max="{{12}}" />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    Do not show number value
    <view slot="extra">
      <stepper onChange="callBackFn" step="{{1}}" readOnly="{{false}}" value="{{value}}" min="{{2}}" max="{{12}}" />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    Disabled
    <view slot="extra">
      <stepper onChange="callBackFn" showNumber value="{{11}}" min="{{2}}" max="{{12}}" disabled />
    </view>
  </list-item>
  <list-item disabled="{{true}}">
    readOnly
    <view slot="extra">
      <stepper onChange="callBackFn" showNumber value="{{11}}" min="{{2}}" max="{{12}}" readOnly />
    </view>
  </list-item>
  <list-item>
    <button onTap="modifyValue">Change the initial value of stepper.</button>
  </list-item>
</list>
```

```
Page({
  data: {
    value: 8,
  },
  callBackFn(value){
   console.log(value);
  },
  modifyValue() {
    this.setData({
      value: this.data.value + 1,
    });
  }
});
```

1.9.8.3. AMIcon

This section describes icon component AMIcon.

This section describes icon component AMIcon.

Property	Description	Type	Default value	Mandatory
type	Type of the icon. For details, scan the QR code above to view.	String	-	true
size	The icon size in pixels.	String	-	false
color	The icon color, which is consistent with the CSS color.	String	-	false

Icon Style	Valid type Value
Basic	`arrow-left` , `arrow-up` , `arrow-right` , `arrow-down` , `cross` , `plus`
Outline	`close-o` , `dislike-o` , `heart-o` , `help-o` , `like-o` , `location-o` , `info-o` , `success-o` , `wait-o` , `warning-o` , `star-o` , `download` , `friends` , `circle` , `delete` , `charge` , `card` , `notice` , `qrcode` , `reload` , `scan` , `money` , `search` , `setting` , `share` , `zoom-in` , `zoom-out`
Solid	`close` , `dislike` , `heart` , `help` , `like` , `location` , `info` , `success` , `wait` , `warning` , `star`

Sample code

```
{
  "defaultTitle": "AntUI component library of the Mini Program",
  "usingComponents": {
    "am-icon": "mini-antui/es/am-icon/index",
  }
}
```

```
<view>
  <am-icon type="like" size="{{24}}" color="#333" />
</view>
```

1.10. API

1.10.1. Overview

The mPaaS framework provides developers with more JSAPI and OpenAPI capabilities, and also provide users with a variety of convenient services through the MINI Framework.

The mPaaS framework provides developers with more JSAPI and OpenAPI capabilities, and also provide users with a variety of convenient services through the MINI Framework.

Description

APIs starting with `my.on` are used to monitor system events and receive a `callback` function as a parameter. When the event is triggered, the `callback` function is called, and the `callback` function can be passed to the corresponding APIs starting with `my.off` to remove the monitoring. If you call the APIs starting with `my.off` directly, all the monitorings are removed. For example:

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    // Remove monitoring when the page is unloaded
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
});
```

Other APIs receive an object as a parameter. You can specify success (call success), fail (call failure), or complete (call success or failure) to receive the interface call result. The result of callback is typically an object unless otherwise specified, and `error` / `errorMessage` indicates that the call is failed. The return value after the call is a `promise` object. For example:

```
my.httpRequest({
  url: '/x.htm',
  success:(res1) => {
  },
}).then((res2) => {
  // res1 === res2
},(res2) => {
  console.log(res.error, res.errorMessage);
})
```

1.10.2. API list

This document includes all the APIs involved in mPaaS Mini Program. See the corresponding API documents for the specific API details.

UI

Navigation bar

Name	Features
my.getTitleColor	Get the background color of the navigation bar.
my.hideBackHome	Hide the “Back to Homepage” icon on the title bar.
my.hideNavigationBarLoading	Hide loading animation on the navigation bar of the current page.
my.showNavigationBarLoading	Show loading animation on the navigation bar of the current page.
my.setNavigationBar	Set navigation bar style including title, background color, bottom border color, and logo image in the upper-left corner of the navigation bar.

tabBar

Name	Features
my.hideTabBar	Hide tabBar.

Name	Features
my.hideTabBarRedDot	Hide the red dot in the upper-right corner of an item on the tabBar.
my.removeTabBarBadge	Remove text in the upper-right corner of an item on the tabBar.
my.setTabBarBadge	Add text in the upper-right corner of an item on the tabBar. You can use red dot reminder to set the number of messages.
my.setTabBarItem	Set the content of an item on the tabBar dynamically.
my.setTabBarStyle	Set the general style of tabBar dynamically, such as font color, tab background color, and label border color.
my.showTabBar	Show tabBar.
my.showTabBarRedDot	Show the red dot in the upper-right corner of an item on the tabBar.
onTabItemTap	You can click a tab to trigger the function and listen to the events of tabBar click.
FAQ regarding tabBar	Answers to FAQ regarding tab.

Routes

Name	Features
my.switchTab	Switch to the specific tabBar page and close all the irrelevant pages.
my.reLaunch	Close all the current pages and relaunch to the specific page in the application.
my.redirectTo	Close the current page and redirect to the specific page in the application.
my.navigateTo	Navigate to the specific page in the application from the current page.
my.navigateBack	Close the current page and navigate back to the previous level or multilevel page.
FAQ regarding routes	Answers to FAQ regarding routes.

Interaction feedback

Name	Features
my.alert	Alert box (alert).
my.confirm	Confirm box (confirm).
my.prompt	You can enter text in the prompting dialogue box.
my.showToast	Show a toast. You can set the time that the toast lasts.
my.hideToast	Hide toasts.
my.showLoading	Show loading prompts.
my.hideLoading	Hide loading prompts.
my.showActionSheet	Show action sheet.

Pull-down refresh

Name	Features
onPullDownRefresh	Listen to the pull-down refresh events that are performed by users on this page.
my.stopPullDownRefresh	Stop the operations of pull-down refresh on the current page.
my.startPullDownRefresh	Start the operations of pull-down refresh.

Contact

Name	Features
my.choosePhoneContact	Select the phone number of a contact in the directory of the local system.

Choose city

Name	Features
my.chooseCity	You can call this operation to open the city list.
my.onLocatedComplete	You can customize `onLocatedComplete` function and listen to the callback after the page located. This operation is only for the case when the attribute `setLocatedCity` in `my.chooseCity` is true.
my.setLocatedCity	You can call this operation to modify the default name of the located city in `my.chooseCity`.

Choose date

Name	Features
my.datePicker	Open the date picker list.

Animation

Name	Features
my.createAnimation	Create animation instances.

Canvas

Name	Features
my.createCanvasContext	Create canvas drawing context.

Keypad

Name	Features
my.hideKeyboard	Hide keypad.

Scroll

Name	Features
my.pageScrollTo	Scroll to the target location on the page.

Node query

Name	Features
my.createSelectorQuery	Get a node query object `SelectorQuery`.

Option selector

Name	Features
my.optionsSelect	A component similar to the native Safari select but with more powerful features, which you can use to replace select or level 2 data. Note: The component does not support interactions among level 2 data.

Multilevel select

Name	Features
my.multiLevelSelect	Business scenarios for relevant multilevel data selection such as the information selection of province, city, and region.

Set background window

Name	Features
my.setBackgroundColor	Set the background color of the settings window dynamically.
my.setBackgroundTextStyle	Set the font of the pull-down background, and the style of loading graphic dynamically.

Set page pulldown

Name	Features
my.setCanPullDown	Set if the page supports pull down operations. By default, the page in the mini program supports pull down operations.

Set optionMenu

Name	Features
my.setOptionMenu	Configure extra icons on the optionMenu navigation bar, and then click to trigger `onOptionMenuClick`.

Multimedia

Image

Name	Features
my.chooseImage	Take photos or select images from your mobile phone album.
my.previewImage	Preview images.
my.saveImage	Save online images to your mobile phone album.
my.compressImage	Compress images.
my.getImageInfo	Get image information.

Storage

Name	Features
my.setStorage	The operation to save data in the specific key of the local storage will overwrite the corresponding data to this key.
my.setStorageSync	Synchronize data in the specific key of the local storage.
my.getStorage	Get data from the storage asynchronously.
my.getStorageSync	Get data from the storage synchronously.
my.removeStorage	Remove asynchronous operations for data storage.
my.removeStorageSync	Remove synchronous operations for data storage.
my.clearStorage	Clear asynchronous operations for local data storage.
my.clearStorageSync	Clear synchronous operations for local data storage.
my.getStorageInfo	Get relevant information for the current storage asynchronously.
my.getStorageInfoSync	Get relevant information for the current storage synchronously.

File

Name	Features
my.saveFile	Save file to local system. The maximum local file size is 10 MB.
my.getFileInfo	Get file information.
my.getSavedFileInfo	Get the saved file information.
my.getSavedFileList	Get all the save files.
my.removeSavedFile	Remove a saved file.

Network

Name	Features
my.request	Network request of the mini program.
my.uploadFile	Upload local resource to the developer server.
my.downloadFile	Download files to the local system.
my.connectSocket	Create a WebSocket connection.
my.onSocketOpen	Listen to the events of WebSocket connection open.
my.offSocketOpen	Stop listening to the events of WebSocket connection off.
my.onSocketError	Listen to WebSocket errors.
my.offSocketError	Stop listening to WebSocket errors.
my.sendSocketMessage	Send data through WebSocket connection.

Name	Features
my.onSocketMessage	Listen to the message events on the server received by WebSocket.
my.offSocketMessage	Stop listening to message events on the server received by WebSocket.
my.closeSocket	Close WebSocket connection.
my.onSocketClose	Listen to close WebSocket connection.
my.offSocketClose	Stop listening to close WebSocket connection.

Device

canIUse

Name	Features
my.canIUse	Determine if the current version supports API operations, input parameters or output values, components, and attributes of the current mini program.

Get base library version number

Name	Features
my.SDKVersion	Get the version number of the base library. For reference only, do not rely on this value for code logic.

System information

Name	Features
my.getSystemInfo	Get mobile phone system information.
my.getSystemInfoSync	Get the synchronous operation for mobile phone system information.

Network status

Name	Features
my.getNetworkType	Get current network status.
my.onNetworkStatusChange	Start listening to network status changes.
my.offNetworkStatusChange	Stop listening to network status changes.

Clipboard

Name	Features
my.getClipboard	Get clipboard data.
my.setClipboard	Set clipboard data.

Shake

Name	Features
my.watchShake	Use shake feature. Every time you call the API, the callback will be triggered after you shake the mobile phone. If you need to listen again, you have to recall this API.

Vibrate

Name	Features
my.vibrate	Use vibrate feature.
my.vibrateLong	Long vibration lasts for 400 ms.
my.vibrateShort	Short vibration lasts for 40 ms.

Accelerometer

Name	Features
my.onAccelerometerChange	Listen to accelerometer data.
my.offAccelerometerChange	Stop listening to accelerometer data.

Gyroscope

Name	Features
my.onGyroscopeChange	Listen to the events of gyroscope data change.
my.offGyroscopeChange	Stop listening to gyroscope data.

Compass

Name	Features
my.onCompassChange	Listen to compass data.
my.offCompassChange	Stop listening to compass data.

Make phone calls

Name	Features
my.makePhoneCall	Make phone calls.

Events of user capture screen

Name	Features
my.onUserCaptureScreen	Listen to the events of active screen capture initiated by the user.
my.offUserCaptureScreen	Stop listening to the events of screen capture.

Screen brightness

Name	Features
my.setKeepScreenOn	Set whether to keep the screen on.
my.getScreenBrightness	Get screen brightness.
my.setScreenBrightness	Set screen brightness.

Add phone contact

Name	Features
my.addPhoneContact	You can add the list to the directory in the mobile phone system by **Create New Contact** or **Add to Existing Contact**.

Scan

Name	Features
my.scan	Use scan feature.

Bluetooth

Name	Features
my.openBluetoothAdapter	Initialize Bluetooth module of the mini program.
my.closeBluetoothAdapter	Turn off the local Bluetooth module.
my.getBluetoothAdapterState	Get the status of the local Bluetooth module.
my.startBluetoothDevicesDiscovery	Start searching for nearby Bluetooth devices.
my.stopBluetoothDevicesDiscovery	Stop searching for nearby Bluetooth devices.
my.getBluetoothDevices	Get all discovered Bluetooth devices, including the Bluetooth device that is already connected.
my.getConnectedBluetoothDevices	Get connected devices.
my.connectBLEDevice	Connect to the Bluetooth Low Energy (BLE) device.
my.disconnectBLEDevice	Disconnect from the BLE device.
my.writeBLECharacteristicValue	Write data into the characteristic value of the BLE device.
my.readBLECharacteristicValue	Read the data from the characteristic value of the BLE device.
my.notifyBLECharacteristicValueChange	Enable the notify feature for the characteristic value change of the BLE device.
my.getBLEDeviceServices	Get all the services of Bluetooth devices.
my.getBLEDeviceCharacteristics	Get all the characteristic values of Bluetooth devices.
my.onBluetoothDeviceFound	This event is triggered when a new Bluetooth device is found.
my.offBluetoothDeviceFound	Stop listening to the events of searches for new Bluetooth devices.
my.onBLECharacteristicValueChange	Listen to the events of characteristic value changes of the BLE device.
my.offBLECharacteristicValueChange	Stop listening to the events of characteristic value changes of the BLE device.
my.onBLEConnectionStateChanged	Listen to the events of BLE connection errors, such as device loss and abnormal disconnection.
my.offBLEConnectionStateChanged	Stop listening to the events of connection status changes of the BLE device.
my.onBluetoothAdapterStateChange	Listen to the events of local Bluetooth status changes.

Name	Features
my.offBluetoothAdapterStateChange	Stop listening to the events of local Bluetooth status changes.
Error code	List for the error code of Bluetooth API.

Data security

Name	Features
my.rsa	Asymmetric encryption.

Share

Name	Features
onShareAppMessage	Customize `onShareAppMessage` function on the page to set the information for sharing on this page.
my.hideShareMenu	Hide Share button.

Current running version

Name	Features
my.getRunScene	Get the running version of the current mini program.

Custom analysis

Name	Features
my.reportAnalytics	Customize the report operation of data analysis.

Mini program redirection

Name	Features
my.navigateToMiniProgram	Navigate to other mini programs.
my.navigateBackMiniProgram	You can navigate back to the previous mini program only when another mini program is navigated to the active mini program.

webview component control

Name	Features
my.createWebViewContext	Create and return `webViewContext` object.
webViewContext	You can bind `WebViewContext` with a web-view component and implement some features through `webviewId` .

1.10.3. UI

1.10.3.1. Navigation bar

my.getTitleColor

Use this API to obtain the background color of the navigation bar.

Version requirement : This interface is supported since basic library version 1.13.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.

Code sample

```
// API-DEMO page/API/get-title-color/get-title-color.json
{
    "defaultTitle": "Obtain navigation bar background color"
}
```

```
<!-- API-DEMO page/API/get-title-color/get-title-color.axml-->
<view>
  <view class="page-section-demo">
    <text>Current background color of the navigation bar:
    </text>
    <input type="text" disabled="{{true}}" value="{{titleColor.color}}">
    </input>
  </view>
  <view class="page-section-btns">
    <view onTap="getTitleColor">Obtain navigation bar background color
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/get-title-color/get-title-color.js
Page({
  data: {
    titleColor: {},
  },
  getTitleColor() {
    my.getTitleColor({
      success: (res) => {
        this.setData({
          titleColor: res
        })
      }
    })
  }
});
```

Input parameters

The input parameters are of Object type, and the attributes are as follows:

Attribute	Type	Required	Description
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Success callback function

Object type, the attributes are as follows:

Attribute	Type	Description
color	HexColor	Returns the background color of the current navigation bar, it is a hexadecimal color value in ARGB format, such as #323239FF.

FAQ

- **Q**: Can I set the color of **Share and Favorite** in the upper right corner of the mini program?

  A: This color is by default and cannot be set.

my.hideBackHome

Use this API to hide the home button in the top navigation bar, and the return-home option in the tab bar in the upper right corner.

Version requirement: This interface is supported since basic library version 1.16.4. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.

> **Note**
>
> - By default, the home button is displayed if the page where an user enters on starting the Mini Program is not the homepage.
> - If the tab bar is configured to redirect to `pages/index/index` in the `app.json`, the **Return-home** option is not displayed.



Code sample

```
//.js
Page({
  onReady() {
    if (my.canIUse('hideBackHome')) {
      my.hideBackHome();
    }
  },
});
```

```
//.js
onLoad(){
    my.reLaunch({
    url:'../swiper/swiper'// An added page other than the homepage
  })

  setTimeout(() => {
    //Hide the home button after 5 seconds
    my.hideBackHome()
  }, 5000)
}
```

my.hideNavigationBarLoading

Use this API to hide the navigation bar loading.

Code sample

```
// API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.json
{
    "defaultTitle": "Loading animation in title bar"
}
```

```
<!-- API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.axml-->
<view class="page">
  <view class="page-section">
    <button type="primary" onTap="showNavigationBarLoading">Show loading animation</button>
    <button onTap="hideNavigationBarLoading">Hide loading animation</button>
  </view>
</view>
```

```
// API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.js
Page({
  showNavigationBarLoading() {
    my.showNavigationBarLoading()
  },
  hideNavigationBarLoading() {
    my.hideNavigationBarLoading()
  }
})
```

```
/* API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.acss */
button + button {
  margin-top: 20rpx;
}
```

my.showNavigationBarLoading

Use this API to show the navigation bar loading.

Example



Code sample

```
// API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.json
{
    "defaultTitle": "Loading animation in title bar"
}
```

```
<!-- API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.axml-->
<view class="page">
  <view class="page-section">
    <button type="primary" onTap="showNavigationBarLoading">Show loading animation</button>
    <button onTap="hideNavigationBarLoading">Hide loading animation</button>
  </view>
</view>
```

```
// API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.js
Page({
  showNavigationBarLoading() {
    my.showNavigationBarLoading()
  },
  hideNavigationBarLoading() {
    my.hideNavigationBarLoading()
  }
})
```

```
/* API-DEMO page/API/navigation-bar-loading/navigation-bar-loading.acss */
button + button {
  margin-top: 20rpx;
}
```

my.setNavigationBar

Use this API to set the navigation bar style: navigation bar title, navigation bar background color, bottom border color of navigation bar, logo image at the upper left corner of navigation bar.

> **Note**
>
> - The logo image in the upper left corner of the navigation bar supports the `.gif` format and must be an HTTPS link.
> - If the navigation bar background color backgroundColor is set, the borderBottomColor at the bottom of the navigation bar will not take effect, and the color will be the same as backgroundColor by default.
> - The background color of the navigation bar does not support gradient colors.

Example



Code sample

```
// API-DEMO page/API/set-navigation-bar/set-navigation-bar.json
{
    "defaultTitle": "Set page navigation bar"
}
```

```
<!-- API-DEMO page/API/set-navigation-bar/set-navigation-bar.axml-->
<view class="page">
  <view class="page-description">Set navigation bar API</view>
  <form onSubmit="setNavigationBar" style="align-self:stretch">
    <view class="page-section">
      <view class="page-section-demo">
        <input class="page-body-form-value" type="text" placeholder="Title" name="title"></input>
        <input class="page-body-form-value" type="text" placeholder="Navigation bar background color" name="backgroundColor"></input>
        <input class="page-body-form-value" type="text" placeholder="Navigation bar bottom border color" name="borderBottomColor"></input>
        <input class="page-body-form-value" type="text" placeholder="Navigation bar image address" name="image"></input>
      </view>
      <view class="page-section-btns">
        <button type="primary" size="mini" formType="submit">Set</button>
        <button type="primary" size="mini" onTap="resetNavigationBar">Reset</button>
      </view>
    </view>
  </form>
  <view class="tips">
    tips:
   <view class="item">1. image: The image link address, which must be an HTTPS address. Please use a 3x HD image. If an image is set, the title parameter is invalid.</view>
    <view class="item">2. backgroundColor: Navigation bar background color, support hexadecimal color value.</view>
    <view class="item">3. borderBottomColor: The color of the bottom border of the navigation bar, which supports hexadecimal color value. If the backgroundColor is set, borderBottomColor will not take effect, and the color be the same as backgroundColor by default.</view>
  </view>
</view>
```

```
// API-DEMO page/API/set-navigation-bar/set-navigation-bar.js
Page({
  setNavigationBar(e) {
    var title = e.detail.value.title;
    var backgroundColor = e.detail.value.backgroundColor;
    var borderBottomColor = e.detail.value.borderBottomColor;
    var image = e.detail.value.image;
    console.log(title)
    my.setNavigationBar({
      title,
      backgroundColor,
      borderBottomColor,
      image,
    })
  },
  resetNavigationBar() {
    my.setNavigationBar({
      reset: true,
      title: 'Reset navigation bar style',
    });
  }
})
```

```
/* API-DEMO page/API/set-navigation-bar/set-navigation-bar.acss */
.page-section-btns {
  padding: 26rpx;
}
```

Input parameters

The input parameters are of Object type, and the attributes are as follows:

Attribute	Type	Required	Description
title	String	No	Navigation bar title.
image	String	No	The image link address (.gif format supported), which must be an HTTPS address. Please use iOS @3x HD image. If an image is set, the title parameter is invalid.
backgroundColor	String	No	The background color of navigation bar, which supports hexadecimal color value.
borderBottomColor	String	No	The bottom border color of navigation bar, which supports hexadecimal color value. If `backgroundColor` is set, `borderBottomColor` will not take effect, and the color will be the same as the `backgroundColor` color by default.
reset	Boolean	No	Whether to reset the color of navigation bar to the default color, the value is false by default.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

FAQ

- **Q**: Can I set the color of **Share and Favorite** in the upper right corner of the mini program?

  A: This color is by default and cannot be set.

Related Information

For more information about iOS @3x resolution standards, see Image Size and Resolution.

Navigation Bar FAQ

- **Q**: Can I set the color of **Share and Favorite** in the upper right corner of the mini program?

  A: This color is by default and cannot be set.

- **Q**: Can I change the menu page in the mini program capsule button?



  A: Currently, the menu page in the mini program capsule button does not support custom modification.

- **Q**: Can I change the font color of the navigation bar?

  A: The font color of the navigation bar cannot be customized and modified, but you can modify the background color to make it automatically adjust the font color of the navigation bar to black or white

1.10.3.2. tabBar

tabBar usage

For a mini program that has multiple pages, you can configure tabBar and the corresponding pages. tabBar is used to switch among the pages.

Note:

- If a page appears after the `my.navigateTo` or `my.redirectTo` operation is performed, tabBar does not appear at the bottom even if the page is defined in tabBar.
- The default page of `tabBar` must be the homepage of the mini program.

Configurations of tabBar

Attribute	Type	Required	Description
textColor	HexColor	No	The text color of a tab icon when the icon is not selected.
selectedColor	HexColor	No	The text color of a tab icon when the icon is selected.
backgroundColor	HexColor	No	The background color.
items	Array	No	The configurations of each tab in tabBar. The following table describes the detailed configurations of each tab.

Sub parameters of items

Attribute	Type	Required	Description
pagePath	String	Yes	The URL of the page that corresponds to the tab.
name	String	Yes	The name of the tab.
icon	String	No	The URL of the tab icon when the corresponding page is not displayed. We recommend that you choose an image of 60 × 60 pixels. Images of other sizes will automatically be adjusted to this size in a non-proportional manner.
activeIcon	String	No	The URL of the highlighted tab icon when the corresponding page is displayed.

Sample code

The following is the sample code of `tabBar`:

```
{
  "tabBar": {
    "textColor": "#dddddd",
    "selectedColor": "#49a9ee",
    "backgroundColor": "#ffffff",
    "items": [
      {
        "pagePath": "pages/index/index",
        "name": "Homepage"      },
      {
        "pagePath": "pages/logs/logs",
        "name": "Logs"      }
    ]
  }
}
```

my.hideTabBar

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

This interface allows you to hide tabBar. If you have any problems, see FAQ.

Sample code

The following is the sample code of `my.hideTabBar`:

```
my.hideTabBar({
    animation: true
})
```

Parameters

The input parameters are in Object type and have the following attributes:

Parameter	Type	Required	Description
animation	Boolean	No	Specifies whether to enable animation. Default value: No.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.hideTabBarRedDot

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to hide the red dot in the upper-right corner of a tab icon in tabBar. If you have any problems, see FAQ.

Effect



Sample code

The following is the sample code of `my.hideTabBarRedDot`:

```
my.hideTabBarRedDot({
  index: 0
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
index	Number	Required	The serial number of the tab icon, counting from the left.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.

Parameter	Type	Required	Description
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.removeTabBarBadge

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to remove the text from the upper-right corner of a tab icon. If you have any problems, see FAQ.

Effect



Sample code

The following is the sample code of `my.removeTabBarBadge`:

```
my.removeTabBarBadge({
  index: 0
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
index	Number	Required	The serial number of the tab icon, counting from the left.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.setTabBarBadge

Version requirement: Base library V1.11.0 or later versions.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to add a badge to the upper-right corner of a tab icon. You can use red dot reminder to set the number of messages. If you have any problems, see FAQ.

Effect



Sample code

The following is the sample code of `my.setTabBarBadge`:

```
my.setTabBarBadge({
  index: 0,
  text: '42'
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
index	Number	Required	The serial number of the tab icon, counting from the left.
text	String	Required	The text to display. If the text exceeds four characters, the text is displayed as the first four characters + "..." . For example, if the text is "Ant Group", the text is displayed as "Ant ..." .
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.setTabBarItem

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to dynamically set the text of a tab icon in tabBar. If you have any problems, see FAQ.

Sample code

The following is the sample code of `my.setTabBarItem`:

```
my.setTabBarItem({
    index: 0,
    text: 'text',
    iconPath: '/image/iconPath',
    selectedIconPath: '/image/selectedIconPath'
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
index	Number	Required	The serial number of the tab icon, counting from the left.
text	String	Required	The text on the tab icon.
iconPath	String	Required	The URL of the tab icon when the corresponding page is not displayed. We recommend that you choose an image of 81 × 81 pixels in the PNG, JPEG, JPG. or GIF format. Images from the Internet are supported.
selectedIconPath	String	Required	The URL of the tab icon when the corresponding page is displayed. We recommend that you choose an image of 81 × 81 pixels in the PNG, JPEG, JPG. or GIF format. Images from the Internet are supported.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.setTabBarStyle

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to dynamically set the overall style of tabBar, including the font color and the background color and border color of tab icons. If you have any problems, see FAQ.

Sample code

The following is the sample code of `my.setTabBarStyle`:

```
my.setTabBarStyle({
    color: '#FF0000',
    selectedColor: '#00FF00',
    backgroundColor: '#0000FF',
    borderStyle: 'white'
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
color	HexColor	Required	The text color of the tab when the tab is not selected.
selectedColor	HexColor	Required	The text color of the tab when the tab is selected.
backgroundColor	HexColor	Required	The background color of the tab.
borderStyle	String	Required	The color of the top border of tabBar. Valid values: `black` and `white`.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.showTabBar

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

This interface allows you to show tabBar. If you have any problems, see FAQ.

Sample code

The following is the sample code of `my.showTabBar`:

```
my.showTabBar({
    animation: true
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
animation	Boolean	No	Specifies whether to enable animation. Default value: No.
success	Function	No	The callback method that indicates a successful call.
fail	Function	No	The callback method that indicates a failed call.
complete	Function	No	The callback method that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

my.showTabBarRedDot

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

Note: IDEs do not support commissioning. Use a physical device for commissioning.

This interface allows you to show the red dot in the upper-right corner of a tab icon in tabBar. If you have any problems, see FAQ.

Sample code

The following is the sample code of `my.showTabBarRedDot`:

```
my.showTabBarRedDot({
  index: 0
})
```

Parameters

The parameters are in object type and have the following properties:

Parameter	Type	Required	Description
index	Number	Required	The serial number of the tab icon, counting from the left.
success	Function	No	The callback function that indicates a successful operation.
fail	Function	No	The callback function that indicates a failed operation.
complete	Function	No	The callback function that indicates the operation is completed. This function is executed regardless of whether the operation succeeds or fails.

onTabItemTap

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you must first handle the incompatibility. For more information, see Compatibility processing.

This interface allows you to listen to the tapping events of tabBar. If you have any problems, see FAQ.

Sample code

The following is the sample code of `onTabItemTap`:

```
//.js
Page({
  onTabItemTap(item) {
    console.log(item.index)
    console.log(item.pagePath)
    console.log(item.text)
  }
})
```

FAQ

Questions about supported features

- **Q**: Can I use input parameters to redirect pages that are defined in tabBar?

  A: Yes.
- **Q**: How do I listen to the tapping events of tabBar?

  A: You can call the onTabItemTap to listen to the tapping events of tabBar in the mini program.
- **Q**: Can I use an image in the SVG format as a tab icon in tabBar?

  A: No. You cannot use images in the SVG format but use the PNG, JPEG, JPG, or GIF format only.
- **Q**: How do I configure the style for a tab?

  A: You can configure the style in the JSON file, as shown in the following code snippet. Alternatively, perform the my.setTabBarStyle to configure the style.

```
"tabBar": {
    "textColor": "#404040",
    "selectedColor": "#108ee9",
    "backgroundColor": "#F5F5F9"
  }
```

Questions about request exceptions

- **Q**: What do I do if the error message "Cannot read property 'getCurrentPages' of undefined" appears when I try to switch to a page in tabBar?

  A: This error occurs because the specified URL of tabBar is wrong. Please check the URL of tabBar.
- **Q**: Why does tabBar disappear after I redirect to a page?

  A: If a page appears after the my.navigateTo or my.redirectTo is performed, tabBar does not appear at the bottom of the page. In addition, the default page of tabBar must be the homepage of the mini program.
- **Q**: Can I redirect pages in tabBar with input parameters?

  A: tabBar does not support redirecting tabs with input parameters. We recommend that you use cache or global variables to redirect with input parameters.
- **Q**: After I go to a page of tabBar, how do I obtain the URL of the upper-level page?

  A: After you go to a page, the URL of the current page is stored globally. When you switch among the tabs in tabBar, you can use the URL property to obtain the global URL of the upper-level page.
- **Q**: During commissioning in an IDE, why are the life cycle functions onshow and onload not executed when I switch among tabs in tabBar?

  A: You must use a physical device to test page switching in tabBar. Life cycle functions are not triggered if you perform commissioning in an IDE.

1.10.3.3. Route

my.switchTab

Jump to the specified tabBar page and close all other pages that are not tabBar.

Note:

- If the Mini Program is an application of multiple tabs (tabs can be switched at the bottom of the client window), it is possible to use tabBar to configure the behavior of the specified tab, and the page to display during the tab switchover.
- On the page to which it jumps through the page jumping (my.navigateTo) or page redirecting (my.redirectTo), even if the page is defined in the tabBar configuration, the bottom tab bar is not displayed. In addition, the first page of the tabBar must be the home page.
- The first page of the tabBar must be the home page.

See Route related FAQs to learn more.

Effect



Sample code

```
// app.json
{
  "tabBar": {
    "items": [{
      "pagePath": "page/home/index",
      "name": "Home"
    },{
      "pagePath": "page/user/index",
      "name": "User"
    }]
  }
}
```

```
//.js
my.switchTab({
  url: 'page/home/index'
})
```

Parameters

The parameters are of Object type, with attributes as follows:

Parameter	Type	Required	Description
url	String	Yes	Path of the jumping tabBar page (page to be defined in the tabBar field in the `app.json`. **Note** The path cannot be followed by parameters.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

tabBar Configuration

Attribute	Type	Required	Description
textColor	HexColor	No	Text color.
selectedColor	HexColor	No	Color of highlighted text.
backgroundColor	HexColor	No	Background color.
items	Array	Yes	Configured for each tab.

configurations for `items`:

Attribute	Type	Required	Description
pagePath	String	Yes	Set page path.
name	String	Yes	Name.
icon	String	No	Normal icon path. The recommended size is 60 × 60 px, and the system will stretch or scale any incoming images non-proportionally.
activeIcon	String	No	Highlighted icon path.

Sample configuration

```
// tabBar Sample configuration
{
  "tabBar": {
    "textColor": "#dddddd",
    "selectedColor": "#49a9ee",
    "backgroundColor": "#ffffff",
    "items": [
      {
        "pagePath": "pages/index/index",
        "name": "Home"
      },
      {
        "pagePath": "pages/logs/logs",
        "name": "Log"
      }
    ]
  }
}
```

my.reLaunch

Close all current pages and jump to the specified page within the application.

Requirement on version: Base library 1.4.0 or later versions. For earlier versions, it is suggested to perform compatibility processing.

See Route related FAQs to learn more.

Effect



Sample code

```
my.reLaunch({
  url: '/page/index'
})
```

Parameters

The parameters are of Object type, with attributes as follows:

Parameter	Type	Required	Description
url	String	Yes	Page path If the page is not a tabBar page, the path can be followed by parameters. Rules for the parameters: The path and parameter are separated with “?”, the parameter key and the parameter value are connected with “=”, and different parameters must be separated with “&”, such as `path?key1=value1&key2=value2`.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

my.redirectTo

Close the current page and jump to the specified page within the application.

See Route related FAQs to learn more.

Note When you use `my.redirectTo` to navigate to a specific page, you will be redirected to the target page on which no back arrow is available, and the current page will be closed.

Effect



Sample code

```
my.redirectTo({
  url: 'new_page?count=100' //The URL can be an absolute or relative path
})
```

With redirecting to the index page of the home page for example:

- Use absolute path: URL is `/pages/index/index` .
- Use relative path: URL is `../index/index` .

Parameters

The parameters are of Object type, with attributes as follows:

Parameter	Type	Required	Description
url	String	Yes	The application for the jumping does not include the destination page path of the tabBar. The path can be followed by parameters. Rules for the parameters: The path and parameter are separated with “?” , the parameter key and the parameter value are connected with “=” , and different parameters must be separated with “&” , such as `path?key1=value1&key2=value2` .
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

my.navigateTo

Maintain the current page and jump to the specified page within the application.

- You can use `my.navigateBack` to return to the original page.
- The maximum page depth is 10. In other words, the `navigateTo` can be called 10 times at most.

`my.navigateTo` and `my.redirectTo` do not support jumping to tabBar page. If neccessary, you can use my.switchTab to redirect to the tabBar page.

See Route related FAQs to learn more.

Effect



Sample code

```
// API-DEMO page/API/navigator/navigator.json
{
    "defaultTitle": "page redirection"
}
```

```
<!-- API-DEMO page/API/navigator/navigator.axml-->
<view class="page">
  <view class="page-section">
    <button type="primary" onTap="navigateTo">Jump to new page</button>
    <button type="primary" onTap="navigateBack">Return to previous page</button>
    <button type="primary" onTap="redirectTo">Open on the current page - obtain user information</button>
    <button type="primary" onTap="switchTab">Jump Tab - component</button>
    <view class="page-description">This Demo only provides instructions on APIs, not covering mini program redirection function. To learn about the mini program redirection, see the relevant documents.</view>
    <button type="primary" onTap="navigateToMiniProgram">Jump to mini program</button>
    <button type="primary" onTap="navigateBackMiniProgram">Jump back to mini program</button>
  </view>
</view>
```

```
// API-DEMO page/API/navigator/navigator.js
Page({
  navigateTo() {
    my.navigateTo({ url: '../get-user-info/get-user-info' })
  },
  navigateBack() {
    my.navigateBack()
  },
  redirectTo() {
    my.redirectTo({ url: '../get-user-info/get-user-info' })
  },
  navigateToMiniProgram() {
    if (my.canIUse('navigateToMiniProgram')) {
      my.navigateToMiniProgram({
        appId: '2017072607907880',
        extraData: {
          "data1": "test"
        },
        success: (res) => {
          console.log(JSON.stringify(res))
        },
        fail: (res) => {
          console.log(JSON.stringify(res))
        }
      });
    }
  },
  navigateBackMiniProgram() {
    if (my.canIUse('navigateBackMiniProgram')) {
      my.navigateBackMiniProgram({
        extraData: {
          "data1": "test"
        },
        success: (res) => {
          console.log(JSON.stringify(res))
        },
        fail: (res) => {
          console.log(JSON.stringify(res))
        }
      });
    }
  },
  switchTab() {
    my.switchTab({
        url: '/page/tabBar/component/index',
        success: () => {
          my.showToast({
            content: 'Succeeded',
            type: 'success',
            duration: 4000
          });
        }
      }
    );
  },
})
```

```
/* API-DEMO page/API/navigator/navigator.acss */
button + button {
  margin-top: 20rpx;
}
```

Parameters

The parameters are of Object type, with attributes as follows:

Parameter	Type	Required	Description
url	String	Yes	The application for thejumping does not include the destination page path of the tabBar. The path can be followed by parameters. Rules for the parameters: The path and parameter are separated with “?”, the parameter key and the parameter value are connected with “=”, and different parameters must be separated with “&”, such as `path?key1=value1&key2=value2`.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.

Parameter	Type	Required	Description
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

my.navigateBack

Close the current page and return to the previous one or upper-level page. It is possible to use `getCurrentPages` to get the current page stack information and decide how many levels to return.

See Route related FAQs to learn more.

Effect



Sample code

```
// API-DEMO page/API/navigator/navigator.json
{
    "defaultTitle": "page redirection"
}
```

```
<!-- API-DEMO page/API/navigator/navigator.axml-->
<view class="page">
  <view class="page-section">
    <button type="primary" onTap="navigateTo">Navigate to new page</button>
    <button type="primary" onTap="navigateBack">Return to previous page</button>
    <button type="primary" onTap="redirectTo">Open on the current page - obtain user information</button>
    <button type="primary" onTap="switchTab">Navigate Tab - component</button>
    <view class="page-description">This Demo only provides instructions on APIs, not covering mini program redirection function. To learn about the mini program redirection, see the relevant documents.</view>
    <button type="primary" onTap="navigateToMiniProgram">Navigate to mini program</button>
    <button type="primary" onTap="navigateBackMiniProgram">Navigate back to mini program</button>
  </view>
</view>
```

```
// API-DEMO page/API/navigator/navigator.js
Page({
  navigateTo() {
    my.navigateTo({ url: '../get-user-info/get-user-info' })
  },
  navigateBack() {
    my.navigateBack()
  },
  redirectTo() {
    my.redirectTo({ url: '../get-user-info/get-user-info' })
  },
  navigateToMiniProgram() {
    if (my.canIUse('navigateToMiniProgram')) {
      my.navigateToMiniProgram({
        appId: '2017072607907880',
        extraData: {
          "data1": "test"
        },
        success: (res) => {
          console.log(JSON.stringify(res))
        },
        fail: (res) => {
          console.log(JSON.stringify(res))
        }
      });
    }
  },
  navigateBackMiniProgram() {
    if (my.canIUse('navigateBackMiniProgram')) {
      my.navigateBackMiniProgram({
        extraData: {
          "data1": "test"
        },
        success: (res) => {
          console.log(JSON.stringify(res))
        },
        fail: (res) => {
          console.log(JSON.stringify(res))
        }
      });
    }
  },
  switchTab() {
    my.switchTab({
        url: '/page/tabBar/component/index',
        success: () => {
          my.showToast({
            content: 'Succeeded',
            type: 'success',
            duration: 4000
          });
        }
      }
    );
  },
})
```

```
/* API-DEMO page/API/navigator/navigator.acss */
button + button {
  margin-top: 20rpx;
}
```

Parameters

The parameters are of Object type, with attributes as follows:

Parameter	Type	Required	Default	Description
delta	Number	No	1	Number of pages to return. If delta is greater than the number of open pages, it returns to the home page.

Route related FAQs

- **Q**: When I use `my.navigateTo` or `my.redirectTo` for redirecting the page, why the page does not show the tabBar at the bottom?

  A: If you redirect to the page through command (my.navigateTo) or command (my.redirectTo), the page will not show tabBar at the bottom even defined in tabBar configuration. To redirect to the tab, use the my.switchTab method.

- **Q**: Does `my.navigateTo` support parameter input?

  A: Yes.

  Parameter rule: Use between the path and the parameter "?" Concatenate the parameter key and the parameter value with "=", and separate different parameters with "&".

Example: `path? key1=value1&key2=value2`

- **Q**: Can I remove the Back button in the upper-left corner when I use `my.redirectTo` for page redirection?

  A: When the page stack depth is 1, you cannot see the Back button in the upper-left corner of the target page after you use my.redirectTo.

  - We recommend that you determine the peak page stack depth through the getCurrentPages method.
  - Alternatively, use my.reLaunch for redirection. When you use my.reLaunch for redirection, you cannot redirect to the tabBar page.

- **Q**: The mini program has been redirected many times by using `my.navigateTo`. However, the redirection fails after another a few attempts. Why?

  A: The maximum page stack depth for mini programs is 10. It is recommended that you determine the peak page stack depth through getCurrentPages and use redirection for page jumping when the maximum depth is exceeded.

- **Q**: Can I hide the Back button in the navigation bar of the mini program?

  A: No, the Back button always appears due to the hierarchy. You can hide the Back button through the my.reLaunch method to close all current pages and redirect to this page.

1.10.3.4. Feedback

my.alert

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

The alert box. You can set the title, content, and button text of the alert box. Setting the styles such as images is currently not supported.

Parameters

Name	Type	Required	Description
title	String	No	The title of the alert box.
content	String	No	The content of the alert box.
buttonText	String	No	Button text, which defaults to **OK**.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Sample code

```
// API-DEMO page/API/alert/alert.json
{
  "defaultTitle": "Alert"
}
```

```
<!-- API-DEMO page/API/alert/alert.axml-->
<view class="page">
  <view class="page-description">The alert box API</view>
  <view class="page-section">
    <view class="page-section-title">my.alert</view>
    <view class="page-section-demo">
      <button type="primary" onTap="alert">Display the alert box</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/alert/alert.js
Page({
  alert() {
    my.alert({
      title: 'Dear',
      content: 'Your monthly bill has been generated',
      buttonText: 'Got it',
      success: () => {
        my.alert({
          title: 'The user has clicked Got it',
        });
      }
    });
  },
})
```

my.confirm

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

The confirmation box. The confirmation box for prompting the user to confirm the current operation. You can set the title, content, and text of the confirmation or cancel button in the confirmation box.

Parameters

Name	Type	Required	Description
title	String	No	The title of the confirmation box.
content	String	No	The content of the confirmation box.
confirmButtonText	String	No	The text of the confirmation button, which defaults to **OK**.
cancelButtonText	String	No	The text of the cancellation button, which defaults to **Cancel**.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function for ended operations. This callback function will be executed for both successful and failed calls.

Return values of a success callback

Name	Type	Description
confirm	Boolean	Clicking **OK** returns `true` and clicking **Cancel** returns `false`.

Sample code

```
// API-DEMO page/API/confirm/confirm.json
{
    "defaultTitle": "Confirm"
}
```

```
<!-- API-DEMO page/API/confirm/confirm.axml-->
<view class="page">
  <view class="page-description">The confirmation box API</view>
  <view class="page-section">
    <view class="page-section-title">my.confirm</view>
    <view class="page-section-demo">
      <button type="primary" onTap="comfirm">Display the confirmation box</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/confirm/confirm.js
Page({
  comfirm() {
    my.confirm({
      title: 'Reminder',
      content: 'Do you want to query the package with this tracking number: \n1234567890',
      confirmButtonText: 'Query now',
      cancelButtonText: 'Not now',
      success: (result) => {
        my.alert({
          title: `${result.confirm}`,
        });
      },
    });
  },
});
```

my.prompt

> **Note** This interface is only supported in base library V1.7.2 (~~119285~~) and later versions and mPaaS 10.1.32 and later versions.

This interface allows you to display a dialog box for the user to enter text in the dialog box.

Parameters

Name	Type	Required	Description
title	String	No	The title of the prompt box.
message	String	No	The text of the prompt box, which defaults to **Please enter content**.
placeholder	String	No	Prompt text in the entry box.
align	String	No	The message alignment method, which can be the enumerated left, center, and right values. For example, `iOS 'center', android 'left'` indicates center alignment on the iOS client and left alignment on the Android client.
okButtonText	String	No	The text of the confirmation button, which defaults to **OK**.
cancelButtonText	String	No	The text of the confirmation button, which defaults to **Cancel**.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return values of a success callback

Name	Type	Description
ok	Boolean	Clicking **OK** returns true and clicking **Cancel** returns false.
inputValue	String	If `ok` is set to true, the user entry is returned.

Sample code

```
my.prompt({
  title: 'The single-line title',
  message: 'Explain the current situation and prompt the user with the solution, which is preferably no more than two lines in length',
  placeholder: 'Leave a message to friends',
  okButtonText: 'OK',
  cancelButtonText: 'Cancel',
  success: (result) => {
    my.alert({
      title: JSON.stringify(result),
    });
  },
});
```

my.showToast

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

This interface allows you to display a toast. You can set the duration of the toast.

Parameters

Name	Type	Required	Description
content	String	No	Text content.
type	String	No	The toast type reflected by the corresponding icon, which defaults to `none`. The possible values are `success`, `fail`, `exception`, and `none`. Among them, textual information is required for the `exception` type.
duration	Number	No	The display duration in milliseconds, which defaults to 2,000.
success	Function	No	The callback function that indicates a successful call.

Name	Type	Required	Description
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Sample code

```
// API-DEMO page/API/toast/toast.json
{
    "defaultTitle": "Toast"
}
```

```
<!-- API-DEMO page/API/toast/toast.axml-->
<view class="page">
  <view class="page-description">Toast API</view>
  <view class="page-section">
    <view class="page-section-title">my.showToast</view>
    <view class="page-section-btns">
      <view type="primary" onTap="showToastSuccess">Display the success prompt text</view>
      <view type="primary" onTap="showToastFail">Display the fail prompt text</view>
    </view>
    <view class="page-section-btns">
      <view type="primary" onTap="showToastException">Display the exception prompt text</view>
      <view type="primary" onTap="showToastNone">Show the "none" toast.</view>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">my.hideToast</view>
    <view class="page-section-btns">
      <view onTap="hideToast">Hide toasts.</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/toast/toast.js
Page({
  showToastSuccess() {
    my.showToast({
      type: 'success',
      content: 'Operation succeeded',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastFail() {
    my.showToast({
      type: 'fail',
      content: 'Operation failed',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastException() {
    my.showToast({
      type: 'exception',
      content: 'Network exception',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastNone() {
    my.showToast({
      type: 'none',
      content: 'Wait a moment',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  hideToast() {
    my.hideToast()
  },
})
```

my.hideToast

Note This interface is only supported in mPaaS 10.1.32 and later versions.

This interface allows you to hide toasts.

Parameters

Name	Type	Required	Description
success	function	No	The callback function that indicates a successful operation.
fail	function	No	The callback function that indicates a failed operation.
complete	function	No	The callback function that indicates the operation is completed. This function is executed regardless of whether the operation succeeds or fails.

Sample code

```
// API-DEMO page/API/toast/toast.json
{
    "defaultTitle": "Toast"
}
```

```
<!-- API-DEMO page/API/toast/toast.axml-->
<view class="page">
  <view class="page-description">Toast API</view>
  <view class="page-section">
    <view class="page-section-title">my.showToast</view>
    <view class="page-section-btns">
      <view type="primary" onTap="showToastSuccess">Display the success prompt text</view>
      <view type="primary" onTap="showToastFail">Display the fail prompt text</view>
    </view>
    <view class="page-section-btns">
      <view type="primary" onTap="showToastException">Display the exception prompt text</view>
      <view type="primary" onTap="showToastNone">Show the "none" toast.</view>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">my.hideToast</view>
    <view class="page-section-btns">
      <view onTap="hideToast">Hide toasts.</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/toast/toast.js
Page({
  showToastSuccess() {
    my.showToast({
      type: 'success',
      content: 'Operation succeeded',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastFail() {
    my.showToast({
      type: 'fail',
      content: 'Operation failed',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastException() {
    my.showToast({
      type: 'exception',
      content: 'Network exception',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  showToastNone() {
    my.showToast({
      type: 'none',
      content: 'Wait a moment',
      duration: 3000,
      success: () => {
        my.alert({
          title: 'Toast disappeared',
        });
      },
    });
  },
  hideToast() {
    my.hideToast()
  },
})
```

my.showLoading

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

This interface allows you to display a transition effect with a loading prompt. This interface can be used together with the my.hideLoading operation.

Parameters

Name	Type	Required	Description
content	String	No	The loading prompt.
delay	Number	No	The delay to display. Unit: ms. Default value: 0. If the `my.hideLoading` operation is performed before the specified time, the loading process is not shown.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Sample code

```
// API-DEMO page/API/loading/loading.json
{
    "defaultTitle": "Loading prompt"
}
```

```
<!-- API-DEMO page/API/loading/loading.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">
      During the loading process, the whole HTML5 page is covered and becomes not interactive.
    </view>
    <view class="page-section-btns">
      <view onTap="showLoading">Show the loading prompt.</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/loading/loading.js
Page({
  showLoading() {
    my.showLoading({
      content: 'Loading...',
      delay: 1000,
    });
    setTimeout(() => {
      my.hideLoading();
    }, 5000);
  },
});
```

```
/* API-DEMO page/API/loading/loading.acss */
.tips{
  margin-left: 10px;
  margin-top: 20px;
  color: red;
  font-size: 12px;
}
.tips .item {
  margin: 5px 0;
  color: #888888;
  line-height: 14px;
}
```

my.hideLoading

Note This interface is only supported in mPaaS 10.1.32 and later versions.

This interface allows you to hide a transition effect that is specified for showing the loading process. This interface can be used together with the my.showLoading operation.

Parameters

Name	Type	Required	Description

Name	Type	Required	Description
page	Object	No	The page instance on which you want the `my.hideLoading` operation to be performed.

Sample code

```
Page({
  onLoad() {
    my.showLoading();
    const that = this;
    setTimeout(() => {
      my.hideLoading({
        page: that, // Specify the page parameter to ensure that the my.hideLoading operation is performed on a specific page as required.
      });
    }, 4000);
  }
})
```

my.showActionSheet

Note This interface is only supported in mPaaS 10.1.32 and later versions.

This interface allows you to show the action sheet.

Parameters

Name	Type	Required	Description	Minimum base library version
title	String	No	The title of the action sheet.	
items	String Array	Required	The string array of the texts that are displayed on the buttons in the action sheet.	
cancelButtonText	String	No	The text that is displayed on the Cancel button. Default value: “Cancel”. Note that for Android, this parameter is not supported and the Cancel button is not shown.	
destructiveBtnIndex	Number	No	The index number of a specific button. The minimum value is 0. This parameter is supported only for iOS. Applicable scenarios: This operation allows you to delete or clear data. By default, index numbers are displayed in red.	
badges	Object Array	No	The array of objects that need to be highlighted. The following table describes the parameters that you need to specify for each object in the array.	1.9.0
success	Function	No	The callback function that indicates a successful call.	
fail	Function	No	The callback function that indicates a failed call.	
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).	

Parameters to be specified for each object in the badges parameter

Name	Type	Description
index	Number	The index number of the object that needs to be highlighted. The minimum value is 0.

Name	Type	Description
type	String	The highlight style. Valid values: none, point, num, text, and more. • If you set this parameter to none, no red dot is displayed when the object is highlighted. • If you set this parameter to point, a red dot without a text is displayed when the object is highlighted. • If you set this parameter to text, a red circle with a text in it is displayed when the object is highlighted. • If you set this parameter to more, you need to further configure a style as required.
text	String	The custom text that appears when the object is highlighted. • If you set the type parameter to none, point, or more, you do not need to set this parameter. • If you set the type parameter to num, or the text you specify is a decimal or a number less than or equal to 0, the text does not appear when the object is highlighted. If the text you specify is a number greater than 100, the text appears as ....

Sample code

```
// API-DEMO page/API/action-sheet/action-sheet.json
{
    "defaultTitle": "Action Sheet"
}
```

```
<!-- API-DEMO page/API/action-sheet/action-sheet.axml-->
<view class="page">
  <view class="page-description">The API operation for showing the action sheet</view>
  <view class="page-section">
    <view class="page-section-title">my.showActionSheet</view>
    <view class="page-section-demo">
      <button type="primary" onTap="showActionSheet">Show the action sheet.</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/action-sheet/action-sheet.js
Page({
  showActionSheet() {
    my.showActionSheet({
      title: 'Alipay-ActionSheet',
      items: ['Menu 1', 'Menu 2', 'Menu 3'],
      cancelButtonText: 'Cancelled',
      success: (res) => {
        const btn = res.index === -1 ? 'Cancel' : 'No.' + res.index;
        my.alert({
          title: `You selected ${btn}`
        });
      },
    });
  },
});
```

1.10.3.5. Pull down refresh

This API is supported in mPaaS 10.1.32 and later versions.

onPullDownRefresh

This API is supported in mPaaS 10.1.32 and later versions.

On the Page, customize the onPullDownRefresh function to listen to the pull-down refresh event from the user.

- To enable the pull-down refresh event, the `"pullRefresh":true` option needs to be configured in the `.json` configuration file of the related page, and the `"allowsBounceVertical":"YES"` option needs to be configured in `window` of `app.json`.
- After calling my.startPullDownRefresh, the pull-down refresh animation is triggered, and the effect is consistent with the user's manual pull-down refresh (the `onPullDownRefresh` monitoring method will be triggered).
- When the data is refreshed, my.stopPullDownRefresh can stop the pull-down refresh of the current page.

Input parameters

Attribute	Type	Required	Description
pullRefresh	Boolean	No	Whether to enable pull-down refresh. The default value is true. Note: Only when the value of `allowsBounceVertical` is YES, the pull-down refresh will take effect.
allowsBounceVertical	String	No	Whether the page supports vertical dragging beyond the actual content. The default value is YES, it supports YES/NO.

Code sample

`onPullDownRefresh` code sample is as follows:

```
// API-DEMO page/API/pull-down-refresh/pull-down-refresh.json
{
    "defaultTitle": "Pull-down refresh",
    "pullRefresh": true
}
```

```
<!-- API-DEMO page/API/pull-down-refresh/pull-down-refresh.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">Pull down the page to refresh</view>
    <view class="page-section-btns">
      <view type="primary" onTap="stopPullDownRefresh">Stop refresh</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/pull-down-refresh/pull-down-refresh.js
Page({
  onPullDownRefresh() {
    console.log('onPullDownRefresh', new Date());
  },
  stopPullDownRefresh() {
    my.stopPullDownRefresh({
      complete(res) {
        console.log(res, new Date())
      }
    })
  }
});
```

my.stopPullDownRefresh

This API is supported in mPaaS 10.1.32 and later versions.

Stop pull-down refresh on current page.

- After calling my.startPullDownRefresh, the pull-down refresh animation is triggered, and the effect is consistent with the user's manual pull-down refresh (the onPullDownRefresh monitoring method will be triggered).
- When the data is refreshed, my.stopPullDownRefresh can stop the pull-down refresh of the current page.

Input parameters

Object type, the attributes are as follows:

Attribute	Type	Required	Description
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Code sample

`my.stopPullDownRefresh` code sample is as follows:

```
// API-DEMO page/API/pull-down-refresh/pull-down-refresh.json
{
    "defaultTitle": "Pull-down refresh",
    "pullRefresh": true
}
```

```
<!-- API-DEMO page/API/pull-down-refresh/pull-down-refresh.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">Pull down the page to refresh</view>
    <view class="page-section-btns">
      <view type="primary" onTap="stopPullDownRefresh">Stop refresh</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/pull-down-refresh/pull-down-refresh.js
Page({
  onPullDownRefresh() {
    console.log('onPullDownRefresh', new Date());
  },
  stopPullDownRefresh() {
    my.stopPullDownRefresh({
      complete(res) {
        console.log(res, new Date())
      }
    })
  }
});
```

my.startPullDownRefresh

This API is supported in mPaaS 10.1.32 and later versions.

Start pull-down refresh on current page.

- After calling my.startPullDownRefresh, the pull-down refresh animation is triggered, and the effect is consistent with the user's manual pull-down refresh (the onPullDownRefresh monitoring method will be triggered).
- When the data is refreshed, my.stopPullDownRefresh can stop the pull-down refresh of the current page.
- `my.startPullDownRefresh` is not affected by the `allowsBounceVertical` and `pullRefresh` parameters.

Input parameters

Object type, the attributes are as follows:

Attribute	Type	Required	Description
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Code sample

`my.startPullDownRefresh` code sample is as follows:

```
my.startPullDownRefresh()
```

1.10.3.6. Contact

This interface is used to select the phone number of a contact from the local directory.

my.choosePhoneContact

This interface is used to select the phone number of a contact from the local directory.

Parameters

Parameter	Type	Required	Description
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Return value on success

Name	Type	Description
name	String	Selected contact name.
mobile	String	Selected phone number of the contact.

Error code

error	Description	Solution
10	No permission.	Check the permission.
11	The user cancels the operation (or the device hasn't been authorized to use the directory).	It is recommended to authorize the device for using directoy.

Sample code

```
my.choosePhoneContact({
  success: (res) => {
    my.alert({
      content: 'Name:' + res.name + '\nPhone number:' + res.mobile
    });
  },
});
```

1.10.3.7. Choose city

You can call this interface to open the city list.

my.chooseCity

You can call this interface to open the city list.

When you use this interface in iOS client, in order to obtain inverse geographic information, you must set the key of AMAP LBS in the `beforeDidFinishLaunchingWithOptions` method. The relevant code is as follows. To obtain the key of AMAP LBS, see Obtain Key.

```
[LBSmPaaSAdaptor sharedInstance].shouldAMapRegeoWhenLBSFailed = YES;
[AMapServices sharedServices].apiKey = @"The key of AutoNavi positioning"
```

Parameters

The input parameters are of Object type and have the following attributes:

Parameter	Type	Required	Description
showLocatedCity	Boolean	No	Whether to display the currently located city. The default is false.
showHotCities	Boolean	No	Whether to display popular cities. The default is true.
setLocatedCity	Boolean	No	Whether to change the currently located city. The default is false. If `showLocatedCity` is set to false, this setting is invalid.
cities	Object Array	No	The list of custom cities. For the object fields in the list, see the cities table below.
hotCities	Object Array	No	List of custom popular cities. The object fields in the list are same as those in `cities`.
success	Function	No	The callback function for successful call.
fail	Function	No	The callback function for failed call.
complete	Function	No	The callback function for ended call. The callback function will be executed for both successful and failed call.

cities

The object fields in `cities` are described as follows.

Parameter	Type	Required	Description
city	String	Yes	The city name.
adCode	String	Yes	The administrative area code. For the list of administrative area codes, see Administrative Area Codes for the County Level and Above in the People's Republic of China.
spell	String	Yes	The city name in Pinyin, which facilitates user search.

Sample code

```
//.js
my.chooseCity({
  cities: [
    {
      city: 'Chaoyang District',
      adCode: '110105',
      spell: 'chaoyang'
    },
    {
      city: 'Haidian District',
      adCode: '110108',
      spell: 'haidian'
    },
    {
      city: 'Fengtai District',
      adCode: '110106',
      spell: 'fengtai'
    },
    {
      city: 'Dongcheng District',
      adCode: '110101',
      spell: 'dongcheng'
    },
    {
      city: 'Xicheng District',
      adCode: '110102',
      spell: 'xicheng'
    },
    {
      city: 'Fangshan District',
      adCode: '110111',
      spell: 'fangshan'
    }
  ],
  hotCities: [
    {
      city: 'Chaoyang District',
      adCode: '110105'
    },
    {
      city: 'Haidian District',
      adCode: '110108'
    },
    {
      city: 'Fengtai District',
      adCode: '110106'
    }
  ],
  success: (res) => {
    my.alert({
      content: res.city + ':' + res.adCode
    });
  },
});
```

Return values of a success callback

If the user does not select any city and directly clicks the Back button, the callback function will not be triggered.

The input parameters are in Object type and have the following attributes:

Attribute	Type	Description
city	String	The city name.
adCode	String	The administrative area code.
longitude	Number	The longitude, which is returned only for the currently located city
latitude	Number	The latitude, which is returned only for the currently located city

Sample code

```
<!-- API-DEMO page/API/choose-city/choose-city.axml-->
<view class="page">
  <view class="page-description">The API for selecting a city</view>
  <view class="page-section">
    <view class="page-section-title">my.chooseCity</view>
    <view class="page-section-demo">
      <button type="primary" onTap="chooseCity">Choose city</button>
    </view>
  </view>
  <view class="page-description">The API for changing the currently located city</view>
  <view class="page-section">
    <view class="page-section-title">my.setLocatedCity</view>
    <view class="page-section-demo">
      <button type="primary" onTap="setLocatedCity">Change the name of the currently located city</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/choose-city/choose-city.js
Page({
  chooseCity() {
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      success: (res) => {
        my.alert({
          title: 'chooseCity response: ' + JSON.stringify(res),
        });
      },
    });
  },
  setLocatedCity() {
    my.onLocatedComplete({
      success: (res) => {
        my.setLocatedCity({
          locatedCityId:res.locatedCityId,//res.locatedCityId
          locatedCityName:'The changed city name',
          success: (res) => {
            my.alert({ content: 'Changed the currently located city successfully' + JSON.stringify(res), });
          },
          fail: (error) => {
            my.alert({ content: 'Failed to change the currently located city' + JSON.stringify(error), });
          },
        });
      },
      fail: (error) => {
        my.alert({ content: 'onLocatedComplete failed' + JSON.stringify(error), });
      }
    });
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      setLocatedCity: true,
      success: (res) => {
        my.alert({
          title: 'chooseCity response: ' + JSON.stringify(res),
        });
      },
    });
  },
});
```

my.onLocatedComplete

When you customize the `onLocatedComplete` function, you can monitor the callback for the completion of locating this page. This feature is valid only for the case where the `setLocatedCity` attribute in my.chooseCity is set to true.

Parameters

Name	Type	Description
success	Function	The callback function upon successful call.
fail	Function	The callback function for failed call.
complete	Function	The callback function for ended call. This callback function is executed for both successful and failed call.

Return value

Name	Type	Description
longitude	Number	The longitude of the currently located city.

Name	Type	Description
latitude	Number	The latitude of the currently located city.
locatedCityId	String	The ID of the currently located city, which is required for changing the default located city (by using `setLocatedCity`).

The sample return value:

```
{
    longitude:100.3,
    latitude:30.1,
    locatedCityId:""
}
```

Sample code

```
<!-- API-DEMO page/API/choose-city/choose-city.axml-->
<view class="page">
  <view class="page-description">The API for selecting a city</view>
  <view class="page-section">
    <view class="page-section-title">my.chooseCity</view>
    <view class="page-section-demo">
      <button type="primary" onTap="chooseCity">Choose city</button>
    </view>
  </view>
  <view class="page-description">The API for changing the currently located city</view>
  <view class="page-section">
    <view class="page-section-title">my.setLocatedCity</view>
    <view class="page-section-demo">
      <button type="primary" onTap="setLocatedCity">Change the name of the currently located city</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/choose-city/choose-city.js
Page({
  chooseCity() {
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      success: (res) => {
        my.alert({
          title: 'chooseCity response: ' + JSON.stringify(res),
        });
      },
    });
  },
  setLocatedCity() {
    my.onLocatedComplete({
      success: (res) => {
        my.setLocatedCity({
          locatedCityId:res.locatedCityId,//res.locatedCityId
          locatedCityName:'The changed city name',
          success: (res) => {
            my.alert({ content: 'Changed the currently located city successfully' + JSON.stringify(res), });
          },
          fail: (error) => {
            my.alert({ content: 'Failed to change the currently located city' + JSON.stringify(error), });
          },
        });
      },
      fail: (error) => {
        my.alert({ content: 'onLocatedComplete failed' + JSON.stringify(error), });
      }
    });
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      setLocatedCity: true,
      success: (res) => {
        my.alert({
          title: 'chooseCity response: ' + JSON.stringify(res),
        });
      },
    });
  },
});undefined
```

my.setLocatedCity

This interface allows you to change the name of the default located city in `my.chooseCity`.

Parameters

Name	Type	Required	Description
locatedCityId	String	Yes	The ID of the currently located city, which is returned by the `onLocatedComplete` of `my.chooseCity`.
locatedCityName	String	Yes	The name of the currently located city.
locatedCityAdCode	String	No	The administrative area code of the currently located city. If no value is specified for this parameter, the value obtained by the control prevails.
locatedCityPinyin	String	No	The name of the currently located city in Pinyin. If no value is specified for this parameter, the value obtained by the control prevails.
success	Function	No	The callback function for successful call.
fail	Function	No	The callback function for failed call.
complete	Function	No	The callback function for ended call. The callback function will be executed for both successful and failed call.

Return values upon callback failure

Name	Type	Description
error	String	The error code.
errorMessage	String	The error description.

Return values upon callback success

Name	Type	Description
locatedCityName	String	The name of the currently located city.

Error code

Error code	Remarks and solution	
11	The parameter type is incorrect.	Check whether the parameter type is correct.
12	A required parameter is empty.	Check whether the `locatedCityId` and `locatedCityName` parameters are specified.
13	The locatedCityId value does not match.	Ensure that the value is consistent with the `locatedCityId` value of `onLocatedComplete` in `my.chooseCity`.

Sample code

```
<!-- .axml -->
<view class="page">
  <view class="page-description">Choose city</view>
  <view class="page-section">
    <view class="page-section-title">chooseCity</view>
    <view class="page-section-demo">
      <button type="primary" onTap="chooseCity">Choose city</button>
      <button type="primary" onTap="noChooseCity">No popular or current city</button>
      <button type="primary" onTap="selfChooseCity">Customize the city selection</button>
      <button type="primary" onTap="self_chooseCity">Customize the city selection</button>
      <button type="primary" onTap="setLocatedCity">setLocatedCity</button>
    </view>
  </view>
</view>
```

```
// .js
Page({
  data: {
    localcity: 'Tianjin',
  },
  chooseCity() {
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      success: (res) => {
```

```
      my.alert({ title: `chooseAlipayContact response: ${JSON.stringify(res)}` })
    },
    fail: (error) => {
      my.alert({ content: `Failed to select the city${JSON.stringify(error)}` })
    },
    complete: () => {
      my.showToast({ content: 'Completion callback' })
    },
  })
},
noChooseCity() {
  my.chooseCity({
    showLocatedCity: false,
    showHotCities: false,
    success: (res) => {
      my.alert({ title: `The operation has succeeded: ${JSON.stringify(res)}` })
    },
    fail: (error) => {
      my.alert({ content: `Failed to select the city${JSON.stringify(error)}` })
    },
  })
},
selfChooseCity() {
  my.chooseCity({
    cities: [
      {
        city: 'Chaoyang District',
        adCode: '110105',
        spell: 'chaoyang',
      },
      {
        city: 'Haidian District',
        adCode: '110108',
        spell: 'haidian',
      },
      {
        city: 'Fengtai District',
        adCode: '110106',
        spell: 'fengtai',
      },
      {
        city: 'Dongcheng District',
        adCode: '110101',
        spell: 'dongcheng',
      },
      {
        city: 'Xicheng District',
        adCode: '110102',
        spell: 'xicheng',
      },
      {
        city: 'Fangshan District',
        adCode: '110111',
        spell: 'fangshan',
      },
    ],
    hotCities: [
      {
        city: 'Chaoyang District',
        adCode: '110105',
      },
      {
        city: 'Haidian District',
        adCode: '110108',
      },
      {
        city: 'Fengtai District',
        adCode: '110106',
      },
    ],
    success: (res) => {
      my.alert({ title: `The operation succeeded: ${JSON.stringify(res)}` })
    },
    fail: (error) => {
      my.alert({ content: `Failed to select the city${JSON.stringify(error)}` })
    },
  })
},

self_chooseCity() {
  my.chooseCity({
    showLocatedCity: true,
    showHotCities: true,
    cities: [
      {
        city: 'Chaoyang District',
        adCode: '110105',
```

```
        spell: 'chaoyang',
      },
      {
        city: 'Haidian District',
        adCode: '110108',
        spell: 'haidian',
      },
      {
        city: 'Fengtai District',
        adCode: '110106',
        spell: 'fengtai',
      },
      {
        city: 'Dongcheng District',
        adCode: '110101',
        spell: 'dongcheng',
      },
      {
        city: 'Xicheng District',
        adCode: '110102',
        spell: 'xicheng',
      },
    ],
    hotCities: [
      {
        city: 'Chaoyang District',
        adCode: '110105',
      },
      {
        city: 'Haidian District',
        adCode: '110108',
      },
      {
        city: 'Fengtai District',
        adCode: '110106',
      },
    ],
    success: (res) => {
      my.alert({ title: `The operation succeeded: ${JSON.stringify(res)}` })
    },
    fail: (error) => {
      my.alert({ content: `Failed to select the city${JSON.stringify(error)}` })
    },
  })
},

multiLevelSelect() {
  my.multiLevelSelect({
    title: 'Please select a city', // The cascaded selection title.
    list: [
      {
        name: 'Hangzhou City', // The item name.
        subList: [
          {
            name: 'Xihu District',
            subList: [
              {
                name: 'Wenyi Road',
              },
              {
                name: 'Wen'er Road'',
              },
              {
                name: 'Wensan Road',
              },
            ],
          },
          {
            name: 'Binjiang District',
            subList: [
              {
                name: 'Binhe Road',
              },
              {
                name: 'Binxing Road',
              },
              {
                name: 'Baima Lake Anime Square',
              },
            ],
          },
        ], // The cascaded sub-data list.
      },
    ],
    success: (result) => {
      console.log(result)
      my.alert({ content: `Cascaded${JSON.stringify(result)}` })
```

```
      },
      fail: (error) => {
        my.alert({ content: `Failed to call${JSON.stringify(error)}` })
      },
    })
  },

  setLocatedCity() {
    my.chooseCity({
      showLocatedCity: true,
      showHotCities: true,
      setLocatedCity: true,
      success: (res) => {
        this.setData({
          localcity: res.city,
        })
        my.alert({ title: `chooseAlipayContact response: ${JSON.stringify(res)}` })
      },
      fail: (error) => {
        my.alert({ content: `Failed to select the city${JSON.stringify(error)}` })
      },
      complete: () => {
        my.showToast({ content: 'Completion callback' })
      },
    })
    my.onLocatedComplete({
      success: (res) => {
        my.setLocatedCity({
          locatedCityId: res.locatedCityId,
          locatedCityName: this.data.localcity,
          success: (result) => {
            console.log(result)
          },
          fail: (error) => {
            my.alert({
              content: `Failed to change the currently located city${JSON.stringify(error)}`,
            })
          },
        })
      },
      fail: (error) => {
        my.alert({
          content: `onLocatedComplete failed${JSON.stringify(error)}`,
        })
      },
    })
  },
})
```

1.10.3.8. Choose date

my.datePicker

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Use this API to open the date selection list.

Parameters

Parameter	Type	Required	Description
format	String	No	Format of the returned date. 1. yyyy-MM-dd (default) 2. HH:mm 3. yyyy-MM-dd HH:mm 4. yyyy-MM (Base library with version lower than 1.1.1 is not supported. Use `canIUse('datePicker.object.format.yyyy')` to query if the current version can be used.) 5. yyyy (Base library with version lower than 1.1.1 is not supported. Use `canIUse('datePicker.object.format.yyyy')` to query if the current version can be used.)
currentDate	String	No	The date and time initially selected. By default, the current time date and time are used.
startDate	String	No	Minimum date and time.

Parameter	Type	Required	Description
endDate	String	No	Maximum date and time.
success	Function	No	The callback function for a successful API call.
fail	Function	No	The callback function for a failed API call.
complete	Function	No	The callback function used when the API call is completed. This function is always executed no matter the call succeeds or fails.

Return value on success

Name	Type	Description
date	String	Selected date.

Error code

Error	Description	Solution
11	The user cancelled the operation.	This is normal user interaction, and you don't have to take any action.

Sample code

```
// API-DEMO page/API/date-picker/date-picker.json
{
    "defaultTitle": "Date Picker"
}
```

```
<!-- API-DEMO page/API/date-picker/date-picker.axml -->
<view class="page">
  <view class="page-description">API for selecting date</view>
  <view class="page-section">
    <view class="page-section-title">my.datePicker</view>
    <view class="page-section-demo">
      <button class="page-body-button" type="primary" onTap="datePicker">Select date-1</button>
      <button class="page-body-button" type="primary" onTap="datePickerHMS">Select date-2</button>
      <button class="page-body-button" type="primary" onTap="datePickerYMDHMS">Select date-3</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/date-picker/date-picker.js
Page({
  datePicker() {
    my.datePicker({
      currentDate: '2016-10-10',
      startDate: '2016-10-9',
      endDate: '2017-10-9',
      success: (res) => {
        my.alert({
          title: 'datePicker response: ' + JSON.stringify(res)
        });
      },
    });
  },
  datePickerHMS() {
    my.datePicker({
      format: 'HH:mm',
      currentDate: '12:12',
      startDate: '11:11',
      endDate: '13:13',
      success: (res) => {
        my.alert({
          title: 'datePicker response: ' + JSON.stringify(res)
        });
      },
    });
  },
  datePickerYMDHMS() {
    my.datePicker({
      format: 'yyyy-MM-dd HH:mm',
      currentDate: '2012-01-09 11:11',
      startDate: '2012-01-01 11:11',
      endDate: '2012-01-10 11:11',
      success: (res) => {
        my.alert({
          title: 'datePicker response: ' + JSON.stringify(res)
        });
      },
    });
  },
});
```

```
/* API-DEMO page/API/date-picker/date-picker.acss */
button + button {
  margin-top: 20rpx;
}
```

Note For iOS users who use the baseline version V10.1.68.35 and later, the latest style of time picker can be set by creating the `AUImplDatePicker` class and overriding the `userNewYearDateAndTime` method to return the `YES` method.

```
@implementation AUImplDatePicker (NewDatePicker)
// External override can use the new UI of year, month, day, hour and minute.
- (BOOL)userNewYearDateAndTime
{
    return YES;
}

@end
```

The latest style of time picker is as follows:



1.10.3.9. Animation

my.createAnimation

> Note
> This interface is only supported in mPaaS 10.1.32 and later versions.

Create an animation instance. Call the instance's methods to describe the animation, and finally export and pass the animation data to the component's `animation` attribute via the animation instance's `export` method.

> Note
> The `export` method will clear the previous animation operation after it is called.

Input parameter

Parameter	Type	Required	Description
duration	Integer	No	The duration of the animation, in ms, 400 by default.
timingFunction	String	No	Define the effect of the animation, the default value is "linear", and the valid values are "linear", "ease", "ease-in", "ease-in-out", "ease-out", "step-start", and "step-end"
delay	Integer	No	Animation delay time, in ms, 0 by default
transformOrigin	String	No	Set transform-origin, "50% 50% 0" by default

```
const animation = my.createAnimation({
  transformOrigin: "top right",
  duration: 3000,
  timeFunction: "ease-in-out",
  delay: 100,
})
```

animation

The animation instance can call the following methods to describe the animation. After the call ends, it will return the instance itself, supporting the chain call.

Style:

Method	Parameter	Description
opacity	value	Transparency, parameter ranging from 0~1
backgroundColor	color	Color value
width	length	Set the width: length values, in px, such as 300 px.
height	length	Set the height: length values, in px, such as 300 px.
top	length	Set the top: length values, in px, such as 300 px.
left	length	Set the left: length values, in px, such as 300 px.
bottom	length	Set the bottom: length values, in px, such as 300 px.
right	length	Set the right: length values, in px, such as 300 px.

Rotate:

Method	Parameter	Description
rotate	deg	deg range -180 ~ 180, rotating deg degree clockwise from the origin
rotateX	deg	deg range -180 ~ 180, rotating deg degree on the X axis
rotateY	deg	deg range -180 ~ 180, rotating deg degree on the Y axis
rotateZ	deg	deg range -180 ~ 180, rotating deg degree on the Z axis
rotate3d	(x, y , z, deg)	Same as transform-function rotate3d.

Scale:

Method	Parameter	Description

Method	Parameter	Description
scale	sx,[sy]	When there is only one parameter, it means scaling by sx times on X and Y axes simultaneously; when there are two parameters, it means scaling by sx times on the X axis and sy times on the Y axis.
scaleX	sx	Scale by sx times on the X axis
scaleY	sy	Scale by sy times on the Y axis
scaleZ	sz	Scale by sy times on the Z axis
scale3d	(sx,sy,sz)	Scale by sx times on the X axis, sy times on the Y axis, and sz times on the Z axis.

Translate:

Method	Parameter	Description
translate	tx,[ty]	When there is only one parameter, it indicates translating by tx on X axis. When there are two parameters, it indicates translating by tx on X axis and ty on Y axis.
translateX	tx	Translate by tx on the X axis, in px
translateY	ty	Translate by ty on the Y axis, in px
translateZ	tz	Translate by tz on the Z axis, in px
translate3d	(tx,ty,tz)	Translate by tx on the X axis, ty on the Y axis, and tz on the Z axis, in px.

Skew:

Method	Parameter	Description
skew	ax,[ay]	The parameter range is -180 ~ 180. When there is only one parameter, the Y-axis coordinate is unchanged, and the X-axis coordinate is skewed ax degrees clockwise; when there are two parameters, it means skewing ax degrees on the X-axis and ay degrees on the Y-axis.
skewX	ax	The parameter range is -180 ~ 180. The Y-axis coordinate is unchanged, and the X-axis coordinate is skewed ax degrees clockwise.
skewY	ay	The parameter range is -180~180. The X-axis coordinates is unchanged, and the Y-axis coordinate is skewed ay degrees clockwise.

Matrix transformation:

Method	Parameter	Description
matrix	(a,b,c,d,tx,ty)	Same as transform-function.
matrix3d		Same as transform-function matrix3d.

Animation queue

When the animation operation method is called, it is required to call `step()` to indicates the completion of a group of animations. Within a group of animation, it is possible to call any number of animation methods. All animations in the group start at the same time. It does not enter into the next group until the current animation group ends. You can pass the same configuration parameter as `my.createAnimation()` to specify the configuration of the current set of animations by using `step()`.

Code example

```
<view animation="{{animationInfo}}" style="background:yellow;height:100rpx;width:100rpx"></view>
```

```
Page({
  data: {
    animationInfo: {}
  },
  onShow(){
    var animation = my.createAnimation({
      duration: 1000,
        timingFunction: 'ease-in-out',
    });

    this.animation = animation;

    animation.scale(3,3).rotate(60).step();

    this.setData({
      animationInfo:animation.export()
    });

    setTimeout(function() {
      animation.translate(35).step();
      this.setData({
        animationInfo:animation.export(),
      });
    }.bind(this), 1500);
  },
  rotateAndScale () {
    // Rotate while zooming in
    this.animation.rotate(60).scale(3, 3).step();
    this.setData({
      animationInfo: this.animation.export(),
    });
  },
  rotateThenScale () {
    // Rotate and then zoom in
    this.animation.rotate(60).step();
    this.animation.scale(3, 3).step();
    this.setData({
      animationInfo: this.animation.export(),
    });
  },
  rotateAndScaleThenTranslate () {
    // Rotate while zooming in, then translate
    this.animation.rotate(60).scale(3, 3).step();
    this.animation.translate(100, 100).step({ duration: 2000 });
    this.setData({
      animationInfo: this.animation.export()
    });
  }
})
```

1.10.3.10. Canvas

my.createCanvasContext(canvasId)

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Create a drawing context for the canvas. This drawing context only works on the `<canvas/>` of the corresponding `canvasId`.

Parameters

Parameter	Type	Description
canvasId	String	ID defined on `<canvas/>`

toTempFilePath

Export the contents of the current canvas to generate an image and return the file path.

Parameters

Parameter	Type	Required	Description	Lowest base library version supported
x	Number	No	X axis starting point of the canvas, the default value is 0.	-
y	Number	No	Y axis starting point of the canvas, the default is 0.	-
width	Number	No	Canvas width, the default value is canvas width - X.	-

Parameter	Type	Required	Description	Lowest base library version supported
height	Number	No	Canvas height, the default value is canvas height - Y.	-
destWidth	Number	No	Output image width, the default value is canvas width.	-
destHeight	Number	No	Output image height, the default value is canvas height.	-
fileType	String	No	Type of the target file. The legal values are jpg and png, and the default value is 1. The parameter is available in mPaaS 10.1.60 and later versions.	1.10
quality	Number	No	Image quality. Currently, the parameter only works on jpg, and the value range is (0, 1]. The parameter is available in mPaaS 10.1.60 and later versions.	1.10
success	Function	No	Interface callback succeeded.	-
fail	Function	No	Interface callback failed.	-
complete	Function	No	Interface callback completed.	-

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.toTempFilePath({
  success() {},
});
```

setTextAlign

`textAlign` is a property of Canvas 2D API that describes the alignment of text when it is drawn. Note that this alignment is based on the x value of `CanvasRenderingContext2D.fillText` method. So, if `textAlign`="center", then the text will be drawn at `x-50%*width`.

Parameters

Parameter	Type	Definition
textAlign	String	Enumeration values: "left", "right", "center", "start" and "end"

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setTextAlign("left");
ctx.fillText("Hello world", 0, 100);
```

setTextBaseline

`textBaseline` is a property of Canvas 2D API that describes the baseline of text when it is drawn.

Parameters

Parameter	Type	Definition
textBaseline	String	Enumeration values: "top", "hanging", "middle", "alphabetic", "ideographic" and "bottom"

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setTextBaseline("top");
ctx.fillText("Hello world", 0, 100);
```

setFillStyle

Set the fill color.

If `fillStyle` is not set, the default color is `black`.

Parameters

Parameter	Type	Definition
color	Color	The color.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setFillStyle('blue');
ctx.fillRect(50, 50, 100, 175);
ctx.draw();
```

setStrokeStyle

Set the border color.

If `strokeStyle` is not set, the default color is `black`.

Parameters

Parameter	Type	Definition
color	Color	The color.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setStrokeStyle('blue');
ctx.strokeRect(50, 50, 100, 175);
ctx.draw();
```

setShadow

Set the shadow style.

If not set, the default values of `offsetX`, `offsetY` and `blur` are all 0, and the default value of `color` is `black`.

Parameters

Parameter	Type	Value range	Definition
offsetX	Number		The offset of the shadow relative to the horizontal direction of the shape
offsetY	Number		The offset of the shadow relative to the vertical direction of the shape
blur	Number	0~100	The blur level of the shadow, the larger the value, the more blurred
color	Color		Shadow color

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setFillStyle('red');
ctx.setShadow(15, 45, 45, 'yellow');
ctx.fillRect(20, 20, 100, 175);
ctx.draw();
```

createLinearGradient

Create a linear gradient.

You need to use `addColorStop()` to specify at least two gradientstops.

Parameters

Parameter	Type	Definition
x0	Number	X coordinate of the starting point
y0	Number	Y coordinate of the starting point
x1	Number	X coordinate of the end point
y1	Number	Y coordinate of the end point

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

const grd = ctx.createLinearGradient(10, 10, 150, 10);
grd.addColorStop(0, 'yellow');
grd.addColorStop(1, 'blue');

ctx.setFillStyle(grd);
ctx.fillRect(20, 20, 250, 180);
ctx.draw();
```

createCircularGradient

Create a circular gradient. The starting point is at the center of the circle and the end point is at the ring.

You need to use `addColorStop()` to specify at least two gradientstops.

Parameters

Parameter	Type	Definition
x	Number	X coordinate of the center of circle
y	Number	Y coordinate of the center of circle
r	Number	Radius of circle

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

const grd = ctx.createCircularGradient(90, 60, 60);
grd.addColorStop(0, 'blue');
grd.addColorStop(1, 'red');

ctx.setFillStyle(grd);
ctx.fillRect(20, 20, 250, 180);
ctx.draw();
```

addColorStop

Create a color gradientstop.

Portions smaller than the minimum `stop` are rendered in the color of the minimum `stop`, and portions larger than the maximum `stop` are rendered in the color of the maximum `stop`.

You need to use `addColorStop()` to specify at least two gradientstops.

Parameters

Parameter	Type	Definition
stop	Number	Indicates that the color gradientstop is positioned between the starting and end points, ranging from 0 to 1
color	Color	Color of gradientstop

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

const grd = ctx.createLinearGradient(40, 20, 230, 40);
grd.addColorStop(0.36, 'orange');
grd.addColorStop(0.56, 'cyan');
grd.addColorStop(0.63, 'yellow');
grd.addColorStop(0.76, 'blue');
grd.addColorStop(0.54, 'green');
grd.addColorStop(1, 'purple');
grd.addColorStop(0.4, 'red');

ctx.setFillStyle(grd);
ctx.fillRect(20, 20, 250, 180);
ctx.draw();
```

setLineWidth

Set the width of line.

Parameters

Parameter	Type	Description
lineWidth	Number	Line width, in px

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.beginPath();
ctx.moveTo(20, 20);
ctx.lineTo(250, 10);
ctx.stroke();

ctx.beginPath();
ctx.setLineWidth(10);
ctx.moveTo(20, 35);
ctx.lineTo(250, 30);
ctx.stroke();

ctx.beginPath();
ctx.setLineWidth(20);
ctx.moveTo(20, 50);
ctx.lineTo(250, 55);
ctx.stroke();

ctx.beginPath();
ctx.setLineWidth(25);
ctx.moveTo(20, 80);
ctx.lineTo(250, 85);
ctx.stroke();

ctx.draw();
```

setLineCap

Set the end point style of line.

Parameters

Parameter	Type	Range	Description
lineCap	String	'round', 'butt' and 'square'	End point style of line

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.beginPath();
ctx.moveTo(10, 10);
ctx.lineTo(150, 10);
ctx.stroke();

ctx.beginPath();
ctx.setLineCap('round');
ctx.setLineWidth(20);
ctx.moveTo(20, 70);
ctx.lineTo(250, 80);
ctx.stroke();

ctx.beginPath();
ctx.setLineCap('butt');
ctx.setLineWidth(10);
ctx.moveTo(25, 80);
ctx.lineTo(250, 30);
ctx.stroke();

ctx.beginPath();
ctx.setLineCap('square');
ctx.setLineWidth(10);
ctx.moveTo(35, 47);
ctx.lineTo(230, 120);
ctx.stroke();

ctx.draw();
```

setLineJoin

Set the intersection style of line.

Parameters

Parameter	Type	Range	Description
lineJoin	String	'round', 'bevel' and 'miter'	End intersection style of line

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.beginPath();
ctx.moveTo(20, 30);
ctx.lineTo(150, 70);
ctx.lineTo(20, 100);
ctx.stroke();

ctx.beginPath();
ctx.setLineJoin('round');
ctx.setLineWidth(20);
ctx.moveTo(100, 20);
ctx.lineTo(280, 80);
ctx.lineTo(100, 100);
ctx.stroke();

ctx.beginPath();
ctx.setLineJoin('bevel');
ctx.setLineWidth(20);
ctx.moveTo(60, 25);
ctx.lineTo(180, 80);
ctx.lineTo(90, 100);
ctx.stroke();

ctx.beginPath();
ctx.setLineJoin('miter');
ctx.setLineWidth(15);
ctx.moveTo(130, 70);
ctx.lineTo(250, 50);
ctx.lineTo(230, 100);
ctx.stroke();

ctx.draw();
```

setMiterLimit

Set the maximum miter length which refers to the distance between the inner and outer corners at the intersection of two lines. Only valid when `setLineJoin()` is miter. When the maximum miter length is exceeded, the intersection will be displayed by using bevel for `lineJoin`.

Parameters

Parameter	Type	Description
miterLimit	Number	Maximum miter length

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.beginPath();
ctx.setLineWidth(15);
ctx.setLineJoin('miter');
ctx.setMiterLimit(1);
ctx.moveTo(10, 10);
ctx.lineTo(100, 50);
ctx.lineTo(10, 90);
ctx.stroke();

ctx.beginPath();
ctx.setLineWidth(15);
ctx.setLineJoin('miter');
ctx.setMiterLimit(2);
ctx.moveTo(50, 10);
ctx.lineTo(140, 50);
ctx.lineTo(50, 90);
ctx.stroke();

ctx.beginPath();
ctx.setLineWidth(15);
ctx.setLineJoin('miter');
ctx.setMiterLimit(3);
ctx.moveTo(90, 10);
ctx.lineTo(180, 50);
ctx.lineTo(90, 90);
ctx.stroke();

ctx.draw();
```

rect

Create a rectangle.

Use `fill()` or `stroke()` method to draw a rectangle on the canvas.

Parameters

Parameter	Type	Description
x	Number	X coordinate of the top left corner of rectangle
y	Number	Y coordinate of the top left corner of rectangle
width	Number	Rectangular path width
height	Number	Rectangular path height

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.rect(20, 20, 250, 80);
ctx.setFillStyle('blue');
ctx.fill();
ctx.draw();
```

fillRect

Fill the rectangle.

Use `setFillStyle()` to set the fill color of rectangle. If not set, default to black.

Parameters

Parameter	Type	Description
x	Number	X coordinate of the top left corner of rectangle
y	Number	Y coordinate of the top left corner of rectangle
width	Number	Rectangular path width
height	Number	Rectangular path height

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.fillRect(20, 20, 250, 80);
ctx.setFillStyle('blue');
ctx.draw();
```

strokeRect

Draw a rectangle (not filled).

Use `setFillStroke()` to set the color of rectangle line. If not set, the default is black.

Parameters

Parameter	Type	Range	Description
x	Number		X coordinate of the top left corner of rectangle
y	Number		Y coordinate of the top left corner of rectangle
width	Number		Rectangular path width
height	Number		Rectangular path height

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setStrokeStyle('blue');
ctx.strokeRect(20, 20, 250, 80);
ctx.draw();
```

clearRect

Clear the content within this rectangular area on the canvas.

`clearRect` It will clear the rectangular instead of drawing a white rectangle. In order to intuitively understand this operation, you can add a layer of background color to the canvas.

```
<canvas id="awesomeCanvas" style="border: 1px solid; background: red;"/>
```

Parameters

Parameter	Type	Description
x	Number	X coordinate of the top left corner of rectangle
y	Number	Y coordinate of the top left corner of rectangle
width	Number	Rectangular width
height	Number	Rectangular height

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setFillStyle('blue');
ctx.fillRect(250, 10, 250, 200);
ctx.setFillStyle('yellow');
ctx.fillRect(0, 0, 150, 200);
ctx.clearRect(10, 10, 150, 75);
ctx.draw();
```

fill

Fill the contents of the current path. The default fill color is black.

- If the current path is not closed, the `fill()` method will connect the starting and end points and then fill them. See sample 1 for details.
- For `fill()`, the fill path is calculated from `beginPath()`, but `fillRect()` is not included. See sample 2 for details.

Code sample

- Sample 1

```
const ctx = my.createCanvasContext('awesomeCanvas')
ctx.moveTo(20, 20)
ctx.lineTo(200, 20)
ctx.lineTo(200, 200)
ctx.fill()
ctx.draw()
```

- Sample 2

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.rect(20, 20, 110, 40);
ctx.setFillStyle('blue');
ctx.fill();

ctx.beginPath();
ctx.rect(20, 30, 150, 40);

ctx.setFillStyle('yellow');
ctx.fillRect(20, 80, 150, 40);

ctx.rect(20, 150, 150, 40);

ctx.setFillStyle('red');
ctx.fill();
ctx.draw();
```

stroke

Draw the border of the current path, the default is black.

`stroke()` The drawn path is calculated from `beginPath()`, but `strokeRect()` is not included. See sample 2 for details.

Code example

- Sample 1

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.moveTo(20, 20);
ctx.lineTo(150, 10);
ctx.lineTo(150, 150);
ctx.stroke();
ctx.draw();
```

- Sample 2

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.rect(10, 10, 100, 30);
ctx.setStrokeStyle('blue');
ctx.stroke();

ctx.beginPath();
ctx.rect(20, 50, 150, 50);

ctx.setStrokeStyle('yellow');
ctx.strokeRect(15, 75, 200, 35);

ctx.rect(20, 200, 150, 30);

ctx.setStrokeStyle('red');
ctx.stroke();
ctx.draw();
```

beginPath

To start creating a path, you need to call `fill` or `stroke` to use the path to fill or draw.

- At the very beginning, it is equivalent to calling `beginPath()` once.
- If you set `setFillStyle()` , `setStrokeStyle()` or `setLineWidth()` multiple times within the same path, the last setting will prevail.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.rect(20, 20, 150, 50);
ctx.setFillStyle('blue');
ctx.fill();

ctx.beginPath();
ctx.rect(20, 50, 150, 40);

ctx.setFillStyle('yellow');
ctx.fillRect(20, 170, 150, 40);

ctx.rect(10, 100, 100, 30);

ctx.setFillStyle('red');
ctx.fill();
ctx.draw();
```

closePath

Close a path.

- Closing a path will connect the starting and end points.
- If `fill()` or `stroke()` is not called after the path is closed and a new path is created, the previous path will not be rendered.

Code example

- Sample 1

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.moveTo(20, 20);
ctx.lineTo(150, 20);
ctx.lineTo(150, 150);
ctx.closePath();
ctx.stroke();
ctx.draw();
```

- Sample 2

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.rect(20, 20, 150, 50);
ctx.closePath();

ctx.beginPath();
ctx.rect(20, 50, 150, 40);

ctx.setFillStyle('red');
ctx.fillRect(20, 80, 120, 30);

ctx.rect(20, 150, 150, 40);

ctx.setFillStyle('blue');
ctx.fill();
ctx.draw();
```

moveTo

Move the path to a specified position on the canvas without creating a line.

Use the `stroke()` method to draw lines.

Parameters

Parameter	Type	Description
x	Number	X coordinate of target position
y	Number	Y coordinate of target position

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.moveTo(20, 20);
ctx.lineTo(150, 15);

ctx.moveTo(20, 55);
ctx.lineTo(120, 60);
ctx.stroke();
ctx.draw();
```

lineTo

Add a new position with the `lineTo` method and then create a line from the last specified position to the target position.

Use the `stroke()` method to draw lines.

Parameters

Parameter	Type	Description
x	Number	X coordinate of target position
y	Number	Y coordinate of target position

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.moveTo(20, 20);
ctx.rect(20, 20, 80, 30);
ctx.lineTo(120, 80);
ctx.stroke();
ctx.draw();
```

arc

Draw an arc.

- To create a circle, you can use the `arc()` method to specify a starting radian of 0 and a ending radian of `2 * Math.PI`.
- Use the `stroke()` or `fill()` method to draw an arc on the canvas.

Parameters

Parameter	Type	Description
x	Number	X coordinate of circle
y	Number	Y coordinate of circle
r	Number	Radius of circle
sAngle	Number	Start radian, in degrees (at 3 o'clock)
eAngle	Number	End radian
counterclockwise	Boolean	Optional, specify whether the direction of the radian is counterclockwise or clockwise. The default value is false.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.arc(200, 75, 50, 0, 2 * Math.PI);
ctx.setFillStyle('#CCCCCC');
ctx.fill();

ctx.beginPath();
ctx.moveTo(50, 65);
ctx.lineTo(170, 80);
ctx.moveTo(200, 35);
ctx.lineTo(200, 235);
ctx.setStrokeStyle('#AAAAAA');
ctx.stroke();

ctx.setFontSize(12);
ctx.setFillStyle('yellow');
ctx.fillText('0', 165, 78);
ctx.fillText('0.6*PI', 96, 148);
ctx.fillText('1*PI', 15, 57);
ctx.fillText('1.7*PI', 94, 20);

ctx.beginPath();
ctx.arc(200, 85, 2, 0, 2 * Math.PI);
ctx.setFillStyle('blue');
ctx.fill();

ctx.beginPath();
ctx.arc(200, 35, 2, 0, 2 * Math.PI);
ctx.setFillStyle('green');
ctx.fill();

ctx.beginPath();
ctx.arc(450, 60, 2, 0, 2 * Math.PI);
ctx.setFillStyle('red');
ctx.fill();

ctx.beginPath();
ctx.arc(150, 35, 50, 0, 1.8 * Math.PI);
ctx.setStrokeStyle('#666666');
ctx.stroke();

ctx.draw();
```

The three key coordinates for `arc(150, 35, 50, 0, 1.8 * Math.PI)` are as follows:

- Green: Center of circle (15, 35)
- Red: Start radian (0)
- Blue: End radian (1.8 * Math.PI)

bezierCurveTo

Create a cubic Bezier curve path.

The starting point of the curve is the previous point in the path.

Parameters

Parameter	Type	Description
cp1x	Number	X coordinate of the first Bezier control point
cp1y	Number	Y coordinate of the first Bezier control point
cp2x	Number	X coordinate of the second Bezier control point
cp2y	Number	Y coordinate of the second Bezier control point
x	Number	X coordinate of the end point
y	Number	Y coordinate of the end point

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.beginPath();
ctx.arc(30, 30, 2, 0, 2 * Math.PI);
ctx.setFillStyle('red');
ctx.fill();

ctx.beginPath();
ctx.arc(250, 25, 2, 0, 2 * Math.PI);
ctx.setFillStyle('blue');
ctx.fill();

ctx.beginPath();
ctx.arc(20, 100, 2, 0, 2 * Math.PI);
ctx.arc(200, 100, 2, 0, 2 * Math.PI);
ctx.setFillStyle('green');
ctx.fill();

ctx.setFillStyle('yellow');
ctx.setFontSize(14);

ctx.beginPath();
ctx.moveTo(30, 30);
ctx.lineTo(30, 100);
ctx.lineTo(150, 75);

ctx.moveTo(250, 30);
ctx.lineTo(250, 80);
ctx.lineTo(70, 75);
ctx.setStrokeStyle('#EEEEEE');
ctx.stroke();

ctx.beginPath();
ctx.moveTo(30, 30);
ctx.bezierCurveTo(30, 150, 250, 150, 180, 20);
ctx.setStrokeStyle('black');
ctx.stroke();

ctx.draw();
```

For `moveTo(30, 30)`, the three key coordinates for `bezierCurveTo(30, 150, 250, 150, 180, 20)` are as follows:

- Red: Start point (20, 20)
- Blue: Two control points (20, 150) and (250, 150)
- Green: End point (180, 20)

clip

Set the currently created path to the current clipping path.

quadraticCurveTo

Create a quadratic Bezier curve path.

The starting point of the curve is the previous point in the path.

Parameters

Parameter	Type	Description
cpx	Number	X coordinate of Bezier control point
cpy	Number	Y coordinate of Bezier control point
x	Number	X coordinate of the end point
y	Number	Y coordinate of the end point

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.beginPath();
ctx.arc(30, 30, 2, 0, 2 * Math.PI);
ctx.setFillStyle('red');
ctx.fill();

ctx.beginPath();
ctx.arc(250, 20, 2, 0, 2 * Math.PI);
ctx.setFillStyle('blue');
ctx.fill();

ctx.beginPath();
ctx.arc(30, 200, 2, 0, 2 * Math.PI);
ctx.setFillStyle('green');
ctx.fill();

ctx.setFillStyle('black');
ctx.setFontSize(12);

ctx.beginPath();
ctx.moveTo(30, 30);
ctx.lineTo(30, 150);
ctx.lineTo(250, 30);
ctx.setStrokeStyle('#AAAAAA');
ctx.stroke();

ctx.beginPath();
ctx.moveTo(30, 30);
ctx.quadraticCurveTo(30, 150, 250, 25);
ctx.setStrokeStyle('black');
ctx.stroke();

ctx.draw();
```

For `moveTo(30, 30)`, the three key coordinates for `quadraticCurveTo(30, 150, 250, 25)` are as follows:

- Red: Start point (30, 30)
- Blue: Control point (30, 150)
- Green: End point (250, 25)

scale

After the `scale` method is called, the horizontal and vertical coordinates for the path created afterwards will be scaled. If `scale` is called multiple times, scale levels will be multiplied.

Parameters

Parameter	Type	Description
scaleWidth	Number	Scale levels for horizontal coordinates (1 = 100%, 0.5 = 50%, 2 = 200%)
scaleHeight	Number	Scale levels for vertical coordinates (1 = 100%, 0.5 = 50%, 2 = 200%)

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.strokeRect(15, 15, 30, 25);
ctx.scale(3, 3);
ctx.strokeRect(15, 15, 30, 25);
ctx.scale(3, 3);
ctx.strokeRect(15, 15, 30, 25);

ctx.draw();
```

rotate

Centered on the origin, and the origin can be modified with the translate method. Rotate the current axis clockwise. If `rotate` is called multiple times, the rotation angles will be added up.

Parameters

Parameter	Type	Description
rotate	Number	Rotation angle in radians (degrees * Math.PI/180; degrees ranging from 0~360)

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.strokeRect(200, 20, 180, 150);
ctx.rotate(30 * Math.PI / 180);
ctx.strokeRect(200, 20, 180, 150);
ctx.rotate(30 * Math.PI / 180);
ctx.strokeRect(200, 20, 180, 150);

ctx.draw();
```

translate

Transform the origin (0, 0) of the current coordinate system. The default origin of the coordinate system is the top left corner of the page.

Parameters

Parameter	Type	Description
x	Number	Settoff for horizontal coordinate
y	Number	Settoff for vertical coordinate

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.strokeRect(20, 20, 250, 80);
ctx.translate(30, 30);
ctx.strokeRect(20, 20, 250, 80);
ctx.translate(30, 30);
ctx.strokeRect(20, 20, 250, 80);

ctx.draw();
```

setFontSize

Set the font size.

Parameters

Parameter	Type	Description
fontSize	Number	Font size

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.setFontSize(14);
ctx.fillText('14', 20, 20);;;
ctx.setFontSize(22)
ctx.fillText('22', 40, 40)
ctx.setFontSize(30);
ctx.fillText('30', 60, 60);
ctx.setFontSize(38);
ctx.fillText('38', 90, 90);

ctx.draw();
```

fillText

Draw the filled text on the canvas.

Parameters

Parameter	Type	Description
text	String	Text
x	Number	X coordinate of the top left corner of the drawn text
y	Number	Y coordinate of the top left corner of the drawn text

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.setFontSize(42)
ctx.fillText('Hello', 30, 30);;
ctx.fillText('alipay', 200, 200)

ctx.draw();
```

drawImage

Draw an image and the image remains in its original size.

Parameters

Parameter	Type	Description
imageResource	String	Image resources; only support online CDN address or offline package address, and the online CDN needs to return header `Access-Control-Allow-Origin: *`.
x	Number	X coordinate of the top left corner of image
y	Number	Y coordinate of the top left corner of image
width	Number	Image width
height	Number	Image height

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.drawImage('https://img.alicdn.com/tfs/TB1GvVMj2BNTKJjy0FdXXcPpVXa-520-280.jpg', 2, 2, 250, 80);
ctx.draw();
```

setGlobalAlpha

Set global brush transparency.

Parameters

Parameter	Type	Range	Description
alpha	Number	0~1	Transparency, 0 indicates completely transparent; 1 indicates opaque.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.setFillStyle('yellow');
ctx.fillRect(10, 10, 150, 100);
ctx.setGlobalAlpha(0.2);
ctx.setFillStyle('blue');
ctx.fillRect(50, 50, 150, 100);
ctx.setFillStyle('red');
ctx.fillRect(100, 100, 150, 100);

ctx.draw();
```

setLineDash

Set the style of dash line.

Parameters

Parameter	Type	Description
segments	Array	A set of numbers describing the length of alternately drawn line segments and spacing (units of coordinate space). If the number of array elements is odd, the array elements are copied and repeated. For example, the [5, 15, 25] will be changed to [5, 15, 25, 5, 15, 25].

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setLineDash([5, 15, 25]);
ctx.beginPath();
ctx.moveTo(0,100);
ctx.lineTo(400, 100);
ctx.stroke();
ctx.draw();
```

transform

A method in which a matrix is superimposed multiple times on the current transformation, and the matrix is described by the parameters of the method. You can scale, rotate, move, and tilt the context.

Parameters

Parameter	Type	Description
scaleX	Number	Scale horizontally.
skewX	Number	Skew horizontally.
skewY	Number	Skew vertically.
scaleY	Number	Scale vertically.
translateX	Number	Translate horizontally.
translateY	Number	Translate vertically.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.rotate(45 * Math.PI / 180);
ctx.setFillStyle('red');
ctx.fillRect(70,0,100,30);
ctx.transform(1, 1, 0, 1, 0, 0);
ctx.setFillStyle('#000');
ctx.fillRect(0, 0, 100, 100);
ctx.draw();
```

setTransform

Reset the transform and invoke the transform by unit matrix. The transform is described by the variable of the method.

Parameters

Parameter	Type	Description
scaleX	Number	Scale horizontally.
skewX	Number	Skew horizontally.
skewY	Number	Skew vertically.
scaleY	Number	Scale vertically.
translateX	Number	Translate horizontally.
translateY	Number	Translate vertically.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.rotate(45 * Math.PI / 180);
ctx.setFillStyle('red');
ctx.fillRect(70,0,100,30);
ctx.setTransform(1, 1, 0, 1, 0, 0);
ctx.setFillStyle('#000');
ctx.fillRect(0, 0, 100, 100);
ctx.draw();
```

getImageData

Get the pixel data of the implicit area of the canvas.

> **Note** The base library with version lower than 1.10 is not supported.

Parameters

Parameter	Type	Required	Description
x	Number	Yes	The x coordinate of left-upper corner of the image rectangular area that need to be get.
y	Number	Yes	The y coordinate of left-upper corner of the image rectangular area that need to be get.
width	Number	Yes	The width of the image rectangular area that need to be gotten.
height	Number	Yes	The height of the image rectangular area that need to be gotten.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.

Parameter	Type	Required	Description
complete	Function	No	Callback function upon call completion.

Callback parameters on success

Parameter	Type	Description
width	Number	The width of the image rect area.
height	Number	The height of the image rect area.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.getImageData({
  x: 0,
  y: 0,
  width: 100,
  height: 100,
  success(res) {
    console.log(res.width) // 100
    console.log(res.height) // 100
    console.log(res.data instanceof Uint8ClampedArray) // true
    console.log(res.data.length) // 100 * 100 * 4
  }
})
```

putImageData

Draw the pixel data into the canvas.

> **Note** The base library with version lower than 1.11 is not supported.

Parameters

Parameter	Type	Required	Description
data	Uint8ClampedArray	Yes	Image pixel data, it is an array, and the four elements represent the rgba data.
x	Number	Yes	The x offset for the original image data in the destination canvas.
y	Number	Yes	The y offset for the original image data in the destination canvas.
width	Number	Yes	The width of the original image data.
height	Number	Yes	The height of the original image data.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion.

Code example

```
const data = new Uint8ClampedArray([255, 0, 0, 1])
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.putImageData({
    x: 0,
    y: 0,
    width: 1,
    height: 1,
    data: data,
    success(res) {}
})
```

save

Save the current drawing context.

Code example

```
//.js
const ctx = my.createCanvasContext('myCanvas')
// save the default fill style
ctx.save()
ctx.setFillStyle('red')
ctx.fillRect(10, 10, 150, 100)
// restore to the previous saved state
ctx.restore()
ctx.fillRect(50, 50, 150, 100)
ctx.draw()
```

restore

Restore the previously saved drawing context.

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.save();
ctx.setFillStyle('red');
ctx.fillRect(20, 20, 250, 80);

ctx.restore();
ctx.fillRect(60, 60, 155, 130);

ctx.draw();
```

draw

Draw the description (path, transformation, style) previously existed in the drawing context onto the canvas.

The drawing context needs to be created by `my.createCanvasContext(canvasId)`.

Parameters

Parameter	Type	Description	Lowest base library version supported
reserve	Boolean	Optional. Whether this drawing will follow the last drawing; that is, if the `reserve` parameter is false, the native layer will clear the canvas first and then continue drawing by calling `drawCanvas`; if the `reserve` parameter is true, the content on the current canvas is retained and then `drawCanvas` is called to override it. The default is false.	-
callback	Function	Required, the callback function to be executed when the drawing is done.	1.10

Code example

```
const ctx = my.createCanvasContext('awesomeCanvas');

ctx.setFillStyle('blue');
ctx.fillRect(20, 20, 180, 80);
ctx.draw();
ctx.fillRect(60, 60, 250, 120);
ctx.draw(true);
```

1.10.3.11. Keyboard

Hide the keyboard.

my.hideKeyboard

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Hide the keyboard.

Code example

```
// API-DEMO page/API/keyboard/keyboard.json
{
  "defaultTitle": "Keyboard"
}
```

```
<!-- API-DEMO page/API/keyboard/keyboard.axml-->
<view class="page">
  <view class="page-description">Input box</view>
  <view class="page-section">
    <view class="form-row">
      <view class="form-row-label">Password keyboard</view>
      <view class="form-row-content">
        <input class="input" password type="text" onInput="bindHideKeyboard" placeholder="Input 123 to automatically hide the keyboard" />
      </view>
    </view>
    <view class="form-row">
      <view class="form-row-label">Numeric keyboard</view>
      <view class="form-row-content">
        <input class="input" type="digit" onInput="bindHideKeyboard" placeholder="Input 123 to automatically hide the keyboard" />
      </view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/keyboard/keyboard.js
Page({
  bindHideKeyboard(e) {
    if (e.detail.value === "123") {
      //Hide keyboard
      my.hideKeyboard();
    }
  },
});
```

1.10.3.12. Scroll

Scroll to the target position of the page.

my.pageScrollTo

Scroll to the target position of the page.

Parameter Description

Parameter name	Type	Description
scrollTop	Number	Scroll to the target position of the page, in px

Code example

```
my.pageScrollTo({
  scrollTop: 100
})
```

1.10.3.13. Node query

my.createIntersectionObserver

Create and return an IntersectionObserver object instance. You need to run `my.createIntersectionObserver()` after `page.onReady`.

Version requirement: Base library V1.11.0 or later versions. For earlier versions, you are suggested to perform Compatibility processing.

Parameters

The input parameters are of Object type and have the following attributes:

Attribute	Type	Default value	Description
thresholds	Array `<Number>`	[0]	A numeric array containing all thresholds.
initialRatio	Number	0	The initial intersection ratio. If the intersection ratio detected during the operation is not equal to this value and reaches the threshold, the callback function of the listener will be triggered.
selectAll	Boolean	false	Whether to monitor multiple target nodes (rather than one) at the same time. If this parameter is set to true, the monitoring targetSelector selects multiple nodes. Note: When you select too many nodes at the same time, this will affect rendering performance.

Return value

```
IntersectionObserver
```

Sample code

```
<!-- .axml -->
<view class="logo" style='width: 200px;height: 200px;background-color:blue'>11</view>
```

```
Page({
  onReady() {
    my.createIntersectionObserver().relativeToViewport({top: 100, bottom: 100}).observe('.logo', (res) => {
      console.log(res, 'intersectionObserver');
        console.log(res.intersectionRatio); // The ratio of the intersection area to the layout area of the target node.
        console.log(res.intersectionRect); // The intersection area.
        console.log(res.relativeRect); // The boundaries of the reference area.
        console.log(res.boundingClientRect); // The target boundaries.
        console.log(res.time); // The timestamp.
        console.log(res.id);
    });
  }
})
```

IntersectionObserver

Infers whether certain nodes can be seen by the user and how many nodes can be seen by the user.

IntersectionObserver.disconnect

Stops the monitoring. The callback function will not be triggered.

IntersectionObserver.observe

Specifies the target node and starts monitoring intersection status changes.

Parameters

The format of input parameters is: `(String targetSelector, function callback)`

- String targetSelector indicates the selector.
- Function callback indicates the callback function for monitoring intersection status changes.

Callback parameters

Object res attributes

Attribute	Type	Description
intersectionRatio	Number	The intersection ratio.
intersectionRect	Object	The boundaries of the intersection area.
boundingClientRect	Object	The target boundaries.
relativeRect	Object	The boundaries of the reference area.
time	Number	The timestamp of intersection detection.

res.intersectionRect attributes

Attribute	Type	Description
left	Number	The left boundary.
right	Number	The right boundary.
top	Number	The upper boundary.
bottom	Number	The lower boundary.

res.boundingClientRect attributes

Attribute	Type	Description
left	Number	The left boundary.
right	Number	The right boundary.
top	Number	The upper boundary.
bottom	Number	The lower boundary.

res.relativeRect attributes

Attribute	Type	Description
left	Number	The left boundary.
right	Number	The right boundary.

Attribute	Type	Description
top	Number	The upper boundary.
bottom	Number	The lower boundary.

IntersectionObserver.relativeTo

Uses the selector to specify a node as a reference area.

Parameters

The format of input parameters is: `(String selector,Object margins)`

- String selector indicates the selector.
- Object margins is used to expand or shrink the boundaries of the layout area of the reference node with the following attributes:

Attribute	Type	Required	Description
left	Number	No	The left boundary of the node layout area.
right	Number	No	The right boundary of the node layout area.
top	Number	No	The upper boundary of the node layout area.
bottom	Number	No	The lower boundary of the node layout area.

IntersectionObserver.relativeToViewport

Specifies a page display area as a reference area.

Parameters

The input parameter is “Object margins”, which is used to expand or shrink the boundaries of the layout area of the reference node with the following attributes:

Attribute	Type	Required	Description
left	Number	No	The left boundary of the node layout area.
right	Number	No	The right boundary of the node layout area.
top	Number	No	The upper boundary of the node layout area.
bottom	Number	No	The lower boundary of the node layout area.

my.createSelectorQuery

> **Note**
> - This interface is supported since basic library version 1.4.0.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

Obtains the node query object `SelectorQuery`.

Parameters

Parameter	Type	Description
params	Object	You can specify the `page` attribute, which defaults to the current page.

Sample code

```
<!-- API-DEMO page/API/create-selector-query/create-selector-query.axml-->
<view class="page">
  <view class="page-description">The node query API</view>
  <view class="page-section">
    <view className="all">Node all1</view>
    <view className="all">Node all2</view>
    <view id="one">Node one</view>
    <view id="scroll" style="height:200px;overflow: auto">
      <view style="height:400px">The independent scrolling area</view>
    </view>
    <button type="primary" onTap="createSelectorQuery">Node query</button>
  </view>
</view>
```

```
// API-DEMO page/API/create-selector-query/create-selector-query.js
Page({
  createSelectorQuery() {
    my.createSelectorQuery()
      .select('#non-exists').boundingClientRect()
      .select('#one').boundingClientRect()
      .selectAll('.all').boundingClientRect()
      .select('#scroll').scrollOffset()
      .selectViewport().boundingClientRect()
      .selectViewport().scrollOffset().exec((ret) => {
      console.log(ret);
      my.alert({
        content: JSON.stringify(ret, null, 2),
      });
    })
  },
});
```

ret structure

```
[
  null,
  {
    "x": 1,
    "y": 2,
    "width": 1367,
    "height": 18,
    "top": 2,
    "right": 1368,
    "bottom": 20,
    "left": 1
  },
  [
    {
      "x": 1,
      "y": -34,
      "width": 1367,
      "height": 18,
      "top": -34,
      "right": 1368,
      "bottom": -16,
      "left": 1
    },
    {
      "x": 1,
      "y": -16,
      "width": 1367,
      "height": 18,
      "top": -16,
      "right": 1368,
      "bottom": 2,
      "left": 1
    }
  ],
  {
    "scrollTop": 0,
    "scrollLeft": 0
  },
  {
    "width": 1384,
    "height": 360
  },
  {
    "scrollTop": 35,
    "scrollLeft": 0
  }
]
```

SelectorQuery

The node query object class, which includes the following methods:

selectorQuery.select(selector)

Selects the first node that matches the selector. The selector supports the ID and class selectors.

selectorQuery.selectAll(selector)

Selects all nodes that match the selector. The selector supports the ID and class selectors.

selectorQuery.selectViewport()

Selects the window object.

selectorQuery.boundingClientRect()

Places the position information of the currently selected node into the query result. This function is similar to `getBoundingClientRect` of dom, and the returned objects are width, height, left, top, bottom, and right. If the current node is a window object, only width and height are returned.

selectorQuery.scrollOffset()

Places the scrolling information of the currently selected node into the query result. The returned objects are scrollTop and scrollLeft.

selectorQuery.exec(callback)

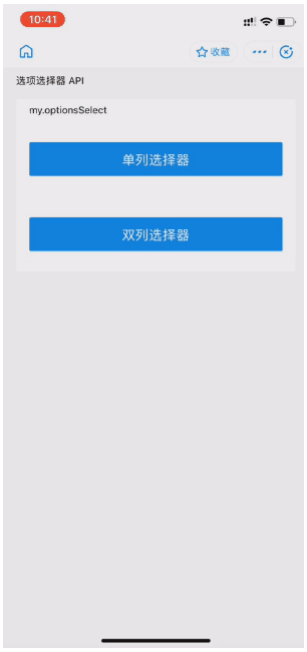
Places the query result into the `callback` for callback. The query result is an array, and each item in the array is the result of a query. If the item is a node list, the result of a single query is also an array. Note that you must perform the `Page onReady` operation before `exec`.

1.10.3.14. Option selector

my.optionsSelect

A component similar to the native Safari `select` but with more powerful features, which you can use to replace the native `select` or level 2 data. Note: The component does not support interactions among level 2 data.

Effect



Sample code

```
// API-DEMO page/API/options-select/options-select.json
{
    "defaultTitle": "Option selector"
}
```

```
<!-- API-DEMO page/API/options-select/options-select.axml-->
<view class="page">
  <view class="page-description">Option selector API</view>
  <view class="page-section">
    <view class="page-section-title">my.optionsSelect</view>
    <view class="page-section-demo">
      <button type="primary" onTap="openOne">Single column selector</button>
    </view>
    <view class="page-section-demo">
      <button type="primary" onTap="openTwo">Double column selector</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/options-select/options-select.js
Page({
  openOne() {
    my.optionsSelect({
      title: "Select the repayment date",
      optionsOne: ["Every Monday", "Every Tuesday", "Every Wednesday", "Every Thursday", "Every Friday", "Every Saturday", "Every Sunday"],
      selectedOneIndex: 2,
      success(res) {
        my.alert({
          content: JSON.stringify(res, null, 2),
        });
      }
    });
  },
  openTwo() {
    my.optionsSelect({
      title: "Select the year and month of birth",
      optionsOne: ["2014", "2013", "2012", "2011", "2010", "2009", "2008"],
      optionsTwo: ["January", 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'],
      selectedOneIndex: 3,
      selectedTwoIndex: 5,
      success(res) {
        my.alert({
          content: JSON.stringify(res, null, 2),
        });
      }
    });
  },
});
```

Parameters

The input parameters are in Object type and have the following attributes:

Name	Type	Description	Required	Default value
title	String	Header title information	No	-
optionsOne	String[]	List for option one	Required	-
optionsTwo	String[]	List for option two	No	-
selectedOneIndex	Number	It means option one is selected by default.	No	0
selectedTwoIndex	Number	It means option two is selected by default.	No	0
positiveString	String	The text that is displayed on the Confirm button	No	Confirm
negativeString	String	The text that is displayed on the Cancel button	No	Cancel

Callback function upon success

The input parameters are in Object type and have the following attributes:

Name	Type	Description	Remarks
selectedOneIndex	Number	Selected value in option one	If you cancel the selection, an empty string will be returned.
selectedOneOption	String	Selected content in option one	If you cancel the selection, an empty string will be returned.
selectedTwoIndex	Number	Selected value in option two	If you cancel the selection, an empty string will be returned.
selectedTwoOption	String	Selected content in option two	If you cancel the selection, an empty string will be returned.

1.10.3.15. Multilevel select

Cascaded selection is mainly used for selecting several levels of associated data, such as province, city and district.

my.multiLevelSelect(Object)

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Cascaded selection is mainly used for selecting several levels of associated data, such as province, city and district.

Input parameters

Parameter	Type	Required	Description
title	String	No	Title
list	JsonArray	Yes	Selection data list
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

list objects

Parameter	Type	Required	Description
name	String	Yes	Entry name
subList	JsonArray	No	Sub-entry list.

Output parameters

Parameter	Type	Description
success	Boolean	Selection completed or not, returning false for cancellation.
result	JsonArray	Selection result, for example, [{ “name” :” Hangzhou” },{ “name” :” Xihu District” }, { “name” :” Gucui Subdistrict” }].

Code sample

```
// API-DEMO page/API/multi-level-select/multi-level-select.json
{
     "defaultTitle": "cascaded selector"
}
```

```
<!-- API-DEMO page/API/multi-level-select/multi-level-select.axml-->
<view class="page">
  <view class="page-description">cascaded selector API</view>
  <view class="page-section">
    <view class="page-section-title">my.multiLevelSelect</view>
    <view class="page-section-demo">
      <button type="primary" onTap="openMultiLevelSelect">cascaded selector</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/multi-level-select/multi-level-select.js
Page({
  openMultiLevelSelect() {
    my.multiLevelSelect({
        title: 'cascaded selector',//Title for cascaded selection
        list: [
        {
            name: "Hangzhou",//Entry name
            subList: [
                {
                    name: "Xihu District",
                    subList: [
                        {
                            name: "Gucui Subdistrict"
                        },
                        {
                            name: "Wenxin Subdistrict "
                        }
                    ]
                },
                {
                    name: "Shangcheng District",
                    subList: [
                        {
                            name: "Yan'an Subdistrict"
                        },
                        {
                            name: "Longxianggiao Subdistrict"
                        }
                    ]
                }
            ]//sub-level data list
        }],//data list
        success:(res)=>{
            my.alert({title:JSON.stringify(res)})
        }
    });
  }
})
```

1.10.3.16. Set background color

This API is supported in mPaaS 10.1.32 and later versions.

my.setBackgroundColor

This API is supported in mPaaS 10.1.32 and later versions.

Dynamically set window background color.

Input parameters

Attribute	Type	Required	Description
backgroundColor	HexColor	Yes	Background color of the window.
backgroundColorTop	HexColor	Yes	Background color of the window top, only supported in iOS.
backgroundColorBottom	HexColor	Yes	Background color of the window bottom, only supported in iOS.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Code sample

```
/*Set window background color*/
my.setBackgroundColor({
    backgroundColor: '#ffffff'
})

/*Set background color of window top and bottom respectively*/
my.setBackgroundColor({
    backgroundColorTop: '#ffffff',
    backgroundColorBottom: '#ffffff'
})
```

my.setBackgroundTextStyle

This API is supported in mPaaS 10.1.32 and later versions.

Dynamically set the font of the pull-down background and the style of loading graphics.

Input parameters

Attribute	Type	Required	Description
textStyle	String	Yes	The font of the pull-down background and the style of loading graphics. Only `dark` and `light` are supported.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Code sample

```
my.setBackgroundTextStyle({
    textStyle: 'dark', // Font of the pull-down background and the style of loading graphics are dark.
})
```

1.10.3.17. Set page pulldown

Set whether to support pulldown on the page (supported by default on Mini Program pages).

my.setCanPullDown

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Set whether to support pulldown on the page (supported by default on Mini Program pages).

Parameters

Parameter	Type	Required	Description
canPullDown	Boolean	Yes	Support pulldown or not.

Code sample

```
my.setCanPullDown({
  canPullDown:true
})
```

1.10.3.18. Settings

my.setOptionMenu

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to configure additional icons on the `optionMenu` navigation bar. Clicking any of these icons triggers `onOptionMenuClick`.

Parameters

Name	Type	Required	Description
icon	String	Required	Customizes the URL, which starts with `https` or `http`, or Base64 string of the icon used by optionMenu. We recommend that you set the size of the icon to 30×30. Base64 images are currently not supported.

Notes on icon attributes

- Subject to iOS App Transport Security (ATS) restrictions, icon URLs must be HTTPS links or in Base64 format, and HTTP links will be ignored.
- If the icon is in Base64 format, only the vector format is supported, and the prefix must not be `data:image/png;base64`.

Sample code

```
my.setOptionMenu({
  icon: 'https://img.alicdn.com/tps/i3/T1OjaVFl4dXXa.JOZB-114-114.png',
});
```

1.10.4. Multimedia

1.10.4.1. Image

my.chooseImage

This API is supported in mPaaS 10.1.32 and later versions.

This interface is used to take a photo or select a photo from the mobile phone album.

> Note
> The array of the image path has a suffix of `.png` on the IDE, and `.image` on the preview of the real device. Please refer to the effect of the real device.

Input parameter

Name	Type	Required	Description
count	Number	No	Maximum number of photos that can be selected, the default value is 1.
sourceType	String Array	No	Select from album or take a photo, the default value is ['camera','album'].
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

success return value

Name	Type	Description
apFilePaths	String Array	The array of image file path.

Error Code

error	Description	Solution
11	User cancel operation	This is a user's normal interaction process and does not require special processing.

Code sample

```
// API-DEMO page/API/image/image.json
{
    "defaultTitle": "Image"
}
```

```
<!-- API-DEMO page/API/image/image.axml -->
<view class="page">
  <view class="page-section">
    <view class="page-section-btns">
      <view onTap="chooseImage">Select image</view>
      <view onTap="previewImage">Preview image</view>
      <view onTap="saveImage">Save image</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/image/image.js
Page({
  chooseImage() {
    my.chooseImage({
      sourceType: ['camera','album'],
      count: 2,
      success: (res) => {
        my.alert({
          content: JSON.stringify(res),

        });
      },
      fail:()=>{
        my.showToast({
          content: 'fail', // Text content
        });
      }
    })
  },
  previewImage() {
    my.previewImage({
      current: 2,
      urls: [
        'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1pfG4IFXXXXc6XXXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1h9xxIFXXXXbKXXXXXXXXXXXX.jpg'
      ],
    });
  },
  saveImage() {
    my.saveImage({
      url: 'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
      showActionSheet: true,
      success: () => {
        my.alert({
          title: 'Saved',
        });
      },
    });
  }
});
```

my.previewImage

This API is supported in mPaaS 10.1.32 and later versions.

This interface is used to preview the image. Currently, previewing local image is not supported.

The base library version 1.0.0 does not support the combined use of `my.previewImage` and `my.chooseImage` on iOS.

Input parameter

Name	Type	Required	Description
urls	Array	Yes	List of image links to be previewed, URL and apfilePath are supported.
current	Number	No	Currently display image index, the default value is 0, which means the first image in `urls`.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).
enablesavephoto	Boolean	No	Support press and hold to download photos (Supported since basic the library 1.13.0).
enableShowPhotoDownload	Boolean	No	Whether to display the download entry in the lower right corner, it needs to be used with the enablesavephoto parameter (Supported since the basic library 1.13.0).

Code sample

```
// API-DEMO page/API/image/image.json
{
    "defaultTitle": "image"
}
```

```
<!-- API-DEMO page/API/image/image.axml -->
<view class="page">
  <view class="page-section">
    <view class="page-section-btns">
      <view onTap="chooseImage">Select image</view>
      <view onTap="previewImage">Preview image</view>
      <view onTap="saveImage">Save image</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/image/image.js
Page({
  chooseImage() {
    my.chooseImage({
      sourceType: ['camera','album'],
      count: 2,
      success: (res) => {
        my.alert({
          content: JSON.stringify(res),

        });
      },
      fail:()=>{
        my.showToast({
          content: 'fail', // Text content
        });
      }
    })
  },
  previewImage() {
    my.previewImage({
      current: 2,
      urls: [
        'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1pfG4IFXXXXc6XXXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1h9xxIFXXXXXbKXXXXXXXXXXXX.jpg'
      ],
    });
  },
  saveImage() {
    my.saveImage({
      url: 'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
      showActionSheet: true,
      success: () => {
        my.alert({
          title: 'Saved',
        });
      },
    });
  }
});
```

my.saveImage

This API is supported in mPaaS 10.1.32 and later versions.

This interface is used to save Web images to your phone's photo album.

Input parameter

Name	Type	Required	Description
url	String	Yes	Link of the image to be saved.
showActionSheet	Boolean	No	Whether to display the image operation menu, the default value is true (Supported since basic the library 1.24.0).
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.

Name	Type	Required	Description
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Error Code

error	Description	Solution
2	Invalid parameter, no url parameter passed	Pass in the correct URL parameter.
15	Permission to access the album is not granted (iOS only)	Grant the permission to access the album.
16	Insufficient storage space for mobile phone album (iOS only)	Clear storage space.
17	Other errors in saving the image	Try again later.

FAQ

- **Q**: Can `my.saveImage` API save Base64 images?

  A: Currently, `my.saveImage` cannot save Base64 images.

Code sample

```
// API-DEMO page/API/image/image.json
{
    "defaultTitle": "Image"
}
```

```
<!-- API-DEMO page/API/image/image.axml -->
<view class="page">
  <view class="page-section">
    <view class="page-section-btns">
      <view onTap="chooseImage">Select image</view>
      <view onTap="previewImage">Preview image</view>
      <view onTap="saveImage">Save image</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/image/image.js
Page({
  chooseImage() {
    my.chooseImage({
      sourceType: ['camera','album'],
      count: 2,
      success: (res) => {
        my.alert({
          content: JSON.stringify(res),

        });
      },
      fail:()=>{
        my.showToast({
          content: 'fail', // Text content
        });
      }
    })
  },
  previewImage() {
    my.previewImage({
      current: 2,
      urls: [
        'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1pfG4IFXXXXc6XXXXXXXXXXX.jpg',
        'https://img.alicdn.com/tps/TB1h9xxIFXXXXbKXXXXXXXXXXX.jpg'
      ],
    });
  },
  saveImage() {
    my.saveImage({
      url: 'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
      showActionSheet: true,
      success: () => {
        my.alert({
          title: 'Saved',
        });
      },
    });
  }
});
```

my.compressImage

This API is supported in mPaaS 10.1.32 and later versions and from the basic library version 1.4.0. Compatibility process is required for lower basic library versions.

This interface is used to compress the images.

Input parameter

Name	Type	Required	Description
apFilePaths	String Array	Yes	Address array of the image to be compressed.
compressLevel	Number	No	Compression level, support integers from 0 to 4, the default value is 4. For details, see compressLevel table.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

success return value

Name	Type	Description
apFilePaths	String Array	Compressed path array

compressLevel table

compressLevel	Description
0	Low quality
1	Medium quality
2	High quality
3	Not compressed
4	Adaptive according to network connection

Code sample

```
<!-- API-DEMO page/API/compress-image/compress-image.axml-->
<view class="page">
  <view class="page-description">Compress image API</view>
  <view class="page-section">
    <view class="page-section-title">my.compressImage</view>
    <view class="page-section-demo">
      <button type="primary" onTap="selectImage" hover-class="defaultTap">Select image</button>
      <image
        src="{{compressedSrc}}"
        mode="{{mode}}" />
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/compress-image/compress-image.js
Page({
  data: {
    compressedSrc: '',
    mode: 'aspectFit',
  },
  selectImage() {
    my.chooseImage({
      count: 1,
      success: (res) => {
        my.compressImage({
          apFilePaths: res.apFilePaths,
          compressLevel: 1,
          success: data => {
            console.log(data);
            this.setData({
              compressedSrc: data.apFilePaths[0],
            })
          }
        })
      }
    })
  },
});
```

my.getImageInfo

This API is supported in mPaaS 10.1.32 and later versions and from the basic library version 1.4.0. Compatibility process is required for lower basic library versions.

This interface is used to get image information.

Input parameter

Name	Type	Required	Description
src	String	No	Image path, currently support: • Online image path • apFilePath path • Relative path
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.

Name	Type	Required	Description
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

success return value

Name	Type	Description
width	Number	Image width (in px).
height	Number	Image height (in px).
path	String	Local path of image.
orientation	String	Direction of the returned image.
type	String	Format of the returned image.

orientation parameter description

Enumeration value	Description
up	Default value
down	180 degree rotation
left	Rotate 90 degrees counterclockwise
right	Rotate 90 degrees clockwise
up-mirrored	Same as `up`, but flipped horizontally
down-mirrored	Same as `down`, but flipped horizontally
left-mirrored	Same as `left`, but flipped vertically
right-mirrored	Same as `right`, but flipped vertically

Code sample

```
//Online image path
my.getImageInfo({
      src:'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
      success:(res)=>{
        console.log(JSON.stringify(res))
      }
    })

//apFilePath
my.chooseImage({
      success: (res) => {
        my.getImageInfo({
          src:res.apFilePaths[0],
          success:(res)=>{
            console.log(JSON.stringify(res))
          }
        })
      },
    })

//Relative path
my.getImageInfo({
      src:'image/api.png',
      success:(res)=>{
        console.log(JSON.stringify(res))
      }
    })
```

1.10.4.2. Video

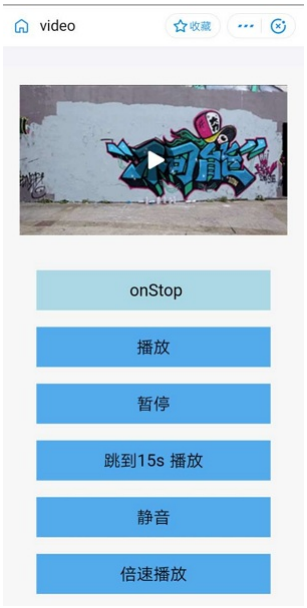
my.createVideoContext

You can call this interface to input video id and return a `videoContext`. video ID refers to the ID attribute customized by the developer in the corresponding video tab.

You can operate a video component through `videoContext`.

Version requirement: Base library V1.14.1 or later versions.

Effect



Sample code

Write the following code to name a video id in `.axml` file. video ID refers to the ID attribute customized by the developer in the corresponding video tab, such as `myVideo` in the following code.

```
<view>

<! -- If the type of onPlay is EventHandle, the play event will be triggered when you start or continue playing. -->
  <video id="myVideo" src="{{src}}" onPlay="{{onPlay}}" enableNative="{{true}}"></video>
    <button type="default" size="defaultSize" onTap="play"> Play </button>
    <button type="default" size="defaultSize" onTap="pause"> Pause </button>
    <button type="default" size="defaultSize" onTap="stop"> stop </button>
    <button type="default" size="defaultSize" onTap="seek"> seek </button>
    <button type="default" size="defaultSize" onTap="requestFullScreen"> requestFullScreen </button>
    <button type="default" size="defaultSize" onTap="exitFullScreen"> exitFullScreen </button>
    <button type="default" size="defaultSize" onTap="mute"> mute </button>
</view>
```

Write the following code in the `.js` file:

```
Page({
  data: {

  // src is the resource address of the video you want to play. Src supports apFilePath: https://resource/xxx.video.
    src: "http://flv.bn.netease.com/tvmrepo/2012/7/C/7/E868IGRC7-mobile.mp4",
  },
  onLoad() {
    this.videoContext = my.createVideoContext('myVideo');
  },

  play() {
    this.videoContext.play();
  },

  pause() {
    this.videoContext.pause();
  },

  stop() {
    this.videoContext.stop();
  },

  seek() {
    this.videoContext.seek(100);
  },

  requestFullScreen() {
    this.videoContext.requestFullScreen({
      direction: 0
    });
  },

  exitFullScreen() {
    this.videoContext.exitFullScreen();
  },

  mute() {
    this.videoContext.mute(false);
  },

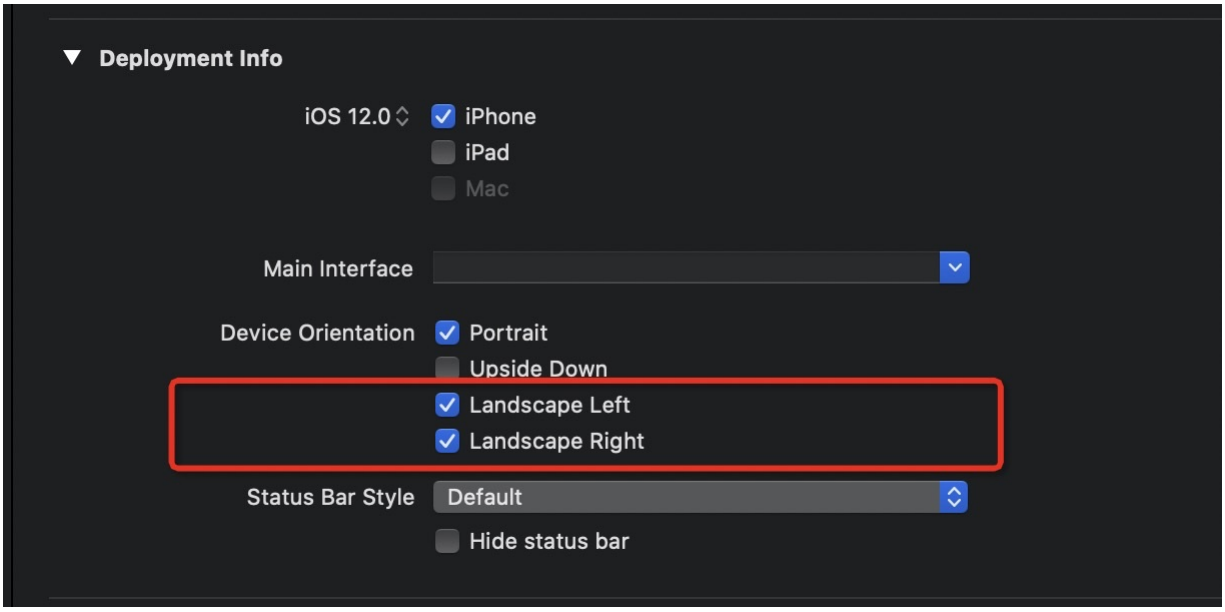

});
```

videoContext methods

Method	Parameter	Type	Description
play	None	-	Play.
pause	None	-	Pause.
stop	None	-	Stop.
seek	position	Number	Redirect to the specific position. The unit is second.
requestFullScreen	direction	Number	Enter full screen mode. • 0 is vertical screen. • 90 is horizontal screen. • -90 is reversed horizontal screen.
exitFullScreen	None	-	Exit full screen mode.
showStatusBar	None	-	Show status bar. The method only works in full screen mode in iOS App.
hideStatusBar	None	-	Hide status bar. The method only works in full screen mode in iOS App.

Method	Parameter	Type	Description
mute	ison	Boolean	Switch to mute status.
playbackRate	rate	Number	Set playback rate (0.5 ≤ rate ≤ 2.0). **This method is not supported by mPaaS currently.**

Note To use iOS full screen, you need to check **Landscape Left** and **Landscape Right** in Xcode> **General**> **Deployment Info**, as shown below.



1.10.5. Cache

This API is used to store the data in the specified key in local cache, which will override the original data that the key coresopnds to.

my.setStorage

This API is used to store the data in the specified key in local cache, which will override the original data that the key coresopnds to.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data
data	Object/String	Yes	Data to be cached
success	Function	No	Callback function for call success
fail	Function	No	Callback function for call failure
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

Code sample

```
my.setStorage({
  key: 'currentCity',
  data: {
    cityName: 'Hangzhou',
    adCode: '330100',
    spell: ' hangzhou',
  },
  success: function() {
    my.alert({content: 'Write successful'});
  }
});
```

Note: After converting a single piece of data into a string, the maximum length of the string is 200×1024 characters. For the same mPaaS mini program user, the total cache limit for a single mini program is 10 MB.

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.setStorageSync

This API is used to synchronously store data in the specified key in local cache.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data
data	Object/String	Yes	Data to be cached

Code sample

```
my.setStorageSync({
  key: 'currentCity',
  data: {
    cityName: 'Hangzhou',
    adCode: '330100',
    spell: ' hangzhou',
  }
});
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.getStorage

This API is used to get the cached data.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data
success	Function	No	Callback function for call success
fail	Function	No	Callback function for call failed
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
data	Object/String	Content that the key corresponds to

Code sample

```
my.getStorage({
  key: 'currentCity',
  success: function(res) {
    my.alert({content: 'Successfully got: ' + res.data.cityName});
  },
  fail: function(res){
    my.alert({content: res.errorMessage});
  }
});
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.getStorageSync

This API is used to synchronously get the cached data.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data

success return value

Name	Type	Description
data	Object/String	Content that the key corresponds to

Code sample

```
let res = my.getStorageSync({ key: 'currentCity' });
 my.alert({
    content: JSON.stringify(res.data),
 });
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.removeStorage

This API is used to delete cached data.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data
success	Function	No	Callback function for call successful
fail	Function	No	Callback function for call failed
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

Code sample

```
my.removeStorage({
  key: 'currentCity',
  success: function(){
    my.alert({content: 'Successfully deleted'});
  }
});
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.removeStorageSync

This API is used to synchronously delete cached data.

Note

Input parameter

Name	Type	Required	Description
key	String	Yes	key of the cached data

Code sample

```
my.removeStorageSync({
  key: 'currentCity',
});
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.clearStorage

This API is used to clear local data cache.

Note

Code sample

```
my.clearStorage()
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.clearStorageSync

This API is used to synchronously clear local data cache.

Note

Code sample

```
my.clearStorageSync()
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.getStorageInfo

This API is used to asynchronously get the information about the current storage.

Note

Input parameter

Name	Type	Required	Description
success	Function	No	Callback function for call successful
fail	Function	No	Callback function for call failed
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
keys	String Array	All keys in the current storage
currentSize	Number	The size of space currently occupied, in KB
limitSize	Number	The size limit of the space, in KB

Code sample

```
my.getStorageInfo({
  success: function(res) {
    console.log(res.keys)
    console.log(res.currentSize)
    console.log(res.limitSize)
  }
})
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

my.getStorageInfoSync

This API is used to synchronously get the information about the current storage.

Note

success return value

Name	Type	Description
keys	String Array	All keys in the current storage
currentSize	Number	The size of space currently occupied, in KB
limitSize	Number	The size limit of the space, in KB

Code sample

```
var res = my.getStorageInfoSync()
console.log(res.keys)
console.log(res.currentSize)
console.log(res.limitSize)
```

Other information

- Cached data is encrypted and stored locally and will be automatically decrypted and returned when read through API.
- iTunes backup supported in iOS client.

1.10.6. File

my.saveFile

> Note
> - This interface is supported since basic library version 1.3.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to save the file to local. The maximum local file capacity is 10 MB. After `my.saveFile` is successfully called, you can find the saved files in the path `storage/alipay/pictures/file location` in Android device. Hidden file path in iOS device cannot be found.

Parameters

Parameter	Type	Required	Description
apFilePath	String	Yes	File path
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Return value on success

Parameter	Type	Description
apFilePath	String	File save path

Sample code

```
my.chooseImage({
  success: (res) => {
    my.saveFile({
      apFilePath: res.apFilePaths[0],
      success: (res) => {
        console.log(JSON.stringify(res))
      },
    });
  },
});
```

my.getFileInfo

> Note
> - This interface is supported since basic library version 1.4.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to get file information.

Parameters

Parameter	Type	Required	Description
apFilePath	String	Yes	File path (local path)
digestAlgorithm	String	No	Digest algorithm, supporting md5 and sha1, md5 by default.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Return value on success

Parameter	Type	Description
size	Number	File size
digest	String	Digest result.

Sample code

```
my.getFileInfo({
    apFilePath:'https://resource/apml953bb093ebd2834530196f50a4413a87.video',
    digestAlgorithm:'sha1',
    success:(res)=>{
        console.log(JSON.stringify(res))
    }
})
```

my.getSavedFileInfo

> **Note**
> - This interface is supported since basic library version 1.3.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to get the information of the saved files.

Parameters

Parameter	Type	Required	Description
apFilePath	String	Yes	File path
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Return value on success

Parameter	Type	Description
size	Number	File size
createTime	Number	Creation time

Sample code

`my.getSavedFileInfo` only works on the file address saved by using my.saveFile.

```
var that = this;
    my.chooseImage({
    success: (res) => {
      console.log(res.apFilePaths[0], 1212)
      my.saveFile({
        apFilePath: res.apFilePaths[0],
        success: (result) => {
          console.log(result, 1212)
          my.getSavedFileInfo({
            apFilePath: result.apFilePath,
            success: (resu) => {
              console.log(JSON.stringify(resu))
              that.filePath = resu
            }
          })
        },
      });
    },
});
```

my.getSavedFileList

> Note
> - This interface is supported since basic library version 1.3.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to get all saved files.

Parameters

Parameter	Type	Required	Description
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Return value on success

Parameter	Type	Description
fileList	List	File list

Attributes of file object

Attribute	Type	Description
size	Number	File size
createTime	Number	Creation time
apFilePath	String	File path

Sample code

```
my.getSavedFileList({
    success:(res)=>{
        console.log(JSON.stringfy(res))
    }
});
```

my.removeSavedFile

> Note
> - This interface is supported since basic library version 1.3.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to delete a saved file.

Parameters

Parameter	Type	Required	Description
apFilePath	String	Yes	File path

Parameter	Type	Required	Description
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Sample code

```
my.getSavedFileList({
        success:(res)=>{
            my.removeSavedFile({
          apFilePath:res.fileList[0].apFilePath,
          success:(res)=>{
            console.log('remove success')
          }
        })
        }
    });
```

1.10.7. Location

my.chooseLocation

This API is used to open the built-in map to choose a location.

- When using this API on the Android client, you need to apply for an AutoNavi key and add it to `AndroidManifest`. For details, see Applying for AutoNavi Key.
- When using this API on iOS, you need to set the AutoNavi key in the `beforeDidFinishLaunchingWithOptions` method. The required code is shown below. Please refer to the Obtaining Key document to obtain the AutoNavi key.

```
[APMapKeySetting getInstance].apiKey = @"AutoNavi key"
```

Restrictions

- There is no overseas map data temporarily, and this API may not be called normally in regions other than Mainland China (excluding Hong Kong, Macao and Taiwan).
- Only supports Autonavi Map Style and National Confidentiality Plugin (AKA Mars coordinate system).

Example



Input parameter

Object type, the attributes are as follows:

Attribute	Type	Required	Description
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

success callback function

Attribute	Type	Description
name	String	Location name.
address	String	Detailed location.
latitude	Number	Latitude, floating point number, the range is -90 ~ 90, negative number means south latitude.
longitude	Number	Longitude, floating point number, the range is -180 ~ 180, negative number means west longitude.
provinceName	String	Province name
cityName	String	City name.

Code sample

`.json` code sample:

```
// API-DEMO page/API/choose-location/choose-location.json
{
    "defaultTitle": "Choose location"
}
```

`.axml` code sample:

```
<!-- API-DEMO page/API/choose-location/choose-location.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <text>Longitude:</text>
      <input value="{{longitude}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Latitude:</text>
      <input value="{{latitude}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Location name:</text>
      <input value="{{name}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Detailed location:</text>
      <input value="{{address}}"></input>
    </view>
    <view class="page-section-btns">
      <view onTap="chooseLocation">Choose location</view>
    </view>
  </view>
</view>
```

`.js` code sample:

```
// API-DEMO page/API/choose-location/choose-location.js
Page({
  data: {
    longitude: '120.126293',
    latitude: '30.274653',
    name: 'Huanglong Center',
    address: 'No. 77, Xueyuan Road',
  },
  chooseLocation() {
    var that = this
    my.chooseLocation({
         success:(res)=>{
          console.log(JSON.stringify(res))
          that.setData({
            longitude:res.longitude,
            latitude:res.latitude,
            name:res.name,
            address:res.address
          })
        },
        fail:(error)=>{
          my.alert({content: 'Call failed:'+JSON.stringify(error), });
        },
    });
    },
})
```

`.acss` code sample:

```
/* API-DEMO page/API/choose-location/choose-location.acss */
.page-body-info {
  height: 250rpx;
}
.page-body-text-location {
  display: flex;
  font-size: 50rpx;
}
.page-body-text-location text {
  margin: 10rpx;
}
.page-section-location-text{
    color: #49a9ee;
}
```

my.getLocation(OBJECT)

This API is used to get the user's current geographic location information.

- When using this API on the Android client, you need to apply for an AutoNavi key and add it to `AndroidManifest`. For details, see Applying for AutoNavi Key.
- When using this API on iOS, you need to set the AutoNavi key in the `beforeDidFinishLaunchingWithOptions` method. The required code is shown below. Please refer to the Obtaining Key document to obtain the AutoNavi key.

```
[LBSmPaaSAdaptor sharedInstance].shouldAMapRegeoWhenLBSFailed = YES;
[AMapServices sharedServices].apiKey = @"AutoNave Key"
```

Restrictions

- This interface is supported since basic library version 1.1.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
- There is no overseas map data temporarily, and this API may not be called normally in regions other than Mainland China (excluding Hong Kong, Macao and Taiwan).
- Only supports Autonavi Map Style and National Confidentiality Plugin (AKA Mars coordinate system).

Example



Input parameter

Name	Type	Required	Description
cacheTimeout	Number	No	Expiration time of mPaaS client's latitude and longitude location cache, in seconds. It defaults to 30 seconds. Using cache will speed up the positioning, and the reposition will be performed in case the cache is expired.
type	Number	No	Get the type of latitude and longitude data. The default value is 0. The earliest version is 1.1.1. • 0: By default, get latitude and longitude • 1: Get latitude and longitude and reverse geocoding data in district and county level • 2: Get latitude and longitude and reverse geocoding data in street level, not recommended • 3: Get latitude and longitude and reverse geocoding data in POI level, not recommended
success	Function	No	Callback function for call success
fail	Function	No	Callback function for call failure
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description	Earliest version
longitude	String	Longitude	-
latitude	String	Latitude	-
accuracy	String	Accuracy, in meters	-
horizontalAccuracy	String	Horizontal accuracy, in meters	-
country	String	Country (valid in case type>0)	1.1.1
countryCode	String	Country code (valid in case type>0)	1.1.1
province	String	Province (valid in case type>0)	1.1.1
city	String	City (valid in case type>0)	1.1.1
cityAdcode	String	Region code at city level (valid in case type>0)	1.1.1
district	String	District & county (valid in case type>0)	1.1.1
districtAdcode	String	Region code at district & county level (valid in case type>0)	1.1.1
streetNumber	Object	This field is returned only when street level reverse geocoding data is required. Street number information in the format of {street, number} (valid in case type>1)	1.1.1
pois	array	This field is returned only when POI level reverse geocoding data is required. POI near the locating position in the format of {name, address} (valid in case type>2)	1.1.1

fail callback function

Object type, the attributes are as follows:

Attribute	Type	Description
error	String	Error code
errorMessage	String	Error information

Code sample

`.json` code sample:

```
// API-DEMO page/API/get-location/get-location.json
{
    "defaultTitle": "Get location"
}
```

`.axml` code sample:

```
<!-- API-DEMO page/API/get-location/get-location.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <view>Longitude and latitude of the current location</view>
      <block a:if="{{hasLocation === false}}">
        <text>Not obtained</text>
      </block>
      <block a:if="{{hasLocation === true}}">
        <view class="page-body-text-location">
          <text>E{{location.longitude[0]}}°{{location.longitude[1]}}'</text>
          <text>N{{location.latitude[0]}}°{{location.latitude[1]}}'</text>
        </view>
      </block>
    </view>

    <view class="page-section-btns">
      <view onTap="getLocation">Get location</view>
      <view onTap="clear">Clear</view>
    </view>
  </view>
</view>
```

`.js` code sample:

```
// API-DEMO page/API/get-location/format-location.js
function formatLocation(longitude, latitude) {
  longitude = Number(longitude).toFixed(2),
  latitude = Number(latitude).toFixed(2)

  return {
    longitude: longitude.toString().split('.'),
    latitude: latitude.toString().split('.')
  }
}

export default formatLocation
```

`.js` code sample:

```
// API-DEMO page/API/get-location/get-location.js
import formatLocation from './format-location.js';

Page({
  data: {
    hasLocation: false,
  },
  getLocation() {
    var that = this;
    my.showLoading();
    my.getLocation({
      success(res) {
        my.hideLoading();
        console.log(res)
        that.setData({
          hasLocation: true,
          location: formatLocation(res.longitude, res.latitude)
        })
      },
      fail() {
        my.hideLoading();
        my.alert({ title: 'Positioning failed' });
      },
    })
  },
  clear() {
    this.setData({
      hasLocation: false
    })
  }
})
```

`.acss` code sample:

```
/* API-DEMO page/API/get-location/get-location.acss */
.page-body-info {
  height: 250rpx;
}
.page-body-text-small {
  font-size: 24rpx;
  color: #000;
  margin-bottom: 100rpx;
}
.page-body-text-location {
  display: flex;
  font-size: 50rpx;
}
.page-body-text-location text {
  margin: 10rpx;
}
```

Error code

Error code	Description	Solution
11	Please confirm that the location-related permissions have been turned on.	Prompt user to turn on the location permissions.
12	Network error, please try again later	Prompt the user to check the current network.
13	Positioning failed, please try again later	Prompt the user to try again.
14	Business positioning timeout	Prompt the user to try again.
2001	The user refused to authorize the mini program.	Prompt the user to accept the mini program authorization.

FAQ

- Q: If the mini program is deleted after the `my.getLocation` authorization is allowed for the first time, will it need to re-authorize after reopening?

  A: Re-authorization is required. After deleting the Mini Program, the authorization relationship for obtaining the positioning will be deleted together.

my.openLocation

This API is used to use the built-in map of mPaaS MINI program to view the location.

Restrictions

- There is no overseas map data temporarily, and this API may not be called normally in regions other than Mainland China (excluding Hong Kong, Macao and Taiwan).
- Only supports Autonavi Map Style and National Confidentiality Plugin (AKA Mars coordinate system).

Example



Input parameter

Name	Type	Required	Description
longitude	String	Yes	Longitude
latitude	String	Yes	Latitude
name	String	Yes	Location name
address	String	Yes	Detailed description of the address
scale	Number	No	Scaling, ranging from 3 ~ 19, it defaults to 15
success	Function	No	Callback function for call success
fail	Function	No	Callback function for call failure
complete	Function	No	Callback function for call end (executed regardless of whether the call is successful or failed)

Code sample

```
// API-DEMO page/API/open-location/open-location.json
{
    "defaultTitle": "View location"
}
```

```
<!-- API-DEMO page/API/open-location/open-location.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <text>Longitude</text>
      <input type="text" disabled="{{true}}" value="{{longitude}}" name="longitude"></input>
    </view>
    <view class="page-section-demo">
      <text>Latitude</text>
      <input type="text" disabled="{{true}}"  value="{{latitude}}" name="latitude"></input>
    </view>
    <view class="page-section-demo">
      <text>Location name</text>
      <input type="text" disabled="{{true}}"  value="{{name}}" name="name"></input>
    </view>
    <view class="page-section-demo">
      <text>Detailed location</text>
      <input type="text" disabled="{{true}}"  value="{{address}}" name="address"></input>
    </view>
    <view class="page-section-btns">
      <view type="primary" formType="submit" onTap="openLocation">View location</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/open-location/open-location.js
Page({
  data: {
    longitude: '120.126293',
    latitude: '30.274653',
    name: 'Huanglong Center',
    address: 'No. 77, Xueyuan Road',
  },

  openLocation() {
    my.openLocation({
      longitude: this.data.longitude,
      latitude: this.data.latitude,
      name: this.data.name,
      address: this.data.address,
    })
  }
})
```

1.10.8. Network

my.request

Network request of a Mini Program.

- `my.request` currently supports GET/POST/DELETE.
- `my.request` currently only supports HTTPS protocol requests.

> **Note**

For more information, see my.request FAQ.

Instructions for use:

- You need to turn on the **Permission control switch** of the mini program in the figure below in **Release Mini Program > Open Platform Mini Program Management** in advance, and configure the domain name whitelist in the **Server domain name whitelist**. The mini program can only communicate with the domain names in the whitelist when calling the following APIs: HTTP request (`my.request`), upload file (`my.uploadFile`).



- After adding the server domain name whitelist, you need to repackage and upload to generate the trial version, and the server domain name will take effect.
- When debugging on the IDE, please use the real device to preview and debug.

Input parameter

Property	Type	Mandatory	Description
url	String	Yes	Target server url.
headers	Object	No	Set the request HTTP header. The default value is `{'content-type': 'application/json'}`. The key and value in the object must be String.
method	String	No	The default value is GET. GET/POST/DELETE are supported.
data	Object	No	Data parameter description.
timeout	Number	No	Timeout period, in ms. The default value is 30000.
dataType	String	No	Expected format of the returned data. The following formats are supported: json text base64 The default format is json.
success	Function	No	Callback function for successful call.
fail	Function	No	Callback function for failed call.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Data parameter description

Data transferred to the server is eventually expressed in String. If the type is not String, the data will be converted into String. Conversion rules are:

- If the method is GET, the data will be converted into query string:
  `encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...`.
- If the method is POST and the `headers['content-type']` is `application/json`, the data will be JSON serialized.
- If the method is POST and the `headers['content-type']` is `application/x-www-form-urlencoded`, the data will be converted into query string:
  `encodeURIComponent(k)=encodeURIComponent(v)&encodeURIComponent(k)=encodeURIComponent(v)...`.

referer description

-The `referer` header of the network request cannot be set.-The format is fixed to `https://urlhost/{appid}/{version}/page-frame.html`, where `{appid}` is the APPID of the mini program and `{version}` is the version number of the mini program.

Success Callback Function

Name	Type	Description
data	String	Response data. The format depends on the value of `dataType` in the request.
status	Number	Response code
headers	Object	Response header

Error code

Error code	Description	Solution
1	The request is not over, and it jumps to another page	It is recommended that the page jump after the request is completed

2	Incorrect parameter.	• It may be caused by the link being too long. It is recommended that the parameters be processed in data. • It is recommended to check whether the data passed in the request is normal and whether the format is correct. You can print the parameter data log before requesting.
11	No cross-domain permission	Check whether the requested domain name is added to the domain name whitelist. For the development version test, click on the upper right corner of the IDE > **Details**, and check **Ignore httpRequest domain name validity check**. **Note**: When a new version is online, you must add the **Server domain name whitelist**, otherwise an exception will occur.
12	Network error	It is recommended to check whether the network environment is normal and whether the server is stable.
13	Timeout	It is recommended to check whether the network environment is normal and whether the server responds normally. If the request takes a long time, the timeout period timeout can be set accordingly.
14	Decoding failed	It is recommended to check whether the front-end and back-end request and response data formats are consistent. If the returned data format text is inconsistent with the input parameter dataType value JSON, which causes the interface to report an error, please modify the backend data format to be JSON.
15	Failed to transfer parameters.	If you use urlencode to pass parameters on the mini program page, you need to encode the overall parameters.
19	HTTP error	• Please confirm whether the request URL address can normally request the HTTPS protocol on the external network. The mini program requests in real devices are formal requests in the online environment and cannot use the local network request. • If you encounter HTTP 404, 500, 504 and other errors, it is recommended to open the IDE debugger > **Network** to view the specific error information, and then deal with it according to the corresponding HTTP error code. • The SSL certificate is incorrect. It is recommended to replace the website SSL certificate.
20	The request has been stopped / Traffic limited in server	Please confirm whether the requesting server can request and respond normally.
23	The proxy request failed.	It is recommended to check whether the proxy configuration is correct.

Note: When the value of the input parameter `dataType` is `json`, the applet framework will first perform the `JSON.prase` operation on the returned result. If the parsing fails, a code 14 error will be returned. When the input parameter `dataType` value is `text`, if the returned content format does not match, a code 14 error will also be returned. When encountering this error, please check whether the dataType setting is correct.

If the call to `my.request` returns that **You are not authorized to call this interface**, you need to configure the domain name whitelist in **Open Platform Mini Program Management > Server domain name whitelist**.



Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.request({
  url: 'https://example.org/post',
  method: 'POST',
  data: {
    from: 'Alipay',
    production: 'AlipayJSAPI',
  },
  dataType: 'json',
  success: function(res) {
    my.alert({content: 'success'});
  },
  fail: function(res) {
    my.alert({content: 'fail'});
  },
  complete: function(res) {
    my.hideLoading();
    my.alert({content: 'complete'});
  }
});

// Return RequestTask, you can call abort method to cancel the request
const task = my.request({url: 'https://example.org/post'})
task.abort()
```

Return value

RequestTask

Network request task object.

Method

`RequestTask.abort()`

my.uploadFile

This API is supported in mPaaS 10.1.32 and later versions.

Upload local resources to the developer server.

Instructions for use:

You need to configure the domain name whitelist in **Open Platform Mini Program Management** > **Server domain name whitelist** in advance. The mini program can only communicate with the domain names in the whitelist when calling the following APIs: HTTP request (`my.request`), upload file (`my.uploadFile`).



Input parameter

Name	Type	Required	Description
url	String	Yes	Developer server address

filePath	String	Yes	Local locator of the file resources to be uploaded
fileName	String	Yes	The file name, which is the corresponding key. The developer can obtain the binary content of the file through the key on the server.
fileType	String	Yes	File type, which can be image, video or audio
header	Object	No	HTTP request header
formData	Object	No	Additional form data in the HTTP request
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
data	String	Data returned by the server
statusCode	String	HTTP status code
header	Object	The header returned by the server

Error Code

Error Code	Description
11	File does not exist
12	File upload failed
13	No access

Code sample

```
my.uploadFile({
  url: 'Please use your own server address',
  fileType: 'image',
  fileName: 'file',
  filePath: '...',
  success: (res) => {
    my.alert({
      content: 'Upload successful'
    });
  },
});
```

UploadTask

Monitor the upload progress change and cancel the upload task object.

Method

Method	Description
UploadTask.abort()	Interrupt upload task

UploadTask.onProgressUpdate(function callback)	Monitor upload progress change event

Code sample

```
const task = my.uploadFile({
  url: 'Please use your own server address',
  fileType: 'image',
  fileName: 'file',
  filePath: '...',
});
task.onProgressUpdate(({progress, totalBytesWritten, totalBytesExpectedToWrite}) => {
})
task.abort()
```

FAQ

- Q: Can images uploaded by the Mini Program be automatically converted to Base64 (based on 64 printable characters to represent binary data)?

  A: Currently, the mini program does not support the conversion of pictures to Base64.
- Q: How can `my.uploadFile` obtain the error message returned by the server?

  A:
  - It can be obtained through the data parameter in the success callback.
  - A log acquisition interface can be added on the server side. If the upload fails, a request is made to the log acquisition interface to obtain detailed failure logs.
- Q: What is the default timeout period of `my.uploadFile` ? Can I extend the default time?

  A: The default timeout period of `my.uploadFile` is 30 s, and the default time cannot be set at present.
- Q: I use `my.uploadFile` to upload files, why is there an error `error:12` ?

  A: The upload failure results in the error `error:12` . The possible reasons for the upload failure are:
  - The file is too large.
  - The upload time exceeds 30s.
  - Permission denied.
- Q: Use `my.uploadFile` to upload a picture to the back end. The binary picture is received, and then the binary picture is sent from the back end to the mini program foreground. How does the mini program foreground parse it?

  A: Uploading a picture means that the back end receives the picture through a binary stream, and then the back end only needs to provide the location address of the corresponding picture on the server.
- Q: When calling `my.uploadfile` , why is there an error of `error: 4` , and I don't have permission to call this interface?

  A: The requested URL is not configured with a whitelist. It is recommended to add the domain name of the URL to the whitelist.
- Q: Does the Mini Program support uploading excel files?

  A: Currently the upload file type of `my.uploadFile` supports image, video, audio (image / video / audio), and other types of files are not currently supported.
- Q: Does `my.uploadFile` support uploading multiple pictures at the same time?

  A: `my.uploadFile` does not support uploading multiple pictures at the same time. Only one picture can be uploaded at a time.

my.downloadFile

This API is supported in mPaaS 10.1.32 and later versions.

Download the file resource to local position.

Input parameter

Name	Type	Required	Description
url	String	Yes	Address of file downloaded
header	Object	No	HTTP request header
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description

apFilePath	String	Temporary storage location of files

Error code

Error code	Description	
12	Download failed	
13	No access	
20	The requested URL does not support HTTP	It is recommended to change the requested URL to HTTPS

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
// API-DEMO page/API/download-file/download-file.json
{
    "defaultTitle": "Download file"
}
```

```
<!-- API-DEMO page/API/download-file/download-file.axml-->
<view class="container">
  <button onTap="download">Download image and display</button>
</view>
```

```
// API-DEMO page/API/download-file/download-file.js
Page({
  download() {
    my.downloadFile({
      url: 'https://img.alicdn.com/tfs/TB1x669SXXXXXbdaFXXXXXXXXXX-520-280.jpg',
      success({ apFilePath }) {
        my.previewImage({
          urls: [apFilePath],
        });
      },
      fail(res) {
        my.alert({
          content: res.errorMessage || res.error,
        });
      },
    });
  },
})
```

my.connectSocket

This API is supported in mPaaS 10.1.60 and later versions.

Create a WebSocket connection.

A mPaaS mini program can only keep one WebSocket connection at a time. If a WebSocket connection already exists, it will be automatically closed, and then a new WebSocket connection will be re-created.

Input parameter

Name	Type	Required	Description
url	String	Yes	Target server URL. **Note**: Some newly released applets only support the wss protocol.
data	Object	No	Parameters requested
header	Object	No	Set request header
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call

complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

Error code

Error code	Description	Solution
1	Unknown error	-
2	The network connection already exists	A mini program can only retain one WebSocket connection for a period of time. If there is already a WebSocket connection, it will be automatically closed and a new WebSocket connection will be created.
3	URL parameter is empty	Replace URL link.
4	Unrecognized URL format	Replace URL link.
5	URL must start with `ws` or `wss`	Replace URL link.
6	Connection to server timed out	Try again later.
7	The https certificate returned by the server is invalid	The mini program must use HTTPS/WSS to initiate network requests. When requesting, the system will verify the HTTPS certificate used by the server domain name. If the verification fails, the request cannot be successfully initiated. Due to system limitations, different platforms have different stringent requirements for certificates. In order to ensure the compatibility of applets, it is recommended that developers configure certificates in accordance with the highest standards and use relevant tools to check existing certificates to ensure that they meet the requirements.
8	Invalid protocol header returned by the server	For newly created mini programs starting in May 2019, the HTTPS and WSS protocols are mandatory by default, and the HTTP and WS protocols are no longer supported.
9	The `Sec-WebSocket-Protocol` request header not specified in WebSocket request	Please specify the `Sec-WebSocket-Protocol` request header.
10	No network connection, and the message cannot be sent	Please call my.sendSocketMessage to send data messages after connecting to the server normally. You can use my.onSocketOpen to monitor events. Determine the correct connection with the server. **Note**: To send data through a WebSocket connection, you need to use my.connectSocket to initiate a connection, and then call my.onSocketOpen](#my.onSocketOpen) after the callback .sendSocketMessage Send data.
11	Message failed to send	Try again later.
12	Cannot apply for more memory to read network data	Please check the memory.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.connectSocket({
  url: 'test.php',
  data: {},
  header:{
    'content-type': 'application/json'
  },
});
```

my.onSocketOpen

This API is supported in mPaaS 10.1.60 and later versions.

Listen to the WebSocket connection establishment event.

Input parameter

Object type, the properties are as follows:

Attribute	Type	Required	Description
callback	Function	Yes	The callback function of the WebSocket connection open event.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.connectSocket({
  url: 'test.php',
});

my.onSocketOpen(function(res) {
  console.log('WebSocket connection is established! ');
});
```

my.offSocketOpen

This API is supported in mPaaS 10.1.60 and later versions.

Cancel listening of WebSocket connection establishment event.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketOpen(this.callback);
  },
  onUnload() {
    my.offSocketOpen(this.callback);
  },
  callback(res) {
  },
})
```

Whether to pass the callback value

- If the callback value is not passed, all event callbacks will be removed. The code sample is as follows:

```
my.offSocketOpen();
```

- If the callback value is passed, only remove the corresponding callback event. The code sample is as follows:

```
my.offSocketOpen(this.callback);
```

my.onSocketError

This API is supported in mPaaS 10.1.60 and later versions.

Listen to WebSocket errors.

Input parameter

Name	Type	Required	Description
callback	Function	Yes	The callback function of WebSocket error event.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.connectSocket({
  url: 'Developer server address'
});

my.onSocketOpen(function(res){
  console.log('WebSocket connection is established! ');
});

my.onSocketError(function(res){
  console.log('WebSocket connection failed to establish, please check! ');
});
```

my.offSocketError

This API is supported in mPaaS 10.1.60 and later versions.

Cancel WebSocket listening error.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onSocketError(this.callback);
  },
  onUnload() {
    my.offSocketError(this.callback);
  },
  callback(res) {
  },
})
```

Whether to pass the callback value

- If the callback value is not passed, all event callbacks will be removed. The code sample is as follows:

  ```
  my.offSocketError();
  ```

- If the callback value is passed, only remove the corresponding callback event. The code sample is as follows:

  ```
  my.offSocketError(this.callback);
  ```

my.sendSocketMessage

This API is supported in mPaaS 10.1.60 and later versions.

To send data over a WebSocket connection, you need to first use my.connectSocket to initiate the connection and send the data after the my.onSocketOpen callback.

Input parameter

Name	Type	Required	Description
data	String	Yes	Content that need to be sent: String type plain text content or string type base64 encoded content.
isBuffer	Boolean	No	If you need to send binary data, the data of input parameter should be base64 encoded to string and then assigned to `data`, meanwhile this field should be set to true. If the data is normal text String, you do not need to set this field.
success	Function	No	Callback function
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.sendSocketMessage({
    data: this.data.toSendMessage, // Content to be sent
    success: (res) => {
        my.alert({content: 'Data sent!' + this.data.toSendMessage});
    },
});
```

my.onSocketMessage

This API is supported in mPaaS 10.1.60 and later versions.

Listen to the event that WebSocket receives messages from the server.

Callback return value

Name	Type	Description
data	String/ArrayBuffer	Message returned by the server: String type plain text content or string type base64 encoded content
isBuffer	Boolean	If this field is `true`, the `data` field indicates that the base64-encoded string was received, otherwise the `data` field indicates that the normal text string was received.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.connectSocket({
  url: 'Server address'
})

my.onSocketMessage(function(res) {
  console.log('Content received from the server:' + res.data)
})
```

my.offSocketMessage

This API is supported in mPaaS 10.1.60 and later versions.

Cancel the listening of event that WebSocket receives messages from the server.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.connectSocket({
  url: 'Server address'
})
my.onSocketMessage(function(res) {
  console.log('Received server content:' + res.data)
})
my.offSocketMessage();
```

Whether to pass the callback value

- If the callback value is not passed, all event callbacks will be removed. The code sample is as follows:

```
my.offSocketMessage();
```

- If the callback value is passed, only remove the corresponding callback event. The code sample is as follows:

```
my.offSocketMessage(this.callback);
```

my.closeSocket

This API is supported in mPaaS 10.1.60 and later versions.

Close the WebSocket connection.

Input parameter

Name	Type	Required	Description
success	Function	No	Callback function
fail	Function	No	Callback function for failed call

complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
my.onSocketOpen(function() {
  my.closeSocket()
})

my.onSocketClose(function(res) {
  console.log('WebSocket is closed! ')
})
```

my.onSocketClose

This API is supported in mPaaS 10.1.60 and later versions.

Listen to WebSocket close.

Input parameter

Name	Type	Required	Description
callback	Function	Yes	The callback function of WebSocket connection closed event.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
onLoad() {
    // Note: The registration of callback method only needs to be done once during the startup phase of the mini program, and there will be several c
allbacks when the callback method is invoked several times.
    my.onSocketClose((res) => {
      my.alert({content: 'Connection is closed! '});
      this.setData({
        sendMessageAbility: false,
        closeLinkAbility: false,
      });
    });
    // Note: The registration of callback method only needs to be done once during the startup phase of the mini program, and there will be several c
allbacks when the callback method is invoked several times.
    my.onSocketOpen((res) => {
      my.alert({content: 'Connection is established! '});
      this.setData({
        sendMessageAbility: true,
        closeLinkAbility: true,
      });
    });

    my.onSocketError(function(res){
      my.alert('WebSocket connection failed to establish, please check!' + res);
    });

    // Note: The registration of callback method only needs to be done once during the startup phase of the mini program, and there will be several c
allbacks when the callback method is invoked several times.
    my.onSocketMessage((res) => {
      my.alert({content: 'Received data!' + JSON.stringify(res)});
    });
  }

connect_start() {
    my.connectSocket({
      url: 'Server address', // The developer's server address must use wss protocol, and the domain name must be a legal domain name configured in t
he background
      success: (res) => {
        my.showToast({
          content: 'success', // Text content
        });
      },
      fail:()=>{
        my.showToast({
          content: 'fail', // Text content
        });
      }
    });
  },
```

my.offSocketClose

This API is supported in mPaaS 10.1.60 and later versions.

Cancel listening of WebSocket shutdown.

Code sample

The code is for reference only, it is recommended to use your own address for testing.

```
Page({
  onLoad() {
  my.onSocketClose(this.callback);
  },
  onUnload() {
    my.offSocketClose(this.callback);
    //    my.offSocketClose();
  },
  callback(res) {
  my.alert({content: 'Connection closed! '});
      this.setData({
        sendMessageAbility: false,
        closeLinkAbility: false,
      });
  },
})
```

Whether to pass the callback value

- If the callback value is not passed, all event callbacks will be removed. The code sample is as follows:

```
my.offSocketClose();
```

- If the callback value is passed, only remove the corresponding callback event. The code sample is as follows:

```
my.offSocketClose(this.callback);
```

1.10.9. Device

1.10.9.1. Can I use

This API is supported in mPaaS 10.1.32 and later versions.

my.canIUse(String)

This API is supported in mPaaS 10.1.32 and later versions.

This interface is used to determine whether the current version supports the API, input parameters or return values, components, attributes, etc. of the current MINI.

Input parameter

The parameters are called by the `${API}.${type}.${param}.${option}` or `${component}.${attribute}.${option}` method.

- API: Indicates the API name, exclude the name of `my.`. For example: if you want to judge `my.getFileInfo`, you only need to pass in `getFileInfo`.
- type: Possible values are object/return/callback, which indicates the judgment type of the API.
- param: Indicates an attribute name of the parameter.
- option: Indicates the specific attribute value of the parameter attribute.
- component: Indicates the component name.
- attribute: Indicates the component attribute name.
- option: Indicates the component attribute value.

Code sample

```
// Whether the new API is available
my.canIUse('getFileInfo')
// Whether the new API attribute is available
my.canIUse('closeSocket.object.code')
// Whether the new API attribute is available
my.canIUse('getLocation.object.type')
// Whether the new attribute of the API return value is available
my.canIUse('getSystemInfo.return.brand')
// Whether the new component "Follow lifestyle account" is available
my.canIUse('lifestyle')
// Whether the new attribute value of the component is available
my.canIUse('button.open-type.share')
```

Return value

Boolean type, indicating whether it is supported.

1.10.9.2. Obtain base library version

my.SDKVersion

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

You can call this interface to get the version number of the base library. The base library version is for reference only, it is not suggested to make your code logic rely on this value.

Sample code

```
<!-- API-DEMO page/API/sdk-version/sdk-version.axml-->
<view class="page">
  <view class="page-description">API for getting base library version number</view>
  <view class="page-section">
    <view class="page-section-title">my.SDKVersion</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getSDKVersion">Get base library version number</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/sdk-version/sdk-version.js
Page({
  getSDKVersion() {
    my.alert({
      content: my.SDKVersion,
    });
  },
});
```

Return value

The return value is a string which is the version number of the base library.

1.10.9.3. System info

my.getSystemInfo

This API is supported in mPaaS 10.1.32 and later versions.

This interface is used for system information.

Input parameter

Name	Type	Required	Description
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description	Minimum Version
model	String	Phone model	-
pixelRatio	Number	Device pixel ratio	-
windowWidth	Number	Window width	-
windowHeight	Number	Window height	-
language	String	Language set by the application	-
version	String	Application version number	-
storage	String	Device disk capacity	1.1.1
currentBattery	String	Percentage of current battery	1.1.1
system	String	System version	1.1.1
platform	String	System name: Android, iOS / iPhone OS	1.1.1
titleBarHeight	Number	Title bar height **Description**: This return value is only supported by version 10.1.60.	1.1.1
statusBarHeight	Number	Status bar height **Description**: This return value is only supported by version 10.1.60.	1.1.1

Name	Type	Description	Minimum Version
screenWidth	Number	Screen width	1.1.1
screenHeight	Number	Screen height	1.1.1
brand	String	Mobile phone brand	1.4.0
fontSizeSetting	Number	Users set font size **Description**: This return value is only supported by version 10.1.60.	1.4.0
app	String	The currently running client.	-

Model parameter

For iPhone, the iPhone internal code (Internal Name) is returned in the model parameter. The iPhone model and the corresponding model return values are shown in the following table:

iPhone model	model return value
iPhone	iPhone11
iPhone 3G	iPhone12
iPhone 3GS	iPhone21
iPhone 4	iPhone31 / iPhone32 / iPhone33
iPhone 4S	iPhone41
iPhone 5	iPhone51 / iPhone52
iPhone 5S	iPhone61 / iPhone62
iPhone 6	iPhone72
iPhone 6 Plus	iPhone71
iPhone 6S	iPhone8,1
iPhone 6S Plus	iPhone8,2
iPhone 7	iPhone9,1 / iPhone9,3
iPhone 7 Plus	iPhone9,2 / iPhone9,4
iPhone 8	iPhone10,1 / iPhone10,4
iPhone 8 Plus	iPhone10,2 / iPhone10,5
iPhone X	iPhone10,3 / iPhone10,6
iPhone XR	iPhone11,8
iPhone XS	iPhone11,2
iPhone 11	iPhone12,1
iPhone 11 Pro	iPhone12,3
iPhone XS Max	iPhone11,6 / iPhone11,4
iPhone 11 Pro Max	iPhone12,5

Code sample

```
// API-DEMO page/API/get-system-info/get-system-info.json
{
    "defaultTitle": "Get phone system info"
}
```

```
<!-- API-DEMO page/API/get-system-info/get-system-info.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <text>Phone model</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.model}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Language</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.language}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Version</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.version}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Window width</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.windowWidth}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Window height</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.windowHeight}}"></input>
    </view>
    <view class="page-section-demo">
      <text>DPI</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.pixelRatio}}"></input>
    </view>
    <view class="page-section-btns">
      <view onTap="getSystemInfo">Get phone system info</view>
      <view onTap="getSystemInfoSync">Get phone system info synchronously</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/get-system-info/get-system-info.js
Page({
  data: {
    systemInfo: {}
  },
  getSystemInfo() {
    my.getSystemInfo({
      success: (res) => {
        this.setData({
          systemInfo: res
        })
      }
    })
  },
  getSystemInfoSync() {
    this.setData({
      systemInfo: my.getSystemInfoSync(),
    });
  },
})
```

my.getSystemInfoSync

This API is supported in mPaaS 10.1.32 and later versions.

A synchronization interface for obtaining mobile phone system information. The return value is the same as the success callback parameter of `getSystemInfo`.

This interface is a synchronous interface and has a timeout judgment. When it times out, the interface returns undefined.

Code sample

```
// API-DEMO page/API/get-system-info/get-system-info.json
{
    "defaultTitle": "Get phone system info"
}
```

```
<!-- API-DEMO page/API/get-system-info/get-system-info.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-demo">
      <text>Phone model</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.model}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Language</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.language}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Version</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.version}}"></input>
    </view>
    <view class="page-section-demo">
      <text>window width</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.windowWidth}}"></input>
    </view>
    <view class="page-section-demo">
      <text>Window height</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.windowHeight}}"></input>
    </view>
    <view class="page-section-demo">
      <text>DPI</text>
      <input type="text" disabled="{{true}}" value="{{systemInfo.pixelRatio}}"></input>
    </view>
    <view class="page-section-btns">
      <view onTap="getSystemInfo">Get phone system info</view>
      <view onTap="getSystemInfoSync">Get phone system info synchronously</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/get-system-info/get-system-info.js
Page({
  data: {
    systemInfo: {}
  },
  getSystemInfo() {
    my.getSystemInfo({
      success: (res) => {
        this.setData({
          systemInfo: res
        })
      }
    })
  },
  getSystemInfoSync() {
    this.setData({
      systemInfo: my.getSystemInfoSync(),
    });
  },
})
```

1.10.9.4. Network status

This API is supported in mPaaS 10.1.32 and later versions.

my.getNetworkType

This API is supported in mPaaS 10.1.32 and later versions.

This API is used to get the current network status.

Input parameter

Name	Type	Required	Description
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
networkAvailable	Boolean	Whether the network is available

Name	Type	Description
networkType	String	The possible network type values are UNKNOWN/NOTREACHABLE/WIFI/3G/2G/4G/WWAN

Code sample

```
Page({
  data: {
    hasNetworkType: false
  },
  getNetworkType() {
    my.getNetworkType({
      success: (res) => {
        this.setData({
          hasNetworkType: true,
          networkType: res.networkType
        })
      }
    })
  },
  clear() {
    this.setData({
      hasNetworkType: false,
      networkType: ''
    })
  },
});
```

my.onNetworkStatusChange(CALLBACK)

This API is supported in mPaaS 10.1.32 and later versions.

Start monitoring changes in network status.

Return value

Name	Type	Description
isConnected	Boolean	Whether the network is available
networkType	String	The possible network type values are UNKNOWN/NOTREACHABLE/WIFI/3G/2G/4G/WWAN

Code sample

```
my.onNetworkStatusChange(function(res){
    console.log(JSON.stringify(res))
})
```

my.offNetworkStatusChange

This API is supported in mPaaS 10.1.32 and later versions.

Cancel monitoring changes in network status.

Code sample

```
my.offNetworkStatusChange()
```

Whether to pass the callback value

- If the callback value is not passed, all event callbacks will be removed. The code sample is as follows:

```
my.offNetworkStatusChange();
```

- If the callback value is passed, only remove the corresponding callback event. The code sample is as follows:

```
my.offNetworkStatusChange(this.callback);
```

1.10.9.5. Clipboard

This interface is used to get the clipboard data.

my.getClipboard

This interface is used to get the clipboard data.

Input parameter

Name	Type	Required	Description
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call

Name	Type	Required	Description
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
text	String	Clipboard data

Sample code

```
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handlePaste() {
    my.getClipboard({
      success: ({ text }) => {
        this.setData({ copy: text });
      },
    });
  },
});
```

my.setClipboard

This interface is used to set the clipboard data.

Input parameter

Name	Type	Required	Description
text	String	Yes	Clipboard data
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

Sample code

```
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handleCopy() {
    my.setClipboard({
      text: this.data.text,
    });
  },
});
```

1.10.9.6. Shake

This interface is used for the shake function. Each time the API is called, the callback will be triggered after shaking the phone. If you need to listen to it again, call this API again.

my.watchShake(OBJECT)

This interface is used for the shake function. Each time the API is called, the callback will be triggered after shaking the phone. If you need to listen to it again, call this API again.

Sample code

```
Page({
  watchShake() {
    my.watchShake({
      success: function() {
        console.log('Moved up')
        my.alert({ title:'Moved up o.o'});
      }
    });
  },
});
```

1.10.9.7. Vibrate

This API is supported in mPaaS 10.1.60 and later versions.

my.vibrate(OBJECT)

This API is supported in mPaaS 10.1.60 and later versions.

This API is used to call the vibration function.

Code sample

```
// API-DEMO page/API/vibrate/vibrate.json
{
    "defaultTitle": "Vibrate"
}
```

```
<!-- API-DEMO page/API/vibrate/vibrate.axml-->
<view class="page">

  <button type="primary" onTap="vibrate">
    Start vibrating
  </button>

  <button type="primary" onTap="vibrateLong">
    Long vibration (400ms)
   </button>

  <button type="primary" onTap="vibrateShort">
    Short vibration (40ms)
  </button>

</view>
```

```
// API-DEMO page/API/vibrate/vibrate.js
Page({
  vibrate() {
    my.vibrate({
      success: () => {
        my.alert({ title: 'Vibrating'});
      }
    });
  },
  vibrateLong() {
    if (my.canIUse('vibrateLong')) {
      my.vibrateLong((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateLong() requires 10.1.35 and later versions'
      });
    }
  },
  vibrateShort() {
    if (my.canIUse('vibrateShort')) {
      my.vibrateShort((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateShort() requires 10.1.35 and later versions'
      });
    }
  }
});
```

my.vibrateLong(OBJECT)

This API is supported in mPaaS 10.1.60 and later versions.

Long vibration (400 ms).

Code sample

```
// API-DEMO page/API/vibrate/vibrate.json
{
    "defaultTitle": "Vibrate"
}
```

```
<!-- API-DEMO page/API/vibrate/vibrate.axml-->
<view class="page">

  <button type="primary" onTap="vibrate">
    Start vibrating
  </button>

  <button type="primary" onTap="vibrateLong">
    Long vibration (400ms)
   </button>

  <button type="primary" onTap="vibrateShort">
    Short vibration (40ms)
  </button>

</view>
```

```
// API-DEMO page/API/vibrate/vibrate.js
Page({
  vibrate() {
    my.vibrate({
      success: () => {
        my.alert({ title: 'Vibrating'});
      }
    });
  },
  vibrateLong() {
    if (my.canIUse('vibrateLong')) {
      my.vibrateLong((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateLong() requires 10.1.35 and later versions'
      });
    }
  },
  vibrateShort() {
    if (my.canIUse('vibrateShort')) {
      my.vibrateShort((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateShort() requires 10.1.35 and later versions'
      });
    }
  }
});
```

my.vibrateShort(OBJECT)

This API is supported in mPaaS 10.1.60 and later versions.

Short vibration (40 ms).

Code sample

```
// API-DEMO page/API/vibrate/vibrate.json
{
    "defaultTitle": "Vibrate"
}
```

```
<!-- API-DEMO page/API/vibrate/vibrate.axml-->
<view class="page">

  <button type="primary" onTap="vibrate">
    Start vibrating
  </button>

  <button type="primary" onTap="vibrateLong">
    Long vibration (400ms)
   </button>

  <button type="primary" onTap="vibrateShort">
    Short vibration (40ms)
  </button>

</view>
```

```
// API-DEMO page/API/vibrate/vibrate.js
Page({
  vibrate() {
    my.vibrate({
      success: () => {
        my.alert({ title: 'Vibrating'});
      }
    });
  },
  vibrateLong() {
    if (my.canIUse('vibrateLong')) {
      my.vibrateLong((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateLong() requires 10.1.35 and later versions'
      });
    }
  },
  vibrateShort() {
    if (my.canIUse('vibrateShort')) {
      my.vibrateShort((res) => { });
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.vibrateShort() requires 10.1.35 and later versions'
      });
    }
  }
});
```

1.10.9.8. Accelerometer

my.onAccelerometerChange(function callback)

> **Note**
> - This interface is supported since basic library version 1.9.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.60 and later versions.

The interface is used to listen to the acceleration data. The callback interval is 500ms. After the interface is called, the listening is automatically started. You can use `my.offAccelerometerChange` to stop listening.

Parameters

Parameter	Type	Description
function	callback	Callback function for the acceleration data change events.

Callback return value

Parameter	Type	Description
x	Number	X axis
y	Number	Y axis
z	Number	Z axis

Sample code

```
my.onAccelerometerChange(function(res) {
  console.log(res.x)
  console.log(res.y)
  console.log(res.z)
})
```

my.offAccelerometerChange()

> **Note**
> - This interface is supported since basic library version 1.9.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.60 and later versions.

This interface is used to stop listening to acceleration data.

Sample code

```
my.offAccelerometerChange()
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

```
my.offAccelerometerChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

```
my.offAccelerometerChange(this.callback);
```

1.10.9.9. Gyroscope

my.onGyroscopeChange(function callback)

> **Note** This interface is only supported in base library 1.9.0 or later versions and mPaaS V10.1.60 or later versions. For earlier versions, you need to perform compatibility processing. See Instructions on Mini Program base library.

This interface is used to listen to the gyroscope data change events. The interface starts listening to the events automatically once called. The callback interval is 500 ms. You can use `my.offGyroscopeChange()` to stop listening.

Parameters

Name	Type	Description
function	callback	Callback function for the events of gyroscope data changes.

Callback reutrn parameters

Name	Type	Description
x	Number	Angular velocity in X-axis direction
y	Number	Angular velocity in Y-axis direction
z	Number	Angular velocity in Z-axis direction

Sample code

```
my.onGyroscopeChange((res)=>{
   console.log('gyroData.rotationRate.x = ' + res.x);
   console.log('gyroData.rotationRate.y = ' + res.y);
   console.log('gyroData.rotationRate.z = ' + res.z);
});
```

my.offGyroscopeChange()

> **Note** This interface is only supported in base library 1.9.0 or later versions and mPaaS V10.1.60 or later versions. For earlier versions, you need to perform compatibility processing. See Instructions on Mini Program base library.

Rhis interface is used to stop listening to gyroscope data.

Sample code

```
my.offGyroscopeChange();
```

Whether to pass callback value

- If no callback value is passed, listeners on all event callback will be removed. See the following code example:

```
my.offGyroscopeChange();
```

- If the callback value is passed, only the corresponding callback events will be removed. See the following code example:

```
my.offGyroscopeChange(this.callback);
```

1.10.9.10. Compass

my.onCompassChange(function callback)

> **Note**
> - This interface is supported since basic library version 1.9.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.60 and later versions.

Use this API to listen to the compass data. After the interface is called, the listening is automatically started. The callback interval is 500ms. You can use `my.offCompassChange` to stop listening.

Parameters

Parameter	Type	Description
function	callback	Callback function for the compass data change events.

Callback return value

Parameter	Type	Description
direction	Number	The degree between the direction that you are facing and geographical north: [0,360)

Sample code

```
my.onCompassChange(function (res) {
  console.log(res.direction)
})
```

my.offCompassChange()

> **Note**
> - This interface is supported since basic library version 1.9.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.60 and later versions.

Use this API to unlisten to the compass data.

Sample code

```
my.offCompassChange()
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

```
my.offCompassChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

```
my.offCompassChange(this.callback);
```

1.10.9.11. Make phone call

Make a phone call.

my.makePhoneCall(OBJECT)

> **Note** This interface is only supported in mPaaS 10.1.32 and later versions.

Make a phone call.

Parameters

Parameter	Type	Required	Description
number	String	Yes	Phone muber

Sample code

```
// API-DEMO page/API/make-phone-call/make-phone-call.json
{
    "defaultTitle": "Make phone call"
}
```

```
<!-- API-DEMO page/API/make-phone-call/make-phone-call.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">my.makePhoneCall</view>
    <view class="page-section-btns">
      <view onTap="makePhoneCall">Make phone call</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/make-phone-call/make-phone-call.js
Page({
  makePhoneCall() {
    my.makePhoneCall({ number: '95888' });
  },
});
```

1.10.9.12. User captures screen

This interface is used to listen to the screen capture events initiated by users, and can receive screen capture event notifications from the system and third-party screen capture tools.

my.onUserCaptureScreen(CALLBACK)

This interface is used to listen to the screen capture events initiated by users, and can receive screen capture event notifications from the system and third-party screen capture tools.

Sample code

```
my.onUserCaptureScreen(function() {
    my.alert({
        content: 'Receive user screen capture event'
    });
});
```

my.offUserCaptureScreen()

This interface is used to cancel the listening of screen capture events. It usually needs to be used with `my.onUserCaptureScreen`.

Sample code

```
my.offUserCaptureScreen();
```

1.10.9.13. Screen brightness

This interface is used to set whether to keep the screen on. The interface only works in the current Mini Program.

my.setKeepScreenOn(OBJECT)

> **Note**
> - This interface is supported since basic library version 1.3.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to set whether to keep the screen on. The interface only works in the current Mini Program.

Parameters

Parameter	Type	Required	Description
keepScreenOn	Boolean	Yes	Whether to keep screen on.
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Sample code

```
<!-- API-DEMO page/API/screen/screen.axml-->
<view class="page">
  <view class="page-description">Screen brightness API</view>
  <view class="page-section">
    <view class="page-section-title">Set whether to keep screen on</view>
    <view class="page-section-demo">
      <switch checked="{{status}}" onChange="switchKeepScreenOn"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Set screen brightness</view>
    <view class="page-section-demo">
      <slider value="{{brightness}}" max="1" min="0" onChange="sliderChange" step="0.02"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Get screen brightness</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getBrightness">Get screen brightness</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/screen/screen.js
Page({
  data: {
    status: false,
    brightness: 1,
  },
  onLoad() {
    my.getScreenBrightness({
      success: res => {
        this.setData({
          brightness: res.brightness
        })
      },
    })
  },
  sliderChange(e) {
    my.setScreenBrightness({
      brightness: e.detail.value,
      success: (res) => {
        this.setData({
          brightness: e.detail.value,
        })
      }
    })
  },
  switchKeepScreenOn(e) {
    my.setKeepScreenOn({
      keepScreenOn: e.detail.value,
      success: (res) => {
        this.setData({
          status: e.detail.value,
        })
      }
    })
  },
  getBrightness() {
    my.getScreenBrightness({
      success: res => {
        my.alert({
          content: `Current screen brightness: ${res.brightness}`
        });
      }
    })
  }
});
```

my.getScreenBrightness(OBJECT)

> **Note**
> - This interface is supported since basic library version 1.4.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
> - This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to get the screen brightness.

Parameters

Parameter	Type	Required	Description
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Sample code

```
<!-- API-DEMO page/API/screen/screen.axml-->
<view class="page">
  <view class="page-description">Screen brightness API</view>
  <view class="page-section">
    <view class="page-section-title">Set whether to keep screen on</view>
    <view class="page-section-demo">
      <switch checked="{{status}}" onChange="switchKeepScreenOn"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Set screen brightness</view>
    <view class="page-section-demo">
      <slider value="{{brightness}}" max="1" min="0" onChange="sliderChange" step="0.02"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Get screen brightness</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getBrightness">Get screen brightness</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/screen/screen.js
Page({
  data: {
    status: false,
    brightness: 1,
  },
  onLoad() {
    my.getScreenBrightness({
      success: res => {
        this.setData({
          brightness: res.brightness
        })
      },
    })
  },
  sliderChange(e) {
    my.setScreenBrightness({
      brightness: e.detail.value,
      success: (res) => {
        this.setData({
          brightness: e.detail.value,
        })
      }
    })
  },
  switchKeepScreenOn(e) {
    my.setKeepScreenOn({
      keepScreenOn: e.detail.value,
      success: (res) => {
        this.setData({
          status: e.detail.value,
        })
      }
    })
  },
  getBrightness() {
    my.getScreenBrightness({
      success: res => {
        my.alert({
          content: `Current screen brightness: ${res.brightness}`
        });
      }
    })
  }
});
```

my.setScreenBrightness(OBJECT)

Note

- This interface is supported since basic library version 1.4.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
- This interface is only supported in mPaaS 10.1.32 and later versions.

This interface is used to set the screen brightness.

Parameters

Parameter	Type	Required	Description
brightness	Number	Yes	The screen brightness to be set, ranging from 0-1
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.
complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

Sample code

```
<!-- API-DEMO page/API/screen/screen.axml-->
<view class="page">
  <view class="page-description">Screen brightness API</view>
  <view class="page-section">
    <view class="page-section-title">Set whether to keep screen on</view>
    <view class="page-section-demo">
      <switch checked="{{status}}" onChange="switchKeepScreenOn"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Set screen brightness</view>
    <view class="page-section-demo">
      <slider value="{{brightness}}" max="1" min="0" onChange="sliderChange" step="0.02"/>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">Get screen brightness</view>
    <view class="page-section-demo">
      <button type="primary" onTap="getBrightness">Get screen brightness</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/screen/screen.js
Page({
  data: {
    status: false,
    brightness: 1,
  },
  onLoad() {
    my.getScreenBrightness({
      success: res => {
        this.setData({
          brightness: res.brightness
        })
      },
    })
  },
  sliderChange(e) {
    my.setScreenBrightness({
      brightness: e.detail.value,
      success: (res) => {
        this.setData({
          brightness: e.detail.value,
        })
      }
    })
  },
  switchKeepScreenOn(e) {
    my.setKeepScreenOn({
      keepScreenOn: e.detail.value,
      success: (res) => {
        this.setData({
          status: e.detail.value,
        })
      }
    })
  },
  getBrightness() {
    my.getScreenBrightness({
      success: res => {
        my.alert({
          content: `Current screen brightness: ${res.brightness}`
        });
      }
    })
  }
});
```

1.10.9.14. Add phone contacts

This interface enables users to write the form into mobile contacts via creating contacts or adding to existing contacts.

my.addPhoneContact()

Note

- This interface is supported since basic library version 1.10.0. Compatibility processing is required for earlier versions. See Mini Program base library to learn more.
- This interface is only supported in mPaaS 10.1.60 and later versions.

This interface enables users to write the form into mobile contacts via creating contacts or adding to existing contacts.

Parameters

Parameter	Type	Required	Description
photoFilePath	String	No	Local file path of avatar.
nickName	String	No	Nickname
lastName	String	No	Surname
middleName	String	No	Middle name
firstName	String	No	First name
remark	String	No	Remarks

mobilePhoneNumber	String	No	Mobile phone number
alipayAccount	String	No	Alipay account
addressCountry	String	No	Country in contact address
addressState	String	No	Province in contact address
addressCity	String	No	City in contact address
addressStreet	String	No	Street in contact address
addressPostalCode	String	No	Postcode in contact address
organization	String	No	Company
title	String	No	Title
workFaxNumber	String	No	Work fax number
workPhoneNumber	String	No	Work phone number
hostNumber	String	No	Company phone number
email	String	No	Email
url	String	No	Website
workAddressCountry	String	No	Country in work address
workAddressState	String	No	Province in work address
workAddressCity	String	No	City in work address
workAddressStreet	String	No	Street in work address
workAddressPostalCode	String	No	Postcode in work address
homeFaxNumber	String	No	Residential fax number
homePhoneNumber	String	No	Residential phone number
homeAddressCountry	String	No	Country in residential address
homeAddressState	String	No	Province in residential address
homeAddressCity	String	No	City in residential address
homeAddressStreet	String	No	Street in residential address
homeAddressPostalCode	String	No	Postcode in residential address
success	Function	No	Callback function upon call success.
fail	Function	No	Callback function upon call failure.

complete	Function	No	Callback function upon call completion (to be executed upon either call success or failure).

The support for above fields of App contacts varies by mobile phone models. Some mobile phones may not support emoji or kaomoji. In such cases, the corresponding option is ignored.

Return value

Success: `addPhoneContact response:{"success":true}`

Error code

Error code	Error message	Description	Solution
3	fail ${detail}	Call failed. `detail` includes the detailed information.	-
11	fail cancel	The user canceled the operation.	This is normal user interaction, and you don't have to take any action.

Sample code

```
// API-DEMO page/API/contact/contact.json
{
   "defaultTitle": "Contact"
}
```

```
<!-- API-DEMO page/API/contact/contact.axml-->
<view class="page">

  <view class="page-description">Contact API</view>

  <view class="page-section">
    <view class="page-section-title">my.choosePhoneContact</view>
    <view class="page-section-demo">
      <button type="primary" onTap="choosePhoneContact">Arouse local directory</button>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">my.chooseAlipayContact</view>
    <view class="page-section-demo">
      <button type="primary" onTap="chooseAlipayContact">Arouse Alipay directory</button>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">my.chooseContact</view>
    <view class="page-section-demo">
      <button type="primary" onTap="chooseContact">Select contacts</button>
    </view>
  </view>

  <view class="page-section">
    <view class="page-section-title">my.addPhoneContact</view>
    <view class="page-section-demo">

      <view style="font-size:18px;margin-top:18px;margin-bottom:18px">
        <text style="font-size:18px;margin-top:18px;margin-bottom:18px">Basic information</text>
      </view>

      <view class="form-row">
        <view class="form-row-label">Nickname</view>
        <view class="form-row-content">
          <input id="nickName" onInput="onInput" class="input" value="Baking July" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Last name</view>
        <view class="form-row-content">
          <input id="lastName" onInput="onInput" class="input" value="Last" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Middle name</view>
        <view class="form-row-content">
          <input id="middleName" onInput="onInput" class="input" value="Middle" />
        </view>
      </view>
```

```
<view class="form-row">
  <view class="form-row-label">Name</view>
  <view class="form-row-content">
    <input id="firstName" onInput="onInput" class="input" value="First" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Remarks</view>
  <view class="form-row-content">
    <input id="remark" onInput="onInput" class="input" value="This is remarks" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Mobile number</view>
  <view class="form-row-content">
    <input id="mobilePhoneNumber" onInput="onInput" class="input" value="13800000000" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Alipay account</view>
  <view class="form-row-content">
    <input id="alipayAccount" onInput="onInput" class="input" value="alipay@alipay.com" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">WeChat account</view>
  <view class="form-row-content">
    <input id="weChatNumber" onInput="onInput" class="input" value="liuhuo" />
  </view>
</view>

<view style="font-size:18px;margin-top:18px;margin-bottom:18px">
  <text style="font-size:18px;margin-top:18px;margin-bottom:18px">Address</text>
</view>

<view class="form-row">
  <view class="form-row-label">Country</view>
  <view class="form-row-content">
    <input id="addressCountry" onInput="onInput" class="input" value="US" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Province</view>
  <view class="form-row-content">
    <input id="addressState" onInput="onInput" class="input" value="California" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">City</view>
  <view class="form-row-content">
    <input id="addressCity" onInput="onInput" class="input" value="San Francisco" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Street</view>
  <view class="form-row-content">
    <input id="addressStreet" onInput="onInput" class="input" value="Mountain View" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Postcode</view>
  <view class="form-row-content">
    <input id="addressPostalCode" onInput="onInput" class="input" value="94016" />
  </view>
</view>

<view style="font-size:18px;margin-top:18px;margin-bottom:18px">
  <text style="font-size:18px;margin-top:18px;margin-bottom:18px">Work</text>
</view>

<view class="form-row">
  <view class="form-row-label">Company</view>
  <view class="form-row-content">
    <input id="organization" onInput="onInput" class="input" value="AntFin" />
  </view>
</view>
```

```
<view class="form-row">
  <view class="form-row-label">Title</view>
  <view class="form-row-content">
    <input id="title" onInput="onInput" class="input" value="Developer" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Work fax</view>
  <view class="form-row-content">
    <input id="workFaxNumber" onInput="onInput" class="input" value="11111111" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Work phone</view>
  <view class="form-row-content">
    <input id="workPhoneNumber" onInput="onInput" class="input" value="11111112" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Company phone</view>
  <view class="form-row-content">
    <input id="hostNumber" onInput="onInput" class="input" value="11111113" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Email</view>
  <view class="form-row-content">
    <input id="email" onInput="onInput" class="input" value="liuhuo01@alipay.com" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Website</view>
  <view class="form-row-content">
    <input id="url" onInput="onInput" class="input" value="www.alipay.com" />
  </view>
</view>

<view style="font-size:18px;margin-top:18px;margin-bottom:18px">
  <text style="font-size:18px;margin-top:18px;margin-bottom:18px">Work address</text>
</view>

<view class="form-row">
  <view class="form-row-label">Country</view>
  <view class="form-row-content">
    <input id="workAddressCountry" onInput="onInput" class="input" value="China" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Province</view>
  <view class="form-row-content">
    <input id="workAddressState" onInput="onInput" class="input" value="Zhejiang" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">City</view>
  <view class="form-row-content">
    <input id="workAddressCity" onInput="onInput" class="input" value="Hangzhou" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Street</view>
  <view class="form-row-content">
    <input id="workAddressStreet" onInput="onInput" class="input" value="Tianmushan Road" />
  </view>
</view>

<view class="form-row">
  <view class="form-row-label">Postcode</view>
  <view class="form-row-content">
    <input id="workAddressPostalCode" onInput="onInput" class="input" value="361005" />
  </view>
</view>

<view style="font-size:18px;margin-top:18px;margin-bottom:18px">
  <text style="font-size:18px;margin-top:18px;margin-bottom:18px">Residence</text>
</view>

<view class="form-row">
  <view class="form-row-label">Fax</view>
```

```
        <view class="form-row-content">
          <input id="homeFaxNumber" onInput="onInput" class="input" value="11111114" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Phone</view>
        <view class="form-row-content">
          <input id="homePhoneNumber" onInput="onInput" class="input" value="11111115" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Country</view>
        <view class="form-row-content">
          <input id="homeAddressCountry" onInput="onInput" class="input" value="Canada" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Province</view>
        <view class="form-row-content">
          <input id="homeAddressState" onInput="onInput" class="input" value="Ontario" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">City</view>
        <view class="form-row-content">
          <input id="homeAddressCity" onInput="onInput" class="input" value="Toronto" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Street</view>
        <view class="form-row-content">
          <input id="homeAddressStreet" onInput="onInput" class="input" value="No.234 Road" />
        </view>
      </view>

      <view class="form-row">
        <view class="form-row-label">Postcode</view>
        <view class="form-row-content">
          <input id="homeAddressPostalCode" onInput="onInput" class="input" value="123456" />
        </view>
      </view>

      <button type="primary" onTap="addPhoneContact">Add to Contacts</button>

    </view>
  </view>

</view>
```

```
// API-DEMO page/API/contact/contact.js
Page({
  data:{
      "photoFilePath": "/sdcard/DCIM/Camera/a.jpg",
      "nickName": "Baking July",
      "lastName": "Last",
      "middleName": "Middle",
      "firstName": "First",
      "remark": "Remarks",
      "mobilePhoneNumber": "13800000000",
      "homePhoneNumber": "11111115",
      "workPhoneNumber": "11111112",
      "homeFaxNumber": "11111114",
      "workFaxNumber": "11111111",
      "hostNumber": "11111113",
      "weChatNumber": "liuhuo",
      "alipayAccount": "alipay@alipay.com",
      "addressCountry": "US",
      "addressState": "California",
      "addressCity": "San Francisco",
      "addressStreet": "Mountain View",
      "addressPostalCode": "94016",
      "workAddressCountry": "China",
      "workAddressState": "Zhejiang",
      "workAddressCity": "Hangzhou",
      "workAddressStreet": "Tianmushan Road",
      "workAddressPostalCode": "361005",
      "homeAddressCountry": "Canada",
      "homeAddressState": "Ontairo",
      "homeAddressCity": "Toronto",
      "homeAddressStreet": "No.234 Road",
      "homeAddressPostalCode": "123456",
```

```
      "organization": "AntFin",
      "title": "Developer",
      "email": "liuhuo01@alipay.com",
      "url": "www.alipay.com",
      success: (res) => {
        my.alert({
          content: 'addPhoneContact response: ' + JSON.stringify(res)
        });
      },
      fail: (res) => {
        my.alert({
          content: 'addPhoneContact response: ' + JSON.stringify(res)
        });
      }
  },
  choosePhoneContact() {
    my.choosePhoneContact({
      success: (res) => {
        my.alert({
          content: 'choosePhoneContact response: ' + JSON.stringify(res)
        });
      },
      fail: (res) => {
        my.alert({
          content: 'choosePhoneContact response: ' + JSON.stringify(res)
        });
      },
    });
  },
  chooseAlipayContact() {
    my.chooseAlipayContact({
      count: 2,
      success: (res) => {
        my.alert({
          content: 'chooseAlipayContact response: ' + JSON.stringify(res)
        });
      },
      fail: (res) => {
        my.alert({
          content: 'chooseAlipayContact response: ' + JSON.stringify(res)
        });
      },
    });
  },
  chooseContact() {
    my.chooseContact({
      chooseType: 'multi', // multiple choices
      includeMe: true,     // Include
      includeMobileContactMode: 'known',//Only include the two-way contacts, that is, the contacts who have each other's phone number in their direct
ories
      multiChooseMax: 3,  // Three contacts at most
      multiChooseMaxTips: 'Exceed the maximum number of contacts allowed',
      success: (res) => {
        my.alert({
          content: 'chooseContact : ' + JSON.stringify(res)
        });
      },
      fail: (res) => {
        my.alert({
          content: 'chooseContact : ' + JSON.stringify(res)
        });
      },
    });
  },
  onInput(e) {
    this.data[e.currentTarget.id] = e.detail.value;
  },
  addPhoneContact() {
    if (my.canIUse('addPhoneContact')) {
      my.addPhoneContact(this.data);
    } else {
      my.alert({
        title: 'Client version is too low',
        content: 'my.addPhoneContact() requires 10.1.32 or later version'
      });
    }
  }
});
```

1.10.9.15. Scan code

Call the scan function.

my.scan

Call the scan function.

Input parameter

Name	Type	Required	Description
type	String	No	Scan style (default is QR): 1. QR, scan frame style is the QR code scan frame 2. Bar, scan frame style is the barcode scan frame
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed call
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

success return value

Name	Type	Description
code	String	Scanned data
qrCode	String	Return QR code data when scanning QR code
barCode	String	Return barcode data when scanning barcode

Error Code

Error Code	Description
10	User canceled
11	Operation failed

Code example

```
Page({
  scan() {
    my.scan({
      type: 'qr',
      success: (res) => {
        my.alert({ title: res.code });
      },
    });
  }
})
```

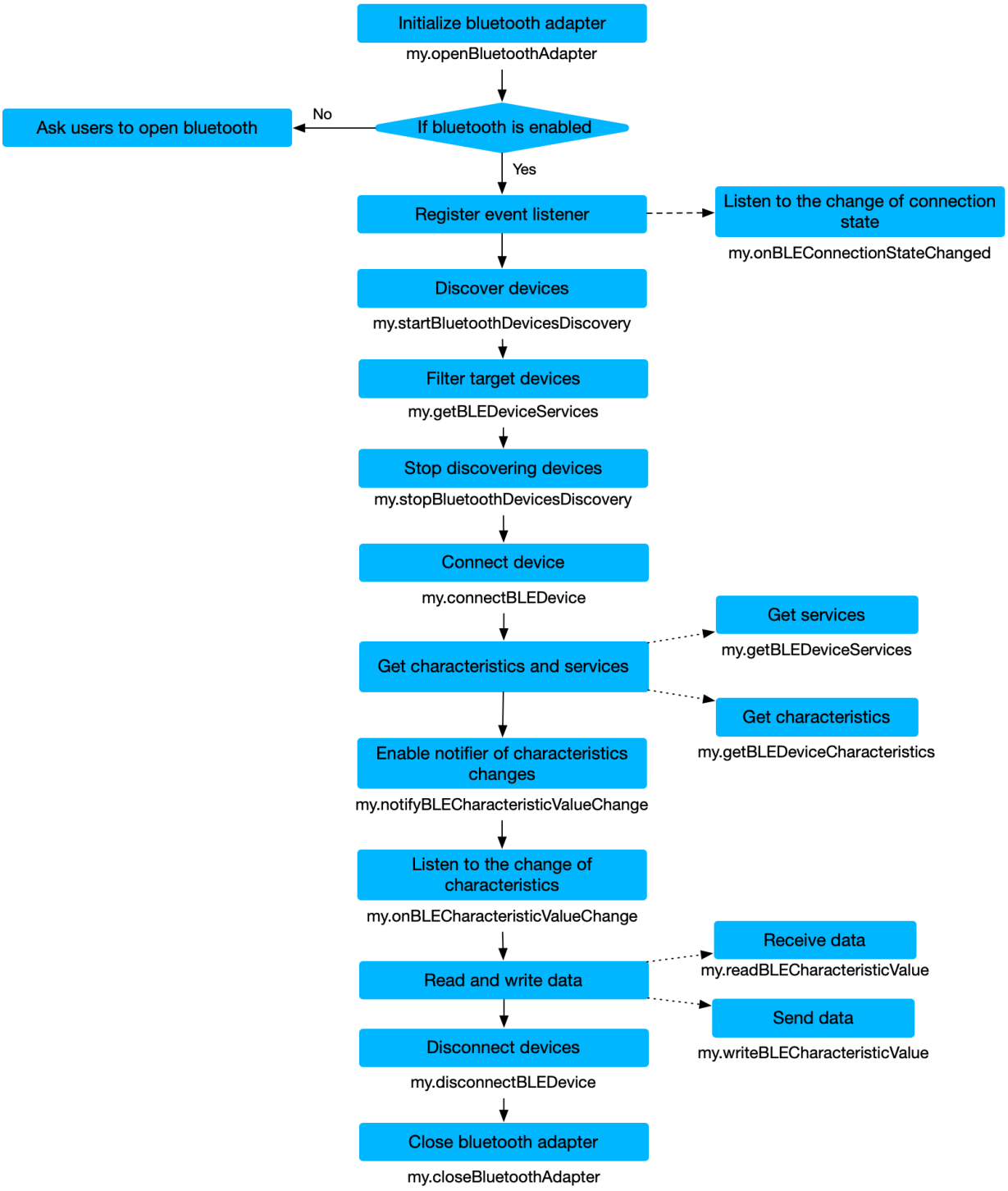
1.10.9.16. Bluetooth API overview

Version requirement

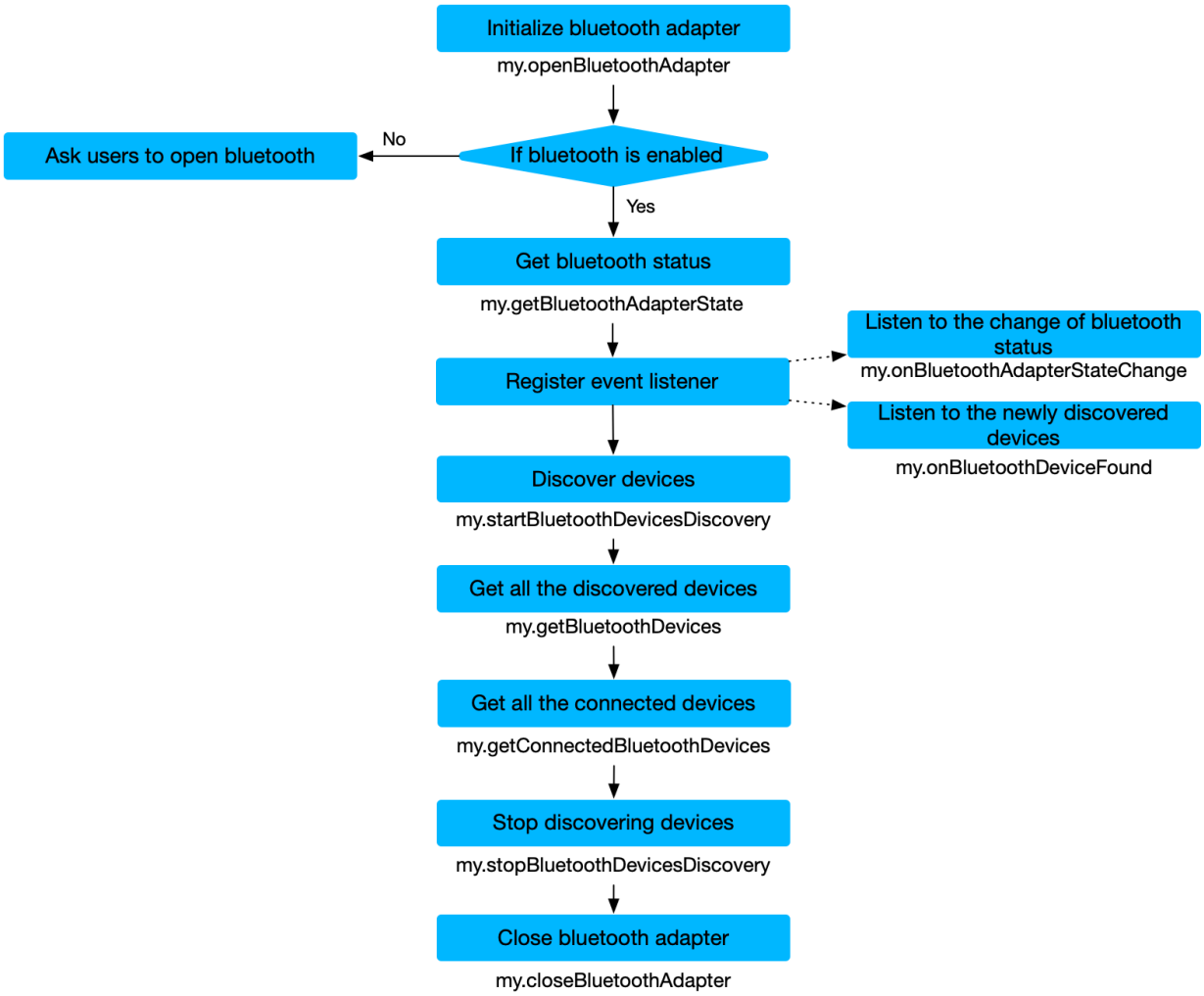
Bluetooth	mPaaS version	Android/iOS
Bluetooth Low Energy (BLE)	mPaaS 10.1.60 or later versions	• Android: 5.0 and later versions • iOS: No requirement
Traditional Bluetooth	mPaaS 10.1.60 or later versions	-

Process flow

BLE



Traditional Bluetooth



Bluetooth API

BLE

API	Description
my.connectBLEDevice	Connect to low energy bluetooth devices.
my.disconnectBLEDevice	Disconnect to low energy bluetooth devices.
my.getBLEDeviceCharacteristics	Get the characteristics of low energy Bluetooth devices.
my.getBLEDeviceServices	Get all the low energy Bluetooth devices that are discovered, including the connected devices.
my.notifyBLECharacteristicValueChange	Enable the function to notify changes to the characteristic value.
my.offBLECharacteristicValueChange	Disable the function to notify changes to the characteristic value.
my.offBLEConnectionStateChanged	Disable the listener for connection status change events.
my.onBLECharacteristicValueChange	Enable the listener for characteristic value change events.
my.onBLEConnectionStateChanged	Enable the listener for connection status change events, such as device lost and device disconnected.
my.readBLECharacteristicValue	Read the characteristic value.
my.writeBLECharacteristicValue	Write data to the characteristic value.

Traditional Bluetooth

API	Description
my.closeBluetoothAdapter	Close the Bluetooth module in the mini program.
my.getBluetoothAdapterState	Get the Bluetooth adapter status in the mini program.

API	Description
my.getBluetoothDevices	Get all the low energy Bluetooth devices that are discovered, including the connected devices.
my.getConnectedBluetoothDevices	Get the Bluetooth devices that are connected.
my.offBluetoothAdapterStateChange	Remove the listener for connection status change events.
my.offBluetoothDeviceFound	Remove the listener for new Bluetooth device discovery events.
my.onBluetoothDeviceFound	Trigger the event when a new Bluetooth device is found.
my.onBluetoothAdapterStateChange	Monitor the Bluetooth status change events.
my.openBluetoothAdapter	Initialize the Bluetooth module in the mini program.
my.startBluetoothDevicesDiscovery	Start discovering Bluetooth devices nearby.
my.stopBluetoothDevicesDiscovery	Stop discovering Bluetooth devices nearby.

Sample code

```
//Initialize
my.openBluetoothAdapter({
  success: (res) => {
      console.log(res);
  }
});
//Register discovery event
my.onBluetoothDeviceFound({
  success: (res) => {
    let device = res.devices[0];
    //Connect the discovered device
    my.connectBLEDevice({
      deviceId: deviceId,
      success: (res) => {
        console.log(res)
      },
      fail:(res) => {
      },
      complete: (res)=>{
      }
    });
    //Stop discovering
    my.stopBluetoothDevicesDiscovery({
      success: (res) => {
        console.log(res)
      },
      fail:(res) => {
      },
      complete: (res)=>{
      }
    });
  }
});
//Register connection event
my.onBLEConnectionStateChanged({
  success: (res) => {
      console.log(res);
    if (res.connected) {
        //Start reading and writing notify and other operations
        my.notifyBLECharacteristicValueChange({
          deviceId: deviceId,
          serviceId: serviceId,
          characteristicId: characteristicId,
          success: (res) => {
              console.log(res)
          },
          fail:(res) => {
          },
          complete: (res)=>{
          }
        });
    }
  }
});
//Register the read or notify data reception events
my.onBLECharacteristicValueChange({
  success: (res) => {
      console.log(res);
  }
});
//Start discovering
my.startBluetoothDevicesDiscovery({
  services: ['fff0'],
```

```
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});

//Disconnect
my.disconnectBLEDevice({
  deviceId: deviceId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});

// Unregister events
my.offBluetoothDeviceFound();
my.offBLEConnectionStateChanged();
my.offBLECharacteristicValueChange();

//Exit bluetooth module
my.closeBluetoothAdapter({
  success: (res) => {
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

1.10.9.17. Bluetooth API list

Note

- Bluetooth APIs are supported in mPaaS 1.60.0 and later versions.
- Currently, debugging on developer tools is not supported. You need to use a physical device to properly call the Bluetooth API of Mini Program.

my.openBluetoothAdapter

This interface is used to initialize the Bluetooth module in the mini program, with the effective period beginning when `my.openBluetoothAdapter` is called and ending when `my.closeBluetoothAdapter` is called or the mini program is destroyed. During the effective period of the Bluetooth adapter module, developers can call the following APIs, and receive the `on` event callback related to the Bluetooth module.

Input parameters

Parameter	Type	Required	Description
autoClose	Boolean	No	Indicates whether to automatically disconnect Bluetooth when you leave the current page. The default value is true. Note, only Android is supported.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Parameter	Type	Description
isSupportBLE	Boolean	Indicates whether BLE is supported.

Error codes

Error	Description	Solution
12	Bluetooth is not turned on.	Try again to turn on Bluetooth.
13	The connection to the system service is temporarily lost.	Try again to reconnect.
14	The client does not have the permission to use Bluetooth.	Authorize app to use Bluetooth.
15	Unknown error	-

Code sample

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary" onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All connected devices/button>
       <button type="primary" onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}" placeholder=" Input the ID of the device to be connected"></input>
       <button type="primary" onTap="connectBLEDevice">Connect the device</button>
       <button type="primary" onTap="getBLEDeviceServices"> Obtain device services </button>
       <button type="primary" onTap="getBLEDeviceCharacteristics"> Obtain read and write characteristics </button>
       <button type="primary" onTap="disconnectBLEDevice"> Disconnect the device </button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary" onTap="notifyBLECharacteristicValueChange">Listen to the characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read data</button>
       <button type="primary" onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary" onTap="offBLECharacteristicValueChange">Cancel listening to characteristic value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary" onTap="bluetoothAdapterStateChange">Bluetooth status change</button>
       <button type="primary" onTap="offBluetoothAdapterStateChange">Cancel listening to Bluetooth status</button>
       <button type="primary" onTap="BLEConnectionStateChanged">Bluetooth connection status changes</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Cancel listening to Bluetooth connection status</button>
    </view>
  </view>
</view>
```

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },
  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Initialization succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
```

```
closeBluetoothAdapter() {
  my.closeBluetoothAdapter({
    success: () => {
      my.alert({ content: 'Bluetooth closed!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
getBluetoothAdapterState() {
  my.getBluetoothAdapterState({
    success: res => {
      if (!res.available) {
        my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Scan the Bluetooth device
startBluetoothDevicesDiscovery() {
  my.startBluetoothDevicesDiscovery({
    allowDuplicatesKey: false,
    success: () => {
      my.onBluetoothDeviceFound({
        success: res => {
          // my.alert({content:'Listen to new device '+JSON.stringify(res)});
          var deviceArray = res.devices;
          for (var i = deviceArray.length - 1; i >= 0; i--) {
            var deviceObj = deviceArray[i];
            //Match the target device by device name or broadcast data, and then record the device ID for later use
            if (deviceObj.name == this.data.name) {
              my.alert({ content: 'Target device is found' });
              my.offBluetoothDeviceFound();
              this.setData({
                deviceId: deviceObj.deviceId,
              });
              break;
            }
          }
        },
        fail: error => {
          my.alert({ content: 'Failed to listen to new device' + JSON.stringify(error) });
        },
      });
    },
    fail: error => {
      my.alert({ content: 'Failed to start scanning' + JSON.stringify(error) });
    },
  });
},
//Stop scanning
stopBluetoothDevicesDiscovery() {
  my.stopBluetoothDevicesDiscovery({
    success: res => {
      my.offBluetoothDeviceFound();
      my.alert({ content: 'Succeeded!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain all devices found
getBluetoothDevices() {
  my.getBluetoothDevices({
```

```
      success: res => {
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  bindKeyInput(e) {
    this.setData({
      devid: e.detail.value,
    });
  },
  //Connect the device
  connectBLEDevice() {
    my.connectBLEDevice({
      deviceId: this.data.devid,
      success: res => {
        my.alert({ content: 'Connected successfully' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Disconnect
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Disconnected successfully!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Obtain the services of the connected devices. The services can only be obtained when the devices are connected.
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  //Obtain the charid of the connected devices. The charid can only be obtained when the devices are connected. The read and write characteristics ar
e respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information about the attributes of the characteristics. Match the read/write characteristics by attr
ibute and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
```

```
    },
  });
},
//Read and write data
readBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.readBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        //1. Android read and write services
        // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
        success: res => {
          my.alert({ content: JSON.stringify(res) });
        },
        fail: error => {
          my.alert({ content: 'Failed to read and write' + JSON.stringify(error) });
        },
      });
    },
  });
},
writeBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.writeBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.charid,
        //Android write service
        //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
        value: 'ABCD',
        success: res => {
          my.alert({ content: 'Data written successfully!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
  });
},
notifyBLECharacteristicValueChange() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.notifyBLECharacteristicValueChange({
        state: true,
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        success: () => {
          //Listen to characteristic change events
          my.onBLECharacteristicValueChange({
            success: res => {
              //  my.alert({content: 'Characteristic change:'+JSON.stringify(res)});
              my.alert({ content: 'Obtain the response data = ' + res.value });
            },
          });
          my.alert({ content: 'Listened successfully' });
        },
        fail: error => {
          my.alert({ content: 'Failed to listen' + JSON.stringify(error) });
```

```
          },
        });
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },
  //Other events
  bluetoothAdapterStateChange() {
    my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Bluetooth status change: ' + JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {
    my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {
    my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Connection status change: ' + JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {
    my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Instructions

- If you call other APIs in the Bluetooth module before calling the API `my.openBluetoothAdapter`, an error will be returned:
  - Error Code: 10000
  - Error description: The Bluetooth adapter is not initialized.
  - Solution: Call the API `my.openBluetoothAdapter`.
- When users do not switch on the Bluetooth or the Bluetooth function is not supported on the user's mobile phone, an error is returned after `my.openBluetoothAdapter` is called. For more information about error codes, see the Bluetooth API error codes. After the Bluetooth module is initialized, you can use the API `my.onBluetoothAdapterStateChange` to monitor changes of the Bluetooth status.

my.closeBluetoothAdapter

This interface is used to close the Bluetooth module in the mini program.

Input parameters

Parameter	Type	Required	Description
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.closeBluetoothAdapter({
  success: (res) => {
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- Calling this method will disconnect all established Bluetooth connections and release system resources.
- We recommend that you call this API in pair with `my.openBluetoothAdapter` when ending the Bluetooth process of the Mini Program.
- Calling `my.closeBluetoothAdapter` to free resources is an asynchronous operation. We do not recommend that you use `my.closeBluetoothAdapter` and `my.openBluetoothAdapter` to handle exception, because it is equivalent to turning Bluetooth off, and then turning on and reinitializing it. The whole process is inefficient and prone to thread synchronization problems.

my.getBluetoothAdapterState

Obtains the status of the Bluetooth module in the mini program.

Input parameters

Parameter	Type	Required	Description
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Parameter	Type	Description
discovering	Boolean	Indicates whether the device is searching for matching devices.
available	Boolean	Indicates whether the Bluetooth module is available (BLE is supported and Bluetooth is on).

Code sample

```
my.getBluetoothAdapterState({
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

my.startBluetoothDevicesDiscovery

You can call this interface to start searching for nearby Bluetooth devices. The search results are returned in the `my.onBluetoothDeviceFound` event.

Input parameters

Parameter	Type	Required	Description
services	Array	No	The UUID list of the Bluetooth device's master service.

Parameter	Type	Required	Description
allowDuplicatesKey	Boolean	No	Whether to report the same device repeatedly. If yes, the `onBluetoothDeviceFound` method will report the same device multiple times, but the RSSI value may be different.
interval	Integer	No	Specifies the interval for reporting devices. Default value is 0, that is, when a new device is found, it is reported immediately. Otherwise, it is reported according to the specified interval.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.startBluetoothDevicesDiscovery({
  services: ['fff0'],
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- This operation consumes more system resources. Please call the `stop` method to stop the search after any device is discovered and connected.

my.stopBluetoothDevicesDiscovery

You can call this interface to stop searching for nearby Bluetooth devices.

Input parameters

Parameter	Type	Required	Description
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.stopBluetoothDevicesDiscovery({
  success: (res) => {
    console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

my.getBluetoothDevices

You can call this interface to obtain all discovered Bluetooth devices, including the Bluetooth device that is already connected.

Input parameters

Parameter	Type	Required	Description
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Parameter	Type	Description
devices	Array	List of discovered devices

device objects

Name	Type	Description
name	String	The name of the Bluetooth device. Some devices may not have a name.
deviceName (compatible with earlier versions)	String	The value is consistent with that of `name`.
localName	String	Name of the advertising device.
deviceId	String	Device ID
RSSI	Number	The signal strength of the device.
advertisData	Hex String	The advertising content of the device.
manufacturerData	Hex String	The manufacturer data of the device.

Code sample

```
my.getBluetoothDevices({
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- The simulator may not be able to obtain advertisData and RSSI. Please use a physical device for debugging.

- For Integrated Development Environment (IDE) and Android devices, the device ID is the MAC address of the device; for iOS device, the device ID is the UUID of the device. Therefore, the device ID cannot be hard coded, and needs to be processed based on the specific platform. iOS can dynamically match the deviceId according to the device properties (such as `localName`, `advertisData`, `manufacturerData`).

my.getConnectedBluetoothDevices

You can call this interface to obtain information of the connected devices.

Input parameters

Parameter	Type	Required	Description
deviceId	String	No	The ID of the Bluetooth device.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.getConnectedBluetoothDevices({
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- If you have searched for a Bluetooth device in the mini program before, you can directly pass in the deviceId obtained by the previous search to connect to the device.
- If the specified bluetooth device is connected, you'll be returned with a success response if the connection is repeated.

my.connectBLEDevice

You can call this interface to connect to the Bluetooth Low Energy (BLE) device.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.connectBLEDevice({
  // The device ID here needs to be obtained in the getBluetoothDevices or onBluetoothDeviceFound APIs above
  deviceId: deviceId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- If you have searched for a Bluetooth device in the mini program before, you can directly pass in the deviceId obtained by the previous search to connect to the device.
- If the specified Bluetooth device is connected, you'll be returned with a success response if the connection is repeated.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.disconnectBLEDevice

You can call this interface to disconnect from the BLE device.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.disconnectBLEDevice({
  deviceId: deviceId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- The Bluetooth device may be disconnected at any time. We recommend that you listen to the `my.onBLEConnectionStateChanged` callback, and reconnect as needed when the Bluetooth device is disconnected.
- If the API for data read/write operation is called for unconnected devices or disconnected devices, 10006 error is returned. See the Bluetooth API error codes for details. It is recommended to perform the reconnection operation.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.writeBLECharacteristicValue

You can call this interface to write data into the characteristic value of the BLE device.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device. For more information, see the `device` objects.
serviceId	String	Yes	The UUID of the `service` to which the Bluetooth characteristic corresponds.

Parameter	Type	Required	Description
characteristicId	String	Yes	The UUID of the Bluetooth characteristic.
value	Hex String	Yes	The value corresponding to the Bluetooth characteristic. It is a hexadecimal string limited to 20 bytes.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.writeBLECharacteristicValue({
  deviceId: deviceId,
  serviceId: serviceId,
  characteristicId: characteristicId,
  value: 'fffe',
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- The characteristic of the device must support “write”, otherwise the interface cannot be successfully called. Please see the `properties` of the characteristic for details.
- The binary data written needs to be hex encoded.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.readBLECharacteristicValue

You can call this interface to read data from the characteristic value of the BLE device. Then, the data is returned in the `my.onBLECharacteristicValueChange()` event.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device. For more information, see the `device` objects.
serviceId	String	Yes	The UUID of the `service` to which the Bluetooth characteristic corresponds.
characteristicId	String	Yes	The UUID of the Bluetooth characteristic.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Parameter	Type	Description
characteristic	Object	The information of the device characteristic.

characteristic objects

The characteristic information of the Bluetooth device.

Name	Type	Description
characteristicId	String	The UUID of the Bluetooth device characteristic.
serviceId	String	The UUID of the service corresponding to the Bluetooth device characteristic.
value	Hex String	The value of the Bluetooth device characteristic.

Code sample

```
my.readBLECharacteristicValue({
  deviceId: deviceId,
  serviceId: serviceId,
  characteristicId: characteristicId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- The characteristic of the device must support “read”, otherwise the interface cannot be successfully called. Please see the `properties` of the characteristic for details.
- Multiple parallel reading and writing calls might cause failures.
- If the reading times out, the error code is 10015. However, `my.onBLECharacteristicValueChange` interface might return data later. Please process the error with reference to Bluetooth API error codes.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.notifyBLECharacteristicValueChange

You can call this interface to enable the “notify” feature for the characteristic value change of the BLE device.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device. For more information, see the `device` objects.
serviceId	String	Yes	The UUID of the `service` to which the Bluetooth characteristic corresponds.
characteristicId	String	Yes	The UUID of the Bluetooth characteristic.
descriptorId	String	No	The UUID of the `descriptor` for the `notify` (This is only used by Android, and is not required. The default value is 00002902-0000-10008000-00805f9b34fb)
state	Boolean	No	Whether to enable `notify` or `indicate`.

Parameter	Type	Required	Description
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Code sample

```
my.notifyBLECharacteristicValueChange({
  deviceId: deviceId,
  serviceId: serviceId,
  characteristicId: characteristicId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- The characteristic of the device must support “notify/indicate”, otherwise the interface cannot be successfully called.
- The “notify” feature must be enabled to listen to the `characteristicValueChange` event of the device.
- After the subscription operation is successful, the device needs to actively update the characteristic value to trigger `my.onBLECharacteristicValueChange`.
- The subscription operation is more efficient. So, it is recommended to use the subscription method instead of the read method.

my.getBLEDeviceServices

You can call this interface to obtain all the Bluetooth devices that are discovered, including the connected devices.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device. For more information, see the `device` objects.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Parameter	Type	Description
services	Array	List of discovered device services

service objects

The service information of the Bluetooth device.

Name	Type	Description
serviceId	String	The UUID of the Bluetooth device service.

Name	Type	Description
isPrimary	Boolean	Indicates whether the service is the primary service. true: primary service false: not primary service

Code sample

```
// Obtain the services of the connected devices. The services can only be obtained when the devices are connected.
 getBLEDeviceServices() {
   my.getConnectedBluetoothDevices({
     success: res => {
       if (res.devices.length === 0) {
         my.alert({ content: 'No connected devices' });
         return;
       }
       my.getBLEDeviceServices({
         deviceId: this.data.devid,
         success: res => {
           my.alert({ content: JSON.stringify(res) });
           this.setData({
             serid: res.services[0].serviceId,
           });
         },
         fail: error => {
           my.alert({ content: JSON.stringify(error) });
         },
       });
     },
   });
 },
```

Example of return value

```
{
    "services": [{
        "isPrimary": true,
        "serviceId": "00001800-0000-1000-8000-00805f9b34fb"
    }, {
        "isPrimary": true,
        "serviceId": "00001801-0000-1000-8000-00805f9b34fb"
    }, {
        "isPrimary": true,
        "serviceId": "d0611e78-bbb4-4591-a5f8-487910ae4366"
    }, {
        "isPrimary": true,
        "serviceId": "9fa480e0-4967-4542-9390-d343dc5d04ae"
    }]
}
```

Instructions

- After the connection is established, you should execute `my.getBLEDeviceServices` and `my.getBLEDeviceCharacteristics` first, and then perform data interaction with the Bluetooth device.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.getBLEDeviceCharacteristics

You can call this interface to obtain all characteristics of the Bluetooth device.

Input parameters

Parameter	Type	Required	Description
deviceId	String	Yes	The ID of the Bluetooth device. For more information, see the device objects.
serviceId	String	Yes	The UUID of the service to which the Bluetooth characteristic corresponds.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.

Parameter	Type	Required	Description
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value on success

Name	Type	Description
characteristics	Array	The list of device characteristics.

characteristic objects

The characteristic information of the Bluetooth device.

Name	Type	Description
characteristicId	String	The UUID of the Bluetooth device characteristic.
serviceId	String	The UUID of the service corresponding to the Bluetooth device characteristic.
value	Hex String	The hexadecimal value corresponding to the Bluetooth device characteristic.
properties	Object	The type of operation supported by the characteristic.

properties objects

Name	Type	Description
read	Boolean	Indicates whether the characteristic supports read operations.
write	Boolean	Indicates whether the characteristic supports write operations.
notify	Boolean	Indicates whether the characteristic supports notify operations.
indicate	Boolean	Indicates whether the characteristic supports indicate operations.

Code sample

```
my.getBLEDeviceCharacteristics({
  deviceId: deviceId,
  serviceId: serviceId,
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Instructions

- After the connection is established, you should execute `my.getBLEDeviceServices` and `my.getBLEDeviceCharacteristics` first, and then perform data interaction with the Bluetooth device.
- The interface is supported by iOS as well as Android 5.0 and later versions.

my.onBluetoothDeviceFound(callback)

This callback is triggered when a new Bluetooth device is found.

Input parameters

Parameter	Type	Required	Description
callback	Function	Yes	The callback that is triggered when the specified event occurs.

Return values of callback

Parameter	Type	Description
devices	Array	The list of newly found devices.

device objects

Name	Type	Description
name	String	The name of the Bluetooth device. Some devices may not have a name.
deviceName (compatible with earlier versions)	String	The value is consistent with that of `name`.
localName	String	The name of the advertising device.
deviceId	String	The ID of the device.
RSSI	Number	The signal strength of the device.
advertisData	Hex String	The advertising content of the device.

Code sample

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBluetoothDeviceFound(this.callback);
  },
  onUnload() {
    my.offBluetoothDeviceFound(this.callback);
  },
  callback(res) {
    console.log(res);
  },
})
```

Instructions

- The simulator may not be able to obtain `advertisData` and RSSI. Please use a physical device for debugging.
- For Integrated Development Environment (IDE) and Android devices, the device ID is the MAC address of the device; for iOS device, the device ID is the UUID of the device. Therefore, the device ID cannot be hard coded, and needs to be processed based on the specific platform. iOS can dynamically match the `deviceId` according to the device properties (such as `localName`, `advertisData`, `manufacturerData`).
- If a Bluetooth device is included in the `my.onBluetoothDeviceFound` callback, the device is added to the array obtained by `my.getBluetoothDevices`.

my.offBluetoothDeviceFound

You can call this interface to remove the listener for new Bluetooth device discovery events.

Code sample

```
my.offBluetoothDeviceFound();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

```
my.offBluetoothDeviceFound();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

```
my.offBluetoothDeviceFound(this.callback);
```

Instructions

- It is recommended that you call the `off` method to close existing event listening before you call the `on` method to listen events to prevent the situation where multiple event listening causes multiple callbacks of an event.

my.onBLECharacteristicValueChange(callback)

You can call this interface to listen to characteristic value change events of BLE device.

Input parameters

Parameter	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Parameter	Type	Description
deviceId	String	The ID of the Bluetooth device. For more information, see the `device` objects.
connected	Boolean	The current state of the connection.

Code sample

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
})
```

Instructions

- It is recommended that you call the `off` method to close existing event listening before you call the `on` method to listen events to prevent the situation where multiple event listening causes multiple callbacks of an event.

my.offBLECharacteristicValueChange

You can call this interface to stop listening to the characteristic value change events of BLE device.

Input parameters

Parameter	Type	Description
deviceId	String	The ID of the Bluetooth device. For more information, see the `device` objects.
serviceId	String	The UUID of the service to which the Bluetooth characteristic corresponds.
characteristicId	String	The UUID of the Bluetooth characteristic.
value	Hex String	The latest hexadecimal value of the characteristic.

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

```
my.offBLECharacteristicValueChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

```
my.offBLECharacteristicValueChange();
```

Code sample

```
Page({
  onLoad() {
    this.callback = this.callback.bind(this);
    my.onBLECharacteristicValueChange(this.callback);
  },
  onUnload() {
    my.offBLECharacteristicValueChange(this.callback);
  },
  callback(res) {
    console.log(res);
  },
})
```

my.onBLEConnectionStateChanged(callback)

You can call this interface to listen to the BLE connection error events, such as device lost and device disconnected.

Input parameters

Parameter	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Parameter	Type	Description
deviceId	String	The ID of the Bluetooth device. For more information, see the `device` objects.
connected	Boolean	The current state of the connection.

Code sample

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary" onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All connected devicese/button>
       <button type="primary" onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}" placeholder=" Input the ID of the device to be connected "></input>
       <button type="primary" onTap="connectBLEDevice">Connect the device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain device services</button>
       <button type="primary" onTap="getBLEDeviceCharacteristics">Obtain read and write characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice"> Disconnect the device </button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary" onTap="notifyBLECharacteristicValueChange">Listen to the characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read data</button>
       <button type="primary" onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary" onTap="offBLECharacteristicValueChange">Cancel listening to characteristic value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary" onTap="bluetoothAdapterStateChange">Bluetooth status change</button>
       <button type="primary" onTap="offBluetoothAdapterStateChange">Cancel listening to Bluetooth status</button>
       <button type="primary" onTap="BLEConnectionStateChanged">Bluetooth connection status changes</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Cancel listening to Bluetooth connection status</button>
    </view>
  </view>
</view>
```

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },
  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Initialization succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
      success: () => {
        my.alert({ content: 'Bluetooth closed!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  getBluetoothAdapterState() {
    my.getBluetoothAdapterState({
      success: res => {
        if (!res.available) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
          return;
        }
        my.alert({ content: JSON.stringify(res) });
```

```
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Scan the Bluetooth device
startBluetoothDevicesDiscovery() {
  my.startBluetoothDevicesDiscovery({
    allowDuplicatesKey: false,
    success: () => {
      my.onBluetoothDeviceFound({
        success: res => {
          // my.alert({content:'Listen to new device'+JSON.stringify(res)});
          var deviceArray = res.devices;
          for (var i = deviceArray.length - 1; i >= 0; i--) {
            var deviceObj = deviceArray[i];
            //Match the target device by device name or broadcast data, and then record the device ID for later use
            if (deviceObj.name == this.data.name) {
              my.alert({ content: 'Target device is found' });
              my.offBluetoothDeviceFound();
              this.setData({
                deviceId: deviceObj.deviceId,
              });
              break;
            }
          }
        },
        fail: error => {
          my.alert({ content: 'Failed to listen to new device' + JSON.stringify(error) });
        },
      });
    },
    fail: error => {
      my.alert({ content: 'Failed to start scanning' + JSON.stringify(error) });
    },
  });
},
//Stop scanning
stopBluetoothDevicesDiscovery() {
  my.stopBluetoothDevicesDiscovery({
    success: res => {
      my.offBluetoothDeviceFound();
      my.alert({ content: 'Succeeded!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain all devices found
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
bindKeyInput(e) {
  this.setData({
    devid: e.detail.value,
  });
},
//Connect the device
connectBLEDevice() {
  my.connectBLEDevice({
    deviceId: this.data.devid,
    success: res => {
```

```
        my.alert({ content: 'Connected successfully' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Disconnect
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Disconnected successfully!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Obtain the services of the connected devices. The services can only be obtained when the devices are connected.
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  //Obtain the charid of the connected devices. The charid can only be obtained when the devices are connected. The read and write characteristics ar
e respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information about the attributes of the characteristics. Match the read/write characteristics by attr
ibute and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  //Read and write data
  readBLECharacteristicValue() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.readBLECharacteristicValue({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          characteristicId: this.data.notifyId,
```

```
        //1. Android read and write services
        // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
        success: res => {
          my.alert({ content: JSON.stringify(res) });
        },
        fail: error => {
          my.alert({ content: 'Failed to read and write' + JSON.stringify(error) });
        },
      });
    },
  });
},
writeBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.writeBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.charid,
        //Android write service
        //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
        value: 'ABCD',
        success: res => {
          my.alert({ content: 'Data written successfully!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
  });
},
notifyBLECharacteristicValueChange() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.notifyBLECharacteristicValueChange({
        state: true,
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        success: () => {
          //Listen to characteristic change events
          my.onBLECharacteristicValueChange({
            success: res => {
              //  my.alert({content: 'Characteristic change:'+JSON.stringify(res)});
              my.alert({ content: 'Obtain the response data = ' + res.value });
            },
          });
          my.alert({ content: 'Listened successfully' });
        },
        fail: error => {
          my.alert({ content: 'Failed to listen' + JSON.stringify(error) });
        },
      });
    },
  });
},
offBLECharacteristicValueChange() {
  my.offBLECharacteristicValueChange();
},
//Other events
bluetoothAdapterStateChange() {
  my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
},
onBluetoothAdapterStateChange() {
  if (res.error) {
    my.alert({ content: JSON.stringify(error) });
  } else {
    my.alert({ content: 'Bluetooth status change: ' + JSON.stringify(res) });
  }
},
```

```
  offBluetoothAdapterStateChange() {
    my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {
    my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Connection status change: ' + JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {
    my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

Instructions

- It is recommended that you call the `off` method to close existing event listening before you call the `on` method to listen events to prevent the situation where multiple event listening causes multiple callbacks of an event.

my.offBLEConnectionStateChanged

You can call this interface to stop listening to the connection status change events of the BLE device.

Code sample

```
my.offBLEConnectionStateChanged();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

  ```
  my.offBLEConnectionStateChanged();
  ```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

  ```
  my.offBLEConnectionStateChanged(this.callback);
  ```

Instructions

- It is recommended that you call the `off` method to close existing event listening before you call the `on` method to listen events to prevent the situation where multiple event listening causes multiple callbacks of an event.

my.onBluetoothAdapterStateChange(callback)

You can call this interface to listen to the native Bluetooth status change events.

Input parameters

Parameter	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Parameter	Type	Description
available	Boolean	Indicates whether the Bluetooth module is available.
discovering	Boolean	Indicates whether the Bluetooth module is in the searching state.

Code sample

```
/* .acss */
.help-info {
  padding:10px;
  color:#000000;
}
.help-title {
  padding:10px;
  color:#FC0D1B;
}
```

```
// .json
{
    "defaultTitle": "Bluetooth"
}
```

```
<!-- .axml-->
<view class="page">
  <view class="page-description">Bluetooth API</view>
  <view class="page-section">
    <view class="page-section-title">Bluetooth state</view>
    <view class="page-section-demo">
       <button type="primary" onTap="openBluetoothAdapter">Initialize Bluetooth</button>
       <button type="primary" onTap="closeBluetoothAdapter">Close Bluetooth</button>
       <button type="primary" onTap="getBluetoothAdapterState">Obtain Bluetooth state</button>
    </view>
    <view class="page-section-title">Scan the Bluetooth device</view>
    <view class="page-section-demo">
       <button type="primary" onTap="startBluetoothDevicesDiscovery">Start searching</button>
       <button type="primary" onTap="getBluetoothDevices">All devices found</button>
       <button type="primary" onTap="getConnectedBluetoothDevices">All connected devices/button>
       <button type="primary" onTap="stopBluetoothDevicesDiscovery">Stop searching</button>
    </view>
    <view class="page-section-title">Connect the device</view>
    <view class="page-section-demo">
       <input class="input" onInput="bindKeyInput" type="{{text}}" placeholder=" Input the ID of the device to be connected "></input>
       <button type="primary" onTap="connectBLEDevice">Connect the device</button>
       <button type="primary" onTap="getBLEDeviceServices">Obtain device services</button>
       <button type="primary" onTap="getBLEDeviceCharacteristics">Obtain read and write characteristics</button>
       <button type="primary" onTap="disconnectBLEDevice">Disconnect the device</button>
    </view>
     <view class="page-section-title">Read and write data</view>
     <view class="page-section-demo">
       <button type="primary" onTap="notifyBLECharacteristicValueChange">Listen to the characteristic data change</button>
       <button type="primary" onTap="readBLECharacteristicValue">Read data</button>
       <button type="primary" onTap="writeBLECharacteristicValue">Write data</button>
       <button type="primary" onTap="offBLECharacteristicValueChange">Cancel listening to characteristic value</button>
    </view>
     <view class="page-section-title">Other events</view>
     <view class="page-section-demo">
       <button type="primary" onTap="bluetoothAdapterStateChange">Bluetooth status change</button>
       <button type="primary" onTap="offBluetoothAdapterStateChange">Cancel listening to Bluetooth status</button>
       <button type="primary" onTap="BLEConnectionStateChanged">Bluetooth connection status changes</button>
       <button type="primary" onTap="offBLEConnectionStateChanged">Cancel listening to Bluetooth connection status</button>
    </view>
  </view>
</view>
```

```
// .js
Page({
  data: {
    devid: '0D9C82AD-1CC0-414D-9526-119E08D28124',
    serid: 'FEE7',
    notifyId: '36F6',
    writeId: '36F5',
    charid: '',
    alldev: [{ deviceId: '' }],
  },
  //Obtain the Bluetooth state
  openBluetoothAdapter() {
    my.openBluetoothAdapter({
      success: res => {
        if (!res.isSupportBLE) {
          my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
          return;
        }
        my.alert({ content: 'Initialization succeeded!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  closeBluetoothAdapter() {
    my.closeBluetoothAdapter({
```

```
    success: () => {
      my.alert({ content: 'Bluetooth closed!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
getBluetoothAdapterState() {
  my.getBluetoothAdapterState({
    success: res => {
      if (!res.available) {
        my.alert({ content: 'Sorry, your mobile Bluetooth is unavailable temporarily' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Scan the Bluetooth device
startBluetoothDevicesDiscovery() {
  my.startBluetoothDevicesDiscovery({
    allowDuplicatesKey: false,
    success: () => {
      my.onBluetoothDeviceFound({
        success: res => {
          // my.alert({content:'Listen to new device'+JSON.stringify(res)});
          var deviceArray = res.devices;
          for (var i = deviceArray.length - 1; i >= 0; i--) {
            var deviceObj = deviceArray[i];
            //Match the target device by device name or broadcast data, and then record the device ID for later use
            if (deviceObj.name == this.data.name) {
              my.alert({ content: 'Target device is found' });
              my.offBluetoothDeviceFound();
              this.setData({
                deviceId: deviceObj.deviceId,
              });
              break;
            }
          }
        },
        fail: error => {
          my.alert({ content: 'Failed to listen to new device' + JSON.stringify(error) });
        },
      });
    },
    fail: error => {
      my.alert({ content: 'Failed to start scanning' + JSON.stringify(error) });
    },
  });
},
//Stop scanning
stopBluetoothDevicesDiscovery() {
  my.stopBluetoothDevicesDiscovery({
    success: res => {
      my.offBluetoothDeviceFound();
      my.alert({ content: 'Succeeded!' });
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain the connected device
getConnectedBluetoothDevices() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices!' });
        return;
      }
      my.alert({ content: JSON.stringify(res) });
      devid = res.devices[0].deviceId;
    },
    fail: error => {
      my.alert({ content: JSON.stringify(error) });
    },
  });
},
//Obtain all devices found
getBluetoothDevices() {
  my.getBluetoothDevices({
    success: res => {
```

```
        my.alert({ content: JSON.stringify(res) });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  bindKeyInput(e) {
    this.setData({
      devid: e.detail.value,
    });
  },
  //Connect the device
  connectBLEDevice() {
    my.connectBLEDevice({
      deviceId: this.data.devid,
      success: res => {
        my.alert({ content: 'Connected successfully' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Disconnect
  disconnectBLEDevice() {
    my.disconnectBLEDevice({
      deviceId: this.data.devid,
      success: () => {
        my.alert({ content: 'Disconnected successfully!' });
      },
      fail: error => {
        my.alert({ content: JSON.stringify(error) });
      },
    });
  },
  //Obtain the services of the connected devices. The services can only be obtained when the devices are connected.
  getBLEDeviceServices() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        my.getBLEDeviceServices({
          deviceId: this.data.devid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            this.setData({
              serid: res.services[0].serviceId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
    });
  },
  //Obtain the charid of the connected devices. The charid can only be obtained when the devices are connected. The read and write characteristics ar
e respectively screened out.
  getBLEDeviceCharacteristics() {
    my.getConnectedBluetoothDevices({
      success: res => {
        if (res.devices.length === 0) {
          my.alert({ content: 'No connected devices' });
          return;
        }
        this.setData({
          devid: res.devices[0].deviceId,
        });
        my.getBLEDeviceCharacteristics({
          deviceId: this.data.devid,
          serviceId: this.data.serid,
          success: res => {
            my.alert({ content: JSON.stringify(res) });
            //See the related document for more information about the attributes of the characteristics. Match the read/write characteristics by attr
ibute and record the value for later use.
            this.setData({
              charid: res.characteristics[0].characteristicId,
            });
          },
          fail: error => {
            my.alert({ content: JSON.stringify(error) });
          },
        });
      },
```

```
  });
},
//Read and write data
readBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.readBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        //1. Android read and write services
        // serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        // characteristicId:'00002a38-0000-1000-8000-00805f9b34fb',
        success: res => {
          my.alert({ content: JSON.stringify(res) });
        },
        fail: error => {
          my.alert({ content: 'Failed to read and write' + JSON.stringify(error) });
        },
      });
    },
  });
},
writeBLECharacteristicValue() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.writeBLECharacteristicValue({
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.charid,
        //Android write service
        //serviceId:'0000180d-0000-1000-8000-00805f9b34fb',
        //characteristicId:'00002a39-0000-1000-8000-00805f9b34fb',
        value: 'ABCD',
        success: res => {
          my.alert({ content: 'Data written successfully!' });
        },
        fail: error => {
          my.alert({ content: JSON.stringify(error) });
        },
      });
    },
  });
},
notifyBLECharacteristicValueChange() {
  my.getConnectedBluetoothDevices({
    success: res => {
      if (res.devices.length === 0) {
        my.alert({ content: 'No connected devices' });
        return;
      }
      this.setData({
        devid: res.devices[0].deviceId,
      });
      my.notifyBLECharacteristicValueChange({
        state: true,
        deviceId: this.data.devid,
        serviceId: this.data.serid,
        characteristicId: this.data.notifyId,
        success: () => {
          //Listen to characteristic change events
          my.onBLECharacteristicValueChange({
            success: res => {
              //  my.alert({content: 'Characteristic change:'+JSON.stringify(res)});
              my.alert({ content: 'Obtain the response data = ' + res.value });
            },
          });
          my.alert({ content: 'Listened successfully' });
        },
        fail: error => {
          my.alert({ content: 'Failed to listen' + JSON.stringify(error) });
        },
      });
```

```
      },
    });
  },
  offBLECharacteristicValueChange() {
    my.offBLECharacteristicValueChange();
  },
  //Other events
  bluetoothAdapterStateChange() {
    my.onBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
  },
  onBluetoothAdapterStateChange() {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Bluetooth status change: ' + JSON.stringify(res) });
    }
  },
  offBluetoothAdapterStateChange() {
    my.offBluetoothAdapterStateChange(this.getBind('onBluetoothAdapterStateChange'));
  },
  getBind(name) {
    if (!this[`bind${name}`]) {
      this[`bind${name}`] = this[name].bind(this);
    }
    return this[`bind${name}`];
  },
  BLEConnectionStateChanged() {
    my.onBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onBLEConnectionStateChanged(res) {
    if (res.error) {
      my.alert({ content: JSON.stringify(error) });
    } else {
      my.alert({ content: 'Connection status change: ' + JSON.stringify(res) });
    }
  },
  offBLEConnectionStateChanged() {
    my.offBLEConnectionStateChanged(this.getBind('onBLEConnectionStateChanged'));
  },
  onUnload() {
    this.offBLEConnectionStateChanged();
    this.offBLECharacteristicValueChange();
    this.offBluetoothAdapterStateChange();
    this.closeBluetoothAdapter();
  },
});
```

my.offBluetoothAdapterStateChange

You can call this interface to stop listening to the native Bluetooth status change events.

Code sample

```
my.offBluetoothAdapterStateChange();
```

Whether to pass callback value or not

- If the callback value is not passed, the callbacks of all events will be removed. The sample code is as follows:

```
my.offBluetoothAdapterStateChange();
```

- If the callback value is passed, only the corresponding callback is removed. The sample code is as follows:

```
my.offBluetoothAdapterStateChange(this.callback);
```

Instructions

- It is recommended that you call the `off` method to close existing event listening before you call the `on` method to listen events to prevent the situation where multiple event listening causes multiple callbacks of an event.

1.10.9.18. Bluetooth API error codes

List of Bluetooth API error codes

Error code	Error message	Solution
10000	The Bluetooth adapter is not initialized.	Call my.openBluetoothAdapter for initialization.
10001	The Bluetooth adapter is not available. Check whether BLE is supported in your device and enable the function if it's supported.	
10002	Device not found	Check the service ID and make sure peripheral broadcast of the target device is enabled.

Error code	Error message	Solution
10003	Connection failed	Check the device ID and make sure peripheral broadcast of the target device is enabled.
10004	Service not found	Check the service ID and make sure the service is available for target devices.
10005	Characteristic not found	Use a correct characteristic ID and make sure the characteristic is enabled for the service.
10006	Connection disconnected	Try to connect the Bluetooth again.
10007	Operation not supported	Check the read, write and notify functions of the current characteristic.
10008	System error	An unknown system error.
10009	BLE is not supported for the Android system with version lower than 4.3.	Remind users that BLE is not supported in the current Android system.
10010	Descriptor not found	Use a correct service ID and characteristic ID.
10011	Invalid device ID, or the device ID is empty	Use a correct device ID.
10012	Invalid service ID, or the service ID is empty	Use a correct service ID.
10013	Invalid characteristic ID, or the characteristic ID is empty	Use a correct characteristic ID.
10014	Data is empty or in incorrect format	Use correct written data or HEX conversion.
10015	Timeout	Try again.
10016	Some parameters missed	Check the parameters and try again.
10017	Failed to write characteristic	Make sure writing is supported for peripheral characteristic, and the Bluetooth is connected.
10018	Failed to read characteristic	Make sure reading is supported for peripheral characteristic, and the Bluetooth is connected.

1.10.9.19. Bluetooth API FAQ

No. Calling this API returns the value of what you write.

Does the API call of my. WriteBLECharacteristicValue return an empty value?

No. Calling this API returns the value of what you write.

Why is listening not supported with the API call of my.onBLECharacteristicValueChange? Is it required to write before listening?

Yes. To call this API, you first need to write before you can listen. In order to prevent multiple callbacks of an event caused by multiple registered event listeners, it is recommended to call off method to listen to an event and close the previous event listener, before you call on method.

Why is the error code 10014 returned after calling my. Why writeBLECharacteristicValue?

The error code 10014 means the data you send are either empty or incorrectly formatted. It is suggested to check for errors in the written data or HEX conversion.

Can the hexadecimal array be used in the process of calling my.writeBLECharacteristicValue to write characteristic value?

No. The characteristic values you write are hexadecimal strings, which are limited to 20 bytes.

What is the deviceId format for Android and iOS devices?

- The Android device gets the MAC address for Bluetooth, for example: `11:22:33:44:55:66`.
- The iOS device gets the UUID of Bluetooth, for example: `00000000-0000-0000-0000-000000000000`.

Why no devices are discovered by calling my.startBluetoothDevicesDiscovery?

Please make sure that the device is discoverable. If the API is passed to services, make sure that the discoverable content of the device contains the UUID of the service.

How to resolve device connection failure?

Please make sure the correct `deviceId` is transmitted with strong signals. If the signal is weak, a device connection failure may occur.

How to resolve write/read data failure?

- `DeviceId`, `serviceId`, and `characteristicId` are transmitted in the correct format.
- `DeviceId` is connected. You can call the API `my.OnBLEConnectionStateChanged` to listen to the connection state changes, and call the API `my.GetConnectedBluetoothDevices` to check for devices that are connected.
- Write a method in the connected state.
- Check and make sure characteristicId belongs to this Service.
- The characteristic supports write/read.

How to receive data notifications?

- Make sure that the API `my.notifyBLECharacteristicValueChange` has been called with correct parameters.
- Notify or indicate features are supported in the transmitted `characteristicID`.
- Make sure the hardware is notified.
- Follow the basic flow, i.e. call the API `my.notifyBLECharacteristicValueChange` once you're connected.

Why are event callbacks called multiple times?

The same event was listened due to multiple anonymous function registrations. It is suggested to call `off` method to disable the previous event listener before you call the `on` method to listen to event.

1.10.10. Data security

my.rsa

> **Note** This interface is only supported by mPaaS 10.1.32 and later versions.

The interface is for asymmetric encryption.

You need to complete encryption and decryption in the client-side and the service-side separately with the private key in the service-side. The operation of putting the private key in the client-side will lead to security problems.

Parameters

Name	Type	Required	Description
action	String	Required	Use RSA to encrypt or decrypt. `encrypt` means encryption and `decrypt` means decryption.
text	String	Required	If you need to process the text, encrypt it as original text and decrypt it in Base64 encoding format.
key	String	Required	RSA private key. Use public key for encryption and private key for decryption.
success	Function	No	The callback function that indicates a successful call.
fail	Function	No	The callback function that indicates a failed call.
complete	Function	No	The callback function that indicates the call is completed (this will be executed regardless of whether the call succeeds or fails).

Return value upon successful callback

Name	Type	Description
text	String	Encrypt the processed text in Base64 encoding format, and decrypt as original text.

Error code

Error code	Description	Solutions
10	Parameter error	Check if the parameter is correct.
11	key error	Check if the key is correct.

Sample code

- Encryption and decryption in the client:

```
Page({
 data: {
   inputValue: '',
   outputValue: '',
 },
 onInput: function (e) {
   this.setData({ inputValue: e.detail.value });
 },
 onEncrypt: function () {
   my.rsa({
     action: 'encrypt',
     text: this.data.outputValue,
     //Set public key
     key: 'MIGfMA0GCSqXXXXXXAQUAA4GNADCBiQKBgQDKmi0dUSVQ04hL6GZGPMFK8+d6\n' +
     'GzulagP27qSUBYxIJfE04KT+OHVeFFb6XXXXXXea5mkmZrIgp022zZXXXXXXNM62\n' +
     '3ouBXXXsfm2ekey8PpQxfXaj8lhM9t8rJlXXXXXXs8Qp7Q5/uYrowQbT9m6t7BFK\n' +
     '3egOO2xOKzLpYSqfbQIDAQAB',
     success: (result) => {
       this.setData({ outputValue: result.text });
     },
     fail(e) {
       my.alert({
         content: e.errorMessage || e.error,
       });
     },
   });
 },
 onDecrypt: function () {
   my.rsa({
     action: 'decrypt',
     text: this.data.inputValue,
     //Set private key
     key: 'MIICdwIXXXXXXgkqhkiG9w0BAQEFAASCAmEwggJdAgEAAoGBAMqaLR1RJVDTiEvo\n' +
     'ZkY8wUrz53obO6VqA/bupJQFjEgl8TTgpP44dVXXXXXX7ydYN5rmaSZmsiCnTbbN\n' +
     'lUOB1Y80zrbeXXXXXXx+bZ6R7Lw+lDF9dqPyWEz23ysmULgURzSzxCntDn+5iujB\n' +
     'BtP2bq3sEUrd6A47bE4rMulhKp9tAgMBAAECgYBjsfRLXXXXXX9hou1Y2KKg+F5K\n' +
     'ZsY2AnIK+6l+XXXXXXAx7e0ir7OJZObb2eyn5rAOCB1r6RL0IH+XXXXXXZANNG9g\n' +
     'pXvRgcZzFY0oqdMZDuSJjpMTj7OXXXXXXGncBfvjAg0zdt9QGAG1at9Jr3i0Xr4X\n' +
     '6WrFhtfVlmQUY1VsoQJBAPK2Qj/ClkZNtrSDfoXXXXXXLcNICqFIIGkNQ+XeuTwl\n' +
     '+Gq4USTyaTOEe68MHluiciQ+QKvRAUd4E1zeZRZ02ikCQQDVscINBPTtTJt1JfAo\n' +
     'wRfTzA0Lvgig136xLLeQXREcgq1lzgkf+tGyUGYoy9BXsV0mOuYAT9ldja4jhJeq\n' +
     'cEulAkEAuSJ5KjV9dyb0XXXXXXC8d8o5KAodwaRIxJkPv5nCZbT45j6t9qbJxDg8\n' +
     'N+vghDlHI4owvl5wwVlAO8iQBy8e8QJBAJe9CVXFV0XJR/XXXXXX66FxGzJjVi0f\n' +
     '185nOXXXXXXCHG5VxxT2PUCo5mHBl8ctIj+rQvalvGs515VQ6YEVDCECQE3S0AU2\n' +
     'BKyFVNtTpPiTyRUWqig4EbSXwjXdr8iBBJDLsMpdWsq7DCwv/ToBoLgXXXXXXc5/\n5DChU8P30EjOiEo=',
     success: (result) => {
       this.setData({ outputValue: result.text });
     },
     fail(e) {
       my.alert({
         content: e.errorMessage || e.error,
       });
     },
   });
 },
});
```

- Encryption and decryption in the server:

```
private static void testJieMi(String miwen, String privateKeyStr) {
    //Switch the private key after Base64 encoding to PrivateKey object
    //Use Base64 to decode the encrypted content
    //Use the private key to decrypt
    try {
        PrivateKey privateKey = RSAUtil.string2PrivateKey(privateKeyStr);
        byte[] base642Byte = RSAUtil.base642Byte(miwen);
        byte[] privateDecrypt = RSAUtil.privateDecrypt(base642Byte, privateKey);
        System.out.println ("Decrypted plaintext:" + new String(privateDecrypt));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
private static void testJiaMi(String message, String publicKeyStr) {
    try {
        //Switch the public key after Base64 encoding to PublicKey object
        PublicKey publicKey = RSAUtil.string2PublicKey(publicKeyStr);
        //Use the public key to encrypt
        byte[] publicEncrypt = RSAUtil.publicEncrypt(message.getBytes(), publicKey);
        //Use Base64 to encode the encrypted content
        String byte2Base64 = RSAUtil.byte2Base64(publicEncrypt);
        System.out.println(byte2Base64);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

1.10.11. Share

This feature is only supported in 10.1.60 or later baseline versions. You can implement the sharing feature in the client-side with native code.

onShareAppMessage

Customize the `onShareAppMessage` function on `page` to set the information for sharing on this page.

- By default, **Share** button shows in the upper-right corner of the menu on each Page. If you rewrite `onShareAppMessage` function, you only customize the content for sharing.
- You can click **Share** button to call this operation.
- This event needs to return an Object used to customize the content for sharing.

Parameters

Parameter	Type	Instruction	The earliest version
from	String	Trigger source: • button: is triggered when a user clicks the Share button on the page. • menu: is triggered when a user clicks the Share button in the upper-right corner of the page.	1.10.0
target	Object	If the `from` value is button, `target` is the button to trigger the sharing feature for this time, otherwise the button will be undefined.	1.10.0
webViewUrl	String	When the page contains web-view component, the URL of web-view will be returned.	1.6.0

Return value

Name	Type	Required	Description	The earliest version
title	String	Yes	Customize the title for sharing.	None
desc	String	No	Customize description for sharing: Because the maximum length for text shared in Weibo is 140 characters, you need to enter the description within the limit.	None
path	String	Yes	Customize the path of the page for sharing. You can get the custom parameters of `path` by the onLoad method in the life cycle of the mini program. Follow the transmission rules for http get to transmit parameters. When you implement the scenario for sharing in the client-side, the name of this field is **page** in native code.	None
imageUrl	String	No	Customize the small icon for sharing. This feature supports the network image path, apFilePath, and relative path.	1.4.0
bgImgUrl	String	No	Customize the big image for sharing preview. The recommended image size is 750 x 825. This feature supports the network image path, apFilePath, and relative path.	1.9.0
success	Function	No	Callback after the sharing completes.	1.4.0
fail	Function	No	Callback after the sharing fails.	1.4.0

Sample code

```
Page({
  onShareAppMessage() {
    return {
      title: 'Applet examples'.
      desc: 'The official demo for the mini program, which shows the supported operation capabilities and components.'
      path: 'page/component/component-pages/view/view?param=123'
    };
  },
});
```

Initiate the sharing feature within the page

Note: This feature is supported on base library version 1.1.0 or later versions.

You can set the attribute `open-type="share"` for button components. This will trigger the `Page.onShareAppMessage()` event after a user clicks the button, and then triggers the sharing pane. If this event is not defined on the current page, the click operation will be invalid. Relevant components: button.

App.onShareAppMessage

In `App(Object)` constructor function, you can set `onShareAppMessage` for global sharing configuration. When you call the sharing feature, global sharing configuration will be applied if you have not set the page-level sharing configuration.

my.hideShareMenu(Object)

> **Note** This feature is only supportedin 1.7.0 or later base library versions. You need to perform Compatibility operation for earlier versions.

Hide **Share** button.

Parameters

Name	Type	Required	Instruction
success	Function	No	The callback function for successful operations.
fail	Function	No	The callback function for failed operations.
complete	Function	No	The callback function for ended operations. The callback function will be executed for both successful and failed operations.

Sample code

```
my.hideShareMenu();
```

Implement extensions in the client

Due to the high customization degree of implementing the sharing feature, the sharing feature on the native mini program is not available at the moment. Thus, you need to implement the feature yourself.

Implementation

1. When you initiate the sharing feature in the mini program, this feature will call JSAPI in `shareTinyAppMsg`. And this feature will also transmit the parameters of object returned by the `onShareAppMessage` method in JS to the client-side.
2. You need to implement the processing of `shareTinyAppMsg` events by customizing the JavaScript (JS) plug-in in the client-side. For the implementation of the JS plug-in, see Android custom JSAPI and iOS custom JSAPI respectively.

Sample code for Android

Note: By default, the Share button on the option menu in the upper-right corner of the mini program is hidden. You can turn on the configuration container switch to show the button. See Container configuration for more details.

- Initiate the following sharing example in the mini program code:

```
  Page({
    data: {
      height: 0,
      title: 'Thanks for your experience. \nIf you have any suggestions, please feel free to give your feedback.'
    },
    onLoad() {
      const { windowHeight, windowWidth, pixelRatio } = my.getSystemInfoSync();
      this.setData({
        height: windowHeight * 750 / windowWidth
      })
    },
    onShareAppMessage() {
      return {
        title: 'Result page',
        desc: 'Result is obtained successfully',
        myprop: 'hello', //Custom parameter. If the field in the document does not meet your requirement, you can add the field yourself. The field
will be transmitted to the client-side.
        path: 'pages/result/result'
      }
    }
  });
```

- See the following HTML5 plug-in example in the client-side:

```
package com.mpaas.demo.nebula;

import com.alibaba.fastjson.JSONObject;
import com.alipay.mobile.antui.dialog.AUNoticeDialog;
import com.alipay.mobile.h5container.api.H5BridgeContext;
import com.alipay.mobile.h5container.api.H5Event;
import com.alipay.mobile.h5container.api.H5EventFilter;
import com.alipay.mobile.h5container.api.H5SimplePlugin;

public class ShareTinyMsgPlugin extends H5SimplePlugin {

    private static final String ACTION_SHARE = "shareTinyAppMsg";

    @Override
    public void onPrepare(H5EventFilter filter) {
        super.onPrepare(filter);
        filter.addAction(ACTION_SHARE);
    }

    @Override
    public boolean handleEvent(H5Event event, final H5BridgeContext context) {
        String action = event.getAction();
        if (ACTION_SHARE.equals(action)) {
            JSONObject param = event.getParam();
            String title = param.getString("title");
            String desc = param.getString("desc");
            String myprop = param.getString("myprop");
            String path = param.getString("page");
            String appId = event.getH5page().getParams().getString("appId");

            // You can call sharing components here to implement further features
            String message = "AppID: " + appId + "\n"
                    + "title: " + title + "\n"
                    + "desc: " + desc + "\n"
                    + "myprop: " + myprop + "\n"
                    + "path: " + path + "\n";

            AUNoticeDialog dialog = new AUNoticeDialog(event.getActivity(),
                    "Sharing result", message, "Sharing successfully", "Sharing failed");
            dialog.setPositiveListener(new AUNoticeDialog.OnClickPositiveListener() {
                @Override
                public void onClick() {
                    JSONObject result = new JSONObject();
                    result.put("success", true);
                    context.sendBridgeResult(result);
                }
            });
            dialog.setNegativeListener(new AUNoticeDialog.OnClickNegativeListener() {
                @Override
                public void onClick() {
                    context.sendError(11, "Sharing failed");
                }
            });
            dialog.show();
            //
            return true;
        }
        return false;
    }
}
```

iOS code example

- By default, the Share button in the upper-right corner of the mini program is hidden. When the container initializes, you can set the following API operations to show the Share button:

> **Note** You need to manually import the corresponding header file `#import <TinyappService/TASUtils.h>`.

```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

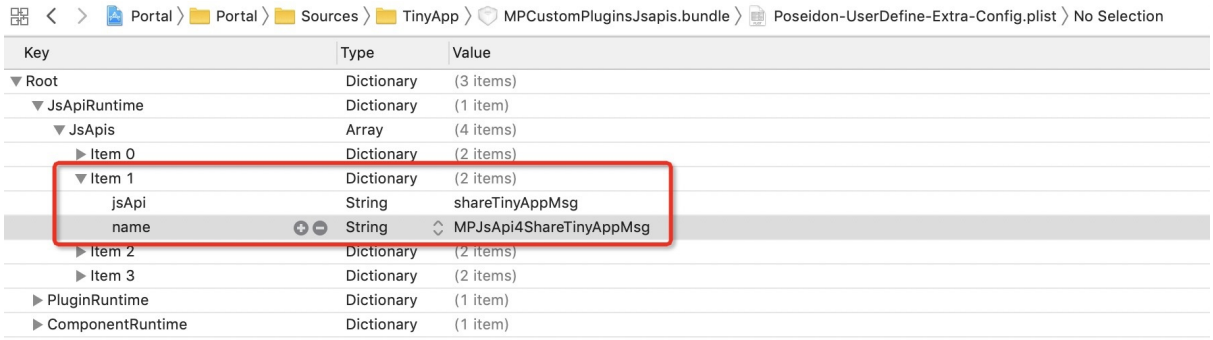
    ...

    [TASUtils sharedInstance].shoulShowSettingMenu = YES;

    ...

}
```

- Custom the implementation of `shareTinyAppMsg` JsApi to accept parameters transmitted from the mini program page:



- In `MPJsApi4ShareTinyAppMsg` implementation type, you can get the parameters shared on the mini program page to process business. See the following code example:

```
#import <NebulaPoseidon/NebulaPoseidon.h>
@interface MPJsApi4ShareTinyAppMsg : PSDJsApiHandler

@end

#import "MPJsApi4ShareTinyAppMsg.h"
#import <MessageUI/MessageUI.h>

@interface MPJsApi4ShareTinyAppMsg()<APSKLaunchpadDelegate>

@property(nonatomic, strong) NSString *shareUrlString;

@end

@implementation MPJsApi4ShareTinyAppMsg

- (void)handler:(NSDictionary *)data context:(PSDContext *)context callback:(PSDJsApiResponseCallbackBlock)callback
{
    [super handler:data context:context callback:callback];

    NSString * appId = context.currentSession.createParam.expandParams[@"appId"];
    NSString * page = data[@"page"]?:@"";
    NSString * title = data[@"title"]?:@"";
    NSString * desc = data[@"desc"]?:@"";

    // Concatenate the content for sharing, and call sharing SDK
    self.shareUrlString = [NSString stringWithFormat:@"http://appId=%@&page=%@&title=%@&desc=desc", appId, page, title, desc];
    [self openPannel];
}

- (void)openPannel {
    NSArray *channelArr = @[kAPSKChannelWeibo, kAPSKChannelWeixin, kAPSKChannelWeixinTimeLine, kAPSKChannelSMS, kAPSKChannelQQ, kAPSKChannelQQZon
e, kAPSKChannelDingTalkSession, kAPSKChannelALPContact, kAPSKChannelALPTimeLine];
    APSKLaunchpad *launchPad = [[APSKLaunchpad alloc] initWithChannels:channelArr sort:NO];
    launchPad.tag = 1000;
    launchPad.delegate = self;

    [launchPad showForView:[[UIApplication sharedApplication] keyWindow] animated:YES];
}

#pragma mark - APSKLaunchpadDelegate
- (void)sharingLaunchpad:(APSKLaunchpad *)launchpad didSelectChannel:(NSString *)channelName {
    [self shareWithChannel:channelName tag:launchpad.tag];
    [launchpad dismissAnimated:YES];
}

- (void)shareWithChannel:(NSString *)channelName tag:(NSInteger)tag{
    APSKMessage *message = [[APSKMessage alloc] init];
    message.contentType = @"url";//The types are "text", "image", and "url".
    message.content = [NSURL URLWithString:self.shareUrlString];
    message.icon = [UIImage imageNamed:@"MPShareKit.bundle/Icon_Laiwang@2x.png"];
    message.title = @"Here is the webpage title";
    message.desc = @"Here is the description information";
    APSKClient *client = [[APSKClient alloc] init];
    client.disableToastDisplay = YES;

    [client shareMessage:message toChannel:channelName completionBlock:^(NSError *error, NSDictionary *userInfo) {
        if(! error) {// Successful
            [AUToast presentToastWithin:[[UIApplication sharedApplication] keyWindow]
                                withIcon:AUToastIconSuccess
                                    text:@"Sharing successfully"
                                duration:2
                                  logTag:@"demo"];
        } else {// Failed
            NSString *desc = error.localizedFailureReason.length > 0 ? error.localizedFailureReason : @"Sharing failed";
            [AUToast presentToastWithin:[[UIApplication sharedApplication] keyWindow]
                                withIcon:AUToastIconNone
                                    text:desc
                                duration:2
                                  logTag:@"demo"];
            NSLog(@"error = %@", error);
        }
    }];
}

@end
```

1.10.12. Mini program version type

This interface is used to obtain the current running version of the Mini program.

my.getRunScene

Note:

- The basic library 1.10.0 and above versions support this interface. The lower versions need to complete compatibility process. For the operation, see Mini program

basic library description.

- mPaaS 1.10.60 and above versions support this interface.

This interface is used to obtain the current running version of the Mini program.

Success return value

Name	Type	Description
envVersion	String	The current running version of the Mini program. Enumerated types include: develop (development version), trial (experience version), release (release version).

Error code

Return code	Meaning
3	Unknown error occurred.

Code sample

```
my.getRunScene({
  success(result) {
    my.alert({
     title: 'Mini program version',
     content:`${result.envVersion}`
    });
  },
})
```

1.10.13. Custom analysis

my.reportAnalytics

Customize reporting interface for analytics data. Before using, you need to create a new event in the event management of the MINI Management Platform and configure the event name and field.

Input parameter

Name	Type	Required	Description
eventName	String	Yes	Custom event name, which you need to apply for
data	Object	Yes	Data reported

Code example

```
my.reportAnalytics('purchase', {
  status: 200,
  reason: 'ok'
});
```

1.10.14. Custom API

If the existing mini program APIs cannot meet your requirements, you can extend customize APIs by yourself. Mini program APIs use the JSAPI plug-in mechanism of the HTML5 container, which means that you can customize APIs according to the plug-in mechanism provided by the HTML5 container, and your mini program can directly call JSAPIs that you have written.

If the existing mini program APIs cannot meet your requirements, you can extend customize APIs by yourself. Mini program APIs use the JSAPI plug-in mechanism of the HTML5 container, which means that you can customize APIs according to the plug-in mechanism provided by the HTML5 container, and your mini program can directly call JSAPIs that you have written.

Custom APIs

To customize APIs, see the Custom JSAPIs topic of the HTML5 container document:

- Custom JSAPIs for Android
- Custom JSAPIs for iOS

Note: The custom mini program APIs only support calling native components from the page, but do not allow the native components to proactively send events to the page.

Call API in the mini program

Use the following method in the mini program to call a customized API:

```
my.call(API, param, callback)
```

In which:

- API: the name of the custom API.
- param: the parameters that are used to call the API.

- callback: the callback function that is executed by the API.

Take calling the `rpc` method as an example, the code sample is as follows:

```
my.call('rpc', {
  operationType: 'com.test.mb1001',
  requestData: [{
    tranCode: 'MB1001',
    customerType: 0,
    customerId: 0,
    UnitType: '7A238BD3-A90B-4458-885E-129230BCF7F1',
    sessionId: 'zzzzzzzzzzzzzzz',
    serverIP: 'zzzzzzzzzzzzzzz',
    mobileNo: username,
    password,
    optionFlag: 3,
  }]
}, (res) => {
    // do your business here.
})
```

To understand the similarities and differences between mini program calls and HTML5 calls, see the Call the RPC API topic of the HTML5 container document.

1.10.15. Mini program jumping

my.navigateToMiniProgram (Object)

This API is supported in mPaaS 10.1.60 and later versions.

This API is used to jump to other Mini programs.

Code sample

```
my.navigateToMiniProgram({
      appId: 'xxxx',
      path: 'page/index/index',
      extraData:{
        "data1":"test"
      },
      success: (res) => {
        console.log(JSON.stringify(res))
      },
      fail: (res) => {
        console.log(JSON.stringify(res))
      }
    });
```

Object parameters

Name	Type	Required	Description
appId	String	Yes	The appId of the target Mini program to jump to.
path	String	No	The path of the page to open. Open the home page if it is null.
extraData	Object	No	The data that needs to be passed to the target Mini program. The target Mini program can get this data in `App.onLaunch()` and `App.onShow()`.
success	Function	No	The callback function of calling succeeded.
fail	Function	No	The callback function of calling failed.
complete	Function	No	The callback function of calling ended (Execute regardless of the success or failure of the call).

FAQ

- Q: How does the target mini program obtain the data passed by the extraData parameter of `my.navigateToMiniProgram`? Can I add multiple parameters to extraData? What symbol is used as a separator among multiple custom parameters?

  A: The above questions are explained as follows

  - The target mini program can obtain extraData data through `App.onLaunch()` and `App.onShow()`.
  - Multiple parameters can be added to extraData, and all custom parameters are passed to the target mini program through extraData.-Use `&` to separate multiple custom parameters.

- Q: How can the mini program jump to the “Add favorite for bonus” page?

  A: You can refer to the following code.

```
my.navigateToMiniProgram({
   appId: '2018122562686742', // The appid of the "Add favorite for bonus" mini program, it is a fixed value, do not modify it
   path: 'pages/index/index?originAppId=2017082508366123&newUserTemplate=20190130000000119123', // Link address and parameter
   success: (res) => {
     // Jumping succeeded
     my.alert({ content: 'success' });
   },
   fail: (error) => {
     // Jumping failed
     my.alert({ content: 'fail' });
   }
 });
```

my.navigateBackMiniProgram (Object)

This API is supported in mPaaS 10.1.60 and later versions.

This API is used to go back to the previous Mini program. The calling successes only when another Mini program jumps to the current program.

Object parameters

Name	Type	Required	Description
extraData	Object	No	The data that needs to be passed to the target Mini program. The target Mini program can get this data in `App.onLaunch()` and `App.onShow()`.
success	Function	No	The callback function of calling succeeded.
fail	Function	No	The callback function of calling failed.
complete	Function	No	The callback function of calling ended (Execute regardless of the success or failure of the call).

Code sample

```
my.navigateBackMiniProgram({
  extraData:{
       "data1":"test"
    },
  success: (res) => {
     console.log(JSON.stringify(res))
    },
    fail: (res) => {
     console.log(JSON.stringify(res))
    }
});
```

1.10.16. Webview component control

You can provide Mini program the ability to send messages to `web-view` by creating `webviewContext`.

Note:

- The basic library 1.8.0 and above versions support this interface. The lower versions need to complete compatibility process. For the operation, see Mini program basic library description.
- mPaaS 1.10.32 and above versions support this interface.

my.createWebViewContext(webviewId)

This interface is used to create and return `web-view` context `webViewContext` object.

Input parameters

Name	Type	Required	Description
webviewId	String	Yes	The ID attribute corresponding to the `web-view` to be created.

webViewContext

`webViewContext` consists of a `webviewId` and a `web-view` component.

The method of `webViewContext` object is as follows:

Method	Parameter	Description	Compatibility

Method	Parameter	Description	Compatibility
postMessage	Object	The Mini program sends message to `web-view` component. Whit the help of `my.postMessage` in `web-view.js`, the two-way communication between Mini program and `web-view` web pages is possible.	1.8.0

Code sample

```
<view>
  <web-view id="web-view-1" src="..." onMessage="onMessage"></web-view>
</view>
```

```
Page({
  onLoad() {
    this.webViewContext = my.createWebViewContext('web-view-1');
  },
  // Receive the message from HTML5
  onMessage(e) {
      console.log(e); //{'sendToMiniProgram': '0'}
    // Send message to HTML5
    this.webViewContext.postMessage({'sendToWebView': '1'});
  }
})
```

```
// In the JS code of HTML5, my.onMessage need to be firstly defined to receive the messages from Mini program.
my.onMessage = function(e) {
    console.log(e); //{'sendToWebView': '1'}
}
// HTML5 sends message to Mini program
my.postMessage({'sendToMiniProgram': '0'});
```

Note：In the two-way communication procedure, HTML5 firstly sends message to Mini program. After receiving the message, Mini program sends message to HTML5.

1.11. Design guide

1.11.1. Principles

1.11.1.1. Clarity

When designing the mPaaS MINI, only if your product is simple and easy to use can it be used by more users. The more users, the wider the market and the greater the benefits.

When designing the mPaaS MINI, only if your product is simple and easy to use can it be used by more users. The more users, the wider the market and the greater the benefits.

One page only does one thing

A page of the app can display very limited information. The mobile phone environment is very unstable, and often affected by various factors, such as people's behaviors (such as walking, riding, etc.), network signals, etc., which further limits the amount of information on the page. It is best for a page to highlight its key point so that users can quickly understand and complete the task. Avoid other distractions on the page that are not related to the user's decision-making and operation.

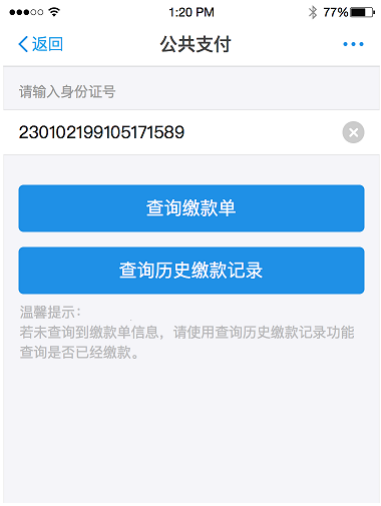
The mPaaS MINI services are mostly task-oriented and are designed to help users achieve certain objectives, such as transfer, payment, etc. This principle is especially important in the task-oriented page, because we want users to focus on and quickly complete the current task.



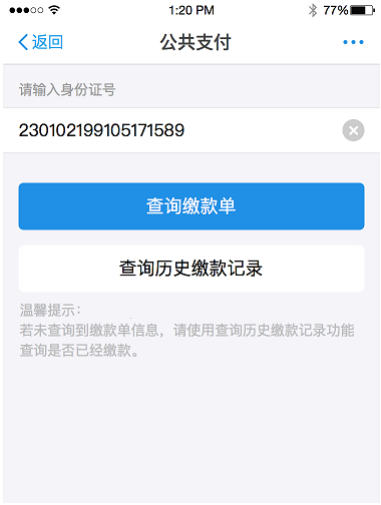
Example

If multiple main action buttons are set in one page and the primary and secondary operations are not distinguished, the user may feel confused and doesn't know how to choose.

Wrong example:



Correct example:



Delete and hide as appropriate

People's ability to process information, learn rules, and memorize details is limited. In practice, people may also face various interruptions and distractions, which further limits their cognitive ability. Like a roadblock, too many minor details in the interface will increase the cognitive burden on users, and reduce their efficiency.

It's better to remove unnecessary features, excess options, redundant text and fancy decorations as well as hide non-core features, thus reducing the cognitive burden on users, and allowing users to focus on what they really want to do.

Clear navigation

Navigation is the most critical factor in ensuring that users don't get lost when browsing through web pages. Navigation needs to tell the users where they are, where to go, and how to go back.

First of all, the navigation bar should be provided on all pages of the mPaaS MINI, so as to resolve problems like where we are and how to go back, providing users with unified experience and interaction awareness in the MINI, without adding learning cost or changing habits when switching between pages.





1.11.1.2. Efficiency

As a service provider, we should help users improve efficiency, create value, and provide a thoughtful experience. As the pace of life is getting faster and faster, high efficiency is a must-have quality of a product, and user caring will undoubtedly make your products stand out from many services, resulting in users rely more and more on your products.

As a service provider, we should help users improve efficiency, create value, and provide a thoughtful experience. As the pace of life is getting faster and faster, high efficiency is a must-have quality of a product, and user caring will undoubtedly make your products stand out from many services, resulting in users rely more and more on your products.

To design an efficient and thoughtful product, you need to consider the following points:

1-second time interval

Reduce waiting time, and keep stable and fast, to help you retain users. Research shows that the average waiting time that users can tolerate is between 6 and 8 seconds. That is to say, 8 seconds is the upper limit. If the page opens too slowly and you need to wait for more than 8 seconds, most users will leave you.



Divert attention

Diverting attention is a typical way to reduce the impact of waiting. In practice, the strategy of diverting attention is very common. For example, some restaurants provide free snacks and leisure services to divert customers' attention when they are waiting to eat, allowing customers to enjoy queuing and reduce anxiety during waiting.

This approach also works in application design.

One click

There are often some unnecessary clicks in the process of using the product. For the user, these unnecessary operations are extra work. Such extra work does not directly achieve the user's goals and wastes the user's time and effect. Eliminating extra work can increase operational efficiency and improve product availability. Interaction designers should be highly sensitive to the extra work in the product in order to design the product more efficiently.

Take the phone top up MINI as an example:



Before Alipay 9.2, the phone top up requires four taps from selecting amount to making payment:

1. Tap the amount to invoke the selector.
2. Swipe to select the amount.
3. Tap **Finish** to close the selector.
4. Tap **Top Up Now** to enter into the payment page.



9.2 or above: The top up amounts are tiled and displayed to the user, and the user only needs to tap once to select the top up amount and enter into the payment page.

Reduce input

The keyboard area of the mobile phone is small and dense, and input difficulties are also likely to cause input errors. Therefore, user input should be minimized when designing the mobile page, and the user input experience can be improved by using existing interfaces or other easy-to-operate selection controls.

Example

Smartphones are equipped with a variety of smart sensors: cameras, microphones, and gyroscopes. In addition, the mPaaS MINI team also opens up various authorization interfaces such as geographic location and account information. Making full use of these interfaces can greatly reduce the manual input by users and enhance the product experience.

Case 1: Login page - Use the face recognition feature of the camera instead of inputting password.



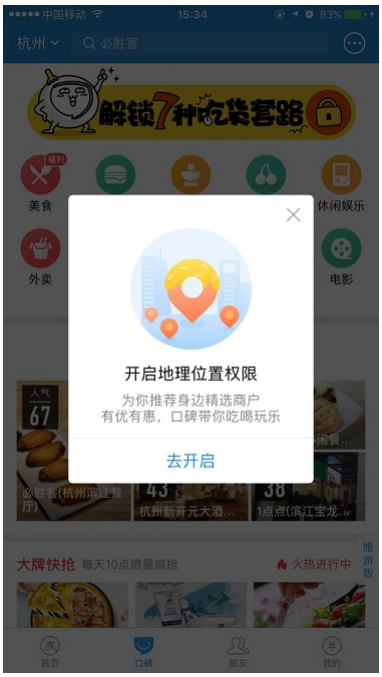
Case 2: Add bank card information page - Name, ID card number and mobile phone number can be obtained by directly calling the authentication information in the user account to avoid manual input.



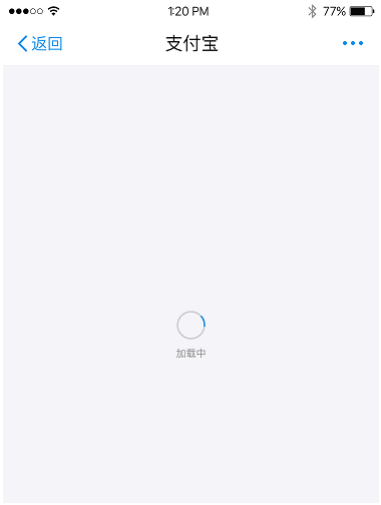
Clear feedback

Timely and appropriate feedback can tell the user what to do next, help the user make judgments and decisions, and let the user know that the system is running well. Therefore, we must provide timely and clear feedback to achieve a harmonious human-computer interaction.

Turn on the location prompt: In case the page cannot perform location service automatically, provide the user with clear feedback and inform the user that the location service should be turned on.



Page loading feedback: When a blank page is loaded, the user may not know what happened, so provide clear loading feedback to inform the user of the loading status.



1.11.1.3. Security

Under the mPaaS MINI account, there are not only user's private information, but also a lot of money. Therefore, account security is very important. To eliminate user concerns, the creation of a sense of security is very important. As a service provider, you must ensure that the services and applications you provide are secure and controllable.

Under the mPaaS MINI account, there are not only user's private information, but also a lot of money. Therefore, account security is very important. To eliminate user concerns, the creation of a sense of security is very important. As a service provider, you must ensure that the services and applications you provide are secure and controllable.

Information masking

Many applications and services require to fill in or display the user's private information, such as account name, password, mobile phone number, ID card number, bank card number, etc. Such information should be masked during display and input in order to hide a portion of information instead of completely exposing it. In this way, users will find that you are protecting their information and they will feel safe.



Input prompt

When the user is required to input sensitive information, they shall be clearly informed of the purpose of the information, thus eliminating the user's concerns.





Show authority

If the service is provided through authoritative channel and it is not widely known online, you can show users the authority that the channel already has in order to quickly win their trust and eliminate their concerns.



Operation controllable

Always keep in mind that it is the user who controls the application, not the application who controls the user, and do not summarily make decisions for the user.

Good design should be trustworthy and easy to believe. When asking the user to perform an action, try to help them understand why this operation is necessary. Each step should be described honestly and clearly in order to build trust. In addition, the trust should be accumulated and improved step by step and bit by bit.

The basis for building trust is that users have their own control.

Double confirmation

To make users feel that they are control the application themselves, some familiar and predictable interaction elements should be used for the application. At the same time, when the user performs a destructive operation, a double confirmation is provided to avoid irreparable damage.



Let the user be the master

You can advise on a range of user behaviors and warn about actions that may have serious consequences, but don't make decisions for users. A good design will strike a balance between allowing the user to make decision and avoiding unwanted results.

Users are used to taking control of their app usage experience. Some users are even beginning to pursue customized experiences and apps that meet their specific needs. Therefore, you should respect the user's right to choose, which is applicable to even the most minor option, such as switches for notification system.

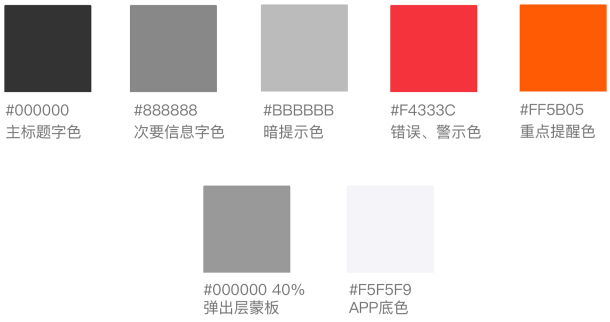


1.11.2. Visual guidelines

1.11.2.1. Color

The mPaaS MINI has a complete color usage specification. The following figure illustrates the basic color of the main text content, background, etc.

The mPaaS MINI has a complete color usage specification. The following figure illustrates the basic color of the main text content, background, etc.



1.11.2.2. Font

To ensure the versatility of the mPaaS MINI, PingFang SC is preferred as a Chinese font for both web and mobile version.

To ensure the versatility of the mPaaS MINI, PingFang SC is preferred as a Chinese font for both web and mobile version.

The font size and usage scenes are as follows:



1.11.2.3. Icon

Icon is an important part of the graphical interface, and has the characteristics such as highly condensed, quickly conveyed and ease to memorize. To make it easier for users to recognize icon information and achieve consistent icon design, the mPaaS MINI provides a unified style for icon design.

Icon is an important part of the graphical interface, and has the characteristics such as highly condensed, quickly conveyed and ease to memorize. To make it easier for users to recognize icon information and achieve consistent icon design, the mPaaS MINI provides a unified style for icon design.

Design principle

It is necessary to design a clear shape and simplify the information process. The geometric shapes and symmetrical icons can be used for design.

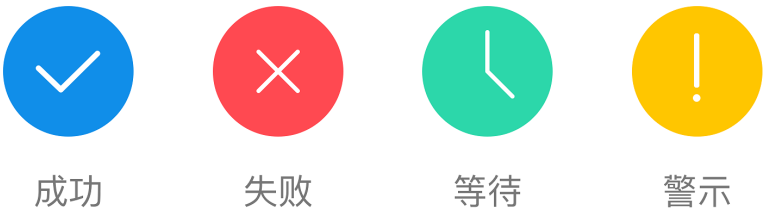
- **Design principle for system icon**

Widely use rounded corners to avoid abrupt and jagged feel. For example, if you zoom in on a 36 x 36px icon, the line structure is 3px and the fillet arc is 4px.



- **Status icon**

Mainly used for status display on result page.



1.11.3. Component guidelines

1.11.3.1. Navigation

Navigation is the most critical factor in ensuring that users don't get lost when browsing through web pages. Navigation needs to tell the users where they are, where to go, and how to go back. The mPaaS MINI provides several navigation components, including navigation bar, segmentation control, and tab bar.

Navigation is the most critical factor in ensuring that users don't get lost when browsing through web pages. Navigation needs to tell the users where they are, where to go, and how to go back. The mPaaS MINI provides several navigation components, including navigation bar, segmentation control, and tab bar.

Navigation bar

Your page is required to embed the navigation framework of mPaaS MINI before being displayed to user. The mPaaS MINI navigation bar directly inherits from the client, so you don't need to or can't customize the content within it. However, you still need to define the jump relationship between the pages so that the navigation system can work properly.

The mPaaS MINI navigation bar is divided into a navigation area, a title area, and an operation area.

- **Navigation area**

The navigation area controls the process of program page. As shown below, it appears different on iOS and Android clients:



In the navigation area, there is usually only one operation, that is, moving up one level. You can only define its contents. When the user enters into more than three levels of your page, both **Back** and **Close** buttons will be displayed; Click **Close** button will leave the current page and return to the mPaaS MINI.

- **Title area**

You can define a title for each page, the text display area of the title is fixed, and the excess length will be omitted. If you use default setting, the name of your service or app is displayed by default.

- **Operation area**

The operation area displays **More** icon by default, and tapping the icon opens an action menu. You can also customize the operation area, defining up to two icon-type action buttons or one text-type (four characters) action button.

Custom color

The mPaaS MINI navigation bar supports for custom basic background color. If satisfied with usability requirements, the selected color needs to be harmoniously matched with three sets of main navigation bar icons provided by the mPaaS MINI. The title color will be automatically adjusted to white or black to match the background color. When you choose colors, it is recommended to refer to the following color schemes:



正确



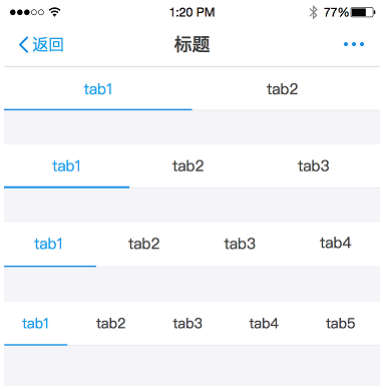
错误

Segmented control

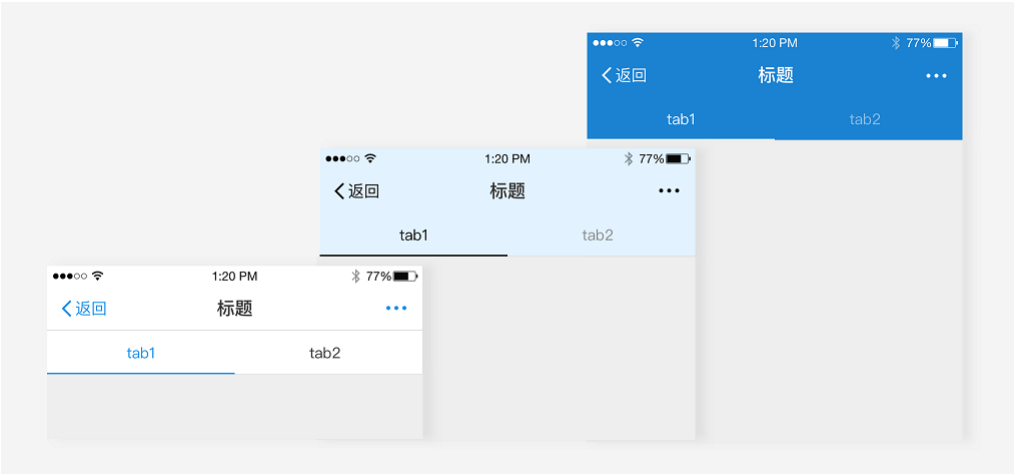
The segmented control typically appears at the top of or within the page, allowing users to quickly switch between different contents within the page.

Design principles:

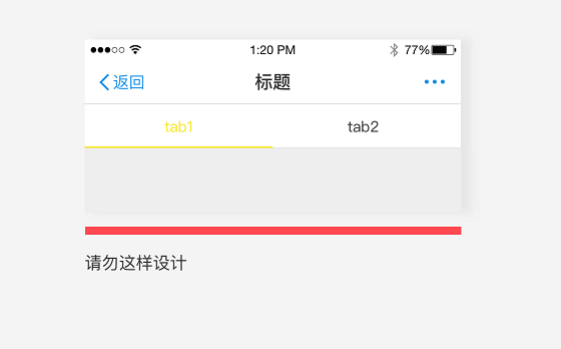
- Highlight the current option to let the users know where they are. You can set up to 5 options, and use a swipe for excess options.



- The color of top segmentation control can be customized. When choosing a custom color, pay careful attention to the usability, visibility, and operability of the page tab bar.



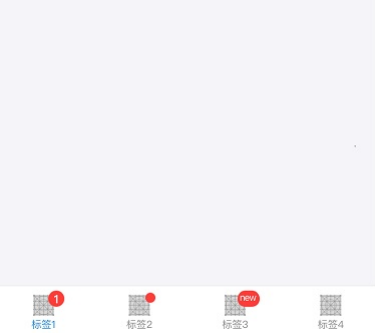
Wrong example:



Tab bar

The tab bar is used to organize the page with information structure in one level. It allows you to design a more flat application information hierarchy, which helps users explore more content.

The tab bar is at the bottom of the page, allowing users to quickly switch between different pages.



Design principles:

- Set up to 5 tabs. When you have 6 or more tabs, use the **More** tab for the last one. The **More** tab can show more navigation options.
- Custom icon and text are supported for the tab bar, but the color cannot be modified. Only the standard link color provided by the mPaaS MINI can be used.
- When switching between pages, you should switch within the current page instead of opening a new page.

1.11.3.2. Enter information

In the interaction between user and application, users often need to input, edit, and delete some information. Diversified and tailored input components can help users c

In the interaction between user and application, users often need to input, edit, and delete some information. Diversified and tailored input components can help users c

Button

The button is used to start an instant operation and submit a set of input data in the form.





Definition and principle

The button acts as the main action point in the page, guiding the user to perform corresponding main actions. The action button should be prominently displayed and ea

The buttons are divided into main button, secondary button, and auxiliary button.

- Main button: Only one main button can appear in a page, indicating the most important user conversion point.
- Secondary button: There may be multiple secondary buttons in a page, which can be used for supplementary operations under the current scene.
- Auxiliary button: An action button located on the right side of the list that guides the user to tap the list.

Visual style

Big button

The main purpose of the big button is to encourage the user to perform operation. The big button's usage specifications are as follows:

- The button text needs to be centered in all directions.
- The button height is fixed at 94px (47pt) and the fillet is 10px (5pt).

> **Note**
> Note: The main button can only appear once in a page.



Small button

The small button is used for the operation/selection of an item or option within a page and can be used multiple times. The small button's usage specifications are as fo

- The button text needs to be centered in all directions.
- The button height is fixed at 60px (30pt), the minimum width is 112px (56pt), the border thickness is 2px (1pt), and the fillet is 6px (3pt).
- The spacing between the text and the border of the button is 30px (15pt). If there is no enough space available for the text, you can extend the left and right sides.



Example

The button and page content should be displayed together to make sense.

Main button:



Auxiliary button:



Checkbox

The multi-select control allows users to select multiple elements at the same time.



Definition and principle

The multi-
select control typically appears in an editable list. After the user completes the selection, the selected element will be edited and processed simultaneously. It has two st

Text input box

The text input box is the simplest input component that allows users to enter a single line data through components such as keyboards, selectors, etc.



Definition and principle

The single-line text input box has a 15-
character limit. You can also limit the types of information that can be entered in the input box, such as Chinese, English, number, email address, and so on.

When activating different types of input boxes, it is necessary to pop up a corresponding type of keyboard: text keyboard, English keyboard, number pad, mailbox keybo

The input box is typically composed of **label area**, **input area**, and **auxiliary operation area**. The **label area** has a limit of up to 8 characters; The **input area** is confi

If the data entered is sensitive, such as password, mobile phone number, etc., data masking should be performed.

Visual style

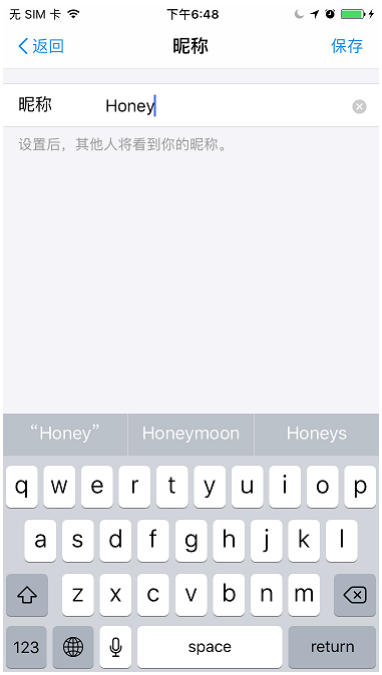
Labels, icons, and auxiliary input buttons together form a variety of input boxes.



Example

Call the keyboard based on the input data type. Both iOS and Android system have keyboards for different types of data input, which helps to improve user input efficie

Text keyboard example:



Number pad example:



Selector

The selector provides a group of preset data that allows the user to complete the input or settings by selecting the data.



Definition and principle

The selector is triggered by tapping an input box on the page. When the selector appears, the page should be covered with a semi-transparent mask to allow the user to focus on the selector.

The data in the selector preferably has certain logical relationship and meets the expectations of the user. Because the selector may not be able to display all data at th

The selector can be configured with a combination of data in multiple columns (up to four columns), but the text of the longest column cannot exceed the width limit.

Date selector

The time selector enables the user to quickly select a time ranging from year, month, day, hour, minute and second.



Radio button

Radio buttons allow the users to select one option from a range of options.



Definition and principle

Radio buttons typically appear on the right side of the list. When a tick appears, it indicates the option has been selected.

Search bar

The search bar allows users to quickly reach the desired content from a large amount of information.



Definition and principle

The search bar is typically located below the navigation bar. The touch keyboard is called when the user taps on an input field and activates it. Tap the **Cancel** button t

If the input box is displayed by default, you can provide a grey hint text (such as a keyword) to help the user input data. Below the search bar, useful labels are provided

Slide switch

A switch is a type of control that visualizes two states.

Definition and principle

The switch control can only be used in list, so switches can only appear in the list to represent two mutually exclusive options.



1.11.3.3. Display information

Properly organized and displayed information helps users better understand and find content, making your application easy to use. The mPaaS MINI provides different information display components, and you can select the appropriate components to display different types of information according to the page requirements.

Properly organized and displayed information helps users better understand and find content, making your application easy to use. The mPaaS MINI provides different information display components, and you can select the appropriate components to display different types of information according to the page requirements.

List

The list is a common form of information organization. It divides the content into a number of rows. Each row can jump to the corresponding details page to display more information.





In the list, you can use a slide to show the information out of view, and group the information.

Visual style

The list consists of a title, a subtitle, and an icon. Icon is optional based on your needs



Example

Take a simple function list page as an example. It uses a list to display various functions on a single page, and the user can find different function items by navigating through the list.

Settings page:



Personal hub page:



Notice bar

The top notice bar is inserted between the navigation bar and the page content to inform the user of important information and announcements.

Definition and principle

The notice bar is used to display relatively important anomalies and announcements, such as the product is about to be maintained, and a bank channel will be unavailable during certain period.

- Do not use the notice bar to display product operation information, otherwise it will reduce the credibility of the announcement.
- The length of the announcement copy should not exceed the screen width, and the portion exceeding the screen width should be omitted by using "..." instead of using the line break.
- After the user taps to view or closes the announcement, the same information should not be appeared again to disturb the user.

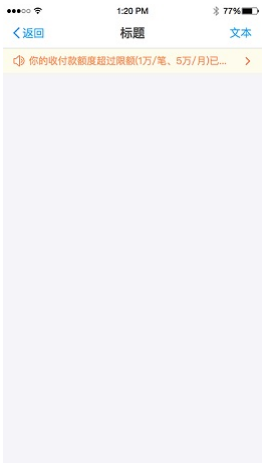
If you have more detailed content, they can be displayed on the details page by tapping the announcement, which should be disappeared after returning.



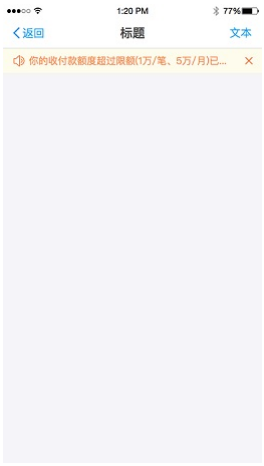
Visual style

You can add an arrow to view details or a button to close the announcement.

Below is the arrow to view details. After the user taps it and enters into the details page, the announcement disappears.



Below is the button to close the announcement. After the user taps **Close** button, the announcement disappears.



Step bar

The step bar shows the number of steps and the current step.



Definition and principle

The step bar shows the user that a task can be split into several steps, and the order of these steps. If these tasks are split into several different pages to display, the step bar can also be used to navigate through these pages.

Example

Taking credit card repayment and transferring to Yu'E Bao as an example, after the user finishes, these two tasks still take some time to complete. Therefore, you need to inform the user to wait through the step bar on the result page.

Result page of credit card repayment:



Result page of transferring to Yu'E Bao:



1.11.3.4. Feedback

One of the most important aspect in the process of human-computer interaction is the operation feedback. We should give timely feedback to the user's operation, and immediate response will improve user's trust in interaction.

One of the most important aspect in the process of human-computer interaction is the operation feedback. We should give timely feedback to the user's operation, and immediate response will improve user's trust in interaction.

mPaaS provides a series of feedback components where you should select the correct one for feedback under different scenes.

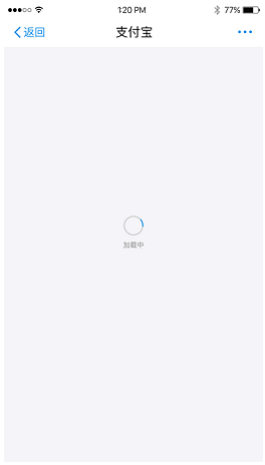
Feedback principle:

1. Provide necessary, positive and immediate feedback for users at all stages of operation;
2. Avoid excessive feedback, so as not to cause unnecessary interruption to the user. It is recommended to provide the feedback that can prompt simple and effective operations. The feedback prompt can be omitted.

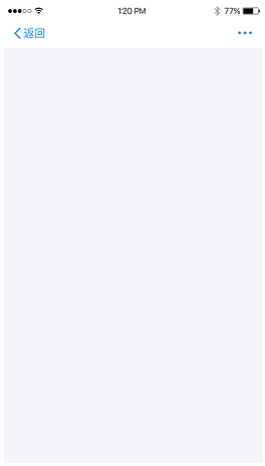
Necessary feedback

Provide necessary, positive and immediate feedback for users at all stages of operation.

- Correct example: When a blank page opens, a prompt appears to inform users that they need to wait



- Error example: When a blank page opens, no waiting prompt appears



Avoid excessive feedback

Excessive feedback can cause unnecessary interruptions to the user and should be avoided.

- Error example 1: When the user actively closes the cashier, a dialog box pops up to let the user confirm whether to quit, which is very unnecessary.



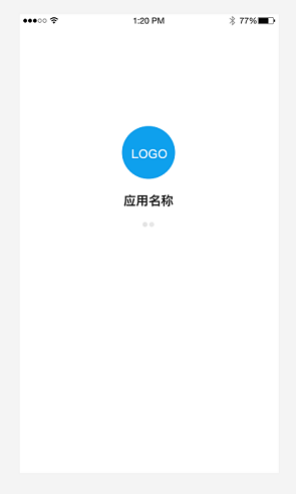
- Error example 2: In the service window, when a service window is deleted, a deleted successfully toast is displayed. At this point the page state has changed significantly, this toast is completely unnecessary.



Start page loading

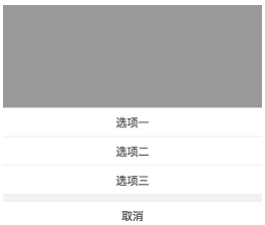
In an application, a start page is one of the pages that are used by the MINI to displays brand characteristics.

This page will highlight the MINI's brand characteristics and loading status. Except for the brand logo, all other elements on the start page, such as loading progress indicator, are provided by mPaaS and cannot be changed. This eliminates the need for the development.



ActionSheet

The action sheet is a special type of pop-up box that displays multiple action options.



Definition and principle

- The action sheet provides the user with multiple action options to choose from.
- The maximum height of the action sheet is limited and the maximum number of action options is 5.
- The action sheet is a special form of pop-up box. When it appears, the page will be covered by a semitransparent mask.

ActivityIndicator

The loading component visualizes the loading process of the page content, reducing the anxiety of the user during the waiting period.

Definition and principle

When a user enters into a new page or performs a submit action, and the page loading requires the user to wait, we should use the progress bar to inform the user of the loading progress. Otherwise, the user will feel absently and the waiting seems endless, thus leaving your page with anger.

We provide two styles, i.e. progress bar and progress ring, and define them separately.



Example

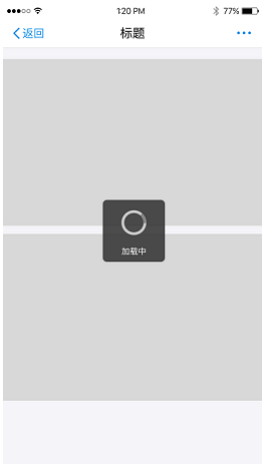
- Loading progress bar on page launch:

  When the user loads a brand new online page using the mPaaS framework, a loading progress bar will appear below the navigation bar to show the progress of the current page content loading. The loading progress bar is provided by mPaaS and cannot be changed. This eliminates the need for the development.



- Modal loading:

  Modal loading can be used when the user has to wait for a page to load after a submission action, and the modal loading style covers the entire page. Since the specific loading location or content cannot be clearly communicated to the user, the modal loading may cause anxiety and should be used with caution. Try not to use modal loading except under some global operations.



- Partial loading:

  The partial loading feedback means that the feedback is only given on the partial page that triggers loading. This feedback mechanism tends to be highly specific and causes minor content jumping, which is a recommended feedback method in mPaaS.



Dialog box (Modal)

A dialog box is used to inform users about critical information, or require users to make necessary decisions.



Definition and principle

A dialog box consists of one or two descriptive sentence(s) and action buttons. It has two usage patterns:

1. Confirm and cancel important operations (such as whether to delete content);
2. Inform the user of very important information (such as a serious error). The bold text are often used to highlight actions that may cause user loss (for example, “Delete”, “Don’t Save”, “Cancel”).

The dialog box must be used with caution. The dialog box is only used to prompt users of the content that they are required to know or the decision that they are required to make. Otherwise, use other weaker feedback methods such as toast.

Visual style

A standard dialog box consists of three parts: a title bar, main instruction, and commit buttons. A special type of dialog box can be configured with images and illustrations to facilitate better communication.





Example

Warning and error

- When the user's operation may cause irreparable damage, a double confirmation dialog box should be appeared to alert users.

  Double confirmation for Delete action:



- When an error occurs in the user's operation, the dialog informs the user of the cause of the error and what to do next.

  Form submission error:



Special dialog with illustration

Under some scenes, you may want to use a dialog box with images to lively convey relevant information, in such case a special dialog with illustration can be used.





Toast

Toast provides lightweight feedback about an operation or message prompt, in other words, it is actually a weakened version of the dialog box. Like a bubble, it disappears on its own after a few seconds (1.5 seconds or 3 seconds) on the interface. It does not force the user to perform any operation or confirmation, so it reduces interruption compared with the dialog box.

Toast is typically used to confirm the status or result of user task, and can also prompt the user of less important information, such as network abnormal condition, loading failure, and the like.



Definition and principle

- If you can timely see the status change on the page, the toast should not be used in order to avoid excessive feedback.
- Toast loading is a block type loading, which should be used sparingly and replaced by other methods if possible.
- The duration of toast is shorter, so the copy can only display up to 15 characters.

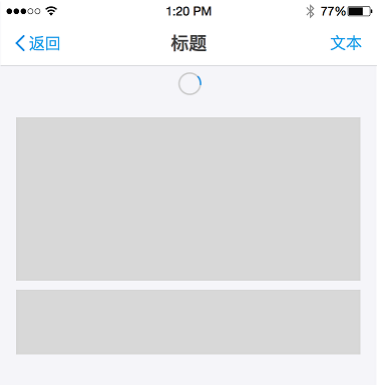
1.11.3.5. Gesture

As hidden shortcuts, gestures can be used to improve the user's operating efficiency. Using gestures can greatly enhance the user experience. iOS and Android systems have already made the users familiar with the operation represented by each gesture, and you can improve the user experience by following and supporting the existing operations of both systems.

As hidden shortcuts, gestures can be used to improve the user's operating efficiency. Using gestures can greatly enhance the user experience. iOS and Android systems have already made the users familiar with the operation represented by each gesture, and you can improve the user experience by following and supporting the existing operations of both systems.

Pull-to-refresh

The refresh of the page data is triggered by a pull-down gesture.



The pull-down gesture performed by user will trigger the client to request data update from the server. After receiving the request, the server will feed back to the client the latest page data.

The mPaaS MINI framework provides standard pull-to-refresh loading capabilities and styles. You can customize the pull-to-refresh interactions. For such interactions, mPaaS MINI will provide standard capabilities and styles.

1.11.3.6. Differentiated design

On the iOS and Android platforms, there exists significant difference in human-computer interaction. We should follow each platform's characteristics and make differentiated designs.

On the iOS and Android platforms, there exists significant difference in human-computer interaction. We should follow each platform's characteristics and make differentiated designs.

Search

There are two types of search bar, i.e. **prominent** and **hidden** search bar.

- The prominent search bar is used on Alipay homepage, industry platform homepage, etc. that have strong search needs or strong marketing needs;
- The hidden search bar is used on pages where search needs are not particularly strong, so the space saved can be used to display more content.

In the iOS platform, the design of prominent and hidden search bar is as follows:

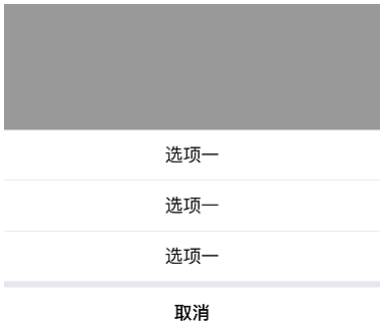


In the Android platform, the design of prominent and hidden search bar is as follows:



Operation list

- The operation list on iOS system is slid out from the bottom of the page, and there is a **Cancel** button at the bottom of the list to close the list;



- The operation list on Android system is popped up from the middle of the page. Since Android devices have physical back button, there is no need to design a cancel or close button in the list. Tap on a blank area outside the list or tap the physical back button to close the list.



Pop-up box

The pop-up box styles on iOS and Android platforms are different, but the interaction method and usage principle are the same:

Principle:

1. Ask whether to perform the current operation in the title;

2. Alert users of risky consequences that may be caused by current operation in the main instruction if necessary;
3. The button to confirm operation should be presented with the operation again.

Don'ts:

1. Don't use ambiguous descriptions like: "Are you sure?";
2. Do not explain or interpret the consequences of the operation;
3. Don't use action buttons that are unclear. For example, there are only "Cancel" or "OK" text on the button, where the "Cancel" button sometimes may cause ambiguity.

The iOS-based pop-up box is as follows:



The Android-based pop-up box is as follows:

1.11.3.7. Combine components

Through the combination of different components, you can build several typical pages.

Through the combination of different components, you can build several typical pages.

Result page



Definition and principle

After completing a task, you need to explicitly inform the user of the current task status. So you can display operation success status, current task status or required time in the page.

Follow-up operations

In addition to showing the final status of the task, the result page can also guide the user to perform other related operations.

Example

The result page is divided into two styles:

- Task succeeded:



- The task is successful at a certain stage, and still waits for operations at next stage:



1.12. FAQ

1.12.1. FAQ of using console

If the mini program package is not published to the application store, can it be launched directly in the App after publishing the test package in the console?

After publishing the test package on the console, you can use the IDE to publish the debugging or preview and obtain the mini program by scanning QR code with a mobile phone.

2.Mini Program Release

2.1. Introduction to Mini Program Release

The mPaaS Mini Program Release service is intended for Mini Program developers. The service supports **Mini Program encryption**, **switch configuration**, **allowlist**, and **release rule** management functions.

After you integrate the Mini Program function into your client, you can generate a package in mPaaS Plug-in, and then deliver the package on the Mini Program Release page in the mPaaS console. The client receives the package and then performs an upgrade. The Mini Program Release service also supports canary release based on an allowlist.

Features

- **Canary release**

  Canary release is a technique to verify whether a new package is as expected before the official release by rolling out the changes to a small subset of users in the allowlist, such as internal employees. You can also perform a canary release based on a time window to release the new package to a specified number of users within a specified time period. If the package is as expected, you can roll it out to all users.

- **Advanced filtering**

  During a canary release, you can use advanced rules to define a more precise allowlist of users. For example, you can send the package only to users of Xiaomi phones. You can also superimpose multiple filtering rules to deliver the package only when all the filtering rules are met.

- **Custom signature verification**

  To ensure security, a custom signature verification procedure is provided to verify script sources. mPaaS Plug-in allows you to generate a resource package and sign the package.

Benefits

- **Supports intelligent canary release and multiple upgrade policies**

  Various rules for controlling the allowlist-based canary release and time window-based canary release are available. The rules can be set based on users, regions, models, and networks.

- **Provides incremental offline packages for updates**

  Data redundancy and bandwidth consumption are reduced. Advantages are evident even when network conditions on the mobile end are unstable.

- **Ensures high sensitivity and availability**

  The RPC API capabilities on the client are upgraded. This service has a 99.999% availability and can be reached online within minutes.

- **Guarantees high performance**

  The Mini Program Release service is 99.999% available and can be accessed by over 200 million unique visitors per day.

2.2. Configure Mini Program package

Before adding a Mini Program package, you need to go to the configuration management interface to add the relevant configuration of the Mini Program package.

About this task

In order to unify the local file address name, when the client loads the local Mini Program package file, the local file is bound with a virtual domain name as the suffix.

The key is an RSA private key generated by OpenSSL, which is used to encrypt the Mini Program package. The corresponding public key is used to decrypt on the client.

Follow the steps below to generate a private key file and a public key file:

```
Generate private key:
  openssl genrsa -out private_key.pem 2048
Generate public key:
  openssl rsa -in private_key.pem -outform PEM -pubout -out public.pem
```

Procedure

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. On the opened Mini Program package list page, click **Configuration management**.
3. In the **Domain name management**, enter the virtual domain name, such as `example.com`.

   > **Note**
   >
   > Restriction: The virtual domain name cannot be a two-level or three-level domain name starting with http or https.

4. In the **Key management**, click **Choose file** to select the key file to upload.

   > **Note**
   >
   > If the client disables the signature verification after receiving the Mini Program package, it is not necessary to upload the key file here.

5. Check **The above information has been confirmed to be accurate and will not be modified after submission**, and click **Upload**.

Follow-up step

Create a Mini Program package

2.3. Create a Mini Program package

After the Mini program package is created in the console, a series of subsequent operations such as release mini program can be performed. When creating a Mini Program package resource, you need to enter basic information and configuration information.

After the Mini program package is created in the console, a series of subsequent operations such as release mini program can be performed. When creating a Mini Program package resource, you need to enter basic information and configuration information.

Prerequisite

You have already completed the relevant configuration of the Mini Program package in the configuration management interface. For details, see Configure Mini Program package.

Procedure

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In **Manage test package** tab, click **New** on the right of the Mini Program package list.

   > **Note**
   > - **Manage test package** is used for internal self-test and verification of test packages before the official release.
   > - **Manage official package** is used for the official package release if the self-test results are correct.

3. In the **New Mini Program** window, enter the ID and the name of the Mini Program, and click **OK**. The Mini Program ID is a 16-digit number.

   > **Note**
   > After creating a Mini Program, the Mini Program will be visible in the Manage official package and Manage test package tabs. You can add a formal package or a test package for the corresponding Mini Program in the corresponding tab page. The steps are the same.

4. Under the Mini Program list, find the new Mini Program and click Add.
5. In the **Basic Information** area, complete the following configuration:
   - **Version number**: Enter the version number of the Mini Program package, such as 1.0.0.1.
   - **Client range**: Select the minimum and maximum version of the iOS or/and Android client which runs the Mini Program.

     > **Note**
     > A client type must be selected.

   - **Icon**: Click **Choose file** to upload the icon of the Mini Program package. The image format must be png, jpeg, jpg. The recommended pixel is 180*180, and the icon core graphic is within the 160 px range.

     > **Note**
     > When creating a Mini Program package for the first time, the icon of the Mini Program must be uploaded. When releasing subsequent versions of the Mini Program package in the future, if you don't upload the icon, the icon of the previous version will be used by default.

   - **Package type**: Define the type of the uploaded Mini Program package.
     - In **Manage test package**, you can choose **Real device test package** or **Real device preview package**.

       > **Note**
       > - Both the real device test package and the real device preview package are test versions uploaded through the IDE and can only be accessed by scanning the QR code generated by the IDE.
       > - The real device test package supports remote debugging by the IDE of Mini Program,and the real machine preview package does not support.

     - In **Manage official package**, you can choose **Feature package** or **Plugin package**.

       > **Note**
       > At present, there is no functional difference between the Feature package and the Plugin package.

   - **File**: Upload the Mini package resource file, the file format is .zip.
6. In the **Configuration information** area, complete the following configuration:
   - **Main entry URL**: Required, the homepage address of the Mini Program package, such as `/index.html#pages/index/index`.

     > **Note**
     > You can get the homepage address from the first item of the `page` array in the Mini Program global configuration file `app.json`. See Application.

   - **Show bottom navigation bar**: Choose whether to display the navigation bar at the bottom of the Mini Program.
   - **Show upper-right function options**: Choose whether to display more function options in the upper right corner of the Mini Program.
   - **Virtual domain name**: Automatically display the virtual domain name filled in when configuring the Mini Program package.
   - **Extended information**: Optional, enter the page loading parameters, the format is KV, separate multiple KVs with comma (,).

> **Note**
> mPaaS supports the configuration of the request interval of Mini Program packages, which can be configured individually or globally.

- **Individual configuration**: Only configure the current Mini Program package. You can fill in `{"asyncReqRate":"1800"}` in the extended information to set the request interval. In which, `1800` represents the interval time, in seconds, and the range is 0 ~ 86400 seconds (that is, 0 ~ 24 hours, 0 means no request interval limit).
- **Global configuration**: Global configuration needs to be done in the client code, see Access Android and Access iOS.

- **Download time**: Select the time when the user downloads the mini program package.
  - If you choose **Wi-Fi only**, the mini program package will be automatically downloaded in the background only when the device is connected to Wi-Fi network.
  - If you choose **Download from all networks**, it will consume user data and download automatically in non-Wi-Fi networks. Use it with caution.
- **Installation time**: Select the time when the user installs the mini program package.
  - If you choose **No preload**, it will only be installed when you enter the mini program package or Mini Program page.
  - If you choose **Preload**, the mini program package or Mini Program will be automatically installed after downloading.

7. Check **The above information has been confirmed to be accurate and will not be modified after submission**.
8. Click **Submit**.

2.4. Release a Mini Program package

To release a Mini Program package you have created, you need to create a release task for the Mini Program package and complete related configurations.

To release a Mini Program package you have created, you need to create a release task for the Mini Program package and complete related configurations.

Procedure

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. On the **Manage official package** page, select the Mini Program package you want to release, and click **Create release** on the right.



3. On the **New release** page, complete the following configuration:
   - **Release type**: Choose **Gray release** or **Official release**.
   - **Release model**: Choose **Whitelist** or **Time window**.
     - If you choose the **Whitelist** release model, select the whitelist in the **Whitelist configuration** below.

       > **Note** When the number of users in a selected whitelist exceeds 100,000, only the first 100,000 will be taken.

     - If you choose **Time window**, select the end time and the number of people in gray release below.
   - **Release description**: Enter the description of the Mini Program package release task.
   - **Advanced rules**: Optional, add one or more advanced rules to the release task.
     - **Type**: Select city, model, network or operation system version.
     - **Operation type**: Choose whether to include the type selected above.
     - **Resource value**: In the drop-down menu, select the resource value corresponding to the selected type.
4. Click **OK** to publish.

Result

On the **Mini Program package list** page, you can see the status of the released Mini Program package is displayed in **Gray release** or **Official release**. Meanwhile, hover the mouse on **View icon** on the details page on the right side of the Mini Program package, you can see the icon of the release Mini Program.

Follow-up step

Manage the released mini program packages.

2.5. Manage Mini Program package

After releasing a Mini Program package, you can manage the released package. Management operations include viewing, suspending, ending, and deleting a Mini Program packages.

After releasing a Mini Program package, you can manage the released package. Management operations include viewing, suspending, ending, and deleting a Mini Program packages.

View a Mini Program package release task

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In the Mini Program package list on the left, select the Mini Program package you want to view.
3. On the right-side page, click the drop-down button (

   +

) on the left of the Mini Program package version to expand more information.
4. Click **View** on the right. You can see the details of the posting task.

Suspend a Mini Program package release task

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In the Mini Program package list on the left, select the Mini Program package you want to suspend release.
3. On the right-side page, click the drop-down button (

   +

) on the left of the Mini Program package version to expand more information.
4. Click **Suspend** on the right.
5. Click **OK** to suspend the release of the Mini Program package.

> **Note** After the suspension, if you want to continue releasing the Mini Program package, click Continue.

End a Mini Program package release task

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In the Mini Program package list on the left, select the Mini Program package you want to end release.
3. On the right-side page, click the drop-down button (

   +

) on the left of the Mini Program package version to expand more information.
4. Click **End** on the right.
5. Click **OK** to end the Mini Program package release.

> **Note** After the ending the release, if you want to release the Mini Program package again, you need to re-create the release task.

Delete a Mini Program package

Enter the mPaaS console and complete the following steps:

1. Click **Mini Program** > **Release Mini Program** in the left navigation bar.
2. In the Mini Program package list on the left, select the Mini Program package you want to delete.
3. Click the delete button (

   🗑

).
4. Click **OK**.

> **Note** After deleting, the Mini Program package resource cannot be retrieved.

2.6. Mini Program permission control

In order to control the access scope of the Mini Program in the App, you can add **Server domain name whitelist**, **API whitelist**, and **Built-in WebView domain name whitelist** for the Mini Program in the mPaaS console. After the **Mini Program permission control switch** in the figure below is turned on, only the resources added to the whitelist can be accessed or used by the current Mini Program.

- **Server domain name whitelist**: Refers to the domain name whitelist of the target server (input URL) in my.request. HTTPS protocol supported, up to 30 domain names can be added.
- **API whitelist**: The API whitelist called by the Mini Program. If permission control is enabled, APIs that are not added to the whitelist cannot be successfully called by the Mini Program.

> **Note** The API provided by the mPaaS official website has been added with permission files by default, and no configuration is required. You only need to configure your custom API here.

- **Built-in WebView domain name whitelist**: The access addresses whitelist of web-view component. HTTPS protocol supported.



Prerequisite

You have created a Mini Program in the Mini Program package management.

Select a Mini Program

At the top of the page, you can select an existing Mini Program through the drop-down list. After selection, the name and AppId of the Mini Program will be displayed below.

> **Note** The Mini Programs created in the Mini Program package management tab on the left will be synchronized to this drop-down list in real time.



Permission control switch

Through the Mini Program permission control switch, you can choose whether to enable the **Server domain name whitelist**, **API whitelist**, and **Built-in WebView domain name whitelist**, so as to realize the permission control of the selected Mini Program.



Server domain name whitelist

In the whitelist configuration area below, you can add server domain names to the whitelist.



Add a server domain name to the whitelist

1. Log in to the mPaaS console and select an application. In the left navigation bar, select **Mini Program** > **Release Mini Program**.
2. Select the **Manage open platform Mini Program** tab, and click **Add** in the **Server domain name whitelist** tab below.
3. In the pop-up **Add server domain name whitelist** window, enter the following information:



- **Domain name**: Required. Only the server domain name of HTTPS protocol is supported here. For non-HTTPS domain names, it will be intercepted when calling.
- **Remark**: Optional, enter the description of this domain name, up to 200 characters.

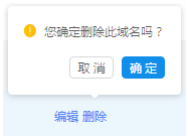
4. Click **OK** to finish.

> **Note** You can add up to 30 server domain names to the whitelist.

Edit and delete a server domain name

All server domain names in the whitelist list can be edited. Click **Edit** in the **Operation** column on the right to change the server domain name and its description.

To delete a server domain name from the whitelist, click **Delete** in the **Operation** column on the right, and click **OK** in the pop-up confirmation box to delete the server domain name.



API whitelist

In the whitelist configuration area below, you can add Mini Program APIs to the whitelist.



Add an API to the whitelist

1. Log in to the mPaaS console and select an application. In the left navigation bar, select **Mini Program** > **Release Mini Program**.
2. Select the **Manage open platform Mini Program** tab, and click **Add** in the **API whitelist** tab below.
3. In the pop-up **Add Mini Program API Whitelist** window, enter the following information:



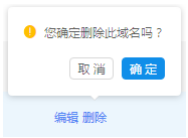
- **API**: The API to be added to the whitelist.
- **Remark**: Optional, enter the description information of this API, up to 200 characters.

4. Click **OK** to finish.

Edit and delete an API

All APIs in the whitelist list can be edited. Click **Edit** in the **Operation** column on the right to change the API and its description.

To delete an API from the whitelist, click **Delete** in the **Operation** column on the right, and click **OK** in the pop-up confirmation box to delete the API.



Built-in WebView domain name whitelist

In the whitelist configuration area below, you can add built-in WebView domain names to the whitelist.



Add a built-in WebView domain name to the whitelist

1. Log in to the mPaaS console and select an application. In the left navigation bar, select **Mini Program** > **Release Mini Program**.
2. Select the **Manage open platform Mini Program** tab, and click **Add** in the **Built-in WebView domain name whitelist** tab below.
3. In the pop-up **Add WebView domain Name whitelist** window, enter the following information:

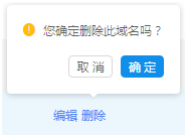


   - **Domain name**: Required. Only the server domain name of HTTPS protocol is supported here. For non-HTTPS domain names, it will be intercepted when calling.
   - **Remark**: Optional, enter the description of this domain name, up to 200 characters.
4. Click **OK** to finish.

Edit and delete a built-in WebView domain name

All WebView domain names in the whitelist list can be edited. Click **Edit** in the **Operation** column on the right to change the WebView domain name and its description.

To delete a WebView domain name from the whitelist, click **Delete** in the **Operation** column on the right, and click **OK** in the pop-up confirmation box to delete the WebView domain name.



3.Mini Program Analytics

3.1. About Mini Program Analytics

mPaaS Mini Program Analytics is a data analytics component designed for mini program developers and operators. This component enables you to perform data statistics for Alipay, WeChat, and mPaaS mini program platforms. It supports comprehensive data analytics for mini programs on these platforms and visualizes statistical and analytics data. In addition, this component helps you make marketing decisions and drives product experience optimization.

The Mini Program Analytics module offers analytics features, such as user analytics, page analytics, and sharing analytics, and visualizes the number of new and active users, user sources, retention length, user retention, sharing spreading of mini programs. These features give you an insight into the overall operations of mini programs, and offer you data to optimize them and make operational decisions.

Features

Mini program management

This feature allows you to visualize statistics and analytics data of mini programs as per app and per mini program. For a mini program, it provides a full tracing analytics path: "Customer acquisition > Activation > Sharing". You can easily add WeChat mini programs, Alipay mini programs, and mPaaS mini programs for data statistics and analytics. You can query, modify, and delete existing mini programs with a few clicks.

User analytics

This feature allows you to analyze the data performance of a mini program in terms of new users, active users, user activity level, and use duration. This feature analyzes the activity level of mini program users by integrating the daily, weekly, and monthly active user (DAU, WAU, and MAU) data, helping you analyze user retention and user churn. In addition, this feature identifies user stickiness based on the average retention duration per time and the average retention duration per user.

Page analytics

This feature displays page views (PVs), unique visitors (UVs), and the average visit duration of each page. You can analyze these view data of the visited page and the entry page to find out the most popular mini program pages and entries.

Sharing analytics

This feature displays key sharing metrics of a mini program, such as the number of people who shared the mini program, the times of sharing the mini program, and sharing-driven visits. This feature gives you an insight into how a mini program is shared and spread.

Scenarios

This component is designed for developers who develop mini programs on multiple platforms. After you integrate this component, the data of mini programs running on different platforms can be viewed in one console.

User activity level analytics

It analyzes the activity level of mini program users. In detail, it analyzes the user behavior habits and preferences based on the user activity level, use frequency, and retention duration. This feature gives you an insight into the user stickiness to the product and helps you optimize products and make better operational decisions.

Sharing spreading analytics

By analyzing the mini program's page sharing and spread effect, you can evaluate how the valuable features or contents help promote the spreading. Through analyzing the sharing times, the number of people sharing the mini program, and sharing-driven visits, and the number of new users after sharing, you can gain an insight into which users shared the most and which users have brought the greatest number of new users to the mini program. In this way, you can specify an operation strategy for these users.

3.2. Data overview

The mini program analytics feature allows you to perform statistics and analytics for the mini programs under this app. On the Data overview page, you can view the historical statistics (with a time limit of T + 1) and real-time data of a single mini program.

To view the data overview, follow these steps:

1. Log on to the mPaaS console, select the target app and choose **Mini Program** > **Mini Program Analytics** > **Data overview.** from the left-side navigation pane.
2. From the drop-down list in the upper-left corner, select the target mini program. By default, it displays the analytics data of the most recently created mini program.
3. Click the **Overview statistics** or **Real-time dashboard** tab to view the historical or real-time statistics of the mini program. The analytics data is illustrated by a metric card and a line trend chart.

Overview statistics

The **Overview statistics** page displays the key metrics and trend changes on "Customer acquisition > Activation > Sharing" of the selected mini program. You can view the current mini program's data performance and value conversion, identify data problems, and locate abnormal data. You can select the metric data for any 1 day, 7 days, or 30 days in the past.

You can export the analytics data as an Excel file. Select the period for analysis, click the **Export** button on the upper-right of the page to export the mini program statistics for that period. In the exported file, the summarized metric data and the line chart data of each metric are displayed on different sheets.

Metrics overview

This page displays historical statistics of the current mini program in customer acquisition, activation, and sharing stages over the specified period. The key metrics are displayed in the form of cards. The metrics include new users, active users, opens, average retention length per time, the number of users sharing the mini program, sharing-driven new users, and the growth rate compared to the previous day, the same period last week, or the same period last month.

Line trend chart

The change trend of all metrics over a specified period is illustrated by a line chart. When you click the metric card, you can view the metric data trend on the metric card.

Real-time dashboard

The real-time dashboard displays the real-time statistics of five key metrics of the selected mini program and their hourly change trends.

You can export the analytics data as an Excel file. Click the **Export** button on the upper-right of the page to export the mini program statistics for that period. In the exported file, the summarized metric data and the line chart data of each metric are displayed on different sheets.

Metrics overview

This page displays the real-time data of each metric of the current mini program in the form of cards. The metrics include active users, opens, PVs, average retention length per time, and average retention length per user, and growth rate compared to the same period yesterday.

Line trend chart

The data change trends of each metric today, yesterday, and seven days ago are illustrated by a line chart. When you click the metric card, you can view the data change trend of the metric on the line chart.

Metrics description

The following table describes the metrics on Data overview.

Metric	Description
New users	The number of users who visited the mini program page for the first time. If a user visits the page more than once, it is counted as one user.
Active users	The total number of users who visited the mini program. If a user visits more than once, it is counted as one user.
Opens	The total times that the mini program was opened. If the mini program is opened and exited from the background, this action is counted as once. If the mini program is opened again within 30 seconds after the first open, this action is counted as once.
Average retention length per time	The total retention length on a mini program page every time a user opens the mini program over the selected period, namely, average retention length per time = total retention length of all users/opens.
Average retention length per user	The total retention length of each user on a mini program page, namely, average retention length per user = total retention length/number of active users.
Number of users sharing the mini program	The total number of users sharing the mini program page. If a user shares the page more than once, it is counted as one user.
Sharing-driven new users	The number of new users who visited the mini program for the first time through the shared link.
Previous day	The growth rate of the cumulative data on the selected day compared to that on the previous day. Comparison with the same period of the previous day = (data on the specified date - data on the previous day)/data on the previous day. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period yesterday	The growth rate of the cumulative hourly data from 00:00 to the time when you view the data compared to that over the same period yesterday.
Same period last week	The growth rate of cumulative data over the past 7 days compared to that over the past 8 to 14 days. Same period last week = (data over the past 7 days - data over the same period last week)/data over the same period last week. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period last month	The growth rate of cumulative data over the past 30 days compared to that over the past 31 to 60 days. Same period last month = (data over the past 30 days - data over the same period last month)/data over the same period last month. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.

3.3. Mini program management

The Mini program management page displays the overall data analysis of all mini programs created through the Mini Program Analytics under this app. The summary data includes four metrics: the number of new users, the number of active users, the number of cumulative users, and the startup times. In addition, you can have a data overview of a single mini program. The metrics displayed on this page are all of the previous day.

You can create mini programs for statistics and analytics purposes and manage the created mini programs.

The following table describes the key metrics.

Metric	Description
New users	The number of cumulative new users of all mini programs under this app (deduplicated). For a single mini program, this metric refers to the number of users who visited the mini program page for the first time. If a user visits more than once, it is counted as one user.
Active users	The number of cumulative active users of all mini programs under this app (deduplicated). For a single mini program, this metric refers to the total number of users visiting the mini program. Multiple visits by the same user will not be counted repeatedly.

Metric	Description
Startup times	The number of cumulative startup times of all mini programs under this app (not deduplicated). For a single mini program, this metric refers to the total times the mini program was opened. If the mini program is opened and exited from the background, this action is counted as once. If the mini program is opened again within 30 seconds after the first open, this action is counted as once.
Cumulative users	The number of cumulative new users of all mini programs under this account (deduplicated). For a single mini program, this metric refers to the number of cumulative users of the mini program since the mini program was released.

Create a mini program

Create a mini program for its statistic and analytics purposes. A user can create multiple apps and multiple mini programs can be created under a single app. A user can create up to 100 mini programs.

> **Important**
>
> The mini programs created under the Mini Program Analytics module are for statistics and analytics purposes only. Actual mini programs must be created on the appropriate mini program development platform or mini program release platform.

To create a mini program, follow the steps:

1. Log on to the mPaaS console and select the target app. On the left-side navigation pane, choose **Mini Program** > **Mini Program Analytics** > **Mini program management**.
2. Click **Create mini program**. On the panel that appears, enter a name for the mini program and select the type. Only mPaaS mini programs, Alipay mini programs, and WeChat mini programs are supported.
3. After you set the mini program, click **OK** to complete the creation. The new mini program appears in the list of mini programs.
4. From the **Code integration** panel, copy the integration code to the `app.js` file for the mini program to integrate. The basic metric statistics for a mini program cannot be performed without code integration.

   The following table describes the parameters in the integration code:

Parameter	Description
appId	mPaaS App ID
workspaceId	Workspace
id	Mini Program ID
reportURL	The tracking point report URL of the mini program, which corresponds to the log collection gateway (MDAP) URL. Contact technical support for the MDAP URL.
debug	Debug mode. By default, debug mode is disabled. Valid values: ○ true: This parameter is enabled. ○ false: This parameter is disabled.

View the data details of the mini program

The list of mini programs displays all mini programs created through the Mini Program Analytics module and their key metric data. You can have the data overview of a mini program by metrics.

In the list of mini programs, find the target mini program and click the mini program name or **View** in the **Operations** column. Then, on the Data overview page, you can view the statistical data details of the mini program.

Edit a mini program

Modify the basic information of the mini program, including its name and type.

In the list of mini programs, find the target mini program and click **Edit** in the **Operations** column. Modify the information and confirm the modification.

Delete a mini program

Delete a mini program from the list.

In the list of mini programs, find the target mini program and click **Delete** in the Operations column. Click **OK** to complete the deletion.

Integrate mini program codes

Integrate mini program codes for metric data statistics of the mini program.

In the list of mini programs, find the target mini program and click **Integrate code** in the **Operations** column. In the pop-up **Code integration** panel, copy the code to integrate it into the mini program.

3.4. User analytics

The User analytics page displays the user visit data to the selected mini program (with a time limit of T + 1), including seven metrics, such as new users, active users, opens, and PVs. These metrics help you analyze user data changes at each stage. You can choose to view statistics for any 1 day, 7 days, or 30 days in the past.

To view user analytics data, follow these steps:

1. Log on to the mPaaS console, select the target app and choose **Mini Program** > **Mini Program Analytics** > **User analytics** from the left-side navigation pane.
2. In the upper-left corner, select the target ation mini program from the drop-down list.
3. Select a time range to view the statistics for that time period.

You can export the user analytics data as an Excel file. Select the period for analysis, click the **Export** button on the upper-right of the page to export the mini program statistics for that period. In the exported file, the summarized metric data and the line chart data of each metric are displayed on different sheets.

User analytics overview

This page displays the statistics of seven metrics of the mini program over a specified period, including new users, active users, opens, PVs, average retention length per time, average retention length per user, and opens per user. In addition, you can view the growth rate compared to the previous day, the same period last week, or the same period last month.

The following table describes the metrics.

Metric	Description
New users	The number of users who visited the mini program page for the first time. If a user visits the page more than once, it is counted as one user.
Active users	The total number of users who visited the mini program. If a user visits the page more than once, it is counted as one user.
Opens	The total times that the mini program was opened. If the mini program is opened and exited from the background, this action is counted as once. If the mini program is opened again within 30 seconds after the first open, this action is counted as once.
PV	The total number of visits to all pages of a mini program. Jumps among multiple pages and repeated visits to the same page are counted as multiple visits.
Average retention length per time	The total retention length on a mini program page every time a user opens the mini program over the selected period, namely, average retention length per time = total retention length of all users/opens.
Average retention length per user	The total retention length of each user on a mini program page, namely, average retention length per user = total retention length/number of active users.
Opens per user	The total times each user opens the mini program over the selected period, namely, opens per user = total opens by all users over a time period/total number of active users.
Previous day	The growth rate of the cumulative data on the selected day compared to that on the previous day. Comparison with the same period on the previous day = (user visit data on the specified date - user visit data on the previous day)/user visit data on the previous day. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period last week	The growth rate of cumulative data over the past 7 days compared to that over the past 8 to 14 days. Same period last week = (user visit data over the past 7 days - user visit data over the same period last week)/user visit data over the same period last week. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period last month	The growth rate of cumulative data over the past 30 days compared to that over the past 31 to 60 days. Same period last month = (user visit data over the past 30 days - user visit data over the same period last month)/user visit data over the same period last month. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.

User visit trend

The change trend of all metrics over a specified period is illustrated by a line chart. When you click the metric card, you can view the data trend of the metric on the line chart.

Details

This page displays hourly or daily user analytics and statistics in tabular form over the selected period. You can view the data by time.

What time granularity the data is displayed depends on the selected time range, that is, when the query time range is 1 day, the data is displayed by hour; and when the time range is 7 or 30 days, the data is displayed by day.

3.5. Page analytics

The Page analytics page displays the page views of the selected mini program (with a time limit of T+1), including PVs, UVs, and the average visit duration of the page, giving you an insight into which pages are most used by users, and how long the pages are used. You can choose to view statistics for any 1 day, 7 days, or 30 days in the past.

To view the page analytics data, follow these steps:

1. Log on to the mPaaS console, select the target app and choose **Mini Program** > **Mini Program Analytics** > **Page analytics** from the left-side navigation pane.

2. In the upper-left corner, select the target mini program from the drop-down list.
3. Select a time range to view the statistics for that time period.
   - **PV**: It refers to the total number of visits to all pages in a mini program. Jumps among multiple pages and repeated visits to the same page are counted as multiple visits.
   - **Page UV**: It refers to the total number of users visiting all pages in a mini program. If a user visits a page more than once, it is counted as one user.
   - **Average time on page**: It refers to the average retention length on a mini program page every time a user opens the page, namely, average visit duration of a page = total retention length of the page over a specified period/total PVs of the page.

You can export the analytics data as an Excel file. Select the period for analysis, click the **Export** button on the upper-right of the page to export the statistics for that period.

3.6. Sharing analytics

The sharing feature is an important way for mini programs to attract new users and acquire customers, and it is an important metric of the health of mini programs. The sharing analytics page displays the sharing data of the selected mini program (with a time limit of T+1), including five metrics, such as the number of users sharing the mini program, the times of sharing the mini program, and sharing-driven visits. The data displayed on this page gives you an insight into the sharing fission effect of a mini program. You can choose to view statistics for any 1 day, 7 days, or 30 days in the past.

To view sharing analytics data, follow these steps:

1. Log on to the mPaaS console, select the target app and choose **Mini Program** > **Mini Program Analytics** > **Sharing analytics** from the left-side navigation pane.
2. In the upper-left corner, select the target mini program from the drop-down list.
3. Select a time range to view the statistics for that time period.

You can export the sharing analytics data as an Excel file. Select the period for analysis, click the **Export** button on the upper-right of the page to export the mini program statistics for that period. In the exported file, the summarized metric data and the line chart data of each metric are displayed on different sheets.

Overview of sharing analytics

This page displays the statistics of the five metrics: the number of users sharing the mini program, the times of sharing the mini program, sharing-driven visits, the share return ratio, and new sharing over the specified period. In addition, this page shows the growth rate compared to the previous day, the same period last week, or the same period last month.

The following table describes the metrics.

Metric	Description
Number of users sharing the mini program	The total number of users sharing the mini program page. If a user shares the page more than once, it is counted as one user.
Sharing times	The total times the mini program page was shared.
Sharing-driven visits	The times that a mini program was visited through the shared link are counted as sharing-driven visits of this sharing.
Sharing-driven ratio	This metric reflects the spreading effect of sharing a mini program. Sharing-driven ratio = sharing-driven visits/times of sharing × 100 %.
New sharing	The number of new users who visited the mini program for the first time through the shared link.
Previous day	The growth rate of the cumulative data on the selected day compared to that on the previous day. Comparison with the same period of the previous day = (sharing data on the specified date - sharing data on the previous day)/sharing data on the previous day. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period last week	The growth rate of cumulative data over the past 7 days compared to that over the past 8 to 14 days. Same period last week = (sharing data over the past 7 days - sharing data over the same period last week)/sharing data over the same period last week. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.
Same period last month	The growth rate of cumulative data over the past 30 days compared to that over the past 31 to 60 days. Same period last month = (sharing data over the past 30 days - sharing data over the same period last month)/sharing data over the same period last month. When the divisor is 0, it is calculated as 1, and the integer is reserved before the percentage sign.

Sharing trend

The change trend of all metrics over a specified period is illustrated by a line chart. When you click the metric card, you can view the data trend of the metric on the line chart.

Details

This page displays hourly or daily sharing analytics and statistics in tabular form over the selected period. You can view the data by time.

What time granularity the data is displayed depends on the selected time range, that is, when the query time range is 1 day, the data is displayed by hour; and when the time range is 7 or 30 days, the data is displayed by day.

4.Mini program monitoring

As an increasing number of mini programs are developed and gaining a growing number of users, mini programs have become a common carrier for users to have access to Internet services. The performance of mini programs greatly affect the user experience and in turn, affects the user retention and conversion rates. The performance of a mini program is reflected in the loading and presentation speed of the mini program and the responsiveness of its interaction with users.

The mini program monitoring feature monitors user access, mini program page stability, and external service calls in real time. The feature provides real-time statistical analysis of the page views (PV), unique visitors (UV), and other user access data and their trends. In addition, the feature monitors performance problems that affect user experience, such as white screens, in-app open exceptions, exceptions in JS API calls, exceptions in requests for app data packets. This feature helps users locate and resolve the problems.

This feature is available for mPaaS mini programs only and will be available for Alipay mini programs and WeChat mini programs in the future.

View monitored mini programs

A user can monitor up to 100 mini programs. In other words, a user can have multiple apps and create multiple mini programs for each app, but cannot monitor more than 100 mini programs in total.

To view the list of monitored mini programs, perform the following steps:

1. Log on to the Alibaba Cloud Management Console. Choose **Products and Services** > **Mobile PaaS**. On the page that appears, select the app that you need to query analysis data.
2. In the left-side navigation pane, choose **Application Performance Management** > **Mini program monitoring**. In the list of monitored mini programs, you can view the number of new users, the number of active users, the number of startup times, the monitoring data of the day, and the monitoring data of the previous day for each mPaaS mini program.

   > **Note**
   >
   > A mini program in the list of monitored mini programs can be downloaded by using the offline mini program package created on the mPaaS mini program platform.

3. In the list of monitored mini programs, click the specified mini program name or click **View** in the

   Operations column. The **Mini program monitoring details** page appears, allowing you to view the monitoring data of the mini program.

   In addition, you can enter keywords in the search box to search for mini programs.

   To export the list of monitored mini programs and the access data of the monitored mini programs that belong to the current app, click **Export** in the upper-right corner. The exported file is in Excel format.

View the mini program monitoring details

On the **Mini program monitoring details** page, you can filter the monitoring data of the current mini program by selecting the platform, client version, and time.

Mini program monitoring is classified into business monitoring and exception monitoring. Each monitoring report displays monitoring data of the last 1 hour, last 6 hours, last 12 hours, and a specified time period of the current day. By default, monitoring data of the current mini program on all platforms and versions for the last 1 hour is displayed.

To export the monitoring data of the current mini program, click **Export** in the upper-right corner. The exported file is in Excel format.

Business monitoring

The business monitoring section displays the number of PVs, the number of active users, and their trends of a mini program within a specified time range.

Exception monitoring

The exception monitoring section displays the monitoring data of exceptions that affect user experience and the trends within a specified time range. The monitoring data includes the number of in-app open exceptions, the white screen rate, the exception rate, the number of exceptions in JS API calls, and the exception rate of requests for app data packets. In addition, the health degree of mini program pages is monitored in three dimensions: the page load speed, page stability, and the success rate of external service calls (API calls).

Metrics

Metric	Description	Judgment description
New users	The number of new users (indicated by device IDs) who visited the mini program.	-
Active users	The total number of users (indicated by device IDs) who visited the mini program. If a user visited the mini program more than once, the user is counted as one user only.	app.launch
Startup times	The total number of times that the mini program was started. The process from a user opening the mini program to the user closing the mini program or the mini program exits due to timeout is counted as one startup time.	-
Time-consuming to start	The time-consuming required for the mini program to open.	tiny_app_launch
Cumulative users	The number of users (indicated by device IDs) who visited the mini program.	-

PVs	The total number of visits to all pages of the mini program.	auto_openPage
Limited number of page visits	The number of times the mini program component webview access link is limited.	H5_AL_NETWORK_PERMISSON_ERROR
In-app open exceptions	The number of times that exceptions occurred when users open mini program pages in an app.	H5_APP_PREPARE (errc!=1)
White screen rate	A mini program page being completely blank is referred to as a white screen. White screen rate = Number of white screens on pages/Number of page opens x 10000‰	-
Number of white screens	The number of times the mini program page is opened with a white screen.	H5_PAGE_ABNORMAL
Number of JS exceptions	Mini-program JS execution exception times.	H5_CUSTOM_ERROR
JS exception rate	JS exception rate = Number of JS exceptions/Number of PVs x 1000‰	-
Exceptions in JS API calls	The total number of exceptions in JS API requests for the mini program.	H5_AL_JSAPI_RESULT_ERROR
request abnormal number	The number of failed my.request/my.httpRequest requests.	H5_AL_JSAPI_RESULT_ERROR
Exception rate of requests for app data packets	Exception rate of requests for app data packets = Number of exceptions in requests for app data packets/Total number of requests for app data packets	-
Application pull request exception-android	The mini program information pull request is abnormal.	H5_APP_REQUEST
Application pull request exception-iOS	The mini program information pull request is abnormal.	H5_APP_REQUEST
The number of resource file request exceptions	The number of failed requests for mini program resource files.	H5_AL_NETWORK_PERFORMANCE_ERROR