Ant Technology

Mini Program User Guide

Document Version: 20250731

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 20
1.1. About Mini Program 20
1.2. Integrate Mriver Tiny App into Android 20
1.2.1. Quick start 20
1.2.2. Start a mini program 23
1.2.3. Installation and debugging 25
1.2.3.1. Real-device preview and debugging 25
1.2.4. Engine management 26
1.2.4.1. Custom APIs 26
1.2.5. Application management 29
1.2.5.1. Obtain the stack and information of a mini prog... 29
1.2.6. Resource management 29
1.2.6.1. Remote management 29
1.2.6.2. Mini program package verification 30
1.2.7. Permission control 30
1.2.7.1. Mini program permission 30
1.2.8. Interface documents 33
1.2.8.1. API 33
1.2.8.2. Switch configuration 36
1.2.8.3. Function configuration 37
1.2.8.4. Startup parameters 59
1.2.9. Switch the map 59
1.2.9.1. Use Google Maps in Mini Program 59
1.2.9.2. Use Baidu Map in Mini Program 59
1.3. Integrate Ariver Tiny App into iOS 59
1.3.1. Baseline cp_change_15200851 Series 59

1.3.2. Upgrade guide 62
1.3.3. Quick start 63
1.3.4. Start mini program 66
1.3.5. Real device preview and debugging 67
1.3.6. Customize UI 68
1.3.6.1. Customize navigation bar 68
1.3.6.2. Customize startup loading page 70
1.3.6.3. Customize error page 71
1.3.7. Engine management 72
1.3.7.1. Custom APIs 72
1.3.7.2. Custom View 73
1.3.8. Resource management 76
1.3.8.1. Mini program package information 76
1.3.8.2. Update mini program package 77
1.3.8.3. Mini program package signature verification 79
1.3.8.4. Preset mini program package 79
1.3.9. Permission management 80
1.3.9.1. Mini program permissions 80
1.3.10. Customize containers 81
1.3.10.1. Custom UserAgent 81
1.3.10.2. Custom WebView Base Class 81
1.3.10.3. Custom Container Base Class 82
1.3.10.4. Custom SDK 82
1.3.11. Switch the map 83
1.3.11.1. Use Google Maps in Mini Program 83
1.3.11.2. Use Baidu Map in Mini Program 84
1.4. Mini Program container - Nebula (not recommended) 86
1.4.1. Integrate Mini Program into Android 86

1.4.1.1. Quick start 86
1.4.1.2. Advanced guide 89
1.4.1.2.1. Integrate real-device preview and debugging ... 89
1.4.1.2.2. Customize navigation bar 90
1.4.1.2.3. Customize bidirectional channel 90
1.4.1.2.4. API permission extension configuration 92
1.4.1.2.5. Customize startup loading page 95
1.4.1.2.6. Extend upper-right pop-up menu 97
1.4.1.2.7. Set entry/exit animation 98
1.4.1.2.8. Specify the redirect page at startup 99
1.4.1.2.9. Pass startup parameters to Android Mini Prog... 99
1.4.1.2.10. Preset an Android mini program in the client 100
1.4.1.2.11. Implement multiple instantiation of Android ... 102
1.4.1.2.12. Customize View in Android Mini Program 102
1.4.1.2.13. Customize rendering parameters in custom ... 105
1.4.1.2.14. Send custom message to custom View 105
1.4.1.2.15. Send custom event by custom View 106
1.4.1.3. Instructions on upgrading Mini Program 106
1.4.1.4. Tutorial 106
1.4.1.4.1. Overview 106
1.4.1.4.2. Create a native project in Android Studio 107
1.4.1.4.3. Create an App in the mPaaS console 107
1.4.1.4.4. Integrate Mini Program to project through Na... 108
1.4.1.4.5. Initialize configuration 108
1.4.1.4.6. Create and release a mini program 109
1.4.1.4.7. Start mini program 111
1.4.1.4.8. Access real-device preview and debugging 111
1.4.1.4.9. Customize bi-directional channel 114

1.4.1.4.10. Customize startup loading page 116
1.4.1.4.11. Custom navigation bar 119
1.4.2. Flutter Project Access Guide 128
1.4.3. Integrate Mini Program into iOS 129
1.4.3.1. Quick start 129
1.4.3.2. Advanced guide 137
1.4.3.2.1. Real-device preview and debugging of iOS Mi... 137
1.4.3.2.2. Customize navigation bar of iOS Mini Program 138
1.4.3.2.3. Custom bidirectional channel for iOS Mini Pro... 140
1.4.3.2.4. Customize startup loading page of iOS Mini P... 140
1.4.3.2.5. Customize error page for iOS mini program 142
1.4.3.2.6. iOS Mini Program supports custom View 143
1.4.3.2.7. API permission extended configuration for iOS... 146
1.4.3.2.8. Preset an iOS mini program in the client 147
1.4.3.2.9. Add extended information for iOS Mini Progra... 148
1.4.3.2.10. Pass startup parameters to iOS Mini Program 149
1.4.3.3. Mini Program upgrade instructions 149
1.5. Develop Mini Program 150
1.5.1. Quick start 150
1.5.2. Advanced guide 155
1.5.2.1. Real-device preview and debugging 155
1.5.2.2. Extensions 155
1.5.2.3. Unregister custom event 159
1.5.2.4. Performance listener for mini programs 160
1.5.2.5. Performance optimization suggestions 164
1.6. Mini Program Development Server API 166
1.6.1. Overview and preparation 166
1.6.2. APIs description 168

1.7. Mini Program Base Library 191
1.8. Frameworks 194
1.8.1. Overview 194
1.8.2. App 195
1.8.3. Page 199
1.8.4. View layer 203
1.8.5. Event 209
1.8.6. Style 212
1.8.7. Global configuration for mini program 213
1.8.7.1. About global configuration for Mini Program 213
1.8.7.2. app-json global configuration 214
1.8.7.3. app acss global style 216
1.8.7.4. app.js registering Mini Program 216
1.8.7.5. getApp method 218
1.8.8. Mini Program page 218
1.8.8.1. About Mini Program Page 218
1.8.8.2. Page configuration 219
1.8.8.3. Page structure 219
1.8.8.4. Page style 219
1.8.8.5. Page registration 220
1.8.8.6. getcurrentpages method 228
1.8.9. AXML 228
1.8.9.1. About AXML 228
1.8.9.2. Data binding 228
1.8.9.3. Conditional rendering 231
1.8.9.4. List rendering 231
1.8.9.5. Template 233
1.8.9.6. Reference 234

1.8.10. SJS syntax reference 235
1.8.10.1. About SJS 235
1.8.10.2. Variable 237
1.8.10.3. Comments 237
1.8.10.4. Operator 237
1.8.10.5. Statement 239
1.8.10.6. Data type 241
1.8.10.7. Basic library 246
1.8.10.8. esnext 248
1.8.11. ACSS syntax reference 249
1.8.12. Event system 250
1.8.12.1. About event 250
1.8.12.2. Event object 252
1.8.13. Custom components 253
1.8.13.1. About custom component 253
1.8.13.2. Create custom component 253
1.8.13.3. Component configuration 254
1.8.13.4. Component template and style 254
1.8.13.5. Component object 257
1.8.13.6. Component lifecycle 260
1.8.13.7. mixins 263
1.8.13.8. Obtain component instances by ref 263
1.8.13.9. Use custom component 264
1.8.13.10. Release custom component 265
1.8.14. Performance optimization suggestions 266
1.9. Basic Components 267
1.9.1. Component overview 268
1.9.2. Component FAQ 270

1.9.3. View container 270
1.9.3.1. View 270
1.9.3.2. Swiper 271
1.9.3.3. Scroll view 273
1.9.3.4. Cover view 275
1.9.3.5. Movable view 275
1.9.4. Basic content 276
1.9.4.1. Text 276
1.9.4.2. Icon 277
1.9.4.3. Progressbar 278
1.9.4.4. Rich text 279
1.9.5. Form component 282
1.9.5.1. button 282
1.9.5.2. form 283
1.9.5.3. label 284
1.9.5.4. input 285
1.9.5.5. textarea 287
1.9.5.6. radio 288
1.9.5.7. checkbox 289
1.9.5.8. switch 291
1.9.5.9. slider 292
1.9.5.10. picker-view 293
1.9.5.11. Bottom scroll selector 294
1.9.6. Navigator 295
1.9.7. Media component 296
1.9.7.1. image 296
1.9.7.2. video 301
1.9.8. Canvas 306

1.9.9. Map 308
1.9.10. Open components 315
1.9.10.1. web-view 315
1.10. Mini Program Extended Component antd-mini 318
1.10.1. Overview 318
1.10.2. Common 318
1.10.2.1. Button 318
1.10.2.2. Icon 323
1.10.3. Navigation 327
1.10.3.1. Tabs 327
1.10.3.2. VTabs 336
1.10.4. Information display 339
1.10.4.1. Avatar 339
1.10.4.2. Collapse 341
1.10.4.3. Container 346
1.10.4.4. FloatPanel 347
1.10.4.5. List 352
1.10.4.6. Steps 355
1.10.4.7. SwipeAction 359
1.10.4.8. Tag 365
1.10.5. Information input 366
1.10.5.1. Checkbox 366
1.10.5.2. CheckboxGroup 369
1.10.5.3. Checklist 372
1.10.5.4. Filter 376
1.10.5.5. Input 380
1.10.5.6. Picker 383
1.10.5.7. RadioGroup 386

1.10.5.8. SearchBar 389
1.10.5.9. Selector 392
1.10.5.10. Stepper 396
1.10.5.11. Switch 398
1.10.5.12. Terms 399
1.10.6. Feedback 402
1.10.6.1. Dialog 402
1.10.6.2. Loading 406
1.10.6.3. Mask 408
1.10.6.4. Modal 409
1.10.6.5. Popover 413
1.10.6.6. Popup 419
1.10.6.7. Result 423
1.10.6.8. Toast 424
1.10.7. Guide and Toast 427
1.10.7.1. Badge 427
1.10.7.2. NoticeBar 430
1.10.7.3. Tips 433
1.10.8. Experimental components 436
1.10.8.1. Form 436
1.10.8.2. SafeArea 448
1.11. API 449
1.11.1. Overview 449
1.11.2. API list 449
1.11.3. UI 457
1.11.3.1. Navigation bar 457
1.11.3.2. tabBar 461
1.11.3.3. Route 467

1.11.3.4. Feedback 473
1.11.3.5. Pull down refresh 481
1.11.3.6. Contact 483
1.11.3.7. Choose city 484
1.11.3.8. Choose date 492
1.11.3.9. Animation 494
1.11.3.10. Canvas 497
1.11.3.11. Keyboard 515
1.11.3.12. Scroll 515
1.11.3.13. Node query 516
1.11.3.14. Option selector 520
1.11.3.15. Multilevel select 522
1.11.3.16. Set background color 523
1.11.3.17. Set page pulldown 524
1.11.3.18. Settings 524
1.11.4. Multimedia 525
1.11.4.1. Image 525
1.11.4.2. Video 532
1.11.5. Cache 534
1.11.6. File 539
1.11.7. Network 542
1.11.8. Device 553
1.11.8.1. Can I use 553
1.11.8.2. Obtain base library version 554
1.11.8.3. System info 554
1.11.8.4. Network status 558
1.11.8.5. Clipboard 559
1.11.8.6. Shake 561

1.11.8.7. Vibrate 562
1.11.8.8. Accelerometer 564
1.11.8.9. Gyroscope 565
1.11.8.10. Compass 566
1.11.8.11. Make phone call 566
1.11.8.12. User captures screen event 567
1.11.8.13. Screen brightness 568
1.11.8.14. Add phone contacts 572
1.11.8.15. Scan codes 579
1.11.8.16. Bluetooth API overview 580
1.11.8.17. Bluetooth API list 584
1.11.8.18. Bluetooth API error codes list 611
1.11.8.19. Bluetooth API FAQ 612
1.11.9. Data security 613
1.11.10. Share 615
1.11.11. The current version type of the Mini Program 620
1.11.12. Custom analysis 621
1.11.13. Custom API 622
1.11.14. Mini Program jumping 623
1.11.15. Webview component control 624
1.11.16. Application-level events 625
1.11.16.1. my.onAppShow 625
1.11.16.2. my.offAppShow 626
1.11.16.3. my.onAppHide 626
1.11.16.4. my.offAppHide 626
1.11.16.5. my.onPageNotFound 627
1.11.16.6. my.offPageNotFound 627
1.11.16.7. my.onUnhandledRejection 628

1.11.16.8. my.offUnhandledRejection 628
1.11.16.9. my.onError 629
1.11.16.10. my.offError 629
1.11.16.11. my.onComponentError 630
1.11.16.12. my.offComponentError 630
1.12. Design Guide 631
1.12.1. Principles 631
1.12.1.1. Clarity 631
1.12.1.2. Efficiency 632
1.12.1.3. Security 633
1.12.2. Visual guidelines 634
1.12.2.1. Color 634
1.12.2.2. Font 634
1.12.2.3. Icon 634
1.12.3. Component guidelines 635
1.12.3.1. Navigation 635
1.12.3.2. Input information 637
1.12.3.3. Display information 639
1.12.3.4. Feedback 640
1.12.3.5. Gesture 644
1.12.3.6. Differentiated design 645
1.12.3.7. Combine components 646
1.12.3.8. aria-component 647
1.12.4. SSO specifications 649
1.12.4.1. SSO user authorization guide 649
1.12.4.2. SSO interface documentation 652
1.13. Deep Analysis of the Technical Architecture of mPaaS ... 654
1.14. FAQ 659

1.14.1. FAQ of using console 659
1.14.2. Mini Program FAQs 659
1.14.3. FAQ about mini programs 660
2.Mini Program Release 661
2.1. About Mini Program Release 661
2.2. Configure Mini Program package 661
2.3. Create a Mini Program package 661
2.4. Release a Mini Program package 663
2.5. Manage Mini Program package 663
2.6. Mini Program permission control 664
3.Mini Program Analysis 666
3.1. About Mini Program Analysis 666
3.2. Mini program management 666
3.3. Data overview 667
3.4. User analytics 668
3.5. Page analytics 669
3.6. Sharing analytics 670
4.Mini Program Monitoring 671
5.Mini Program Open Platform (Public Preview) 673
5.1. Introduction to the Mini Program Open Platform 673
5.2. Developer backend operation manual 673
5.2.1. Register for the developer console 673
5.2.2. Quick logon 674
5.2.3. Homepage 674
5.2.4. Member Management 674
5.2.5. Mini Program management 676
5.2.5.1. My Mini Program 676
5.2.5.1.1. Overview 676

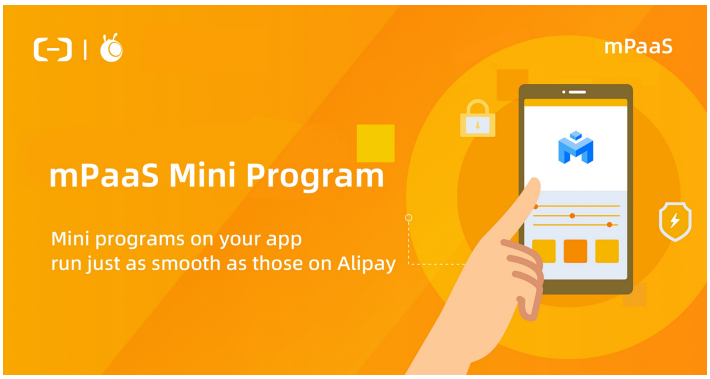
5.2.5.1.2. Create a mini program 676
5.2.5.1.3. View mini program information details 676
5.2.5.1.4. Mini program version management 676
5.2.5.1.5. Mini Program code package management 677
5.2.5.1.6. Mini Program Audit History 678
5.2.5.1.7. Mini program phased release 678
5.2.5.1.8. Custom API management 678
5.2.5.1.9. Edit a mini program 678
5.2.5.1.10. Enable/disable mini programs 679
5.2.5.2. H5 application 679
5.2.5.2.1. Overview 679
5.2.5.2.2. Create an H5 application 679
5.2.5.2.3. View H5 Application Details 679
5.2.5.2.4. Manage H5 application version 680
5.2.5.2.5. H5 Application Code Package Management 681
5.2.5.2.6. H5 Application Audit History 682
5.2.5.2.7. Grayscale Release of H5 Application 682
5.2.5.2.8. Edit H5 application 682
5.2.5.2.9. Enable/Deactivate H5 Application 682
5.2.5.3. My mini game 682
5.2.5.3.1. Overview 682
5.2.5.3.2. Add a mini game 682
5.2.5.3.3. View details of mini games on the intelligent... 682
5.2.5.3.4. Mini game version management 683
5.2.5.3.5. Mini game package management 685
5.2.5.3.6. Mini game review history 685
5.2.5.3.7. Mini game qualification verification 685
5.2.5.3.8. Mini game phased release 685

5.2.5.3.9. Custom API management 685
5.2.5.3.10. Edit mini game 686
5.2.5.3.11. Enable/disable mini games 686
5.2.6. Development configuration 686
5.2.6.1. Release rule management 686
5.2.6.2. Whitelist management 687
5.2.6.3. Domain name management 687
5.2.6.4. IDE Configuration Management 687
5.2.7. Operation logs 688
5.2.8. Data Statistics 688
5.2.8.1. Mini Program management 688
5.2.8.2. Data overview 689
5.2.8.3. User Analysis 690
5.2.8.4. Page analysis 690
5.2.8.5. Share analysis 691
5.3. Operations management console user guide 691
5.3.1. Log on 691
5.3.2. Basic management 691
5.3.2.1. Member Management 692
5.3.2.2. Role Management 692
5.3.2.3. Review configuration 693
5.3.2.3.1. On-shelf review 693
5.3.3. Developer group management 693
5.3.4. Mini Program management 693
5.3.4.1. Mini program listing approval 693
5.3.4.2. Domain name management 694
5.3.4.3. Custom API request review 694
5.3.4.4. Mini game qualification verification 695

5.3.5. Application Management 695
5.3.5.1. Host applications 695
5.3.5.2. Custom API 697
5.3.6. Information publishing center 698
5.3.7. Data statistics 698
5.3.7.1. Mini Program management 698
5.3.7.2. Data overview 699
5.3.7.3. User Analysis 700
5.3.7.4. Page analysis 700
5.3.7.5. Share analysis 701
5.3.8. Operation logs 701
5.3.9. Custom settings 701

1.Mini Program Development

1.1. About 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.

- **Comprehensively realize refined operations on the App side**

  Monitor App operation data such as crashe, stuck, lag, battery percentage, and traffic. Provides multiple indicator statistics such as user activation, login, and new additions, and supports multi-dimensional analysis and comparison by platform, version, region, time, etc. Provides application log diagnosis, including individual user diagnosis and diagnostic log collection. Quickly integrates mobile terminal message push function to maintain interaction with users, thereby effectively improving user retention rate. Provides personalized advertising and event delivery within the App, and customizes the delivery population to help the App reach users accurately and timely, so as to achieve the purpose of promoting activation, retention, and business growth.

How to use



1. Activate the service. Activate mPaaS, then 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 Mriver Tiny App into Android

1.2.1. Quick start

Mriver Tiny App is only provided in the baseline version of mobile PaaS (mPaaS) 10.2.3, and only supports mPaaS native AAR integration. For more information, see About native AAR integration.

Prerequisites

Before you integrate Mriver Tiny App into Android, make sure that you have activated mPaaS and integrated mPaaS using native AAR integration. Please refer to Prerequisites.

Procedure

The procedure of integrating Mriver Tiny App contains the following steps:

1. Select the baseline.
   i. Add the baseline 10.2.3.
   ii. Add the mini program component.
2. Initialize configuration.
   i. Initialize mPaaS.
   ii. Configure mini program signature verification.
   iii. Use mPaaS kernel.
3. Release a mini program.
   i. Enter the mini program backend.
   ii. Configure a virtual domain name.
   iii. Create a mini program.
   iv. Release the mini program.
4. Start the mini program.

The following section describes each step in details.

Select the baseline

1. Add the baseline 10.2.3.

   Click **mPaaS** > **Native AAR Integration**. In the popup window, click **Start Configuration** under Integrate/Upgrade Baseline and add the 10.2.3 baseline.

2. Add the mini program component.

   Install the Mriver Tiny App component through Component Management (AAR) in the project.

Initialize configuration

Initialize mPaaS

Initialize mPaaS using the mPaaS framework (recommended)

1. Add the following initialization code to `Application`.

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();
        // mPaaS initialization
        MP.init(this);
    }
}
```

   For more details, see Initialize mPaaS.

2. Add the meta configuration to `AndroidManifest.xml`.

```
<meta-data
    android:name="mpaas.init.param"
    android:value="com.xxx.xxx.MriverInitImpl" />
```

3. Add `com.xxx.xxx.MriverInitImpl` to implement `MPInitParamManifest`.

```
import android.content.Context;

import com.mpaas.core.MPInitParam;
import com.mpaas.mriver.api.init.MriverInitParam;

public class MriverInitImpl implements com.mpaas.MPInitParamManifest {
    @Override
    public MPInitParam initParam(Context context) {
        MriverInitParam mriverInitParam = MriverInitParam.getDefault();
        mriverInitParam.setMriverInitCallback(new MriverInitParam.MriverInitCallback() {
            @Override
            public void onInit() {
                if (com.alibaba.ariver.kernel.common.utils.ProcessUtils.isMainProcess()) {
                    // 小程序相关配置，比如自定义jsapi，titlebar等

                }
            }

            @Override
            public void onError(Exception e) {

            }
        });
        return MPInitParam.obtain().addComponentInitParam(mriverInitParam);
    }
}
```

Initialize mPaaS using MPInit

Initialize mPaaS by adding the following code to `Application`.

```
public class MyApplication extends Application implements MPInitParam.MPCallback {
    @Override
    public void onCreate() {
        super.onCreate();
        // Initializes mPaaS.
        MriverInitParam mriverInitParam = MriverInitParam.getDefault();
        mriverInitParam.setMriverInitCallback(new MriverInitCallback() {
            @Override
            public void onInit() {
                if (com.alibaba.ariver.kernel.common.utils.ProcessUtils.isMainProcess()) {
                    // Configure the Mriver Tiny App, such as customizing jsapi and titlebar.

                }
            }

            @Override
            public void onError(Exception e) {

            }
        });
        MP.init(this, MPInitParam.obtain().setCallback(this).addComponentInitParam(mriverInitParam));
    }

    @Override
    public void onInit() {
        // init success
    }
}
```

Configure mini program signature verification

Mriver Tiny App provides the feature of package signature verification. By default, the signature verification of the debug package and is disabled, and that of the release package is enabled. The signature verification can be controlled by an API.

```
// Disable signature verification.
MriverResource.disableVerify();

// Enable signature verification, where xx is the public key corresponding to the private key configured in the backend.
MriverResource.enableVerify(MriverResource.VERIFY_TYPE_YES,"xx");
```

> **Note**
> Before you release a mini program, we recommend that you enable signature verification. For specific operations about how to configure mini program signature verification, see Configure Mini Program package.

Configure the interval for requesting a mini program **package**

mPaaS supports configuring the interval for requesting a mini program package, which can be controlled by an API.

```
Mriver.setConfig("h5_nbmngconfig", "{\"config\":{\"al\":\"3\",\"pr\":{\"4\":\"86400\","
    + "\"common\":\"864000\"},\"ur\":\"1800\",\"fpr\":{\"common\":\"3888000\"}},"
    + "\"switch\":\"yes\"}");
```

Here, `\"ur\":\"1800\"` specifies the global update interval, where `1800` is the default value in seconds. Modify the value to set the interval for requesting a global mini program package. The value ranges from 0 to 86400 seconds, that is, 0 to 24 hours. 0 indicates that the interval is not limited.

> **Important**
> Keep other parameters unchanged when possible.

Use mPaaS kernel

The same-layer rendering capability of the mPaaS kernel based on the mini program can improve the experience of embedded WebView, maps, videos and other components. For the integration and application of the mPaaS kernel, please refer to Integrate mPaaS kernel.

Release a mini program

Before you start a mini program, you must release the mini program in the mPaaS console. Perform the following steps:

1. Enter the mini program backend. Log on to the mPaaS console, and select the target app. In the left-side navigation pane, choose **Mini Program** > **Mini Program Release**.
2. Configure a virtual domain name. To configure a virtual domain name for the first time, choose **Mini Program** > **Mini Program Release** > **Manage configuration**. The virtual domain name can be any of your domain names. We recommend that you use your enterprise domain name, such `example.com`.
3. Create a mini program. Log on to the mPaaS console and perform the following steps:
   i. In the left-side navigation pane, choose **Mini Program** > **Mini Program Release**.
   ii. On the mini program list page, click **Create**.
   iii. In the **Create Mini Program** window, enter the mini program ID and name, and then click **OK**. The mini program ID is a 16-digit number, for example 2018080616290001.
   iv. On the mini program list, find the new mini program, and click **Add**.

v. In the Basic information section, set the following parameters:
- Version: Enter 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 the client app within the range can start the mini program. The minimum version can be `0.0.0` while the maximum version can be left empty, indicating that the mini program can start regardless of the client version.

> **Note**
> Enter the version of the Android client, not that of the mini program.

- Icon: Click **Select file** to upload the icon of the mini program package. The icon is required when you create a mini program package for the first time. Sample icon:



- File: Upload the resource file of the mini program package, which must be a `.zip` file. Here is an mPaaS mini program demo (click here to download), and you can upload it directly.

> **Note**
> Before you upload it, rename the `zip` file and its folder to the 16-digit ID of your mini program.

vi. In the Configuration information section, set the following parameters:
- Entry URL: Required. It is the homepage of the mini program. The format is `/index.html#xxx/xxx/xxx/xxx`, where `xxx/xxx/xxx/xxx` following `#` is the first value under `pages` in `app.json`. The entry of the mPaaS mini program demo is `/index.html#page/tabBar/component/index`.
- For other parameters, retain the default values.

vii. Check **I confirm that the information provided is true and correct**.

viii. Click **Submit**.

4. Release the mini program. Log on to the mPaaS console and perform the following steps:

i. In the left-side navigation pane, choose **Mini Program > Mini Program Release > Mini Program Official Package Management**.

ii. On the mini program list page, select the target mini program and version, and click **Create release task**.

iii. In the Create release task section, set the following parameters:
- Release type: Select **Official**.
- Release description: Optional.

Click **OK** to create the release task.

Start the mini program

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

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

> **Note**
> The `2018080616290001` in the code refers to the mini program ID. The ID here is for reference only, so enter the actual program ID.

1.2.2. Start a mini program

This topic describes the APIs to start a mini program and demonstrates how to start a mini program.

APIs

Mriver.startApp(String appid)

This API is used to redirect to a mini program.

Sample code

```
Mriver.startApp("2021022320210223");
```

Parameter description

Parameter	Type	Description	Required
appId	String	The AppID of the target mini program to be redirected.	Yes

Mriver.startApp(Activity activity,String appid)

This API is used to redirect to a mini program. We recommend redirecting to the activity page where the mini program starts.

Sample code

```
Mriver.startApp(activity, "2021022320210223");
```

Parameter description

Parameter	Type	Description	Required
appId	String	The AppID of the target mini program to be redirected.	Yes
activity	Activity	The activity where the mini program starts.	No (but we recommend you specify it)

Mriver.startApp(Activity activity,String appid,Bundle bundle)

This API is used to redirect to a mini program. We recommend redirecting to the activity page where the mini program starts.

Sample code

```
Bundle bundle = new Bundle();
bundle.putString("page", "pages/index/index");// Set the path.
bundle.putString("query", "name=123&pwd=456");// Set the parameters.
Mriver.startApp(activity, "2021022320210223", bundle);
```

Parameter description

Parameter	Type	Description	Required
appId	String	The AppID of the target mini program to be redirected.	Yes
activity	Activity	The activity where the mini program starts.	No (but we recommend you specify it)
bundle	Bundle	The parameters to start the mini program.	No

Start a mini program

Start a mini program and pass custom parameters

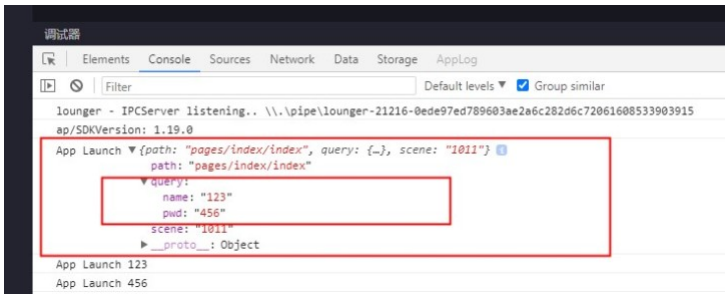
`Mriver.startApp(Activity activity,String appid,Bundle bundle)`

In certain scenarios, you need to pass parameters to the default receiving page of a mini program. The path is `pages/index/index`. This topic describes the scenario by passing the `name` and `pwd` parameters as an example.

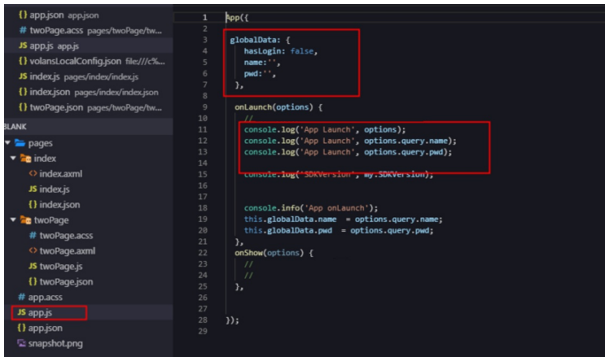
1. Add the parameters for the page to be redirected during startup on the client.

   ```
   Bundle bundle = new Bundle();
   bundle.putString("query", "name=123&pwd=456");// Set the parameters.
   Mriver.startApp(activity, "2021022320210223", bundle);
   ```

2. The `query` field is used to pass parameters in a URL. The `query` field is parsed to obtain the parameters. Description of `startApp`:
   - appId: The app ID of the mini program, which can be viewed in the mPaaS console.
   - bundle: Pass request parameters to the `Bundle` object in the format of `key="query"`, and `value="key-value pair"`. Separate parameters with `&`.
3. Obtain the parameters from the `options` in the `onLaunch/onShow(options)` method in the mini program.



When the `app.js` file is saved, it obtains the parameters passed from the client to the mini program and stores them to the `globalData` global variable. To use them, obtain the values from `globalData` or update the values. Obtain the values directly after parameters such as `token` and `user_id` in the request header are passed from `Native` and stored in `globalData`.



Start a mini program and redirect to a specified page

`Mriver.startApp(Activity activity,String appid,Bundle bundle)`

Redirect to a specified page of the mini program. If not specified, the page redirects to the configured homepage path by default.

1. Add the parameters for the page to be redirected during startup on the client.

```
Bundle bundle = new Bundle();
bundle.putString("page", "pages/index/index");// Set the path.
Mriver.startApp(activity, "2021022320210223", bundle);
```

2. When you pass URL parameters, the `query` field is used. To obtain the parameters, parse the `query` field. Description of `startApp`:
   - activity: The `activity` page where the mini program starts.
   - appId: The app ID of the mini program, which can be viewed in the mPaaS console.
   - bundle: The `Bundle` object. Pass request parameters to the `Bundle` object in the format of `key="page"` , `value="the Mriver Tiny App path to be opened"` .

FAQs

Q: Why is `activity` passed when a mini program starts?

A: If you do not pass the `activity` object, the container uses `ApplicationLifecycle` and records `activity` with weak reference. If the memory is insufficient, weak reference is recycled by JVM, and the `activity` object is `null` . As a result, `activity.startActivity` inside the container fails to start the mini program.

1.2.3. Installation and debugging

1.2.3.1. Real-device preview and debugging

Mini Program IDE supports real-device preview and debugging, so you can preview the actual effect of the current code or debug on your mobile client. This topic describes how to preview and debug a mini program on a physical device, and describes the APIs used in the process.

Prerequisites

Before you use the preview and debugging feature, make sure that you have integrated this feature into your Android mini program. For information about how to integrate this feature, see MriverDebug API.

Procedure

1. Open the Mini Program IDE configuration file named `config.json` that you downloaded from the **mPaaS console** and find the `debug_url` field.An example 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"
}
```

2. Set the debugging address through `setWssHost` and add `/host/` at the end. The example is as follows.

```
MriverDebug.setWssHost("wss://cn-hangzhou-mproxy.cloud.alipay.com/host/");
```

3. Click **Preview** or **Real Device Debugging** in the upper right corner of the IDE.
   i. The IDE will generate a `.zip` package of the current code and upload it to the console.
   ii. The console automatically creates a publishing task, generates a QR code and returns it to the IDE.

> **Note**
> Failure to properly set up the whitelist may cause QR code building and generation to fail. See Whitelist settings for more information.

4. Use the mobile client to scan the QR code displayed in the IDE. After scanning the code, the console will be triggered to deliver the mini program package. The QR code is valid for 15 minutes, and a **refresh** button will be displayed after timeout.
5. After the mobile client receives the mini program package, it can enter the preview or debugging interface on the mobile client.

APIs

The following section describes the APIs and parameters used during preview and debugging, and provides sample code.

MriverDebug.debugAppByScan(Activity activity)

This API is used for preview and debugging. Use mPaaS's built-in code scanning and preview mini programs.

Sample code

```
MriverDebug.debugAppByScan(MainActivity.this);
```

Parameter description

Parameter	Type	Description	Required
activity	Activity	The current activity.	Yes

MriverDebug.debugAppByUri(Activtiy activity,Uri uri)

This interface is used to jump to the mini program. It is recommended to jump to the activity of the page where the mini program is started.

Sample code

```
MriverDebug.debugAppByUri(MainActivity.this,intent.getData());
```

Parameter description

Parameter	Type	Description	Required
activity	Activity	The current activity.	No (but we recommend you specify it)
uri	Uri	The data returned by scanning the QR code.	No

1.2.4. Engine management

1.2.4.1. Custom APIs

This topic describes custom APIs from the following two aspects:

- The mini program calls a custom API from the client.
- The client sends a custom event to the mini program.

The mini program calls a custom API from the client

Perform the following steps:

1. Customize an API on the client.
   - The custom class inherits from `SimpleBridgeExtension`.

     ```
     public class CustomApiBridgeExtension extends SimpleBridgeExtension{
     }
     ```

     > **Important**
     > Do not confuse the class with and the method.

   - Customize a method. Sample code:

     ```
     public class CustomApiBridgeExtension extends SimpleBridgeExtension{
         @ActionFilter
         public void tinyToNative(@BindingApiContext ApiContext apiContext,
                                  @BindingRequest JSONObject params,
                                  @BindingParam("param1") String param1,
                                  @BindingParam("param2") String param2,
                                  @BindingCallback BridgeCallback callback) {
             ...
         }
     }
     ```

     - `tinyToNative` indicates the call method in the mini program, which needs to be added to `@ActionFilter`.
     - The parameters in the method are passed from the mini program and some context and bridge on the client. Add or delete the parameters as needed. For more information, see Customize method parameters.

   - Generate a JSONObject result and return the result to the mini program. Sample code:

```
public class CustomApiBridgeExtension extends SimpleBridgeExtension{
    @ActionFilter
    public void tinyToNative(@BindingApiContext ApiContext apiContext,
                             @BindingRequest JSONObject params,
                             @BindingParam("param1") String param1,
                             @BindingParam("param2") String param2,
                             @BindingCallback BridgeCallback callback) {
        ...
        JSONObject result = BridgeResponse.SUCCESS.get();
        result.put("custom_message","value");
  // Return the result to the Mriver Tiny App.
        callback.sendJSONResponse(result);
    }
}
```

Sample code remarks:

- If the JSONObject result is generated, call:

  ```
  JSONObject result = BridgeResponse.SUCCESS.get();
  ```

  If the JSONObject result is not generated, call:

  ```
  JSONObject result = BridgeResponse.Error.newError(errorCode,errorMessage).get();
  ```

- Add custom parameters to the result.

  ```
  result.put("custom_message","value");
  ```

- The callback that returns the final JSONObject result to the mini program is `@BindingCallback BridgeCallback callback` in the custom method.

  ```
  callback.sendJSONResponse(result);
  ```

- The custom parameters are described as follows:
  - `@BindingId`: The String type. Pass in the workId of the current communication.
  - `@BindingNode`: The Scope type, which can be App.class or Page.class. Pass in the current app or page according to the framework type.
  - `@BindingApiContext`: The ApiContext type. Use the instance to obtain the context of the current API.
  - `@BindingExecutor`: The Executor type. Select a different Executor and execute the corresponding logic in its thread. The following table shows the name, description, and priority of the Executors.

Executor type	Description	Priority
ExecutorType.SYNC	Execute directly without switching the thread to facilitate encapsulation.	None
ExecutorType.UI	Execute the backend task with the highest priority that the frontend UI depends on. Queuing is not allowed.	`{@link Thread#NORM_PRIORITY}`
ExecutorType.URGENT_DISPLAY	Execute the backend task with the highest priority that the frontend UI depends on. Queuing is not allowed.	`{@link Thread#MAX_PRIORITY}`
ExecutorType.URGENT	Execute the backend task with the highest priority that the frontend UI depends on. Queuing is not allowed.	`{@link Thread#NORM_PRIORITY}`
ExecutorType.NORMAL	Execute the backend task that is normal and not urgent. Queuing is allowed.	`{@link Thread#MIN_PRIORITY}`
ExecutorType.IO	Execute the task of persistent file I/O operations. The time consumption can be estimated. Either the task succeeds or an exception occurs.	`{@link Thread#MIN_PRIORITY}`
ExecutorType.NETWORK	Execute the backend task related to the network. The time consumed depends on the condition. A typical scenario is that an RPC request is initiated.	`{@link Thread#MIN_PRIORITY}`
ExecutorType.IDLE	The idle thread pool, which is used for time-insensitive tasks such as tracking points. The task can be executed in a single thread pool.	None

  - `@BindingRequest`: The JSONObject type. The parameters from the mini program are passed in as JSONObject.
  - `@BindingParam`: The String type, which corresponds to the custom parameter key in the mini program. The parameter value types include string, int, long, float, double, and boolean. The default value can be customized.

    ```
    @BindingParam(value = "stringParam", stringDefault = "default") String stringParam
    @BindingParam(value = "intParam", intDefault = 1) int intParam
    @BindingParam(value = "longParam", longDefault = 9223372036854775807L) long longParam
    @BindingParam(value = "floatParam", floatDefault = 1.0F) float floatParam
    @BindingParam(value = "doubleParam", doubleDefault = 1.79769313486231570e+308d) double doubleParam
    @BindingParam(value = "booleanParam", booleanDefault = true) boolean booleanParam
    ```

  - `@BindingCallback`: The BridgeCallback type. The result can be sent to the mini program using BridgeCallback.

- Sample code

```
public class CustomApiBridgeExtension extends SimpleBridgeExtension {

    private static final String TAG = "CustomApiBridgeExtension";

    @ActionFilter
    public void tinyToNative(@BindingId String id,
                             @BindingNode(App.class) App app,
                             @BindingNode(Page.class) Page page,
                             @BindingApiContext ApiContext apiContext,
                             @BindingExecutor(ExecutorType.UI) Executor executor,
                             @BindingRequest JSONObject params,
                             @BindingParam("param1") String param1,
                             @BindingParam("param2") String param2,
                             @BindingCallback BridgeCallback callback) {
        RVLogger.d(TAG, "id: "+id+
                "\napp: "+app.toString()+
                "\npage: "+page.toString()+
                "\napiContext: "+apiContext.toString()+
                "\nexecutor: "+executor.toString());
        RVLogger.d(TAG, JSONUtils.toString(params));
        JSONObject result = BridgeResponse.SUCCESS.get();
        result.put("message", "the client receives parameters: " + param1 + ", " + param2 + "\n return the current package name of the dem
o: " + apiContext.getActivity().getPackageName());
        // Return the result to the Mriver Tiny App.
        callback.sendJSONResponse(result);
    }

}
```

2. Register the custom API on the client. After the container is initialized, you can call the registration method to register the custom API. Do not confuse class names.

```
MriverEngine.registerBridge(CustomApiBridgeExtension.class);
```

3. Call the API in the mini program. Sample code:

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

Sample code remarks:

- The first parameter is action, which is the same as the annotation in the custom `@ActionFilter` method on the client.
- The second parameter is a custom one. In the sample code, the client can receive the parameters using `@BindingRequest JSONObject` or `@BindingParam("param1") String param1`, in which the param1 and param2 values are strings composed of letters and digits. Here, param1 and param2 are only examples. The value types of custom parameters include string, int, long, boolean, float, and double.
- The third parameter is the client callback.

The client sends a custom event to the mini program

Perform the following steps:

1. Register an event in the mini program.

```
// The first parameter is the name of the custom event, and the second parameter is the callback.
my.on('nativeToTiny', (res) => {
    my.showToast({
        type: 'none',
        content: JSON.stringify(res),
        duration: 3000,
        success: () => {},
        fail: () => {},
        complete: () => {}
    });
})
```

2. Send the event on the client.

```
// Simulate the data sent.
JSONObject jo = new JSONObject();
jo.put("index", index);
AppManager appManager = RVProxy.get(AppManager.class);
if (null != appManager) {
    // Obtain the instance of the currently running Mriver Tiny App through AppManager.
    App app = appManager.findAppByAppId("2018080616290001");
    if(null!=app){
        // Send information to the Mriver Tiny App.
        // The first parameter is the current activity page of the Mriver Tiny App, the second parameter is the name of the custom event, an
d the third parameter is the data sent.
        MriverEngine.sendToRender(app.getActivePage(), "nativeToTiny", jo, null);
    }
}
```

1.2.5. Application management

1.2.5.1. Obtain the stack and information of a mini program

To obtain the stack and information of a mini program, perform the following steps:

1. Create the `CustomApiBridgeExtension` class to inherit the `SimpleBridgeExtension` class. Sample code:

```
public class CustomApiBridgeExtension extends SimpleBridgeExtension {

    private static final String TAG = "CustomApiBridgeExtension";

    @ActionFilter
    public void tinyToNative(@BindingId String id,
                             @BindingNode(App.class) App app,
                             @BindingNode(Page.class) Page page,
                             @BindingApiContext ApiContext apiContext,
                             @BindingExecutor(ExecutorType.UI) Executor executor,
                             @BindingRequest JSONObject params,
                             @BindingParam("param1") String param1,
                             @BindingParam("param2") String param2,
                             @BindingCallback BridgeCallback callback) {
        RVLogger.d(TAG, "id: " + id +
                "\napp: " + app.toString() +
                "\npage: " + page.toString() +
                "\napiContext: " + apiContext.toString() +
                "\nexecutor: " + executor.toString());
        RVLogger.d(TAG, JSONUtils.toString(params));

        JSONObject result = BridgeResponse.SUCCESS.get();

        // Return the result to the Mriver Tiny App.

        Stack stack = MriverApp.getAppStack();
        Enumeration enumerationLists = stack.elements();


        JSONArray jsonArray = new JSONArray();
        while (enumerationLists.hasMoreElements()) {
            JSONObject jsonObject = new JSONObject();
            MRApp o = (MRApp) enumerationLists.nextElement();
            jsonObject.put("AppId", o.getAppId());
            jsonObject.put("AppVersion", o.getAppVersion());
            jsonArray.add(jsonObject);
        }
        String tinyappStr = jsonArray.toJSONString();
        // result.put("message", "the client receives parameters: " + param1 + ", " + param2 + "\n return the current package name of the de
mo: " + apiContext.getActivity().getPackageName());
        result.put("message", tinyappStr);
        callback.sendJSONResponse(result);
    }

}
```

2. Register the class before you start the mini program. Sample code:

```
MriverEngine.registerBridge(CustomApiBridgeExtension.class);
```

3. Start the mini program. Sample code:

```
Mriver.startApp(activity, "2021042620210426");
```

1.2.6. Resource management

1.2.6.1. Remote management

This topic introduces the methods of remote management to actively update specified mini programs, actively update all mini programs, and download mini programs.

Update a mini program

MriverResource.updateApp(String appid)

Update a specific mini program. Sample code:

```
MriverResource.updateApp("2021042520210425", new UpdateAppCallback() {
    @Override
    public void onSuccess(List<AppModel> list) {
        showToast("the Mriver Tiny App with appid=2021042520210425 is updated");
    }

    @Override
    public void onError(UpdateAppException e) {
        showToast(e.getMessage());
    }
});
```

Download a Mriver Tiny App

MriverResource.downloadAppPackage(String appId)

Download a specific mini program without a callback function. Sample code:

```
MriverResource.downloadAppPackage("2021042520210425")
```

MriverResource.downloadAppPackage(String appId, PackageDownloadCallback callback)

Download a specific mini program with a callback function. Sample code:

```
MriverResource.downloadAppPackage("2021042520210425", new PackageDownloadCallback() {
        @Override
        public void onPrepare(String s) {
           // Print logs or do other work.
        }

        @Override
        public void onProgress(String s, int i) {
            // Display the download progress.
            showToast("i="+i);
        }

        @Override
        public void onCancel(String s) {
           // Cancel the internal network library.
        }

        @Override
        public void onFinish(@Nullable String s) {
            showToast(s);
        }

        @Override
        public void onFailed(String s, int i, String s1) {
            showToast("onFailed--"+s);
        }
    });
```

1.2.6.2. Mini program package verification

This topic describes how to enable and disable mini program package verification.

By default, the signature verification of the debug package is disabled, and that of the release package is enabled. The signature verification can be controlled by an API.

Enable mini program verification

Sample code:

```
MriverResource.enableVerify(MriverResource.VERIFY_TYPE_YES, "xxx")
```

Disable mini program verification

Sample code:

```
MriverResource.disableVerify();
```

1.2.7. Permission control

1.2.7.1. Mini program permission

Some mini program APIs such as location, camera, and album APIs prompt the user to authorize access before the APIs are called.

Mriver Tiny App allows you to extend API calls as follows:

1. Customize text prompt, and control the text and display style.
2. Configure read and write permission.

> **Note**
> The extension works only when Mini Program permission control is enabled in the backend.

Configure permission

For an API that requires user authorization, you must configure a permission key. Multiple APIs can share the same key. For example, picture selection and code scanning can use the same camera key.

Mini program provides default keys and the corresponding APIs, as shown in the following table:

Permission	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 following permission configuration class passed in by the access party to process API call permissions:

```
public static class Config {
    public String action; // The API name.
    public String scope; // The permission key.
    public String desc; // The text displayed for calling the API.
    public Config() {
    }
}
```

> **Note**
> When **action** is **chooseImage** or **chooseVideo**, the key configured in Mriver Tiny App is invalid. The container has special logic to handle the two APIs, but text can be customized.

Load configurations

To load permission configurations, you need to use the **ApiPermissionConfigProxy** API provided by the container.

API class:

```
package com.mpaas.mriver.base.proxy;

import com.alibaba.ariver.kernel.common.Proxiable;
import com.mpaas.mriver.base.permission.Config;
import java.util.List;

public interface ApiPermissionConfigProxy {
    List<Config> getConfigurationList();
}
```

Call method:

```
// Custom configuration.
Mriver.setProxy(ApiPermissionConfigProxy.class, new CustomApiPermissionConfigProxy());
```

Sample code:

```
public class CustomApiPermissionConfigProxy implements ApiPermissionConfigProxy {
    @Override
    public List<Config> getConfigurationList() {
        List<Config> permissionList = new ArrayList<>();
        permissionList.add(new Config("saveFile", "file", "use your file storage"));
        permissionList.add(new Config("getFileInfo", "file", "use your file storage"));
        return permissionList;
    }
}
```

Customize display

Customize authorization information and allow the user to confirm access. Procedure:

1. Customize display.

   ```
   Mriver.setProxy(LocalPermissionDialogProxy.class, new CustomAuthDialogProxy());
   ```

2. When the user accepts or rejects the call, call the method in the PermissionPermitListener API. Sample code:

```
public class CustomAuthDialogProxy implements LocalPermissionDialogProxy {
    @Override
    public LocalPermissionDialog create(Context context) {
        return new CustomPermissionDialog(context);
    }

    public static class CustomPermissionDialog implements LocalPermissionDialog {
        private AUNoticeDialog mDialog;

        private final Context mContext;

        private PermissionPermitListener mPermissionPermitListener;

        public CustomPermissionDialog(Context context) {
            mContext = context;
        }

        @Override
        public void setDialogContent(String content, String title, String icon) {
            mDialog = new AUNoticeDialog(mContext, title, content,
                "accept",
                "reject");
            mDialog.setPositiveListener(new AUNoticeDialog.OnClickPositiveListener() {
                @Override
                public void onClick() {
                    if (mPermissionPermitListener != null) {
                        mPermissionPermitListener.onSuccess();
                    }
                }
            });
            mDialog.setNegativeListener(new AUNoticeDialog.OnClickNegativeListener() {
                @Override
                public void onClick() {
                    if (mPermissionPermitListener != null) {
                        mPermissionPermitListener.onFailed(-1, "", true);
                    }
                }
            });
        }

        @Override
        public void setPermissionPermitListener(PermissionPermitListener permissionPermitListener) {
            mPermissionPermitListener = permissionPermitListener;
        }

        @Override
        public void show() {
            if (mDialog != null && mContext instanceof Activity) {
                if (!((Activity) mContext).isFinishing()) {
                    mDialog.show();
                }
            }
        }
    }
```

Read and write to configurations

To read configurations, call the following method:

```
RVProxy.get(ApiPermissionConfigManagerProxy.class).getAllPermissionStates(appId);
```

> **Note**
>
> - Mini program configurations are stored in app and user dimensions, so make sure that the app has called the**MPLogger.setUserId** method.
> - If an API is not called, the authorization status of the API key value cannot be obtained.

To write to configurations, call the following method:

```
RVProxy.get(ApiPermissionConfigManagerProxy.class).setPermissionState(appId, key, true);
```

Sample code:

```
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 = RVProxy.get(ApiPermissionConfigManagerProxy.class).getAllPermissionStates(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) {
                            RVProxy.get(ApiPermissionConfigManagerProxy.class).setPermissionState(appId, key, isChecked);
                        }
                    });
                    mScrollView.addView(item, new ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
H5Utils.dip2px(PermissionDisplayActivity.this, 48)));
                }
            }
        });
    }
}
```

1.2.8. Interface documents

1.2.8.1. API

Initialize an API

Method 1

1. You can use the mPaaS infrastructure component initialization method to set additional `mriver` parameters.

```
// Initialization
public class MyApplication extends Application {

    @Override
    protected void attachBaseContext(Context base) {
        super.attachBaseContext(base);
        // mPaaS initialization callback settings
        QuinoxlessFramework.setup(this, new IInitCallback() {
            @Override
            public void onPostInit() {
                // Initialize the logic of other mPaaS components

            }
        });
    }

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

2. Add the `meta` configuration to the `AndroidManifest.xml` :

```
<meta-data
    android:name="mpaas.init.param"
    android:value="com.mpaas.demo.MriverInitImpl" />
```

3. Add `com.mpaas.demo.MriverInitImpl` classes to implement `MPInitParamManifest`.

```
public class MriverInitImpl implements com.mpaas.MPInitParamManifest {
    @Override
    public MPInitParam initParam() {
        MriverInitParam mriverInitParam = createInitParams();
        return MPInitParam.obtain().addComponentInitParam(mriverInitParam);
    }
}
```

Method 2

Initialize the API by using the `MPInit.init(Application application, MPInitParam param)` method of `MPInit`.

```
MP.init(this, createInitParams());

MPInitParam createInitParams() {
        final MriverInitParam mriverInitParam = MriverInitParam.getDefault();
        mriverInitParam.setMriverInitCallback(new MriverInitParam.MriverInitCallback() {
            @Override
            public void onInit() {
                MPLogger.setUserId("MPTestCase");
                if (com.alibaba.ariver.kernel.common.utils.ProcessUtils.isMainProcess()) {
                    // Configure the mini program, such as JSAPI and titlebar.
                    Log.i("MriverApp", "init1");
                    Mriver.setConfig("mr_use_inner_net", "YES");
                    Mriver.setConfig("mr_request_support_gzip", "true");
                    Mriver.setConfig("mr_showShareMenuItem", "YES");
                    Mriver.setConfig("mriver_openlocation_hidden_default", "0");
                    Mriver.setConfig("mriver_support_chooseFile", "YES");
                    Mriver.setConfig("ta_worker_init_low_version_compat", (Build.VERSION.SDK_INT == 22 || Build.VERSION.SDK_INT == 21) ? "YES"
: "NO");
                    MriverEngine.registerBridge(ShareApiBridgeExtension.class);
                    MriverEngine.registerBridge(SnapshotScreenApiBridgeExtension.class);
                    Mriver.enableAPM();

                    List<String> miniAppPoint = new ArrayList<>();
                    miniAppPoint.add(PageResumePoint.class.getName());
                    miniAppPoint.add(PageEnterPoint.class.getName());
                    Mriver.registerPoint(PageLifeCycleExtension.class.getName(), miniAppPoint);

                    RVProxy.set(PrepareNotifyProxy.class, new PrepareNotifyProxy() {

                        @Override
                        public void notify(String s, PrepareStatus prepareStatus) {

                        }

                        @Override
                        public void apmEvent(final String s, final String s1, final String s2, final String s3, final String s4) {
                            mUIHandler.post(new Runnable() {
                                @Override
                                public void run() {
                                    if (APMActivity.logs == null) {
                                        APMActivity.logs = new StringBuilder();
                                    }
                                    APMActivity.logs.append(s).append(" ").append(s1).append(" ").append(s2).append(" ").append(s3).append(" ").
append(s4).append("\n");
                                    if (TextUtils.equals(s, "MiniAppStart")
                                    || TextUtils.equals(s, "MiniPage_Load_T2")) {
                                        Toast.makeText(MRiverApp.sApp, "startTime: " + s4, Toast.LENGTH_SHORT).show();
                                    }
                                }
                            });

                        }
                    });
                }

                Log.i(TAG, "registerPlugin");
                MPNebula.registerH5Plugin(
                        PagePlugin.class.getName(),
                        null,
                        "page",
                        new String[]{"myapi2", H5Plugin.CommonEvents.H5_SESSION_EXIT, H5Plugin.CommonEvents.H5_PAGE_CLOSED,
H5Plugin.CommonEvents.H5_PAGE_FINISHED, H5Plugin.CommonEvents.H5_PAGE_SHOULD_LOAD_URL}
                );


                MriverEngine.enableDebugConsole();

                Mriver.setConfig("mriver_show_debug_menu_all", "YES");
                Log.i("TTAATT", "hasInited");
            }

            @Override
            public void onError(Exception e) {
```

```
                Log.i("MriverApp", "init2");
            }
        });
        mriverInitParam.setUCInitCallback(new MriverInitParam.UCInitCallback() {
            @Override
            public void onInit() {
                sHasUCInit = true;
                Log.i("TTAATT", "hasInitedUC");
            }

            @Override
            public void onError(Exception e) {

            }
        });
        return MPInitParam.obtain().setCallback(this).addComponentInitParam(mriverInitParam);
    }
```

Startup and Configuration

Mriver Framework Global API

Method	Description	Must be called
void setUserId(String userId)	Set the `userId` for debugging and preview.	Yes
void startApp(String appId)	Start the mini program as `appId`.	Yes
void startApp(Activity activity, String appId, Bundle startParams)	Start the mini program according to the `appId` and parameters.	
void setConfig(String key, String value)	The configuration of the mini program container. For more information about the configuration parameters, see Switch configuration.	No
void forcePermissionCheck()	Forced permission verification. After you enable this feature, a pop-up window appears when mini programs call permission-related APIs.	No
void setProxy(Class<T> proxyClazz, T proxyImpl)	Configure the container proxy. For more information about the custom logic, see Function configuration.	No
void registerPoint(String className, List<String> pointsClassName)	Listen to various facets of the container. For more information, seeFunction configuration.	No

MriverResource Resource Management

Method	Description	Must be called
void updateApp(Map<String, String> appList, UpdateAppCallback callback)	Update a specific mini program. The `appList` is the mini program ID and local version number. The background configuration is downloaded and automatically downloaded and installed after the update is complete.	As needed, you can update in advance to optimize the first load performance.
void updateApp(String appId, String targetVersion, UpdateAppCallback callback)	Updates to the specified version.	As needed
void downloadAppPackage(String appId)	Download the mini program.	As needed
AppModel getAppModel(String appId)	You can call this operation to obtain information about an existing local mini program.	As needed
void enableVerify(String type, String publicRsa)	Enable signature verification. • `debug`: Disable by default. • `release`: enabled by default.	Must be called, turn on or off depending on the choice
void disableVerify()	Disable the package experience tab.	
void deleteApp(String appId)	Deletes a local mini program.	As needed
void enableAPM()	Enable APM reporting.	As needed

Map<String, List<AppModel>> getAllApp()	Obtains all local mini programs.	As needed

MriverEngine API

Method	Description	Must be called
void registerBridge(Class<? extends BridgeExtension> bridgeClass)	Custom JSAPI.	As needed
void sendToRender(Page page, String event, @Nullable JSONObject data, @Nullable SendToRenderCallback callback)	Native sends a message to the mini program.	As needed
void setUserAgent(final String customUa)	Set `useragent`.	As needed
void enableDebugConsole	Open the debug panel.	As needed

MriverDebug API

Method	Description	Must be called
void setWssHost(String wssHost)	Set the WSS address for real machine debugging.	You must call the
void debugAppByScan(Activity activity)	Preview /real machine debugging scan code.	You must call the
void debugAppByScan(final Activity activity, Bundle bundle)	Preview /real machine debugging scan code, and add startup parameters.	

1.2.8.2. Switch configuration

key	value	Description
enable_keep_alive	YES/NO, default NO	Enable the keep-alive feature
mriver_keep_alive_time	The number of milliseconds. Default value: 60000.	Backstage keep-alive duration
mriver_keepalive_max	1 to 5. Default value: 2.	Maximum number of background keep-alive mini programs
mriver_force_perm_check	YES/NO, default NO	Forcibly enable permission verification
mriver_custom_appxloading	The resource address. This parameter is empty by default.	The custom URL of the appending operation.
mriver_custom_appxloading_size	10 to 60. Default value: 30.	Customizes the size of the appxloading control. 30 indicates that the width and height are 30% of the screen width.
enable_back_perform	YES/NO, default NO	Turn on the hold return key
mr_showOptionMenu	YES/NO. Default value: YES	titlebar display menu
mr_showShareMenuItem	YES/NO, default NO	Menu bar display sharing
mr_request_support_gzip	YES/NO, default NO	httpRequest JSAPI support for gzip
h5_nbmngconfig	json, default :{\"config\":{\"al\":\"3\",\"pr\":{\"4\":\"86400\"," + "\"common\":\"864000\"},\"ur\":\"1800\",\"fpr\":{\"common\":\"3888000\"}}," + "\"switch\":\"yes\"}	The mini program automatically updates the cycle and changes the ur value. The default is 1800 seconds (half an hour)
miniapp_location_enable_search	YES/NO, default NO	Select a location to support search

miniapp_location_enable_all_search	YES/NO, default NO	Select a location to support nationwide search
ariver_appcompact_compatible	true/false. Default value: false.	The mini program page switching animation is compatible with androidx,appcompact 1.3.+
ta_systeminfo_update_setting	YES/NO, default NO	Whether getSystemInfo obtains permission application information
mriver_show_debug_menu_all	YES/NO, default NO	All mini programs show debug panel entry
mr_use_inner_net	YES/NO, default NO	Mini program httpRequest parallelism optimization
mriver_check_foreground_change	YES/NO, default NO	Whether to check keep alive when returning to the desktop for recovery, multi-stack keep alive does not need to be set
ariver_supportStatusBar	true/false. Default value: true.	By default, the color of the statusBar is set to false for immersion.
mr_tiny_video_type	3 indicates that the new video player is used. Default value: 0	The type of the video player. If you set this parameter to 3, the new version of the player is used.
ta_worker_init_low_version_compat	YES/NO, default NO	(Build.VERSION.SDK_INT == 22 || Build.VERSION.SDK_INT == 21) ? "YES" : "NO"

1.2.8.3. Function configuration

Support for experience edition and non-experience edition

1. Enable the function.

```
Mriver.setConfig("mr_experience_required", "YES");
```

2. Open the mini program experience/test version.

```
MriverResource.de leteApp(appId); // Delete the local version.

Bundle bundle = new Bundle();
bundle.putString(RVStartParams.LONG_NB_UPDATE, "synctry"); // The version is forcibly updated. You can use this parameter in combination.
bundle.putInt(RVStartParams.LONG_NB_EXPERIENCE_REQUIRED, 1); // 1 indicates the experience version. If no parameter is passed, the official version is used.
Mriver.startApp(this, appId, bundle);
```

Support mini program page return query dialog box

1. Turn on the return interception switch.

```
Mriver.setConfig("enable_back_perform", "YES");
```

2. Upgrade the Appx version.

```
// Add the value to build.gradle.
api ('com.mpaas.mriver:mriverappxplus-build:2.7.18.20230130001@aar') {
      force=true
}
```

3. Call related APIs during mini program development.

```
// Enable basic retouching.
my.enableAlertBeforeUnload({
      message: 'Are you sure you want to leave this page?',
});

// Disable basic retouching.
my.disableAlertBeforeUnload()
```

Real machine debug /preview

```
MriverDebug.setWssHost("Real WSS address");
MriverDebug.debugAppByScan(activity);
```

Custom title bar

```
Mriver.setProxy(TitleViewFactoryProxy.class, new TitleViewFactoryProxy() {
               @Override
               public ITitleView createTitle(Context context, App app) {
                   return new CustomTitleView(context);
               }
           });

public class CustomTitleView implements ITitleView, View.OnClickListener {
```

```
public static final String TAG = MRConstants.INTEGRATION_TAG + ":MRTitleView";
protected TextView tvTitle;
protected ImageView ivImageTitle;
protected ImageView btBack;
protected TextView btBackToHome;
protected RelativeLayout rlTitle;
protected View statusBarAdjustView;

protected List<ImageButton> btIconList = new ArrayList<>();


// The container view of the entire TitleBar.
protected TitleBarFrameLayout contentView;

// The number of OptionMenu items in the upper-right corner. Default value: 1.
protected int visibleOptionNum;
protected Page mPage;

// Bottom line separator
protected View mDivider;
protected Context mContext;

protected TitleViewIconSpec mTitleViewIconSpec;

protected TitleViewStyleSpec mDarkStyleSpec;

protected TitleViewStyleSpec mLightStyleSpec;

//    protected  ProgressBar mNavLoadingBar;
protected ITitleEventDispatcher mTitleEventDispatcher;

public CustomTitleView(Context context) {
    mContext = context;
    ViewGroup parent = null;
    if (context instanceof Activity && ((Activity) context).getWindow() != null) {
        parent = ((Activity) mContext).findViewById(android.R.id.content);
    }

    mTitleViewIconSpec = TitleViewSpecProvider.g().getIconSpec();
    mDarkStyleSpec = TitleViewSpecProvider.g().getDarkSpec();
    mLightStyleSpec = TitleViewSpecProvider.g().getLightSpec();

    contentView = (TitleBarFrameLayout) LayoutInflater.from(context).inflate(R.layout.mriver_title_bar_demo, parent, false);
    tvTitle = contentView.findViewById(R.id.h5_tv_title);
    ivImageTitle = contentView.findViewById(R.id.h5_tv_title_img);
    statusBarAdjustView = contentView.findViewById(R.id.h5_status_bar_adjust_view);
    ivImageTitle.setVisibility(View.GONE);
    tvTitle.setOnClickListener(this);
    ivImageTitle.setOnClickListener(this);

    btBack = contentView.findViewById(R.id.h5_tv_nav_back);
    btBackToHome = contentView.findViewById(R.id.h5_tv_nav_back_to_home);


    mDivider = contentView.findViewById(R.id.h5_h_divider_intitle);

    rlTitle = contentView.findViewById(R.id.h5_rl_title);
    visibleOptionNum = 1;

    // ad view
    //        adViewLayout.setTag(H5Utils.TRANSPARENT_AD_VIEW_TAG);

    btBack.setOnClickListener(this);
    btBackToHome.setOnClickListener(this);

    applyViewStyleAndIcon();

}

protected void applyViewStyleAndIcon() {
    boolean useBackSpec = false;
    boolean useHomeSpec = false;
    if (mTitleViewIconSpec != null) {
        TitleViewIconSpec.IconSpecEntry btHomeSpec = mTitleViewIconSpec.getHomeButton();
        if (btHomeSpec != null) {
            btBackToHome.setTypeface(btHomeSpec.getKey());
            btBackToHome.setText(btHomeSpec.getValue());
            useHomeSpec = true;
        }

    }

    if (!useHomeSpec) {
        Typeface iconFont = Typeface.createFromAsset(mContext.getAssets(), "mrv_iconfont.ttf");
        btBackToHome.setTypeface(iconFont);
    }
```

```
    btBackToHome.setTextColor(StateListUtils.getStateColor(mLightStyleSpec.getHomeButtonColor()));
}


protected void setButtonIcon(Bitmap btIcon, int index) {
    if (isOutOfBound(index, btIconList.size())) {
        return;
    }
    btIconList.get(index).setImageBitmap(btIcon);
}

@Override
public void setTitle(String title) {
    if (title != null && enableSetTitle(title)) {
        tvTitle.setText(title);
        tvTitle.setVisibility(View.VISIBLE);
        ivImageTitle.setVisibility(View.GONE);
    }
}

protected boolean enableSetTitle(String title) {
    return !title.startsWith("http://") && !title.startsWith("https://");
}

// view visible control
protected boolean isOutOfBound(int num, int length) {
    return length == 0 || length < num;
}

@Override
public void showBackButton(boolean show) {
    btBack.setVisibility(show ? View.VISIBLE : View.GONE);
    if (show && btBackToHome != null) {
        btBackToHome.setVisibility(View.GONE);
    }
    addLeftMarginOnTitle();
}

@Override
public void showOptionMenu(boolean b) {

}

public void showHomeButton(boolean show) {
    btBackToHome.setVisibility(show ? View.VISIBLE : View.GONE);
    if (show) {
        btBack.setVisibility(View.GONE);
    }
    addLeftMarginOnTitle();
}

@Override
public void setTitleEventDispatcher(ITitleEventDispatcher dispatcher) {
    mTitleEventDispatcher = dispatcher;
}

@Override
public void addCapsuleButtonGroup(View view) {
    if (view == null) {
        return;
    }
}

protected void addLeftMarginOnTitle() {
    boolean needAdd = btBack.getVisibility() != View.VISIBLE &&
        btBackToHome.getVisibility() != View.VISIBLE;
    RelativeLayout.LayoutParams rlTitleLayoutParams =
        (RelativeLayout.LayoutParams) rlTitle.getLayoutParams();
    rlTitleLayoutParams.setMargins(!needAdd ? 0 : DimensionUtil.dip2px(mContext, 16), 0, 0, 0);

}

@Override
public void showTitleLoading(boolean show) {
}

@Override
public View getContentView() {
    return contentView;
}

@Override
public void onClick(View view) {
    RVLogger.d(TAG, "onClick " + view);
    if (mPage == null) {
        return;
```

```
        }
        if (view.equals(btBack)) {
            if (mTitleEventDispatcher != null) {
                mTitleEventDispatcher.onBackPressed();
            }
        } else if (view.equals(tvTitle) || view.equals(ivImageTitle)) {
            if (mTitleEventDispatcher != null) {
                mTitleEventDispatcher.onTitleClick();
            }
        } else if (view.equals(btBackToHome)) {
            if (mTitleEventDispatcher != null) {
                mTitleEventDispatcher.onHomeClick();
            }
        }
    }

    @Override
    public void setPage(Page page) {
        mPage = page;
        tvTitle.setOnLongClickListener(new View.OnLongClickListener() {
            @Override
            public boolean onLongClick(View v) {
                mPage.getApp().restartFromServer(null);
                return false;
            }
        });
    }


    public View getDivider() {
        return mDivider;
    }

    protected void switchToLightTheme() {
        tvTitle.setTextColor(mLightStyleSpec.getTitleTextColor());

        btBackToHome.setTextColor(StateListUtils.getStateColor(mLightStyleSpec.getHomeButtonColor()));

    }

    protected void switchToDarkTheme() {
        tvTitle.setTextColor(mDarkStyleSpec.getTitleTextColor());

        btBackToHome.setTextColor(StateListUtils.getStateColor(mDarkStyleSpec.getHomeButtonColor()));


    }

    public void onRelease() {
        btIconList.clear();
    }


    /***
    * Turn on immersive status bar support
    */
    @Override
    public void setStatusBarColor(int color) {
        if (StatusBarUtils.isSupport()) {
            int statusBarHeight = StatusBarUtils.getStatusBarHeight(mContext);

            If (statusBarHeight == 0) { // protection, in case rom cannot get the height of the status bar, it will not take effect here.
                return;
            }
            LinearLayout.LayoutParams layoutParams =
                (LinearLayout.LayoutParams) statusBarAdjustView.getLayoutParams();
            layoutParams.height = statusBarHeight;
            statusBarAdjustView.setLayoutParams(layoutParams);
            statusBarAdjustView.setVisibility(View.VISIBLE);

            try {
                StatusBarUtils.setTransparentColor((Activity) mContext, color);
            } catch (Exception e) {
                RVLogger.e(TAG, e);
            }
        }
    }

    @Override
    public void setBackgroundColor(int color) {
        contentView.getContentBgView().setColor(color);
    }

    @Override
    public void setAlpha(int alpha, boolean titleTextAlphaEnabled) {
        contentView.getContentBgView().setAlpha(alpha);
        if (titleTextAlphaEnabled) {
```

```
            tvTitle.setAlpha(alpha);
        }
    }

    @Override
    public void setOptionMenu(Bitmap bitmap) {
        visibleOptionNum = 2;
        setButtonIcon(bitmap, 1);
    }

    @Override
    public void setTitleImage(Bitmap image, String contentDesc) {
        if (!TextUtils.isEmpty(contentDesc)) {
            ivImageTitle.setContentDescription(contentDesc);
        }
        if (image != null) {
            RVLogger.d(TAG, "imgTitle width " + image.getWidth() + ", imgTitle height " + image
                    .getHeight());
            ivImageTitle.setImageBitmap(image);
            ivImageTitle.setVisibility(View.VISIBLE);
            tvTitle.setVisibility(View.GONE);
            RVLogger.d(TAG, "ivImageTitle width " + ivImageTitle
                    .getWidth() + ", ivImageTitle height " + ivImageTitle.getHeight());
        }
    }

    @Override
    public void setTitlePenetrate(boolean enable) {
        contentView.setPreventTouchEvent(!enable);
    }

    @Override
    public void applyTheme(TitleBarTheme theme) {
        if (theme == TitleBarTheme.DARK) {
            switchToDarkTheme();
        } else if (theme == TitleBarTheme.LIGHT) {
            switchToLightTheme();
        }
    }
}
```

Custom mini program loading animation

```
Mriver.setProxy(SplashViewFactoryProxy.class, new SplashViewFactoryProxy() {

                @Override
                public ISplashView createSplashView(Context context) {
                    return new CustomLoadingView(context);
                }
            });

public class CustomLoadingView extends FrameLayout implements ISplashView {

    private static final String TAG = "CustomLoadingView";

    private static final int defaultAlphaColor=855638016;//Color.argb(51, 0, 0, 0);// Default transparent color
    private static final long TIME_DELAY_FOR_SHOW_PERCENTAGE=2000; // Show the percentage of the delay time 2s.

    public final static String MSG_UPDATE_APPEARANCE = "UPDATE_APPEARANCE";
    public final static String DATA_UPDATE_APPEARANCE_BG_COLOR = "UPDATE_APPEARANCE_BG_COLOR"; // Page background color# RGB
    public final static String DATA_UPDATE_APPEARANCE_LOADING_ICON = "UPDATE_APPEARANCE_LOADING_ICON"; // Loading icon Drawable
    public final static String DATA_UPDATE_APPEARANCE_LOADING_TEXT = "UPDATE_APPEARANCE_LOADING_TEXT"; //loading copy
    public final static String DATA_UPDATE_APPEARANCE_LOADING_TEXT_COLOR = "UPDATE_APPEARANCE_LOADING_TEXT_COLOR"; //loading copy color# RGB
    public final static String DATA_UPDATE_APPEARANCE_LOADING_BOTTOM_TIP = "UPDATE_APPEARANCE_LOADING_BOTTOM_TIP"; // Prompt copy at bottom

    public final static String ANIMATION_STOP_LOADING_PREPARE = "ANIMATION_STOP_LOADING_PREPARE";

    private Context mContext;

    protected ImageView mLoadingIcon;
    protected TextView mLoadingTitle;
    protected TextView mLoadingPercentTip;
    protected TextView mBottomTip;
    protected TextView mBackButton;

    private Paint mDotPaint;
    private Timer mTimer;
    private TimerTask mTimerTask;
    private boolean mPlayingStartAnim;
    private int mDarkDotX;
    private int mDarkDotY;
    private int mDarkGap;
    private int mDotSize;
    private int mLightDotIndex = 0;
    private int mPercentValue;
    private long mStartLoadingTime = 0;
```

```
private OnCancelListener onCancelListener;
private Activity hostActivity;

public interface OnCancelListener {
    void onCancel();
}

public CustomLoadingView(Context context) {
    this(context, null);
}

public CustomLoadingView(Context context, AttributeSet attrs) {
    this(context, attrs, 0);
}

public CustomLoadingView(final Context context, AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);

    mContext = context;

    hostActivity = (Activity) context;

    initView();

    mBackButton.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View view) {
            cancel();
            if (context instanceof Activity) {
                RVLogger.d(TAG, "user want close app when splash loading");
                ((Activity) context).finish();
            }
        }
    });
}

public final void cancel() {
    if (this.onCancelListener != null) {
        this.onCancelListener.onCancel();
    }

}

public void initView() {
    mLoadingIcon = new ImageView(mContext);
    mLoadingIcon.setScaleType(ImageView.ScaleType.FIT_XY);
    mLoadingIcon.setImageResource(R.drawable.ic_launcher_foreground);
    mLoadingTitle = new TextView(mContext);
    mLoadingTitle.setGravity(Gravity.CENTER);
    mLoadingTitle.setTextColor(Color.BLACK);
    mLoadingTitle.setSingleLine();
    mLoadingTitle.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 18);
    mLoadingTitle.setEllipsize(TextUtils.TruncateAt.END);
    ViewGroup.LayoutParams lp = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT);
    mLoadingTitle.setLayoutParams(lp);
    addView(mLoadingIcon);
    addView(mLoadingTitle);

    mBackButton = new TextView(mContext);
    mBackButton.setGravity(Gravity.CENTER);
    addView(mBackButton);

    // Load percentage
    mPercentValue = 0;
    mLoadingPercentTip = new TextView(mContext);
    mLoadingPercentTip.setGravity(Gravity.CENTER);
    mLoadingPercentTip.setSingleLine();
    mLoadingPercentTip.setTextSize(TypedValue.COMPLEX_UNIT_DIP, 12);
    mLoadingPercentTip.setEllipsize(TextUtils.TruncateAt.END);
    lp = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT);
    mLoadingPercentTip.setLayoutParams(lp);
    mLoadingPercentTip.setText("");
    addView(mLoadingPercentTip);

    mBottomTip = new TextView(mContext);
    mBottomTip.setTextSize(12);
    mBottomTip.setGravity(Gravity.CENTER);
    lp = new ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT, ViewGroup.LayoutParams.WRAP_CONTENT);
    mBottomTip.setLayoutParams(lp);
    addView(mBottomTip);

    mDotSize = 30;
    mDotPaint = new Paint();
    mDotPaint.setStyle(Paint.Style.FILL);
    mDarkGap = 10;
```

```
}

@Override
protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
    int size = 150;
    mLoadingIcon.measure(makeMeasureSpec(size), makeMeasureSpec(size));

    int height = 200;
    int width = 500;
    mLoadingTitle.measure(MeasureSpec.makeMeasureSpec(width, MeasureSpec.AT_MOST), makeMeasureSpec(height));

    height = 200;
    width = 500;
    mLoadingPercentTip.measure(MeasureSpec.makeMeasureSpec(width, MeasureSpec.AT_MOST), makeMeasureSpec(height));

    width = 200;
    height = 100;
    mBottomTip.measure(makeMeasureSpec(width), MeasureSpec.makeMeasureSpec(height, MeasureSpec.AT_MOST));

    width = 200;
    height = 200;
    mBackButton.measure(makeMeasureSpec(width), makeMeasureSpec(height));

    setMeasuredDimension(widthMeasureSpec, heightMeasureSpec);
}

@Override
protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
    int offsetX = 0;
    int offsetY = 0;

    mBackButton.layout(offsetX, offsetY, mBackButton.getMeasuredWidth(), mBackButton.getMeasuredHeight() + offsetY);

    offsetX = (getMeasuredWidth() - mLoadingIcon.getMeasuredWidth()) / 2;
    mLoadingIcon.layout(offsetX, offsetY, offsetX + mLoadingIcon.getMeasuredWidth(),
            offsetY + mLoadingIcon.getMeasuredHeight());

    offsetX = (getMeasuredWidth() - mLoadingTitle.getMeasuredWidth()) / 2;
    offsetY = offsetY + mLoadingIcon.getMeasuredHeight();
    mLoadingTitle.layout(offsetX, offsetY, offsetX + mLoadingTitle.getMeasuredWidth(),
            offsetY + mLoadingTitle.getMeasuredHeight());

    mDarkDotX = getMeasuredWidth() / 2 - mDotSize - mDarkGap;
    mDarkDotY = offsetY + mLoadingTitle.getMeasuredHeight();

    offsetX = (getMeasuredWidth() - mLoadingPercentTip.getMeasuredWidth()) / 2;
    offsetY = offsetY + mLoadingPercentTip.getMeasuredHeight();
    mLoadingPercentTip.layout(offsetX, offsetY, offsetX + mLoadingPercentTip.getMeasuredWidth(),
            offsetY + mLoadingPercentTip.getMeasuredHeight());

    offsetX = (getMeasuredWidth() - mBottomTip.getMeasuredWidth()) / 2;
    offsetY = getMeasuredHeight() - mBottomTip.getMeasuredHeight();
    mBottomTip.layout(offsetX, offsetY, offsetX + mBottomTip.getMeasuredWidth(), offsetY + mBottomTip.getMeasuredHeight());
}

@Override
protected void dispatchDraw(Canvas canvas) {
    super.dispatchDraw(canvas);
    if (mPlayingStartAnim) {
        mDotPaint.setColor(Color.BLACK);
        mDarkDotX = getMeasuredWidth() / 2 - mDotSize - mDarkGap;
        for (int i = 0; i < 3; i++) {
            mDotPaint.setColor(mLightDotIndex == i ? Color.WHITE : Color.BLACK);
            canvas.drawCircle(mDarkDotX, mDarkDotY, mDotSize / 2, mDotPaint);
            mDarkDotX = mDarkDotX + mDarkGap + mDotSize;
        }
    }
}

@Override
public boolean onTouchEvent(MotionEvent ev) {
    super.onTouchEvent(ev);
    return true;
}

public void startLoadingAnimation() {
    if (mPlayingStartAnim) return;
    mPlayingStartAnim = true;

    if (mTimerTask == null) {
        mTimerTask = new TimerTask() {
            @Override
            public void run() {
                mLightDotIndex++;
                if (mLightDotIndex > 2) {
                    mLightDotIndex = 0;
```

```
                }
                ExecutorUtils.runOnMain(new Runnable() {
                    @Override
                    public void run() {
                        invalidate();
                        // Update the value of the percentage.
                        if (isCanShowPercentage()) {
                            if (mPercentValue == 0) {
                                mPercentValue = 52;
                            } else if (mPercentValue < 99) {
                                mPercentValue++;
                            }
                            mLoadingPercentTip.setText(String.format("%d%%", mPercentValue));
                        }

                    }
                });
            }
        };
    }

    if (mTimer == null) {
        try {
            mTimer = new Timer();
            mTimer.schedule(mTimerTask, 0, 200);
        } catch (Throwable throwable) {
            RVLogger.e(TAG, "printMonitor error", throwable);
        }
    }

    RVLogger.d(TAG, "SplashLoadingView... startLoading Animation");
}

public void stopLoadingAnimation() {
    mPlayingStartAnim = false;

    if (mTimer != null) {
        mTimer.cancel();
    }
    if (mTimerTask != null) {
        mTimerTask.cancel();
    }
    invalidate();

    RVLogger.d(TAG, "SplashLoadingView... stopLoading Animation");
}

private int getDimen(int id) {
    return mContext.getResources().getDimensionPixelSize(id);
}

private int makeMeasureSpec(int size) {
    return MeasureSpec.makeMeasureSpec(size, MeasureSpec.EXACTLY);
}

public void onStart() {
    updateStatusBar();
    startLoadingAnimation();
}

public void onStop() {
    stopLoadingAnimation();
    mLoadingPercentTip.setVisibility(GONE);
    RVLogger.d(TAG, "SplashLoadingView... stop");
}

@Override
public void onFail() {
    onStop();
    Map<String, Object> msgData = new HashMap<>();
    msgData.put(CustomLoadingView.DATA_UPDATE_APPEARANCE_LOADING_BOTTOM_TIP, "");
    sendMessage(CustomLoadingView.MSG_UPDATE_APPEARANCE, msgData);
}

public void onHandleMessage(String msg, Map<String, Object> data) {
    if (MSG_UPDATE_APPEARANCE.equals(msg)) {

        String bgColor = (String) data.get(DATA_UPDATE_APPEARANCE_BG_COLOR);
        if (!TextUtils.isEmpty(bgColor)) {
            setBackgroundColor(Color.parseColor(bgColor));
        }

        Drawable loadingIcon = (Drawable) data.get(DATA_UPDATE_APPEARANCE_LOADING_ICON);
        if (loadingIcon != null) {
            mLoadingIcon.setImageDrawable(loadingIcon);
        }
```

```
            String text = (String) data.get(DATA_UPDATE_APPEARANCE_LOADING_TEXT);
            if (text != null) {
                mLoadingTitle.setText(text);
            }

            String textColor = (String) data.get(DATA_UPDATE_APPEARANCE_LOADING_TEXT_COLOR);
            if (!TextUtils.isEmpty(textColor)) {
                mLoadingTitle.setTextColor(Color.parseColor(textColor));
            }

            String bottomTip = (String) data.get(DATA_UPDATE_APPEARANCE_LOADING_BOTTOM_TIP);
            if (bottomTip != null) {
                mBottomTip.setText(bottomTip);
            }
        }
    }

    public void performAnimation(final String animationType, final Animator.AnimatorListener animationListener) {
        if (Looper.myLooper() == Looper.getMainLooper()) {
            doPerformAnimation(animationType, animationListener);
        } else {
            post(new Runnable() {
                @Override
                public void run() {
                    doPerformAnimation(animationType, animationListener);
                }
            });
        }
    }

    private void doPerformAnimation(final String animationType, final Animator.AnimatorListener animationListener) {

        if (getParent() == null) {
            RVLogger.e(TAG, "loading view has not added to parent container");
            return;
        }

        if (ANIMATION_STOP_LOADING_PREPARE.equals(animationType)) {
            mPlayingStartAnim = false;

            int offsetTargetY = 0;
            float titleTargetX = 0f;
            if (isBackButtonVisible()) {
                titleTargetX = mBackButton.getX() + mBackButton.getMeasuredWidth();
            } else {
                titleTargetX = getTitleLeftMargin();
            }
            float titleTargetY = (200 - mLoadingTitle.getMeasuredHeight()) / 2;

            AnimatorSet prepareStopLoadingAnimator = new AnimatorSet();
            prepareStopLoadingAnimator.setDuration(400);
            if (animationListener != null) {
                prepareStopLoadingAnimator.addListener(animationListener);
            }
            prepareStopLoadingAnimator.play(ObjectAnimator.ofFloat(mLoadingIcon, "y", mLoadingIcon.getY(), offsetTargetY))
                    .with(ObjectAnimator.ofFloat(mLoadingIcon, "scaleX", mLoadingIcon.getScaleX(), 0))
                    .with(ObjectAnimator.ofFloat(mLoadingIcon, "scaleY", mLoadingIcon.getScaleY(), 0))
                    .with(ObjectAnimator.ofFloat(mLoadingTitle, "x", mLoadingTitle.getX(), titleTargetX))
                    .with(ObjectAnimator.ofFloat(mLoadingTitle, "y", mLoadingTitle.getY(), titleTargetY));

            prepareStopLoadingAnimator.start();
        } else {
            performAnimation(animationType, animationListener);
        }
    }

    @Override
    public void updateLoadingInfo(EntryInfo entryInfo) {
        Map<String, Object> msgData = new HashMap<>();
        msgData.put(CustomLoadingView.DATA_UPDATE_APPEARANCE_LOADING_TEXT, entryInfo.title);

        sendMessage(CustomLoadingView.MSG_UPDATE_APPEARANCE, msgData);

        H5ImageUtil.loadImage(entryInfo.iconUrl, null, new H5ImageListener() {
            @Override
            public void onImage(Bitmap bitmap) {
                RVLogger.d(TAG, "onBitmapLoaded!");
                Map<String, Object> msgData = new HashMap<>();
                int dimen = 100;
                Bitmap displayBitmap = ImageUtil.scaleBitmap(bitmap, dimen, dimen);
                msgData.put(CustomLoadingView.DATA_UPDATE_APPEARANCE_LOADING_ICON, new BitmapDrawable(displayBitmap));
                sendMessage(CustomLoadingView.MSG_UPDATE_APPEARANCE, msgData);
            }
        });
    }

    @Override
```

```
    public View getView() {
        return this;
    }

    @Override
    public void onExit() {
        performAnimation(CustomLoadingView.ANIMATION_STOP_LOADING_PREPARE, new Animator.AnimatorListener() {
            @Override
            public void onAnimationStart(Animator animation) {
                RVLogger.d(TAG, "onAnimationStart");
            }

            @Override
            public void onAnimationEnd(Animator animation) {
                RVLogger.d(TAG, "onAnimationEnd");
            }

            @Override
            public void onAnimationCancel(Animator animation) {
                RVLogger.d(TAG, "onAnimationCancel");
            }

            @Override
            public void onAnimationRepeat(Animator animation) {

            }
        });
    }


    private void updateStatusBar() {
        if (hostActivity != null && hostActivity.getClass().getName().equals("com.alipay.mobile.core.loading.impl.LoadingPage")) {
            StatusBarUtils.setTransparentColor(hostActivity, defaultAlphaColor);
        }
    }

    protected boolean isBackButtonVisible() {
        return true;
    }

    protected float getTitleLeftMargin() {
        return 0f;
    }

    private boolean isCanShowPercentage() {
        if (mStartLoadingTime == 0) {
            mStartLoadingTime = System.currentTimeMillis();
        }
        long time = System.currentTimeMillis();
        return ((time - mStartLoadingTime) > TIME_DELAY_FOR_SHOW_PERCENTAGE);
    }

    public final void sendMessage(final String msg, final Map<String, Object> data) {
        this.post(new Runnable() {
            public void run() {
                try {
                    CustomLoadingView.this.onHandleMessage(msg, data);
                } catch (Throwable e) {
                    RVLogger.e(TAG, e);
                }

            }
        });
    }
}
```

Supports the debug panel

```
// Call when Mriver is initialized. By default, only the preview and real-machine debugging mini programs are displayed.
MriverEngine.enableDebugConsole();

// Force all mini programs to display the debugging panel.
Mriver.setConfig("mriver_show_debug_menu_all", "YES");
```

Customize more menu bars

```
Mriver.setProxy(MRTinyMenuProxy.class, new MRTinyMenuProxy() {
                @Override
                public ITinyMenuPopupWindow createTinyMenuPopupWindow(Context context, TinyMenuViewModel tinyMenuViewModel) {
                    return new DemoTinyMenuPopupWindow(context, tinyMenuViewModel);
                }
            });

// DemoTinyMenuPopupWindow implementation references internal TinyMenuModalWindow
```

Listener interception return

```
Mriver.setConfig("enable_back_perform", "YES");
List<String> tt = new ArrayList<String>();
tt.add(BackInterceptPoint.class.getName());// The name of the interface class.
Mriver.registerPoint(DemoBackInterceptPointProviderImp.class.getName(), tt);


public class DemoBackInterceptPointProviderImp implements BackInterceptPoint {

    @Override
    public boolean intercepted(final Render render, int i, CommonBackPerform.BackHandler backHandler, GoBackCallback goBackCallback) {
        new Handler(Looper.getMainLooper()).post(new Runnable() {
            @Override
            public void run() {
                Toast.makeText(render.getActivity(), "Return key" ,Toast.LENGTH_LONG).show();
            }
        });
        return false; // The value true.
    }

    @Override
    public void onInitialized() {
        Log.i("BackPoint", "BackInterceptPoint--onInitialized--:");
    }

    @Override
    public void onFinalized() {
        Log.i("BackPoint", "BackInterceptPoint--onFinalized--:");
    }
}
```

Listener and interception disable

```
Mriver.setProxy(AppCloseInterceptProxy.class, new AppCloseInterceptProxy() {
 @Override
 public boolean intercept(Context context, Page page) {
 showToast("Close key");
 return false; // The value true.
 }
});
```

Custom appx loading animation (only GIF)

Add `"nonLoadingIndicator": false` in the `mini.project.json` file of the mini program.

The code example is as follows:

```
Mriver.registerPoint(MriverResourceInterceptor.class.getName(),
                    Arrays.asList("com.alibaba.ariver.resource.api.extension.ResourceInterceptPoint"));
            Mriver.setConfig("mriver_custom_appxloading", CUSTOM_LOADING_RESOURCE);
      // Loading size: the proportion of the screen width. 20 means the loading control size is 20% of the screen width.
            Mriver.setConfig("mriver_custom_appxloading_size", "20");
            ResourcePackage resourcePackage = new GlobalResourcePackage("00000001") {
                @Override
                protected boolean needWaitSetupWhenGet() {
                    return false;
                }

                @Override
                public boolean needWaitForSetup() {
                    return false;
                }

                @Override
                protected boolean canHotUpdate(String hotVersion) {
                    return false;
                }

                @Override
                public Resource get(ResourceQuery query) {
                   // All resources can be intercepted
                    if (TextUtils.equals(CUSTOM_LOADING_RESOURCE, query.pureUrl)) {
                        return getPresetImageResource(UIStyleActivity.this, query);
                    }
                    return null;
                }
            };
GlobalPackagePool.getInstance().add(resourcePackage);

private Resource getPresetImageResource(Context application, ResourceQuery query) {

    Resource sResource = null;
    if (sResource == null) {

        InputStream inputStream = null;
        AssetManager am = application.getAssets();
        try {
            inputStream = am.open("preset/custom_loading.gif");
            int length = inputStream.available();
            byte[] buffer = new byte[length];
            inputStream.read(buffer);
            sResource = new OfflineResource(query.pureUrl, buffer);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (inputStream != null) {
                try {
                    inputStream.close();
                } catch (Throwable t) {
                    t.printStackTrace();
                }
            }
        }
    }
    return sResource;
}
```

Resource management

```
// Delete local mini programs.
MriverResource.deleteApp("xxxx");

// Obtain the information about all mini programs.
Map<String, List<AppModel>> allApp = MriverResource.getAllApp();


// Actively update a specific mini program.
Map<String, String> updateApp = new HashMap<>();
            updateApp.put("xxx", "");
            MriverResource.updateApp(updateApp, new UpdateAppCallback() {
                @Override
                public void onSuccess(List<AppModel> list) {
                    showToast("The mini program with appid=2021042520210425 is updated successfully");
                }

                @Override
                public void onError(UpdateAppException e) {
                    showToast(e.getMessage());
                }
            });

// Actively download the mini program.
MriverResource.downloadAppPackage("xxx", new PackageDownloadCallback() {
                @Override
                public void onPrepare(String s) {
                    // Do some auxiliary work, such as logging
                }

                @Override
                public void onProgress(String s, int i) {
                    // The progress.
                    showToast("i=" + i);
                }

                @Override
                public void onCancel(String s) {
                    // The user does not need to worry. Cancellation is the cancellation api for the internal network library
                }

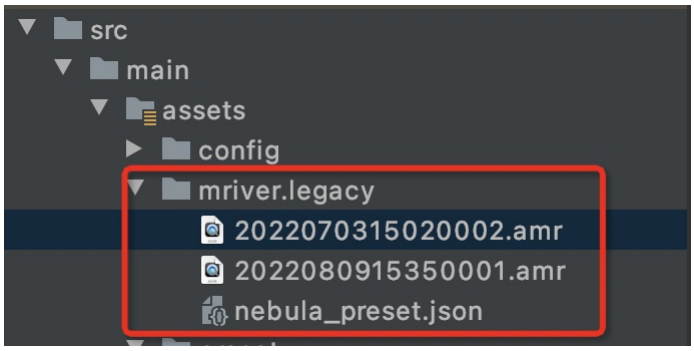
                @Override
                public void onFinish(String s) {
                    showToast(s);
                }

                @Override
                public void onFailed(String s, int i, String s1) {
                    showToast("onFailed--" + s);
                }
            });
```

Preset mini program

- Put the mini program `.amr` package and mini program information into the `assets/mriver/legacy` directory.

> **Important**
> The file name rule is: appId.amr. Do not include the version number.



The mini program information is put into the `nebula_preset.json`, and the reference is as follows:

```
{
 "config":{
  "updateReqRate":16400,
  "limitReqRate":13600,
  "appPoolLimit":3,
  "versionRefreshRate":86400
 },
 "data":[
  {
   "app_desc":"Preset mini program",
   "app_id":"2022080915350001",
   "auto_install":1,
   "extend_info":{
    "launchParams":{
     "enableTabBar":"YES",
     "enableKeepAlive":"NO",
     "enableDSL":"YES",
     "nboffline":"sync",
     "enableWK":"YES",
     "page":"page/tabBar/component/index",
     "tinyPubRes":"YES",
     "enableJSC":"YES"
    },
    "usePresetPopmenu":"YES"
   },
   "fallback_base_url":"https://xxx/2022080915350001/1.0.1.0_all/nebula/fallback/",
   "global_pack_url":"",
   "installType":1,
   "main_url":"/index.html#page/tabBar/component/index",
   "name":"Preset mini program",
   "online":1,
   "package_url":"https://xxx/2022080915350001/1.0.1.0_all/nebula/2022080915350001_1.0.1.0.amr",
   "patch":"",
   "sub_url":"",
   "version":"1.0.1.0",
   "vhost":"https://2022080915350001.h5app.com"
  }
 ],
 "resultCode":100,
 "resultMsg":"The operation is successful.",
 "state":"success"
}
```


- The following code provides an example on how to install the package in advance.

```
private void checkPresetInstalled() {
        Map<String, AppModel> appModelMap = RVProxy.get(RVResourcePresetProxy.class).getPresetAppInfos();
        Map<String, RVResourcePresetProxy.PresetPackage> packageMap = RVProxy.get(RVResourcePresetProxy.class).getPresetPackage();
        Set<String> stringSet = packageMap.keySet();
        for (String key: stringSet) {
            RVResourcePresetProxy.PresetPackage presetPackage = packageMap.get(key);
            AppModel presetModel = appModelMap.get(key);
            if (presetModel != null && presetPackage != null && presetPackage.getInputStream() != null) {
                AppModel appModel = MriverResource.getAppModel(presetModel.getAppId());
                boolean available = ((RVResourceManager)RVProxy.get(RVResourceManager.class)).isAvailable(appModel);
                if (!available) {
                    if (TextUtils.equals(appModel.getAppVersion(), presetModel.getAppVersion())) {
                        InternalUtils.installApp(presetModel, presetPackage.getInputStream());
                    } else {
                        // Determine whether to install the online version in advance.
                    }
                }

            }
        }
    }
```

Resource load interception

```
ResourcePackage resourcePackage = new GlobalResourcePackage("00000001") {
                @Override
                protected boolean needWaitSetupWhenGet() {
                    return false;
                }

                @Override
                public boolean needWaitForSetup() {
                    return false;
                }

                @Override
                protected boolean canHotUpdate(String hotVersion) {
                    return false;
                }

                @Override
                public Resource get(ResourceQuery query) {
                   // All resources can be blocked.
                    if (TextUtils.equals("Specify the resource path", query.pureUrl)) {
                        return getLocalResource(UIStyleActivity.this, query);
                    }
                    return null;
                }
            };
GlobalPackagePool.getInstance().add(resourcePackage);
```

Signature verification

```
// Enable the signature.
MriverResource.enableVerify(MriverResource.VERIFY_TYPE_YES,"public key");

// Disable the signature.
MriverResource.disableVerify( );
```

Enable keep-alive

```
Mriver.setConfig("enable_keep_alive", "YES");
Mriver.setConfig("mriver_keep_alive_time", "120000"); // The keep-alive time of 2 minutes.
Mriver.setConfig("mriver_keepalive_max", "3"); // Maximum number of keep-alive 3
```

Android mini programs are implemented based on the Activity Task Stack. If the mini program jumps to the native page (such as login) or other App pages (such as Third party payment and sharing), note that:

- There is no risk if it is in the same stack as the mini program page.
- If these pages are separate activity stacks, the return stack order may be affected and relevant logic needs to be regressed.

By default, if a page is specified during a keep-alive wake-up and the specified page is not the current page of the mini program, the specified page will be refreshed during wake-up.

You can set the refererBiz parameter to ignore the refresh and directly evoke the current page:

```
 Bundle intent  = new Bundle();
intent.putString("page", "page/component/view/view");
intent.putString("refererBiz", "home");
Mriver.startApp(appId, intent);
```

> **Note**
>
> - If the refererBiz value is fixed to home, the specified page is ignored during the keep-alive wake-up. If no mini program is alive, the specified page opens.
> - If the refererBiz value is set to another value, the referer value is compared with the referer value that was opened last time. If the refererBiz value is the same, the specified page is ignored. If no mini program is alive, the specified page opens.

Start a mini program of a specified version

```
MriverResource.de leteApp("2022080918000001"); // Delete the local mini program.

Bundle bundle = new Bundle();
bundle.putString(RVStartParams.LONG_NB_TARGET_VERSION, "the specified version number");
Mriver.startApp(TargetVersionActivity.this, "2022080918000001", bundle);
```

Custom JSAPI

```
// Customize the jsapi of tinyToNative.
MriverEngine.registerBridge(CustomApiBridgeExtension.class);


public class CustomApiBridgeExtension extends SimpleBridgeExtension {

    private static final String TAG = "CustomApiBridgeExtension";

    @ActionFilter
    public void tinyToNative(@BindingId String id,
                             @BindingNode(App.class) App app,
                             @BindingNode(Page.class) Page page,
                             @BindingApiContext ApiContext apiContext,
                             @BindingExecutor(ExecutorType.UI) Executor executor,
                             @BindingRequest JSONObject params,
                             @BindingParam("param1") String param1,
                             @BindingParam("param2") String param2,
                             @BindingCallback BridgeCallback callback) {
        RVLogger.d(TAG, "id: "+id+
                "\napp: "+app.toString()+
                "\npage: "+page.toString()+
                "\napiContext: "+apiContext.toString()+
                "\nexecutor: "+executor.toString());
        RVLogger.d(TAG, JSONUtils.toString(params));
        JSONObject result = BridgeResponse.SUCCESS.get();
        // result.put("message", "The client receives parameters:" + param1 + ", " + param2 + "\nThe current package name of the demo:" +
apiContext.getActivity().getPackageName());
        // Return the result to the mini program.
        Stack stack = MriverApp.getAppStack();
        Enumeration enumerationLists = stack.elements();

        JSONArray jsonArray = new JSONArray();
        while (enumerationLists.hasMoreElements()) {
            JSONObject jsonObject = new JSONObject();
            MRApp o = (MRApp) enumerationLists.nextElement();
            jsonObject.put("AppId", o.getAppId());
            jsonObject.put("AppVersion", o.getAppVersion());
            jsonArray.add(jsonObject);
        }
        String tinyappStr = jsonArray.toJSONString();
        // result.put("message", "The client receives parameters:" + param1 + ", " + param2 + "\nThe current package name of the demo:" +
apiContext.getActivity().getPackageName());
        result.put("message", tinyappStr);
        callback.sendJSONResponse(result);
    }

}
```

Enable the sharing feature

```
Mriver.setConfig("mr_showShareMenuItem", "YES");

// Implement ShareApiBridgeExtension
public class ShareApiBridgeExtension extends SimpleBridgeExtension {
    private static final String TAG = "CustomApiBridgeExtension";

    @ActionFilter
    public void shareTinyAppMsg(@BindingId String id,
                                @BindingNode(App.class) App app,
                                @BindingNode(Page.class) Page page,
                                @BindingApiContext ApiContext apiContext,
                                @BindingExecutor(ExecutorType.UI) Executor executor,
                                @BindingRequest JSONObject params,
                                final @BindingCallback BridgeCallback callback) {
        Log.i("ShareApiBridge", "share: " + (params == null ? "null" : params.toJSONString()));


        String title = params.getString("title");


        String desc = params.getString("desc");


        String myprop = params.getString("myprop");


        String path = params.getString("page");


        String appId = app.getAppId();



        // You can call the sharing component to implement subsequent features.
```

```
        String message = "Application ID: " + appId + "\n"
            + "title: " + title + "\n"
            + "desc: " + desc + "\n"
            + "myprop: " + myprop + "\n"
            + "path: " + path + "\n";

        AUNoticeDialog dialog = new AUNoticeDialog(apiContext.getActivity(),
            "Sharing result", message, "Sharing successful", "Sharing failed");
        dialog.setPositiveListener(new AUNoticeDialog.OnClickPositiveListener() {
            @Override
            public void onClick() {
                JSONObject result = BridgeResponse.SUCCESS.get();
                result.put("success", true);
                callback.sendJSONResponse(result);
            }
        });
        dialog.setNegativeListener(new AUNoticeDialog.OnClickNegativeListener() {
            @Override
            public void onClick() {
                callback.sendBridgeResponse(BridgeResponse.newError(11, "Sharing failed"));
            }
        });
        dialog.show();
    }
}
```

Permission pop-up window

```
// Customize the pop-up window for permission control alerts.
Mriver.setProxy(LocalPermissionDialogProxy.class, new LocalPermissionDialogProxy() {
                    @Override
                    public LocalPermissionDialog create(Context context) {
                        return new DemoLocalPermissionDialog(context);
                    }

                    @Override
                    public boolean interceptPermission(String appId, String page, String action, String scope, List<String> permissions) {
                        showToast("jsapi: " + action + "mini program: " + appId + "page: " + page);
                        return false;
                    }

                });
// Example of a pop-up window
public class DemoLocalPermissionDialog implements LocalPermissionMultiDialog {
    private Dialog mDialog;
    private final Context mContext;
    private PermissionPermitListener mPermissionPermitListener;

    public DemoLocalPermissionDialog(Context context) {
        this.mContext = context;
    }

 public void setExtData(String[] permissions, AppModel appModel, Page page, String action, String scope) {
    // Supports customization based on these parameters, including multi-level pop-up windows and custom copy styles.
     // permissions: the list of all permissions required by the current action.
     // appModel: the appModel of the current action, which can obtain the relevant copy defined by the mini program.
     // action: the action of the corresponding JSAPI.
     // scope: the scope of the JSAPI.
 }

    public void setDialogContent(String content, String title, String icon) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this.mContext);
        builder.setTitle("Permission pop-up window");
        builder.setMessage(content);
        builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface pDialogInterface, int pI) {
                if (DemoLocalPermissionDialog.this.mPermissionPermitListener != null) {
                    DemoLocalPermissionDialog.this.mPermissionPermitListener.onSuccess();
                }
            }
        });
        builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface pDialogInterface, int pI) {
                if (DemoLocalPermissionDialog.this.mPermissionPermitListener != null) {
                    DemoLocalPermissionDialog.this.mPermissionPermitListener.onFailed(-1, "", true);
                }
            }
        });
        this.mDialog = builder.create();
        this.mDialog.show();
    }

    public void setPermissionPermitListener(PermissionPermitListener permissionPermitListener) {
        this.mPermissionPermitListener = permissionPermitListener;
    }

    public void show() {
        if (this.mDialog != null && this.mContext instanceof Activity && !((Activity)this.mContext).isFinishing()) {
            this.mDialog.show();
        }

    }
```

> **Note**
> Added more capabilities to the setExtData support permission pop-up window.

Custom view

1. Upgrade appx to the `2.7.18` version.

```
api ('com.mpaas.mriver:mriverappxplus-build:2.7.18.20220825001@aar') {
    force=true
}
```

2. Call related APIs.

```
RVProxy.set(RVEmbedProxy.class, new RVEmbedProxy() {

                @Override
                public Class<?> getEmbedViewClass(String type) {
                    if ("custom_barrage".equalsIgnoreCase(type)) {
                        // The type must correspond to the type of the mini program. The corresponding custom view is returned based on the t
ype.
                        return EmbedCustomView.class;
                    }
                    return null;
                }
            });
```

3. Implement a custom view.

```
package com.mpaas.demo.tinyapp.engine;

import android.content.Context;
import android.text.TextUtils;
import android.util.Log;
import android.view.View;
import android.widget.FrameLayout;

import com.alibaba.ariver.app.api.Page;
import com.alibaba.ariver.engine.api.bridge.extension.BridgeCallback;
import com.alibaba.ariver.engine.api.bridge.extension.BridgeResponse;
import com.alibaba.ariver.engine.api.embedview.IEmbedView;
import com.alibaba.fastjson.JSONArray;
import com.alibaba.fastjson.JSONObject;
import com.alipay.mobile.beehive.video.h5.live.MRLivePlayerHelper;
import com.mpaas.mriver.integration.embed.IMREmbedView;

import java.util.Map;

public class EmbedCustomView implements IMREmbedView {
    private Context mContext;
    private Page mPage;
    private CustomBarrageView mCustomBarrageView; // The example of the pop-up view.

    @Override
    public void onCreate(Context context, Page page, IEmbedView iEmbedView) {
        mContext = context;
        mPage = page;
    }

    @Override
    public View getView(int width, int height, final String viewId, String type, Map<String, String> params) {
        Log.i("EmneCustomV", "getView: " + mCustomBarrageView + " " + viewId + " " + type + " " + params);
        if (mCustomBarrageView == null) {
            mCustomBarrageView = new CustomBarrageView(mContext);
        }
        FrameLayout.LayoutParams layoutParams = new FrameLayout.LayoutParams(width, height);
        mCustomBarrageView.setLayoutParams(layoutParams);
        return mCustomBarrageView;
    }

    // Receive the data sent by the mini program.
    @Override
    public void onReceivedMessage(String actionType, JSONObject data, BridgeCallback bridgeCallback) {
        Log.i("EmneCustomV", "onReceivedMessage: " +  actionType);
        if ("mpaasCustomEvent".equalsIgnoreCase(actionType)) {
            String innerAction = data.getString("actionType");
            if (TextUtils.equals(innerAction, "bindLivePlayer")) {
                JSONObject dataJSON = data.getJSONObject("data");

                // Set the barrage data.
                JSONArray barrages = dataJSON.getJSONArray("barrages");
                mCustomBarrageView.setData(barrages);

                // Bind the liveplayer.
                String bindId = dataJSON.getString("id");
                if (!TextUtils.isEmpty(bindId)) {
                    MRLivePlayerHelper.bind(bindId, mCustomBarrageView);
                }
            }

        }
    }

    protected void notifySuccess(final BridgeCallback bridgeContext) {
        if (bridgeContext != null) {
            bridgeContext.sendBridgeResponse(BridgeResponse.SUCCESS);
        }
    }


    @Override
```

```
    public void onReceivedRender(JSONObject params, BridgeCallback bridgeCallback) {
        Log.i("EmneCustomV", "onReceivedRender: " +  params);
        notifySuccess(bridgeCallback);
    }

    @Override
    public void onWebViewResume() {

    }

    @Override
    public void onWebViewPause() {

    }

    @Override
    public void onAttachedToWebView() {

    }

    @Override
    public void onDetachedToWebView() {

    }

    @Override
    public void onDestroy() {

    }

    @Override
    public void onRequestPermissionResult(int i, String[] strings, int[] ints) {

    }

    @Override
    public void onEmbedViewVisibilityChanged(int i) {

    }

    @Override
    public void initElementId(String s) {

    }

}
```

Mini program-side implementation

```
//page.axml
<mpaas-component
          id="mpaas-barrage"
          type="custom_barrage" // The value of the type parameter. The value must be the same as the value of native.
          style="{{ width: 400, height: 200 }}" // You can only configure the width and height.
          onMpaasCustomEvent="onMpaasCustomEvent" // Receive native events
/>

//page.js

barrageContext = my.createMpaasComponentContext('mpaas-barrage');
// Send data to native.
barrageContext.mpaasCustomEvent({
        actionType: 'bindLivePlayer',
        data: {
          "id": "liveplayer",
          "barrages": ["Interesting", "Sofa", "Barrage 1", "Barrage 2", "Barrage 3"]
        }
      });
  }, 100)
```

Page lifecycle listener

1. The following code is called during initialization.

```
List<String> miniAppPoint = new ArrayList<>();
miniAppPoint.add(PageResumePoint.class.getName());
miniAppPoint.add(PagePausePoint.class.getName());
miniAppPoint.add(PageEnterPoint.class.getName());
miniAppPoint.add(AppExitPoint.class.getName());
Mriver.registerPoint(PageLifeCycleExtension.class.getName(), miniAppPoint);
```

2. Achieve `PageLifeCycleExtension.java`.

```
public class PageLifeCycleExtension implements PageResumePoint, PageEnterPoint, PagePausePoint, AppExitPoint {

    private static final String TAG = "PageLifeCycleExtension";

    @Override
    public void onPageResume(Page page) {
    }

    @Override
    public void onInitialized() {

    }

    @Override
    public void onFinalized() {

    }

    @Override
    public void onPageEnter(Page page) {

    }

    @Override
    public void onPagePause(final Page page) {
    }

    @Override
    public void onAppExit(App app) {

    }
}
```

APM listeners

> **Important**
> If the blank area of the page is large, the `MiniPage_Load_T2` is not called back.

1. Set the following content during initialization.

```
RVProxy.set(PrepareNotifyProxy.class, new PrepareNotifyProxy() {

                        @Override
                        public void notify(String s, PrepareStatus prepareStatus) {

                        }

                        @Override
                        public void apmEvent(final String s, final String s1, final String s2, final String s3, final String s4) {
                          // Non-UI thread
                          Log.i("MiniStartTime", "apmE: " + s + " " + s4);
                          if ("MiniAppStart".equalsIgnoreCase(s) || "MiniPage_Load_T2".equalsIgnoreCase(s)) {
                              boolean isT2 = "MiniPage_Load_T2".equalsIgnoreCase(s);
                              String apmData = s4;
                              if (!TextUtils.isEmpty(apmData)) {
                                  parseTime(isT2, apmData);
                              }
                          }
                      });
                }



private void parseTime(boolean isT2, String s4) {
        String[] kvArrs = s4.split("\\^");
        long miniStart = 0; // The time when the mini program is clicked.
        long miniPrepared = 0; // The time when the mini program preparation phase is completed. The first time the mini program is downloade
d.
        long miniAppStarted = 0; // The time when the mini program core phase is completed.
        long miniT2 = 0; // The time when the T2 mini program is rendered.
        boolean needIgnore=false; // true indicates the second-level page rendering callback when the internal second-level page jumps. The
first screen needs to be ignored.

        for (String kvItem : kvArrs) {
            String[] kv = kvItem.split("=");
            if (kv.length == 2) {
                String key = kv[0];
                String value = kv[1];
                if ("mini_st_ts0".equalsIgnoreCase(key)) {
                    // The time when the mini program is clicked.
                    miniStart = Long.parseLong(value);
                } else if ("mini_st_ts7".equalsIgnoreCase(key)) {
                    // The time when the mini program preparation phase is completed.
                    miniPrepared = Long.parseLong(value);
                } else if ("mini_st_end_ts".equalsIgnoreCase(key)) {
                    // The time when the core phase of the mini program was completed.
                    miniAppStarted = Long.parseLong(value);
                } else if ("mini_t2_ts".equalsIgnoreCase(key)) {
                    // The time when the T2 mini program is rendered.
                    miniT2 = Long.parseLong(value);
                } else if (isT2 && "isFirstPage".equalsIgnoreCase(key)) {
                    if ("false".equalsIgnoreCase(value)) {
                        needIgnore = true;
                    }
                }
            }
        }

        if (!needIgnore && miniStart > 0 && miniPrepared > 0 && miniAppStarted > 0 && miniT2 > 0) {
            final String toastStr = "Preparation time=" + (miniPrepared-miniStart) + "Core time=" + (miniAppStarted-miniPrepared) + "Business
time=" + (miniT2-miniAppStarted) + "Total time=" + (miniT2-miniStart);
            Log.i("MiniStartTime", toastStr);
            mUIHandler.post(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(MRiverApp.sApp, toastStr, Toast.LENGTH_LONG).show();
                }
            });
        }
    }
```

2. Add the timestamp parameter when you start the mini program.

```
Bundle intent  = new Bundle();
intent.putString("miniapp_start_ts", Long.toString(System.currentTimeMillis()));
Mriver.startApp(FastStartActivity.this, “appId”, intent);
```

Support for Google Maps

1. Turn on the switch to switch to Google Maps.

```
Mriver.setConfig("ta_map_type", "1");
```

2. Add the Google Maps dependency and configure the key for Google Maps.

1.2.8.4. Startup parameters

key	value	Description
query	Example: `a=xx&c=xx`	Mini program parameter transfer
page	Example: `pages/twoPage/twoPage`	Specify the page to open and pass the
RVStartParams.LONG_NB_UPDATE	`synctry` , `async` , default `async`	synctry: forcibly update the version async: Do not force version update
RVStartParams.LONG_NB_TARGET_VERSION	Example: `1.0.0.0`	Specify the version number

1.2.9. Switch the map

1.2.9.1. Use Google Maps in Mini Program

1. After initialization is complete, switch the configuration switch to Google Maps:

```
Mriver.setConfig("ta_map_type", "1");
```

2. Add Google Maps dependencies and API keys as needed.

1.2.9.2. Use Baidu Map in Mini Program

Introduce Baidu Map SDK

Refer to Baidu Map official documentation to integrate the Map LBS SDK. Ensure the SDK version is 7.5.9 or higher. After adding the SDK, you must delete the `libc++_shared.so` file to prevent conflicts with mPaas.

Configure Baidu Map API Key in AndroidManifest.xml

Insert the following code into <application> to set up the developer key (AK), or dynamically configure the AK by `SDKInitializer.setApiKey("ak")` .

```
<application>
    <meta-data
        android:name="com.baidu.lbsapi.API_KEY"
        android:value="developer key" />
</application>
```

Initialize Baidu Map and set coordinate system

```
// After agreeing to the App privacy, set the map privacy protocol
SDKInitializer.setAgreePrivacy(this, true);
// After agreeing to the App privacy, set the Baidu location privacy protocol
LocationClient.setAgreePrivacy(true);
// Initialize context information before using SDK components, passing in ApplicationContext
SDKInitializer.initialize(this);
// Since version 4.3.0, all interfaces of Baidu Map SDK support both Baidu coordinates and National Survey Bureau coordinates. Use this method
to set the coordinate type you use.
// Including BD09LL and GCJ02 coordinates, the default is BD09LL coordinates.
SDKInitializer.setCoordType(CoordType.GCJ02);
```

Configure the mini program to use Baidu Map

```
Mriver.setConfig("ta_map_type","2");
```

1.3. Integrate Ariver Tiny App into iOS

1.3.1. Baseline cp_change_15200851 Series

This release note provides the updates of the 10.2.3 baseline iOS SDK after the release of the mobile development platform (mPaaS) in **reverse time order**.

The mPaaS client verifies the identity of Alibaba Cloud public cloud customers. Users who fail the verification will no longer be able to use the capabilities of mPaaS. To ensure that you can continue to use mPaaS smoothly, please retrieve the `.config` file in the mPaaS console and import it to the project. For more information, see mPaaS user authentication.

- **Added** feature
- **Updated** feature
- **Fixed** feature
- **Removed** feature
- **Known issues**

cp_change_15200851.53 (June 30, 2025)

Mini Program

- **Added** Internationalization adaptation (English).
- **Fixed** Fixed the issue that cover-view was covered by custom native view.
- **Fixed** Fixed the issue that the image selected by chooseImage cannot be previewed.

mPaaS Framework

- **Fixed** Fixed the issue where the framework would be initialized when switching to the background and calling up the App through Scheme before agreeing to the privacy agreement.

SmartScene

- **Fixed** Fixed the issue of repeated uploading of tracking logs caused by loading multiple identical ID cards.
- **Removed** Removed the old version of RPC component.

cp_change_15200851.52 (June 9, 2025)

Mini Program

- **Added** Support setting whether to automatically download when fully updating.
- **Fixed** Fixed the issue where the numeric keyboard of the mini program on iOS 18.4 could not pop up.
- **Fixed** Fixed the bug that showTitleBar and hideTitlebar did not take effect.

Ant Cube Card

- **Updated** Update mp-swiper display.

cp_change_15200851.49 (April 28, 2025)

Mini Program

- **Added** Support rollback feature.
- **Added** Support the delisting feature.
- **Added** Support scanning QR code to open the open platform test version.
- **Added** Support custom tabbar function.
- **Added** Compatible with templateconfig.
- **Added** Support intercepting and modifying AppInfo.
- **Added** getSystemInfo supports the language attribute.
- **Fixed** Fixed the bug that the page was not released when the offline package was exited.

Ant Cube Card

- **Added** Added mpaas_jsapi external implementation.
- **Updated** Decoupling of Ant Cube Card and Mini Program.

SmartScene

- **Added** Event tracking is added.
- **Removed** mpaas_jsapi internal implementation is removed.

Other

- **Updated** mPaaS.framework symbol conflict is modified.

cp_change_15200851.48 (April 10, 2025)

mPaaS Framework

- **Added** Adaption to Xcode 16. For details, see mPaaS 10.2.3 Adaptation to Xcode 16.

Mobile Gateway Service

- **Updated** Adaption to Xcode 16.

Data Synch Service

- **Updated** Adaption to Xcode 16.

Message Push Service

- **Updated** Adaption to Xcode 16.

H5 Container & Mini Program

- **Updated** Adaption to Xcode 16.

Common UI

- **Updated** Adaption to Xcode 16.

Social Sharing

- **Updated** Adaption to Xcode 16.

Upgrade detection

- **Updated** Adaption to Xcode 16.

Mobile Analysis Service

- **Updated** Updated automated click tracking and added support for a custom API turn-off.

cp_change_15200851.46 (March 4, 2025)

SmartScene

- **Fixed** Fixed the occasional crash in SmartScene.

cp_change_15200851.45 (February 26, 2025)

Mini Program

- **Updated** Optimized the plugin authorization check logic to solve the issue of occasional prompting "The plugin has no permissions".

cp_change_15200851.44 (February 24, 2025)

Scan

- **Added** Supports multiple code recognition in album.

H5 Container

- **Updated** Optimized innerXML XSS attack protection.

cp_change_15200851.43 (February 11, 2025)

Ant Cube Card

- **Fixed** Fixed unknown crash caused by layout "for in".
- **Fixed** Fixed fix302 Loading error and occasional crash on iOS12.

cp_change_15200851.42 (January 14, 2025)

Ant Cube Card

- **Added** Added a layout cache cleaning interface.
- **Fixed** Fixed the underlying rendering issue of antcube.

cp_change_15200851.36 (November 26, 2024)

Ant Cube Card

- **Fixed** Fixed the issue that data was not returned due to not transmitting the card version.
- **Fixed** Fixed the issue that cell0 does not update due to height change and overall refresh.
- **Fixed** Fixed the conflict with Telecom SDK.

cp_change_15200851.33 (November 15, 2024)

Ant Cube Card

- **Updated** Optimized layout experience.

cp_change_15200851.33 (November 12, 2024)

Ant Cube Card

- **Added** Mars supports loading multiple resources.

cp_change_15200851.29 (October 24, 2024)

Ant Cube Card

- **Updated** Optimized swiper component.

cp_change_15200851.26 (October 15, 2024)

Ant Cube Card

- **Updated** Optimized the layout.
- **Updated** Optimized mp-video and mp-swiper components.

cp_change_15200851.25 (September 18, 2024)

Ant Cube Card

- **Updated** Optimized layout experience.

cp_change_15200851.23 (August 23, 2024)

Ant Cube Card

- **Updated** Optimized custom component mp-swiper of Cube.
- **Updated** Optimized layout experience.

cp_change_15200851.21 (August 16, 2024)

Mini Program

- **Updated** Support for Google map.

cp_change_15200851.19 (August 13, 2024)

Ant Cube Card

- **Updated** Optimized layout event tracking logic.
- **Updated** Optimized custom component mp-tab of Cube.

cp_change_15200851.16 (June 12, 2024)

Mini Program

- **Updated** Optimized international site language.

cp_change_15200851.15 (May 31, 2024)

Ant Cube Card

- **Updated** The swiper transformation speed supports dynamic updates.

cp_change_15200851.14 (May 22, 2024)

Ant Cube Card

- **Fixed** Fixed the issue where feedcard could not find the corresponding main card after refreshing.

cp_change_15200851.12 (May 16, 2024)

Ant Cube Card

- **Added** Add basic cards.

cp_change_15200851.10 (May 8, 2024)

mPaaS

- **Updated** Adapt to Xcode 15.

cp_change_15200851.7 (April 8, 2024)

Ant Cube Card

- **Updated** Add non-first rendering callback.

cp_change_15200851.6 (April 7, 2024)

Mini programs

- **Updated** Upgrade video player.
- **Updated** Adapt to the new version of Amap.
- **Updated** Adapt to iOS17.
- **Fixed** Fixed the occasional crash during video playback.

cp_change_15200851.5 (January 11, 2024)

Mini Program

- **Fixed** Optimize the copywriting when loading exceptions.

Social Sharing

- **Update** Upgraded WeChat and Weibo sharing SDK.

cp_change_15200851.4 (December 21, 2023)

Ant Cube Card

- **Fix** Fixed the UI display issue.

cp_change_15200851.3 (December 14, 2023)

Ant Cube Card

- **Added** Support for card components.

cp_change_15200851.2 (November 16, 2023)

Mini Program

- **Updated** Upgrade the mini program container.

1.3.2. Upgrade guide

If you have connected offline packages or mini program components of the old Nebula container, you must make the following modifications after upgrading to the new container baseline of Ariver.

Update the SDK

You can update the SDK with two methods:

- use the mPaaS Xcode Extension plug-in.
- use the Cocoapods plug-in.

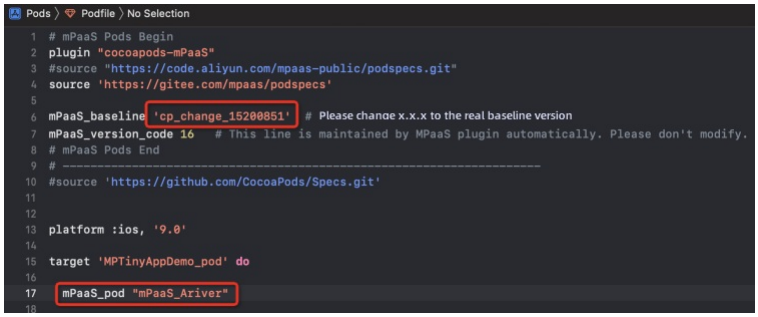
Use mPaaS Xcode Extension plug-in

1. Start Xcode and open an existed project which is developed based on the native iOS framework.
2. Run mPaaSPlugin in the application on your computer. Select **Edit Project** to open the existed project.
3. Click **Upgrade Baseline**, select Custom Baseline, and then enter a custom baseline `cp_change_15200851`.
4. Click the **Edit Components** tab, delete the **Offline package** or **Mini program** component, and then select the **Ariver mini program** component.

Use the Cocoapods plug-in

1. Change the baseline number to `cp_change_15200851`, and introduce the SDK for offline packages or mini programs by `mPaaS_pod "mPaaS_Ariver"`.

> **Note**
> Delete the mPaaS_TinyApp or mPaaS_Nebula component. Otherwise, an SDK conflict error will occur.



2. In the project directory, run the following command to update the new container baseline of Ariver:

```
pod mpaas update cp_change_15200851
```

3. Run the following command to install the SDK:

```
pod update
```

Update configurations

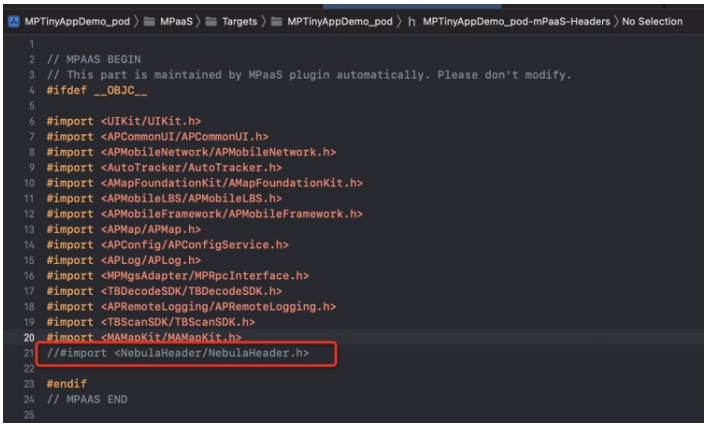
Compared with the old Nebula container, the new Ariver container has some configuration updates. Please check them one by one as described below.

Update referenced header files

If an error occurs when you reference the header files of Nebula-related libraries in your project after you update the baseline, add the Ariver prefix to the SDK name. For example, change `#import <NebulaPoseidon/PsdPluginConfig.h>` to `#import <AriverNebulaPoseidon/PsdPluginConfig.h>`, and save the modified header file to the pch file.

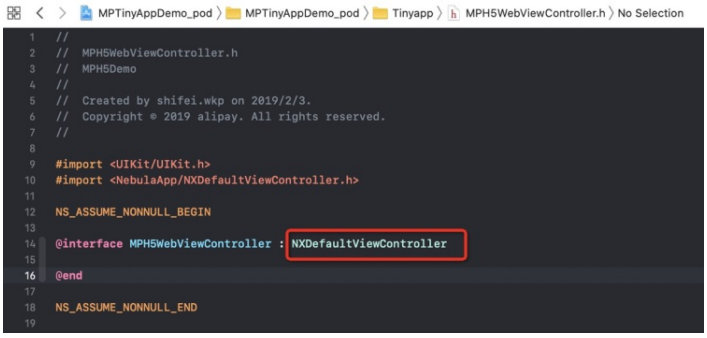


If the `#import <NebulaHeader/NebulaHeader.h>` is referenced in the `mPaaS-Headers.pch` header file, please delete it.



Update custom container base class

If the base class of the container is customized in the project, you need to change the inherited parent class from the `H5WebViewController` to the `NXDefaultViewController` class after upgrading to the new Ariver container.



1.3.3. Quick start

Prerequisites

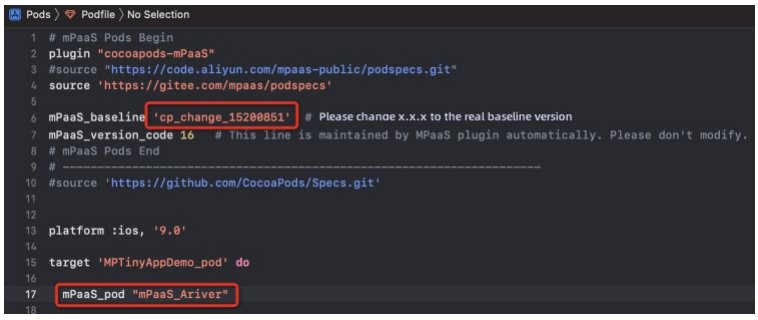
You have connected the project to mPaaS. Related topics:

- Integrate by using CocoaPods based on the existing project

Add an SDK

Select an add method based on your integration method.

- **Use mPaaS Xcode Extension** This method is applicable to **mPaaS-based integration** or **existing projects that use mPaaS plug-ins**.
  i. Click the Xcode menu item **Editor** > **mPaaS** > **Edit Project** to open the Edit Project page.
  ii. Select **Ariver mini program** and click **Edit**.
- **Use cocoapods-mPaaS plug-ins** This method is applicable to the integration mode that uses **CocoaPods based on existing projects**.
  i. In the Podfile file, specify the baseline number as `cp_change_15200851` and use the `mPaaS_pod "mPaaS_Ariver"` to add the dependency of the Ariver mini program component.



ii. Run the `pod mpaas update cp_change_15200851` command to update the baseline.

iii. Run `pod install` on the command line to complete the connection.

The following section describes how to use a mini program based on the official demo of the mini program. The whole process is divided into the following three steps:

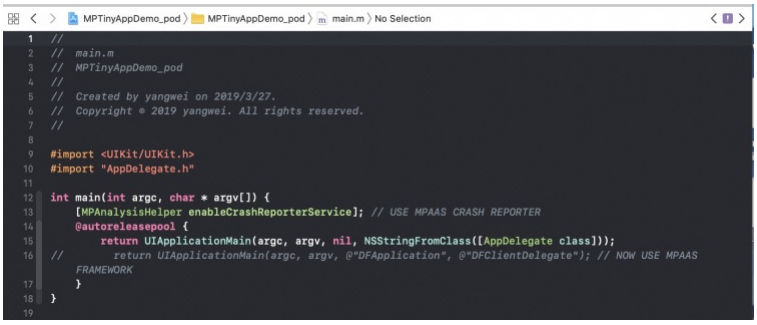
1. Initialize the SDK
2. Release Miniapp
3. Start mini program

Initialize the SDK

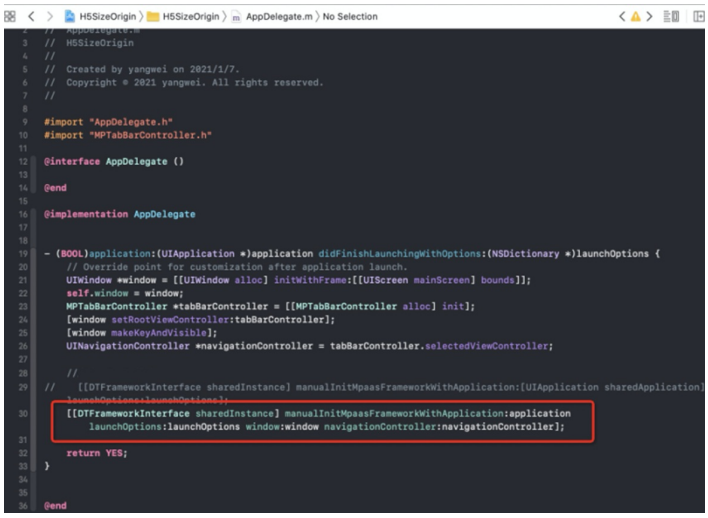
Initialize the mPaaS framework

If the lifecycle of the app is not managed by the mPaaS framework, but is specified as a delegate defined by yourself, as shown in the following figure, you must manually initialize the mPaaS framework.

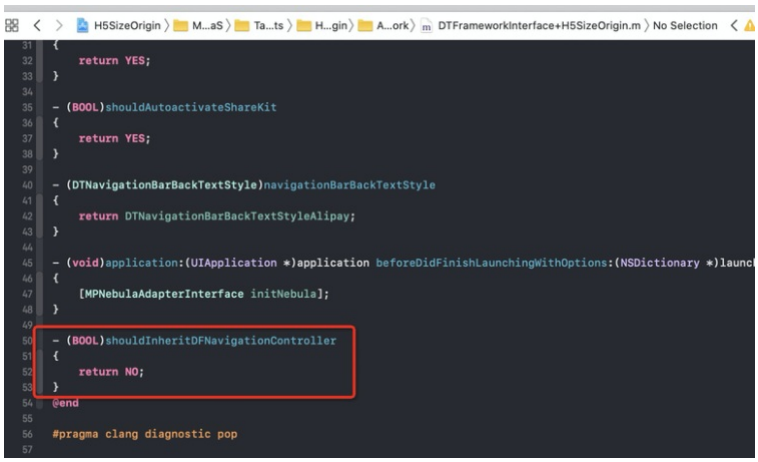
> **Note**
> mPaaS framework hosting means that the app's delegate is set to DFClientDelegate. In this case, you do not need to manually initialize the mPaaS framework.



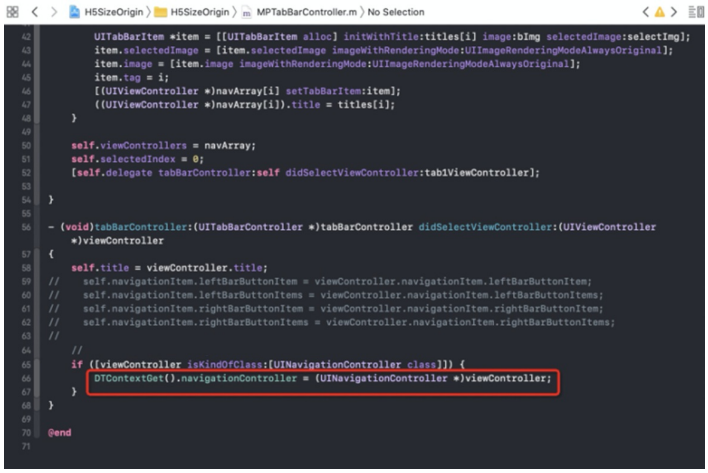
1. After the `window` and `navigationController` of the application are created, call the following method to initialize the mPaaS framework.



2. Overriding the `shouldInheritDFNavigationController` method in a `DTFrameworkInterface` `category` and returning a `NO`, the support navigation bar controller may not inherit the `DFNavigationController`.



3. If an app has multiple navigation bars and you need to open different mini programs in different navigation bars, you must reset the navigation bar of the container after you switch the navigation bar.




```
UITabBarItem *item = [[UITabBarItem alloc] initWithTitle:titles[i] image:bImg selectedImage:selectImg];
item.selectedImage = [item.selectedImage imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
item.image = [item.image imageWithRenderingMode:UIImageRenderingModeAlwaysOriginal];
item.tag = i;
[(UIViewController *)navArray[i] setTabBarItem:item];
((UIViewController *)navArray[i]).title = titles[i];
}

self.viewControllers = navArray;
self.selectedIndex = 0;
[self.delegate tabBarController:self didSelectViewController:tab1ViewController];

}

- (void)tabBarController:(UITabBarController *)tabBarController didSelectViewController:(UIViewController
*)viewController
{
self.title = viewController.title;
//    self.navigationItem.leftBarButtonItem = viewController.navigationItem.leftBarButtonItem;
//    self.navigationItem.leftBarButtonItems = viewController.navigationItem.leftBarButtonItems;
//    self.navigationItem.rightBarButtonItem = viewController.navigationItem.rightBarButtonItem;
//    self.navigationItem.rightBarButtonItems = viewController.navigationItem.rightBarButtonItems;
//
//
if ([viewController isKindOfClass:[UINavigationController class]]) {
DTContextGet().navigationController = (UINavigationController *)viewController;
}
}

@end
```

Init Containers

To correctly start the mini program, you need to call the SDK interface to initialize the container after the app is started. Must be initialized in a `DTFrameworkInterface` `- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions`.

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

Precautions

In the `cp_change_15200851` baseline, if the hosting mode and privacy pop-up box of the mPaaS framework is used and the switch configuration agent `[MPNebulaAdapterInterface shareInstance].configDelegate = self;` is set, you need to set the switch proxy in the following two methods at the same time; if the switch configuration agent is not set, please ignore it.

```
- (DTFrameworkCallbackResult)application:(UIApplication *)application
    privacyAuthDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
    completionHandler:(void (^)(void))completionHandler
{
    //Set switch proxy
    [MPNebulaAdapterInterface shareInstance].configDelegate = self;

```

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

    //Set switch proxy
    [MPNebulaAdapterInterface shareInstance].configDelegate = self;

    //Other ...

}
```

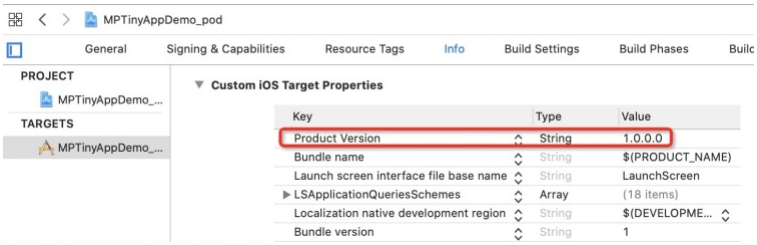
Release mini program

Before starting the mini program, you need to release the mini program in the mPaaS console. The steps are as follows:

1. Enter the background of the mini program. Log on to the mPaaS console. In the left-side navigation pane, choose **Mini Program** > **Release**.
2. Configure the virtual domain name. If this is the first time you configure a virtual domain name, you must first configure a virtual domain name in the **Mini Program** > **Mini Program Release** > **Configuration Management**. The virtual domain name must be hung under the enterprise domain name to prevent it from being hijacked by third parties, such as `example.com`.
3. Create a mini program. Go to the mPaaS console and perform the following operations:
   i. In the left-side navigation pane, choose **Mini Program** > **Mini Program Release**.
   ii. On the page that appears, click **Create**.
   iii. In the **Create Mini Program** dialog box, enter the ID and name of the mini program and click **OK**. The mini program ID is any 16-bit number, for example, `2018080616290001`.
   iv. In the mini app list, find the new mini app and click **Add**.

v. In the Basic Information section, configure the following information:
   - Version: Enter the version number of the small package, for example, `1.0.0.0`.
   - Client Range: Select the minimum and maximum versions of the iOS client corresponding to the mini program app. The client App within this range can start the corresponding small program, otherwise it cannot be started. Here, the minimum version can be filled in `0.0.0`, and the maximum version can be left unfilled, 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. For more information, see the `Product Version` field in the project `Info.plist`.



   - Icon: Click the **Select File** icon to upload the small package. You must upload the icon when you create the mini program for the first time. An example icon is as follows:



   - Upload the small package resource file in the `.zip` format. We have prepared a small mPaaS sample program (click here to download), which can be uploaded directly.

     > **Note**
     > Before you upload the mini program, you must change the `.zip` file name and folder name in the package to the 16-digit ID of the mini program.

vi. In the configuration information bar, complete the following configurations:
   - Main Portal URL: required. The homepage of the mini program. The format of the main entry URL is: `/index.html#xxx/xxx/xxx/xxx`, where the `xxx/xxx/xxx/xxx` after `#` is the first value in the `pages` in the `app.json` of the mini program. The main entry point of the mPaaS mini program is: `/index.html#page/tabBar/component/index`.
   - Keep the default settings for other configurations.

vii. Select the **Confirm that the preceding information is accurate and do not modify it after you submit it** check box.

viii. Click **Submit**.

4. Release the mini program. Go to the mPaaS console and complete the following steps:

i. In the left-side navigation pane, choose **Mini Program > Mini Program Release > Official Mini Program Package Management**.

ii. On the Small Packages page, select the small package and version that you want to release, and click **Create Release**.

iii. In the Create Release panel, configure the following settings:
   - Release Type: Select **Official**.
   - Release Description: Optional.

iv. Click **OK** to create the release.

Start mini program

After you complete the preceding steps, run the following code to start the sample mini program in the iOS project:

```
[MPNebulaAdapterInterface startTinyAppWithId:@"2018080616290001" params:nil];
```

> **Note**
> The `2018080616290001` in the above code is the ID of the mini program. This is only an example in this article. Enter the ID of your mini program.

1.3.4. Start mini program

This topic describes how to start a mini program and provides examples on how to start a mini program.

Usage notes

This interface is used to jump to mini programs.

```
@interface MPNebulaAdapterInterface : NSObject

/**
 Open small package

 @ param appId The ID of the mini program.
 @ param params The parameters of the mini program.
 */
+ (void)startTinyAppWithId:(NSString*)appId params:(NSDictionary*)params;

@end
```

Sample code

```
[MPNebulaAdapterInterface startTinyAppWithId:@"1234567891234567" params:@{}];
```

Description

Parameter	Type	Describe	Required
appId	NSString	The AppID of the target mini program to be redirected.	Yes
params	NSDictionary	The parameters that are passed when the mini program is opened.	No

Description of the params field

Parameter	Type	Describe	Required
query	NSString	The custom parameters passed when the mini program is opened. You can obtain the parameters in the mini program.	No
page	NSString	The page specified when opening the mini program. If it is not passed, the main page of the mini program is opened by default.	No

Start mini program

Start mini program and pass custom parameters

- The parameter information for adding the page jump at startup on the client is as follows:

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

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

Description

- appId: the ID of the mini program, which is obtained from the mPaaS console.
- params:params mini program parameters. The custom parameter field is query. Separate multiple values with ampersand (`&`).

Usage notes

- The mini program framework will decode the values of key-value pairs for each pair of custom input parameters. If the key-value pair of your input parameter contains the special character `&` , call the following method to encode the input parameter. If the input parameter does not contain the special character, you do not need to use encode.

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

- The mini program framework does not process the keys of key-value pairs of custom input parameters. Therefore, do not set special characters for the key to prevent the mini program from being unable to recognize the custom parameters.

- Parameters can be obtained from the parameter `options` of the `onLaunch` or `onShow(options)` method in the mini program. You can store the passed parameters in the `app.js` global variable `globalData` and use them to directly take values from the `globalData` .

Start the mini program and jump to the specified page

If the mini program has multiple pages, you can set the page parameter to open the specified page of the mini program, shown as follows. If the page parameter is not set, the configured homepage path is turned on by default.

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

1.3.5. Real device preview and debugging

This topic describes how to preview and debug a mini program on an iOS client.

> **Note**
> The preview and debugging feature of mini programs is supported only in mPaaS 10.1.60 and later.

Follow these steps to access the preview and debugging features:

1. Obtain the content string based on the QR code of the IDE (for example, by scanning the QR code).
2. Call the mini program preview and debugging interface.
   - Pass in the QR code content string:

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

   - or custom parameter interface:

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

     If you need to pass parameters when opening the mini program, you can set the parameters by `param` the parameters. The `param` contains the `page` and `query` fields:

     - `page` : Specifies the path to a specific page to open.
     - `query` : specifies custom parameters. Concatenate multiple key-value pairs with ampersand (`&`).

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

Configure a 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
```

1.3.6. Customize UI

1.3.6.1. Customize navigation bar

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 specific 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

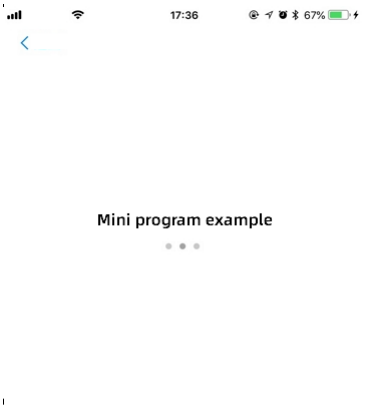
If you want to modify the right button style globally, you need to override the `rightBarButtonItem` of the current page in the `viewwillAppear` of the HTML5 base class.

```
- (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 ac
tion:@selector(onClickClose)];
    }
 }

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

1.3.6.2. Customize startup loading page

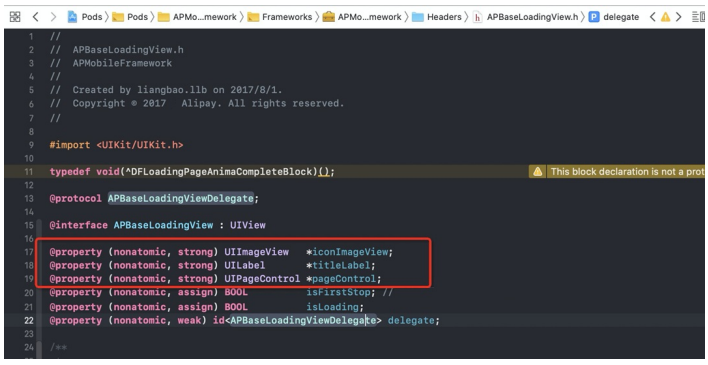
When the Mini Program is started, if the Mini Program has not been downloaded to the device, the Mini Program container will launch the loading page (as shown in the image below) to prompt the user to wait. After the Mini Program is installed on the device, the loading page will be closed and the page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

1.3.6.3. Customize error page

When loading the mini program, if it fails to load the page or the website cannot be opened, an error similar to the following will appear: "Network can't connect (-1009)"

This article introduces how to customize the error 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];

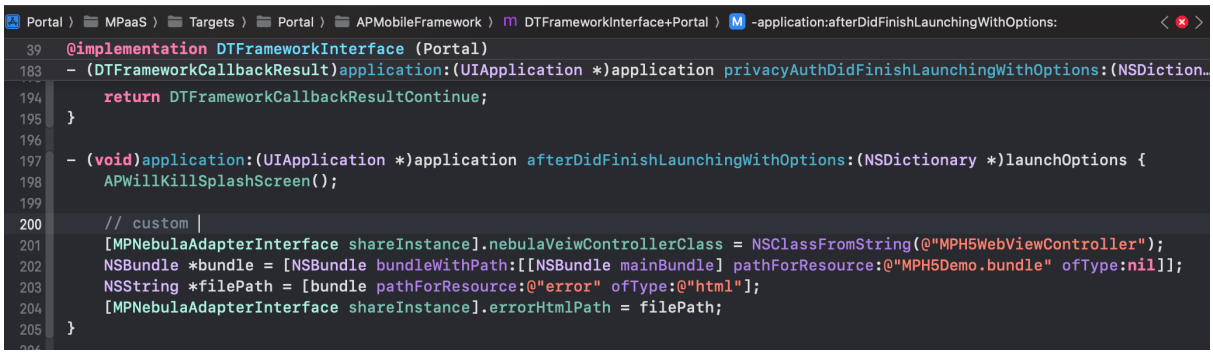
        if ([kEvent_Navigation_Error isEqualToString:event.eventType]) {
        [self handleContentViewDidFailLoad:(id)event];
    }
}
```

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

   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>
```

1.3.7. Engine management

1.3.7.1. Custom APIs

If the existing mini program APIs or events cannot meet your development requirements, you can extend them.

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 calls the API.

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

Native applications 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.

1.3.7.2. Custom View

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"} callback:^(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"
  ...
/>
```

> **Important**
> - 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
```

1.3.8. Resource management

1.3.8.1. Mini program package information

This topic describes how to obtain information about a specific mini package, all installed mini packages, and all mini program appIDs, and delete mini packages on iOS.

Obtain all mini program packages appId

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject

/**
 * Query the appId list of all applications
 *
 * @param array array [appId,appId...]
 */
- (NSArray *)allAppIds;
```

- Sample code

```
NSArray *res = [[MPNebulaAdapterInterface shareInstance] allAppIds];
```

Obtain information about a specified mini program package

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject

/**
 * Get package information for a specified application
 *
 * @param appId Specify application ID array
 * @return NSDictionary All APP instances {appId:[NAMApp, ...], ...}
 */
- (NSDictionary *)allAppsForAppId:(NSArray *)arrAppId;
```

- Sample code

```
NSDictionary *res = [[MPNebulaAdapterInterface shareInstance] allAppsForAppId:@[@"2020012000000001"]];
```

Obtain the installed mini program package information

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject

/**
 * Get package information for installed apps
 *
 * @param list dictionary {appId:version}, return all installed package information when nil is passed
 */
- (NSDictionary *)installedApps:(NSDictionary *)list;
```

- Sample code

```
NSDictionary *res = [[MPNebulaAdapterInterface shareInstance] installedApps:@{@"2020012000000001":@""}];
```

Deletes information about a specific mini program package.

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject
/**
 *  @brief Delete the information of the local specified application (including
 *         package information, amr and installation directory)
 *
 *  @param appId appId of the application
 *
 *  @return
 */
-(void)clearAllAppInfo:(NSString *)appId;
```

- Sample code

```
[[MPNebulaAdapterInterface shareInstance] clearAllAppInfo:@"2020012000000001"];
    NSDictionary *app = [[MPNebulaAdapterInterface shareInstance] allAppsForAppId:@[@"2020012000000001"]];
    NSString *res = @"Failed to delete";
    if(!app){
        res = @"Deleted successfully";
    }else{
        res = @"Failed to delete";
    }

    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"message" message:[NSString stringWithFormat:@"%@", res] delegate:nil
cancelButtonTitle:nil otherButtonTitles:@"ok", nil];
    [alert show];
```

1.3.8.2. Update mini program package

This topic describes how to remotely manage a specific mini program, update all mini programs, and set the update frequency of mini program packages.

Update a specific mini program actively

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject
/**
 *
 * Note:
 *      Before 9.9.9: After the request is successful, the offline package will be automatically downloaded under Wifi. If it is not Wifi, on
ly the offline package with auto_install set to YES will be downloaded.
 *      9.9.9 and later: The download timing can be configured for each application, through server configuration, and the default is WIFI d
ownload.
 *
 * @param params Request list format: {appid:version}, multiple appids can be passed. If version is not specified, it will be passed empty.
The highest version will be taken by default.
 *              Supports fuzzy matching of version numbers, e.g. '*' matches the highest version number '1.*' matches version numbers starti
ng with 1, total highest version number, etc., up to 4 digits
 */
-(void)requestNebulaAppsWithParams:(NSDictionary)
```

- Sample code

```
[[MPNebulaAdapterInterface shareInstance] requestNebulaAppsWithParams:@{@"2020012000000001":@"*"} finish:^(NSDictionary *data, NSError
*error) {
    if (!error) {
        NSLog(@"[mpaas] nebula rpc data :%@", data[@"data"]);
        dispatch_async(dispatch_get_main_queue(), ^{
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"message" message:[NSString stringWithFormat:@"Updated:%@", data[@"data"
]] delegate:nil cancelButtonTitle:nil otherButtonTitles:@"ok", nil];
            [alert show];
        });
    }
}];
```

Update all mini programs actively

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject

/**
 *  Fully update local offline package
 *
 *  @param finish  Complete callback
 *
 */
- (void)requestAllNebulaApps:(NAMRequestFinish)finish;
```

- Sample code

```
[[MPNebulaAdapterInterface shareInstance] requestAllNebulaApps:^(NSDictionary *data, NSError *error) {
    if (!error) {
        NSLog(@"[mpaas] nebula rpc data :%@", data[@"data"]);
        dispatch_async(dispatch_get_main_queue(), ^{
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"message" message:[NSString stringWithFormat:@"Updated:%@", data[@"data"
]] delegate:nil cancelButtonTitle:nil otherButtonTitles:@"ok", nil];
            [alert show];
        });
    }
}];
```

Set the update frequency of mini program packages

By default, each time you open an application, the mini program SDK attempts to check for an updatable version. To reduce the pressure on the server, this check has a time limit. The default time interval is 30 minutes. To adjust the frequency of mini program package updates, you can call the following API operation:

- Interface description

```
@interface MPNebulaAdapterInterface : NSObject

/**
 Set the frequency of offline package or applet request updates. The default is 1800, unit: second, upper limit is 24 hours.
 */
@property(nonatomic, assign) NSTimeInterval nebulaUpdateReqRate;
```

- Sample code

```
// Set the update frequency to 10 min.
[MPNebulaAdapterInterface shareInstance].nebulaUpdateReqRate = 600;
```

1.3.8.3. Mini program package signature verification

This topic describes how to enable and disable mini program package verification. By default, signature verification is enabled. You can call the API operation to modify whether signature verification is required on the endpoint.

Interface Description

```
@interface MPNebulaAdapterInterface : NSObject

/**
Whether to verify the signature of offline packages, the default is YES, that is, the Nebula container verifies the signature of downloaded of
fline packages by default. If you need to turn off signature verification during debugging, set this property to NO.
 */
@property (nonatomic, assign) BOOL nebulaNeedVerify;

/**
The public key path for offline package signature verification, the default is nil. If the offline package signature verification switch is YE
S, the corresponding public key must be set here, otherwise the local signature verification fails and the offline package loading fails.
 */
@property (nonatomic, strong) NSString *nebulaPublicKeyPath;
```

Enable a mini program signature verification

- Turn on the switch of signature verification

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

- Configure a public key for signature verification

```
NSString *path = [[NSBundle mainBundle].bundlePath stringByAppendingFormat:@"%@", @"public_pem.html"];
[MPNebulaAdapterInterface shareInstance].nebulaPublicKeyPath = path;
```

> **Note**
> Call the MPNebulaAdapterInterface interface before you open the mini program package for the first time. Otherwise, the public key fails to be initialized. For more information about the public and private keys, see Configure Mini Program package.

Disable mini program signature verification

Sample code

```
[MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = NO;
```

1.3.8.4. Preset mini program package

This paper introduces the principle and implementation process of the preset mini program.

What is a preset mini program?

Preset mini programs refer to the process of packaging static resources such as rendering, logic, and configuration of mini programs into a compressed package and integrating them into the client app. The mini program container can directly load resources from the local machine. Preset mini programs to minimize the impact of network environment on the mPaaS mini program page. Prebuilt packages provide the following benefits:

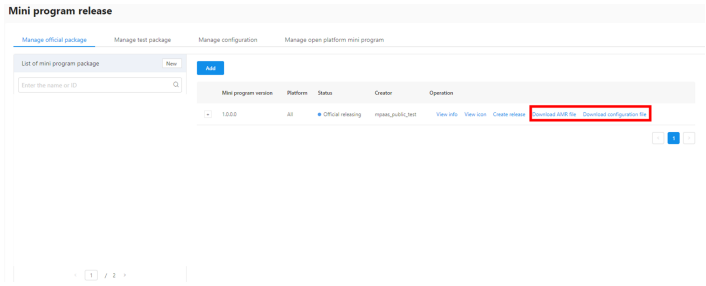
- **Improve user experience**. You can embed static resources on a page into an application and publish the resources along with the application by using a preset package. When you open an application for the first time, you do not need to rely on the network environment to download resources.
- **Implement dynamic update** When a new version is released or an emergency release is made, you can perform iterative development in the mini program IDE and release it in the mPaaS console. The integrated mini program SDK in the client automatically updates the mini program to the latest version. This release does not require App Store review and allows users to receive updates early.

Prerequisites

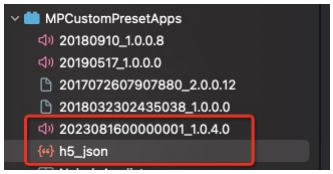
You have connected to the mini program component. For more information about how to install mini programs, see Quick start.

Procedure

1. Release the mini program package in the mPaaS console and download the AMR file and configuration file.



2. Create a separate bundle. If `DemoCustomPresetApps.bundle`, add the AMR offline package and `h5_json.json` files downloaded from the releasing platform to this bundle.



> **Note**
> Currently, the publishing platform only supports downloading the `h5_json.json` configuration file of a single offline package. When multiple mini program packages are preset, you need to manually merge data from different `h5_json.json` into one configuration file.

3. Specifies the path of the out-of-box mini program when initializing the mini program. Use the `initNebulaWithCustomPresetApplistPath` to initialize the container and set the offline package path of the preset mini program to the bundle created in the previous step.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
 // Initialize RPC.
 [MPRpcInterface initRpc];

  // Initialize the container and customize the preset mini program package information.
 NSString *presetApplistPath = [[NSBundle mainBundle] pathForResource:[NSString
stringWithFormat:@"DemoCustomPresetApps.bundle/h5_json.json"] ofType:nil];
 NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
 [MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath customPresetAppPackagePath:appPackagePath
customPluginsJsapisPath:@""];
}
```

4. Start the mini program. Similar to a non-preset mini program, when entering the corresponding page, call the interface method provided by the container to load the mini program.

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

1.3.9. Permission management

1.3.9.1. Mini program permissions

Some special APIs of mini programs, such as location, camera, and photo album, usually prompt the user for authorization. These APIs can be executed only after the user allows it.

The mini program container allows the following extensions for API calls:

1. Custom text prompts, the access party can control the text and display style.
2. Allows the access party to read and write permissions configuration.

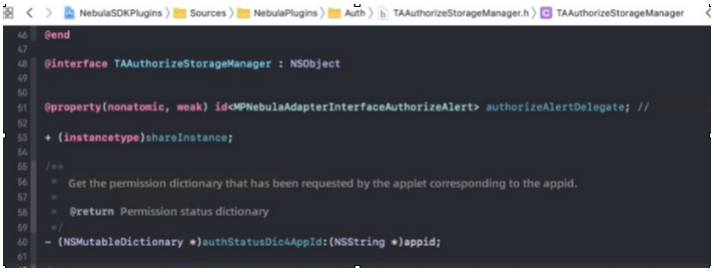
> **Note**
> This extended configuration is available only when mini program permission control is enabled in the background.

Permission configuration

The mini programs' default key and the corresponding API are shown in the following table:

Permission	key	API
camera	camera	scan, chooseImage, chooseVideo
photo album	album	saveImage, saveVideosToPhotosAlbum
location	location	getLocation, getCurrentLocation
microphone	audioRecord	startAudioRecord, stopAudioRecord, cancelAudioRecord

You can obtain a dictionary of permissions that the current mini program has requested:



Custom display

mPaaS supports to customize the dialog box of permissions.

The steps are as follows:

1. After the container is initialized, specify the delegate in the custom permissions dialog box.

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

    // Manage mini program API permissions.
   [TAAuthorizeStorageManager shareInstance].authorizeAlertDelegate = self;

    ...
}
```

2. The method of implementing a custom permissions dialog box.

```
#pragma mark mini program API permission control
- (void)showAlertWithTitle:(NSString *)title appName:(NSString *)appName storageKey:(NSString *)storageKey callback:(void (^)(NSInteger ind
ex))callback
{
    if ([title length] > 0) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:title
                                                        message:nil
                                                       delegate:self
                                              cancelButtonTitle:@"Cancel"
                                              otherButtonTitles:@"OK", nil];

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

}

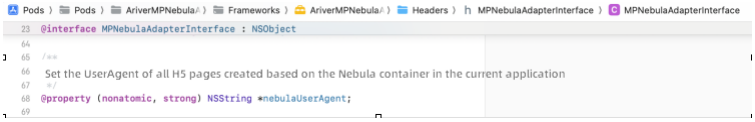
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:(NSInteger)buttonIndex
{
    if (self.callback) {
        if (buttonIndex == alertView.cancelButtonIndex) {
          // Authorization is allowed for the user. The callback parameter is 0.
          self.callback(0);
        }else{
    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

}
```

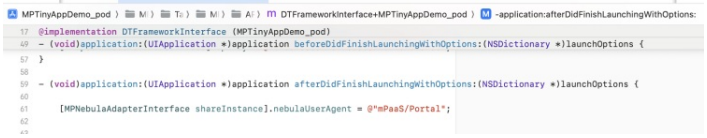
1.3.10. Customize containers

1.3.10.1. Custom UserAgent

The container allows you to add a custom userAgent, which allows you to set the userAgent for all H5 pages of the current application.



You can call in a `- (void)application:(UIApplication )application afterDidFinishLaunchingWithOptions:(NSDictionary )launchOptions` from `DTFrameworkInterface`.



Sample code:

```
[MPNebulaAdapterInterfaceshareInstance].nebulaUserAgent = @"mPaaS/Portal";
```

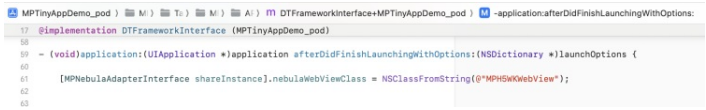
1.3.10.2. Custom WebView Base Class

The default webView class of the container is H5WKWebView. The steps to customize the webView class are as follows:

1. The custom webView class must inherit H5WKWebView.



2. In the `- (void)application:(UIApplication )application afterDidFinishLaunchingWithOptions:(NSDictionary )launchOptions` of `DTFrameworkInterface`, set the base class for the WebView container.



Sample code:

```
[MPNebulaAdapterInterface shareInstance].nebulaWebViewClass = NSClassFromString(@"MPH5WKWebView");
```

1.3.10.3. Custom Container Base Class

The default VC base class for the container is `NXDefaultViewController`. If you need to customize the VC class for custom development, you can perform the steps below:

1. The VC base class to customize must inherit `NXDefaultViewController`.



2. In the `- (void)application:(UIApplication )application afterDidFinishLaunchingWithOptions:(NSDictionary )launchOptions` of `DTFrameworkInterface`, set the base class of the container.

Sample code:

```
[MPNebulaAdapterInterfaceshareInstance].nebulaVeiwControllerClass = NSClassFromString(@"HXH5WebViewController");
```

1.3.10.4. Custom SDK

By default, the mini program component integrates all the API capabilities. You can remove APIs that are not used by the current mini program as needed. including:

Components supported by remove_pod	Component description	Included APIs
mPaaS_TinyApp_Audio	Mini program audio	audio
mPaaS_TinyApp_Device	Mini program device	my.getClipboard my.setClipboard my.watchShake(OBJECT) my.vibrate(OBJECT) my.vibrateLong(OBJECT) my.vibrateShort(OBJECT) my.onAccelerometerChange(function callback) my.onGyroscopeChange(function callback) my.offGyroscopeChange() my.onCompassChange(function callback) my.offCompassChange() my.makePhoneCall(OBJECT) my.onUserCaptureScreen(CALLBACK) my.offUserCaptureScreen() my.setKeepScreenOn(OBJECT) my.getScreenBrightness(OBJECT) my.setScreenBrightness(OBJECT) my.addPhoneContact()

mPaaS_TinyApp_Multimedia	Mini program multimedia	my.chooseImage my.previewImage my.saveImage my.compressImage my.getImageInfo
mPaaS_TinyApp_Player	Mini program player	video
mPaaS_TinyApp_Scan	Mini program scan code	my.scan

Procedure

1. Install the beta version of the cocoapods-mPaaS plugin.

   > **Note**
   > The cocoapods-mPaaS plugin beta is only supported in baselines 10.2.3 and above.

   ```
   sh <(curl -s http://mpaas-ios-test.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaSCocoaPodsPlugin.sh)
   ```

   After the installation is complete, run the command `pod mpaas version --plugin` to confirm that it is the beta version.

2. Rerun the command to update the local baseline: `pod mpaas update 10.2.3` .

   > **Note**
   > If your baseline is `cp_change_15200851` , run `pod mpaas update cp_change_15200851` to update the baseline.

3. Before using the `mPaaS_pod` command, introduce `remove_pod "mPaaS_xxx"` in `podfile` . For example, use `remove_pod "mPaaS_TinyApp_Audio"` before `mPaaS_pod "mPaaS_Ariver"` to remove the mini program audio.

   ```
   remove_pod "mPaaS_TinyApp_Audio"

   mPaaS_pod "mPaaS_Ariver"

   pod 'xxx' # Corresponding third-party native libraries
   ```

1.3.11. Switch the map

1.3.11.1. Use Google Maps in Mini Program

1. When the client is integrated, add the following default switch configuration under `nebulaCustomConfig` method:

   ```
   - (NSDictionary *)nebulaCustomConfig
   {
        return @{
                @"ta_map_engine": @"1"
        };
   }
   ```

2. Add the `MPGoogleMap.framework` into the project directory.

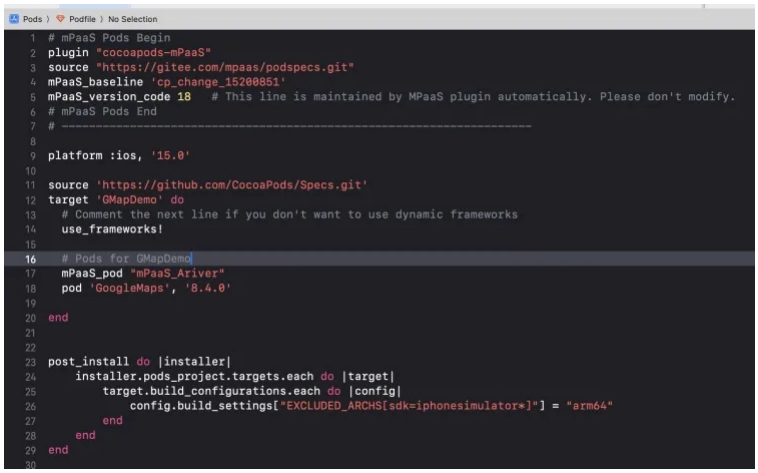
   Simulator architecture:

   MPGoogleMap.framework.zip

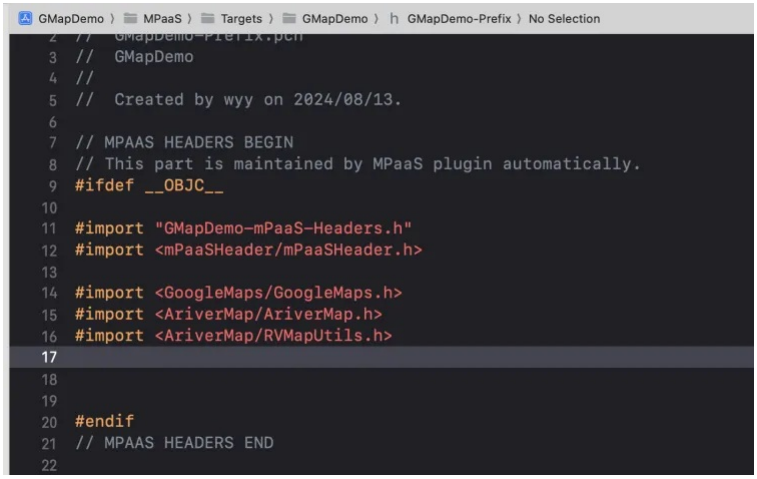
   Real machine architecture:

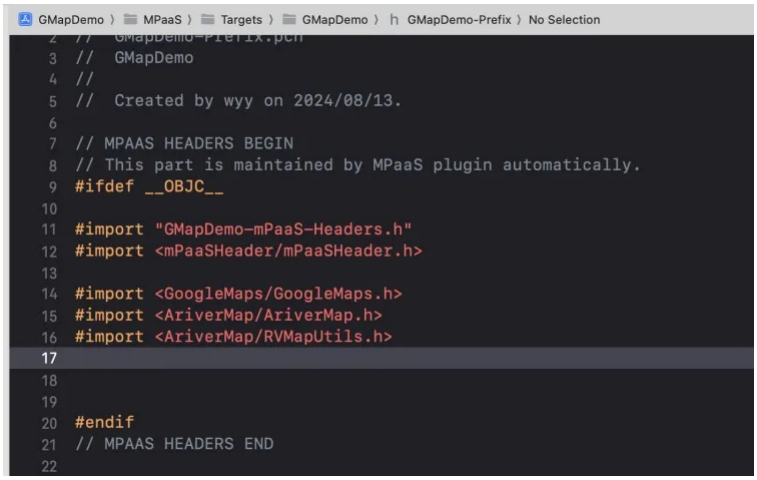
   MPGoogleMap.framework.zip

3. Configure the pod to integrate the Ariver component and GoogleMap, and configure the location-related privacy field in the `info plist` .



4. Configure header files and initialization.





1.3.11.2. Use Baidu Map in Mini Program

my.getLocation supports Baidu SDK

1. Set the switch in the initialize method.

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
 [LBSmPaaSAdaptorsharedInstance].shouldSkipLBSLocation = YES;
}
```

2. Implement the `AriverMapEngineDelegate` and `BMKGeoCodeSearchDelegate` protocols on the page. Import the header file `#import <APMobileLBS/LBSmPaaSAdaptor.h>`.

```
@interfaceViewController : UIViewController<AriverMapEngineDelegate,BMKGeoCodeSearchDelegate>
@property (nonatomic, copy) mapReGeocodeCompletedBlockreGeoBlock;
@end
- (void)viewDidLoad {
 [superviewDidLoad];
 [LBSmPaaSAdaptorsharedInstance].ariverMapEngineDelegate = self;
}
```

3. Implement the methods in the protocol.

```
#pragma mark - AriverMapEngineDelegate
-(void)mPaaSReverseWithLocation:(APMapReGeoRequest*)reGeoRequest completion:(mapReGeocodeCompletedBlock)completion {
 BMKGeoCodeSearch *search = [[BMKGeoCodeSearchalloc] init];
  search.delegate = self;
  BMKReverseGeoCodeSearchOption *reGeoCodeSearchOption = [[BMKReverseGeoCodeSearchOptionalloc]init];
  reGeoCodeSearchOption.location = reGeoRequest.coordinate;
  self.reGeoBlock = completion;
  BOOL flag = [search reverseGeoCode:(BMKReverseGeoCodeSearchOption *)reGeoCodeSearchOption];
}
```

```
#pragma mark - BMKGeoCodeSearchDelegate
/**
*Return reverse geocoding search results
*@param searcher Search object
*@param result Search result
*@param error Error code, @see BMKSearchErrorCode
*/
- (void)onGetReverseGeoCodeResult:(BMKGeoCodeSearch *)searcher result:(BMKReverseGeoCodeSearchResult *)result errorCode:
(BMKSearchErrorCode)error {
  if (result) {
    NSDictionary* resultDic = [selfparseLBSAddressInfo:result searcher:searcher];
    self.reGeoBlock(resultDic, nil);
  }else{
    NSMutableDictionary *infoDic = [[NSMutableDictionaryalloc] initWithCapacity:1];
    infoDic[NSLocalizedDescriptionKey] =    __TEXT(@"APMobileLBS",@"APMobileLBS:APMapService.requestFail", @"Request result is empty");

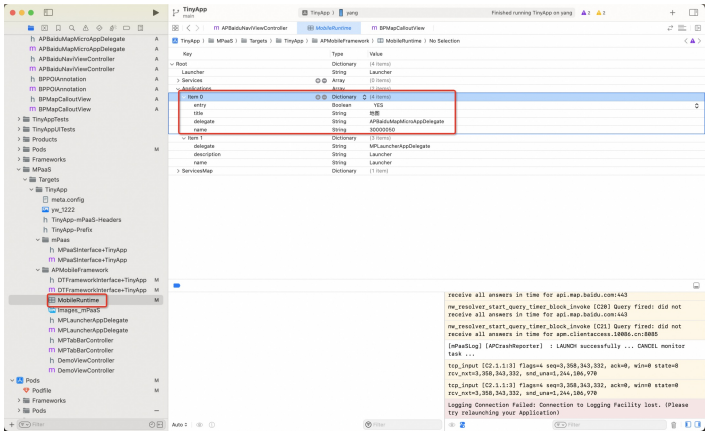
    NSError *resultError = [[NSErroralloc] initWithDomain:@"APMapService"code:error userInfo:infoDic];
    self.reGeoBlock(nil, resultError);
  }
}
```

```
 (NSDictionary )parseLBSAddressInfo:(BMKReverseGeoCodeSearchResult )result
searcher:(BMKGeoCodeSearch *)searcher
{
  if (!result) {
   return @{};
  }
  NSMutableDictionary* resultDic = [NSMutableDictionary dictionary];
  resultDic[@"longitude"] = [NSNumber numberWithDouble:result.location.longitude];
  resultDic[@"latitude"] = [NSNumber numberWithDouble:result.location.latitude];
  resultDic[@"accuracy"] = [NSNumber numberWithInteger:result.confidence];
  if (result.addressDetail) {
    resultDic[@"country"] = result.addressDetail.country;
    resultDic[@"countryCode"] = result.addressDetail.countryCode;
    resultDic[@"city"] = result.addressDetail.city;
    resultDic[@"cityAdcode"] = result.addressDetail.adCode;
    resultDic[@"province"] = result.addressDetail.province;
    resultDic[@"district"] = result.addressDetail.district;
    resultDic[@"street"] = result.addressDetail.streetName;
    resultDic[@"number"] = result.addressDetail.streetNumber;
  }
  resultDic[@"timestamp"] = [NSDate date];
  resultDic[@"pois"] = result.poiList;
  resultDic[@"roads"] = result.roads;
  return resultDic;
}
```

Map component /my.openLocation supports Baidu Map

> **Important**
> The cp_change_15200851 baseline version must be 40 or higher.

1. After extracting MPBaiduMap.framework.zip, add it to the project.
2. Create the `APMobileFramework/MobileRuntime.plist` file in the project folder if it does not exist.
3. Add a new item under the Applications node in MobileRuntime.plist:
   - delegate: APBaiduMapMicroAppDelegate
   - name: 30000050
   - title: Map
   - entry: YES (BOOL)



1.4. Mini Program container - Nebula (not recommended)

1.4.1. Integrate Mini Program into Android

1.4.1.1. Quick start

> **Important**
> - Android Nebula Mini Program will gradually stop maintenance. Please use the new container for mini programs. For more information about the integration, please refer to Quick start.
> - If you have already used Nebula Mini Program, we recommend that you migrate to the new mini program container so that you can continue to enjoy our high-quality services. Thank you for your understanding and support.
> - The new mini program container has been fully upgraded to provide more features including basic library 2.0, debug panel, plugins, etc., and is compatible with the new version of mPaaS Web kernel.

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

> **Note**
> - You can access Mini Program SDK to Android client through Native AAR or Portal & Bundle accessing method. For more information, see the Accessing method introduction.
> - Mini Program is only available in 10.1.60 and higher baseline versions.

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.

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, you need to initialize mPaaS.You need to add the following code in `Application`:

```
public class MyApplication extends Application {

    @Override
    public void onCreate() {
        super.onCreate();

        MP.init(this,
            MPInitParam.obtain().setCallback(new MPInitParam.MPCallback() {
                @Override
                public void onInit() {
                    // Initialize the public resource package of the mini program
                    H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                }
            })
        );
    }

}
```

For more details, see Initialize mPaaS.

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

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

> **Note**
> 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 package.

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.

- **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 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`.

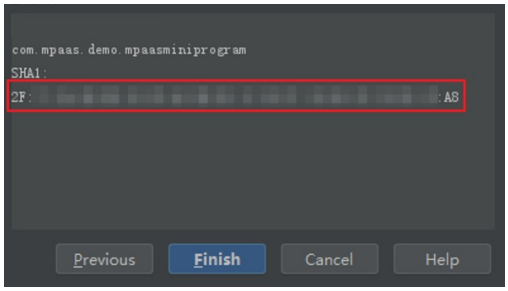
```
<application>
    ...
<meta-data android:name="nebula.android.meta.enable" android:value="true"/>
  ...
</application>
```

1.4 Apply for UC kernel

Before using the mini program, you need to apply for and configure the UC kernel. Without the UC kernel, you will not be able to use some capabilities of the Android mini program.

> **Note**
> Due to changes in product strategy, UC is no longer fully open for applications. Public application for UC Key will not be supported starting from 2022.12.01. You need to submit a UC Key application form, and the staff will review it and feedback the results of the application.

1. Click **mPaaS** > **Basic Tools** > **Generate UC Key Signature Information** to open the **Query Signature Information** window.
2. In the **Query Signature Information** window, fill in the relevant configuration information and click **Next**.
3. Copy the obtained SHA1 information.



4. Fill in the Key you applied for in the previous step into the project's `AndroidManifest.xml` file:

```
<meta-data android:name="UCSDKAppKey" android:value="The Key you applied for"/>
```

> **Note**
> The authorization information of UC SDK is bound to the package name and signature of the apk. Therefore, if UCWebView does not take effect, check whether the signature and package name are consistent with the information used when applying.

Using the UC kernel, mini programs can have the same layer capabilities, such as embedding webviews, embedding maps, etc., and have 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, enter the target application and select **Mini Program** > **Release Mini Program** 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.

> **Note**
> Be sure to use a domain name that you registered yourself.

2.3 Create a mini program

Enter the mPaaS console, and complete the following steps:

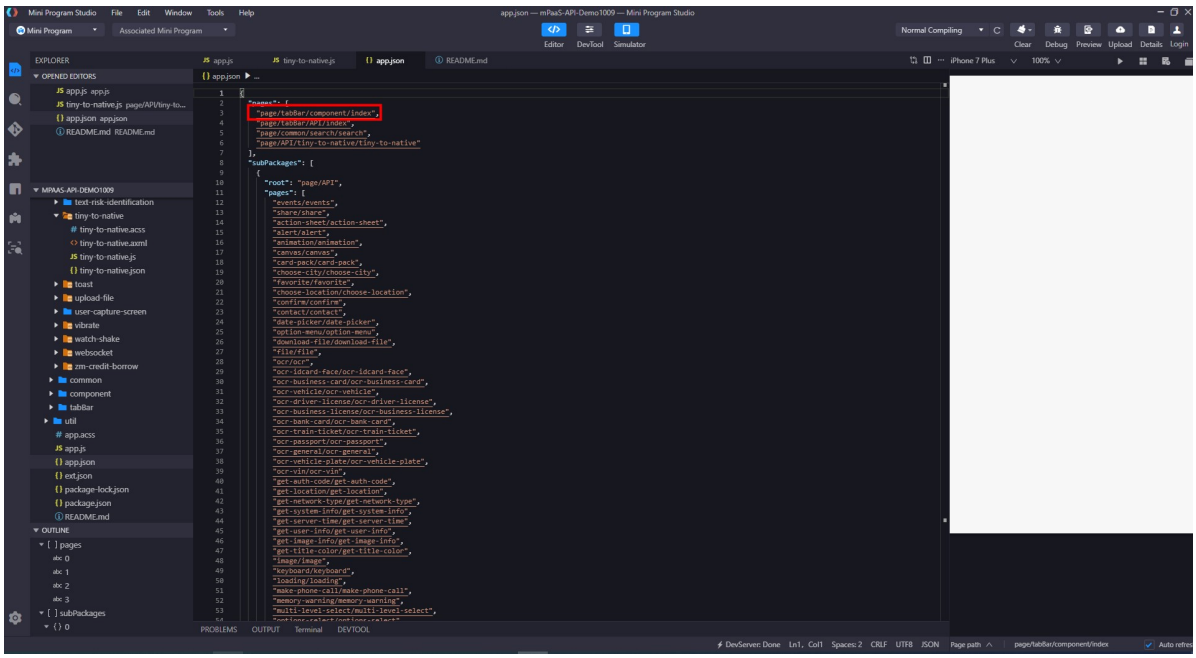
1. Click the **Mini Program > Release Mini Program** from the left navigation pane.
2. On the mini program list page, click **Create**.
3. In the **Create New Mini Program** popup, enter the mini program ID and name, then click **Confirm**. The mini program ID is a 16-digit number, for example `2018080616290001`.
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. On the configuration information page, 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 other 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. On the release task creation page, 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.

1.4.1.2. Advanced guide

1.4.1.2.1. Integrate real-device preview and debugging function

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

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

1. Add the value of `h5_remote_debug_host` in HTML5 container configuration file (`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. In Domain Name Management, you can obtain the virtual domain name you filled in previously.
   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 parame
ter is available, just fill new Bundle().

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

   - If you are using a baseline lower than 10.1.68.6 or 10.1.60, use the following code for parsing.

```
// 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);
}
```

1.4.1.2.2. Customize navigation bar

Prerequisites

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

```
{
    "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

1.4.1.2.3. Customize bidirectional channel

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. The client customizes and registers the API.
   - 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:" + context.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.

1.4.1.2.4. API permission extension configuration

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 m
ini program 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 display

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
permissionCheckCallback) {
        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**
>
> - Mini Program configuration is stored in two dimensions: application and user. Therefore, make sure that the application has called the `MPLogger.setUserId` method.
> - If the 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;

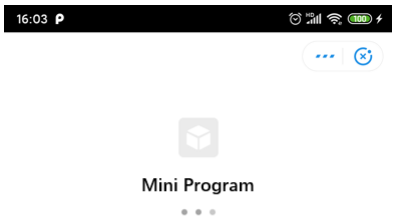
    private AUSearchBar mSearchInputBox;

    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(PermissionDisplayActivity.this, 48)));
                }
            }
        });
    }
}
```

1.4.1.2.5. 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.



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.

1.4.1.2.6. Extend upper-right pop-up menu

This section describes how to extend custom options in the pop-up menu and respond user's clicking 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;
    }
});
```

1.4.1.2.7. Set entry/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.

Enable the function of entry and exit animation

Add the parameter `needAnimInTiny` after starting the Mini Program, and set the value as true. The code example is as follows:

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

Set entry animation

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 exit animation

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>
```

1.4.1.2.8. Specify the redirect page at startup

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 Getting started.

Procedure

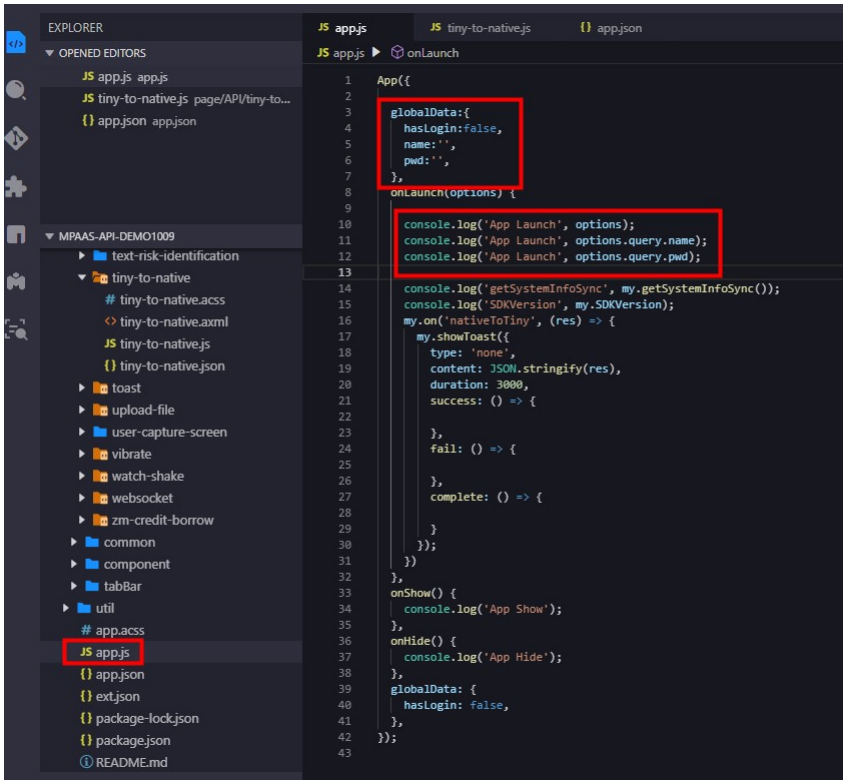
1. Add parameter information of page redirection during startup on the client. The parameter passing method is as follows:

```
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.



1.4.1.2.9. 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 Getting started.

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);
```

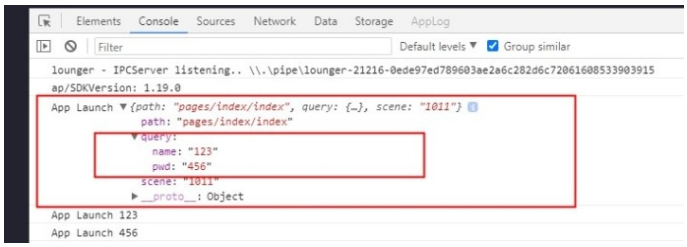
When the URL starts passing parameters, the field for passing parameters is `query`; when obtaining parameters, they are obtained by parsing the `query` field.

Description of startApp parameter:

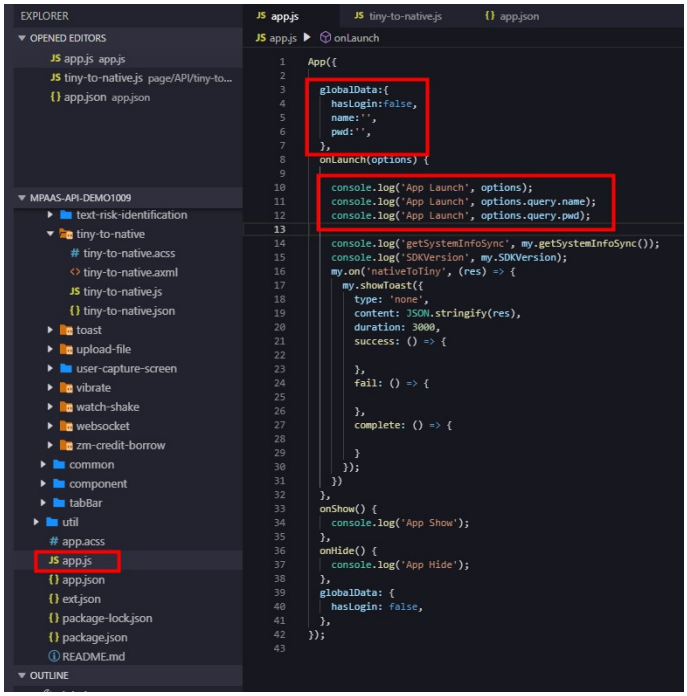
- `appId`: The Mini Program ID, which can be viewed in the mPaaS console.
- `param`: The Bundle object. You can pass request parameters to the Bundle object in the format of `key="query",value="Key-value pair"`. Separate parameters with ampersand (&).

> **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.



1.4.1.2.10. 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 description 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 Getting 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();
```

> **Note**
> - 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.

1.4.1.2.11. Implement 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 mini program can be started after setting these two properties.
   param.putBoolean("appClearTop", false);
   param.putString("startMultApp", "YES");
   MPNebula.startApp("2021042820210428",param);
   ```

1.4.1.2.12. Customize View in Android Mini Program

> **Note**
>
> 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
    }
    ...
}
```

Implement 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");
```

Important

- 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 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

1.4.1.2.13. Customize rendering parameters in custom View

> **Note**
> 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);
    }
    ...
}
```

1.4.1.2.14. Send custom message to custom View

> **Note**
> 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

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

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

Client receives message and processes native View

```
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")));
        }
    }
    ...
}
```

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).

1.4.1.2.15. Send custom event by custom View

> **Note**
> 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`.

Process 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.

1.4.1.3. Instructions on upgrading Mini Program

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 API.

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

- Set App version: Obtain the App version through the 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 integrate Mini Program through the Portal & Bundle method, you should ensure:

- The minimum version of `com.alipay.android:android-gradle-plugin` is 3.0.0.8.0.
- To use mini programs and H5 under Portal&Bundle, you must configure `enableNebulaMetaInfo` and `useMetaInfoClass` to `true` in the `portal` field of the `build.gradle` file of the main module of the project. The example is as follows:

```
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
}
```

1.4.1.4. Tutorial

1.4.1.4.1. Overview

In the process of integrating mPaaS for development, the most common scenario is to integrate mPaaS based on the existing native Android project.

Scenario

In the process of integrating mPaaS for development, the most common scenario is to integrate 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, integrating mPaaS, integrating Mini Program for development to finally releasing a mini program and starting it.

Select integration method

mPaaS supports multiple integration method, refer to Integration method introduction. If you want to easily integrate and use mPaaS just like using other SDK, you can adopt the **Native AAR integration method**.

Native AAR integration 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 integration 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) integration method**, and want to quickly use mPaaS capabilities. The tutorial adopts Native AAR integration method to integrate mPaaS.

What you will 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 integration 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.

1.4.1.4.2. 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.

Write 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.

1.4.1.4.3. 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 Management** > **Code Configuration** > **Android**, 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.

1.4.1.4.4. 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.

1.4.1.4.5. Initialize configuration

The configuration initialization covers the following 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
    public void onCreate() {
        super.onCreate();

        MP.init(this,
            MPInitParam.obtain().setCallback(new MPInitParam.MPCallback() {
                @Override
                public void onInit() {
                    // Initialize the public resource package of the mini program
                    H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                }
            })
        );
    }

}
```

For more details, see Initialize mPaaS.

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**
> Due to product policy changes, UC is no longer fully open for applications. Public application for UC Key will not be supported from December 1, 2022. The UC Key application form needs to be submitted, and the staff will review and give feedback on the results of the application.

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. 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.

1.4.1.4.6. 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.

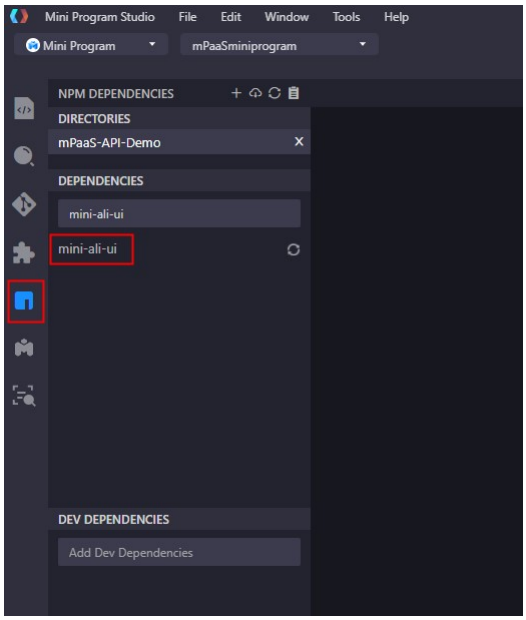
> **Note**
> The downloaded configuration file is named config.json by default.

Log in to Mini Program IDE

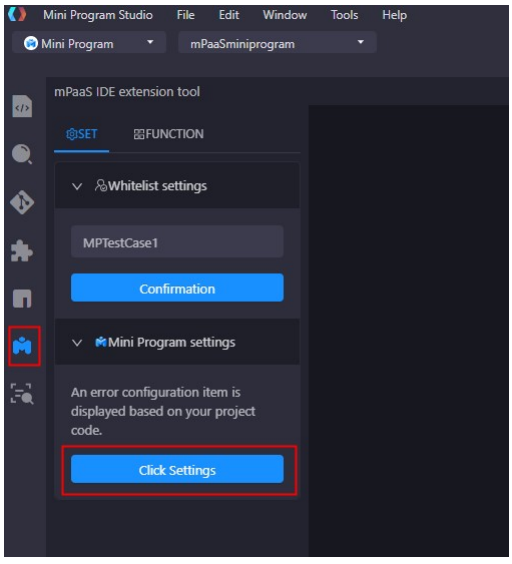
1. In the Mini Program IDE, click the upper-right **Log in**.
2. In the pop-up **New login environment** window, enter the environment name "mPaaSminiprogram", upload the **Mini Program IDE configuration file** (`config.json` file) downloaded from the mPaaS console, and then click **OK**.
3. In the login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

1.4.1.4.7. Start mini program

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.

1.4.1.4.8. 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 6 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.

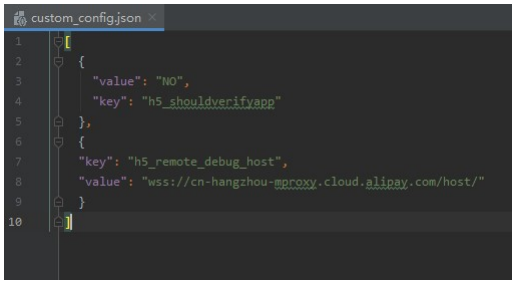
```
{
 "login_url":"https://mpaas-mappcenter.aliyuncs.com/ide/login",
 "uuid_url":"http://cn-hangzhou-mproxy.cloud.alipay.com/switch/uuid",
 "debug_url":"wss://cn-hangzhou-mproxy.cloud.alipay.com",
 "appId":"ONEX0D29541291400",
 "sign":"674f12adf7205358108839eb79f8a487",
 "tenantId":"AUDQKVYH",
 "upload_url":"https://mpaas-mappcenter.aliyuncs.com/ide/mappcenter/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



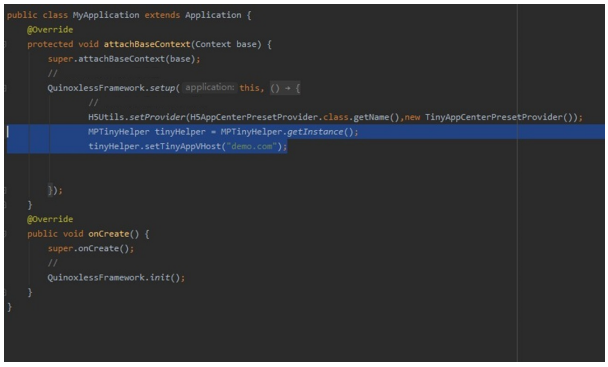
value: “NO” indicates that the mini program signature verification is disabled, while “YES” indicates that the it is enabled (it defaults to “YES” if you do not fill in). In the development and debugging stage, you can turn off the signature verification for quick access; before going online, it is recommended 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:

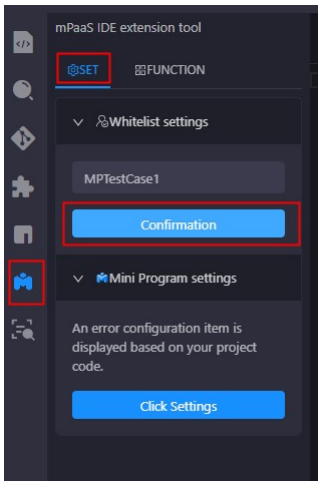


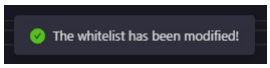
Set the VHost and user whitelist of the mini program

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);
        //
        QuinoxlessFramework.setup( application: this, () -> {
                //
                H5Utils.setProvider(H5AppCenterPresetProvider.class.getName(),new TinyAppCenterPresetProvider());
                MPTinyHelper tinyHelper = MPTinyHelper.getInstance();
                tinyHelper.setTinyAppVHost("demo.com");
                MPLogger.setUserId("MPTestCase1");

        });
    }
    @Override
    public void onCreate() {
        super.onCreate();
        //
        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 {

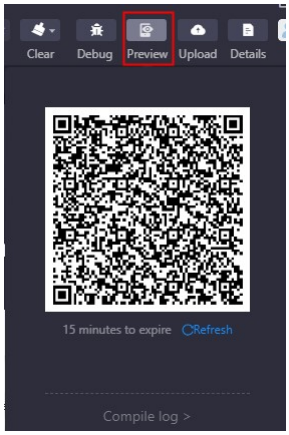
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.my_tv).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                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()){
                            //
                            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

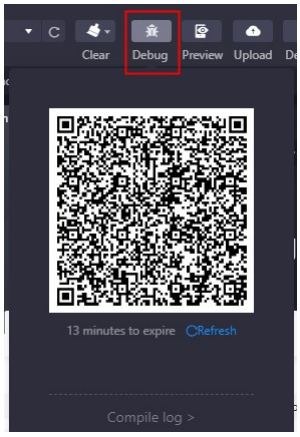
1. Click **Preview** in the navigation bar of mini program development tool to generate a QR code.



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.

1.4.1.4.9. 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 {

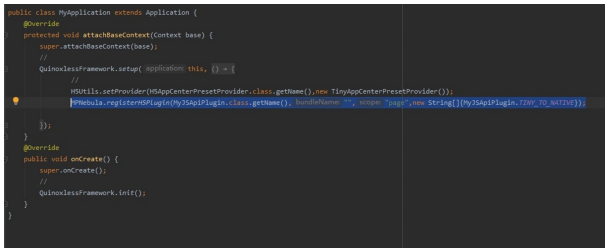
    /**
     * Custom API
     */
    public static final String TINY_TO_NATIVE = "tinyToNative";

    @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 De
mo:" + 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});
```



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

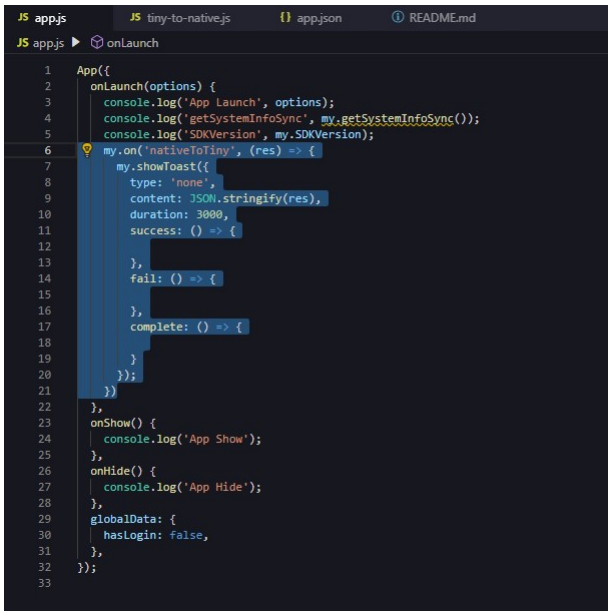
   

   to run the application on real device.
5. In the application running on the real device, click **Hello World!** to start up the mini program.
6. Click **API** tab to open the API page.
7. Click **Custom API** to open the custom API page.
8. Click **Click to trigger custom API** button, a toast pops up 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` click 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("20200929000000001");
             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

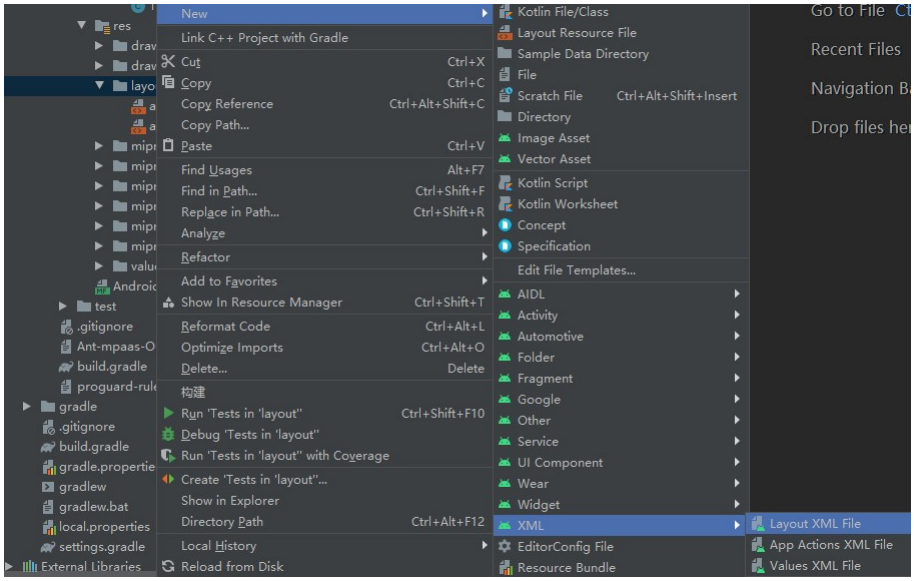
   to run the application on real device.
4. In the application running on the real device, click **Hello World!** to start up the mini program.
5. The mini program receives an event in every 5 seconds, and the passed information is displayed in a toast.

1.4.1.4.10. Customize startup loading page

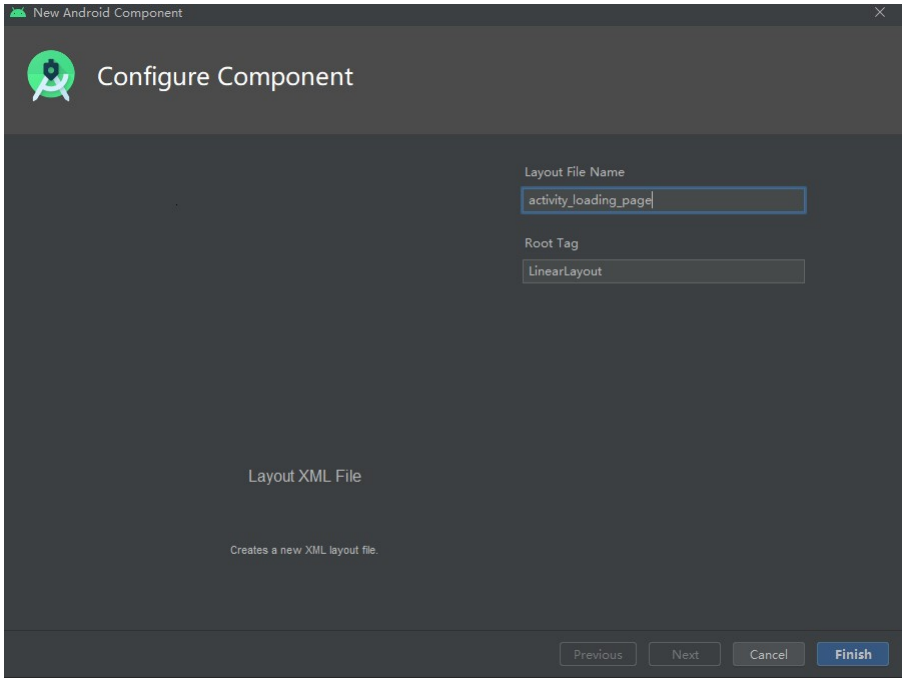
When you start up the mini program, if the mini program has not been downloaded to the device, the mini program container will launch a loading page to prompt the user to wait until the mini program is installed on the device. After that, the loading page will be closed and the mini program will appear.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

Procedure

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



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. Click **Hello World!** to start up the mini program. The loading page after starting up the mini program will be shown.

1.4.1.4.11. 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

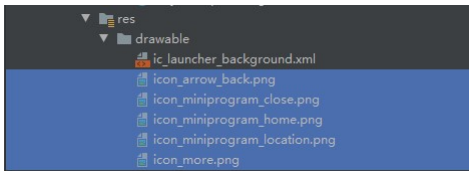
Procedure

Set the navigation bar of the mini program

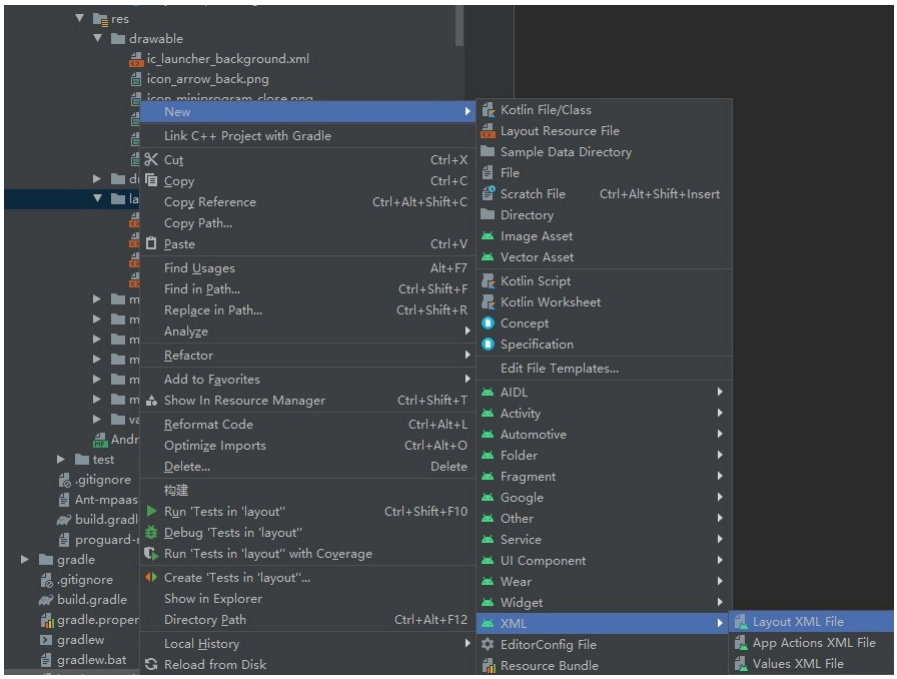
1. Open Android Studio, find the `custom_config.json` file under **assets** > **config** folder, and add the following code to disable the native navigation bar of the mini program.

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

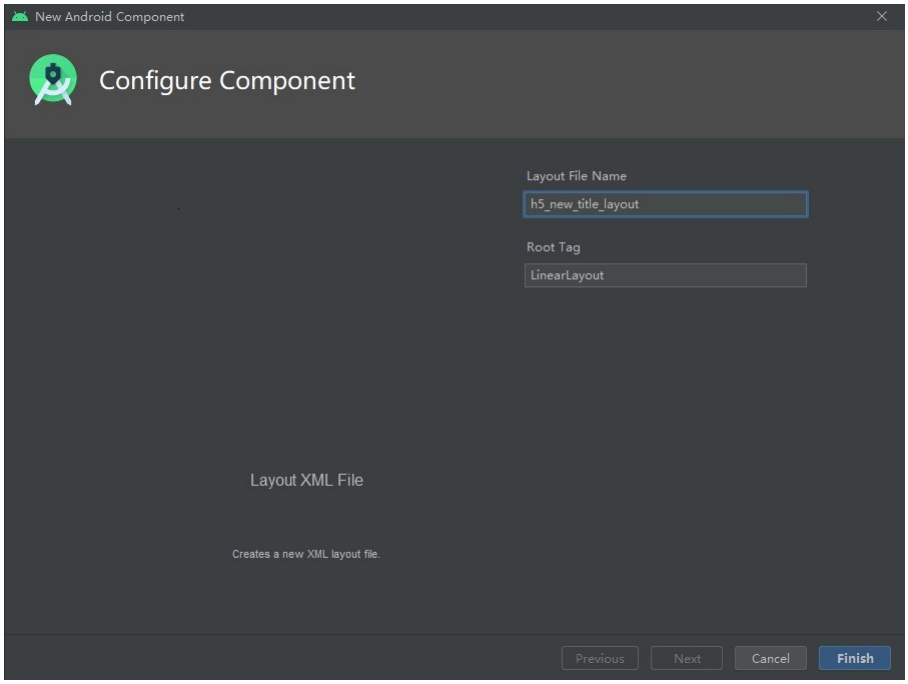
2. In **res** > **drawable** folder, add the image resource as shown in the following figure. Click here to download the image resource.



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" />

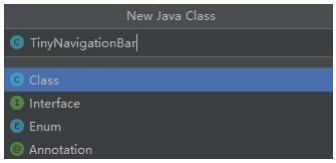
     </LinearLayout>

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

         <ImageView
             android:id="@+id/o1image"
             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;

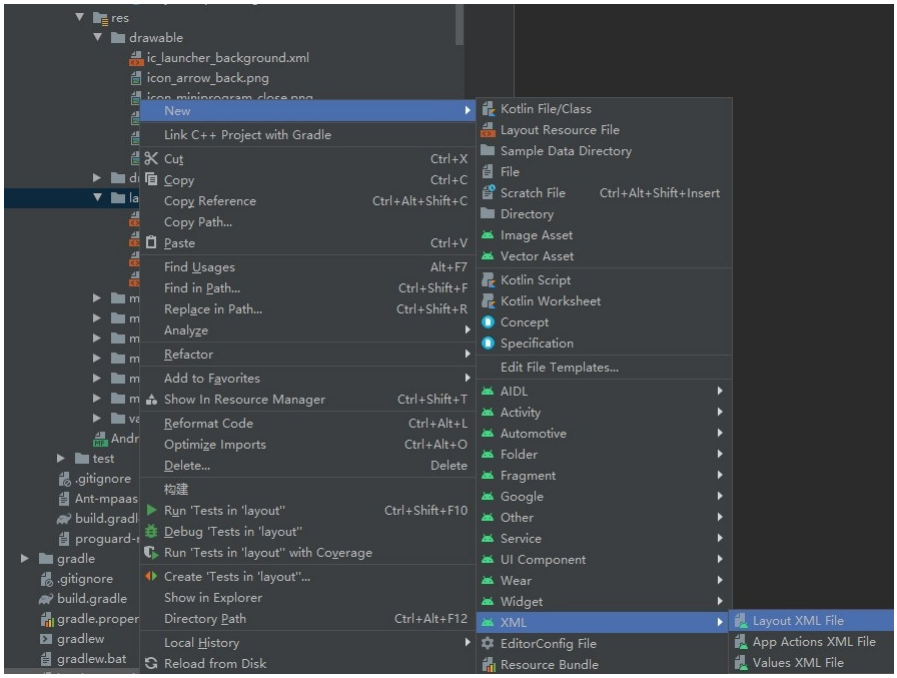
    private View btHome;

    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);
```

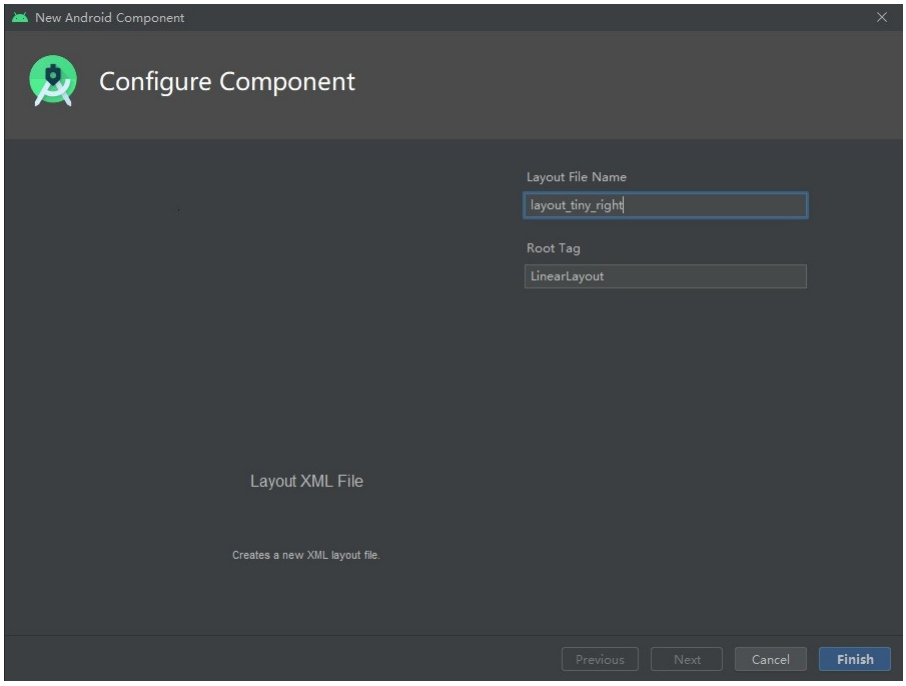
```
        mainTitleView.setText(s);
    }
    @Override
    public void setSubTitle(String s) {
        subTitleView.setText(s);
    }
    @Override
    public void setTitleImage(Bitmap bitmap) {
    }
    @Override
    public TextView getMainTitleView() {
        return mainTitleView;
    }
    @Override
    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.



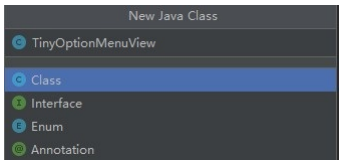
2. Enter **Layout File Name**, and click **Finish**.



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);
                    }
                });
```

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

1. Click

   to run the application on real device.
2. Click **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. Click the **Icon** under **Basic component**, then you can see the the layout of the custom navigation bar in **Icon** page.

1.4.2. Flutter Project Access Guide

For more information about how to configure and initialize an mPaaS environment for a Flutter project, see Flutter Project Integration Operation Guide.

Use mini program

1. Register the mini program in the `configureFlutterEngine` of `FlutterActivity`.

```
val messenger = flutterEngine.dartExecutor.binaryMessenger
// Create a Channel object
val channel = MethodChannel(messenger, "mpaas_mini_app")

// Set a callback for the channel
channel.setMethodCallHandler { call, res ->
    // Distribute different processing based on the method name
    when(call.method) {

        "mpaas_mini_app" -> {
            // Obtain the passed parameters
            val msg = call.argument<String>("msg")
            MPNebula.startApp(msg)
            // Notify that the execution is successful
            res.success("This is the result of the execution.")
        }

        else -> {
            // If an unrecognized method name is found, the execution fails.
            res.error("error_code", "error_message", null)
        }
    }
}
```

2. Use mini program in Flutter.

```
// Create a channel.
const channel = const MethodChannel("mpaas_mini_app");


Widget buttonView() {
  return  TextButton(
    child: Text("Open the mini program"),
    onPressed: () {
      callNativeMethod("2023072011111112");
    },
  );
}

void callNativeMethod(String msg) {
  try {
    // Call the native code method through the channel.
    Future future = channel.invokeMethod("mpaas_mini_app", {"msg": msg} );
    // Print the execution result.
    print(future.toString());
  } on PlatformException catch(e) {
    print(e.toString());
  }
}
```

1.4.3. Integrate Mini Program into iOS

1.4.3.1. 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:

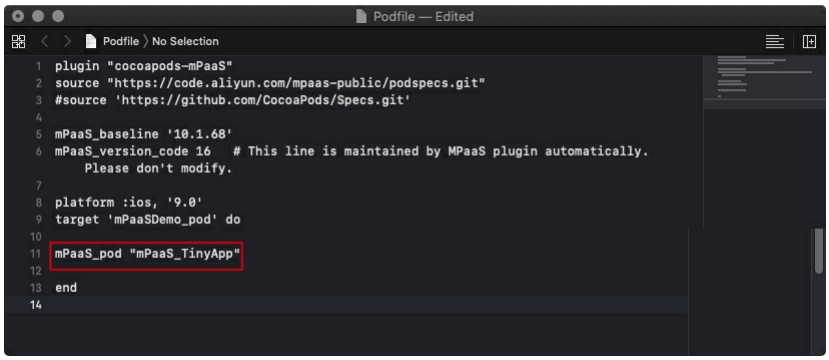
Access based on existing projects and using CocoaPods

Add the SDK

Use CocoaPods-mPaaS plugin to add mini program SDK.

The steps are as follows:

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



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

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 customPlu
ginsJsapisPath: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, 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`.

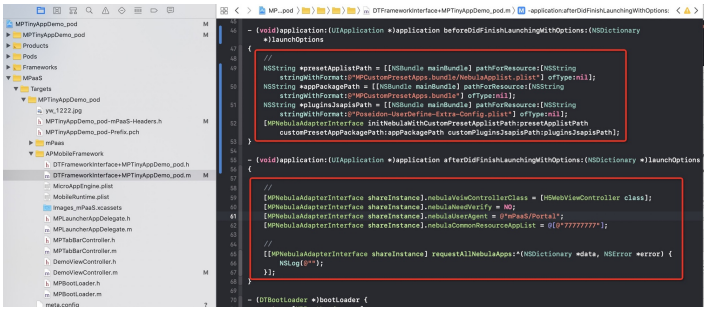
nebulaUserAgent	Set the UserAgent of the application	The set UserAgent will be added as a suffix to the default UA of the container.
nebulaNeedVerify	Whether to verify or not, the default value is YES	If the private key file is not uploaded when configuring the mini program package, this value must be 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`.

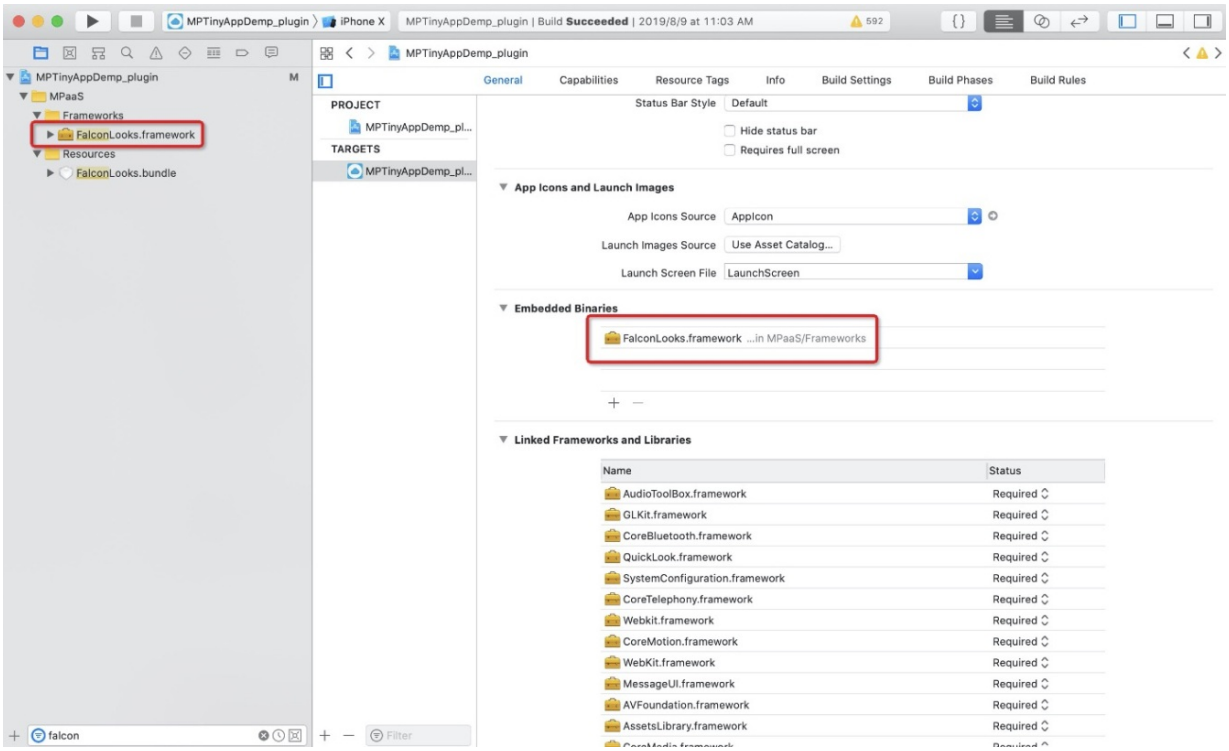
```
- (void)application:(UIApplication *)application afterDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Customize the container
    [MPNebulaAdapterInterface shareInstance].nebulaVeiwControllerClass = [MPH5WebViewController class];
    [MPNebulaAdapterInterface shareInstance].nebulaNeedVerify = 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:



Add the `FalconLooks` library in **General** > **Embedded Binaries** of the current project's **TARGETS**.

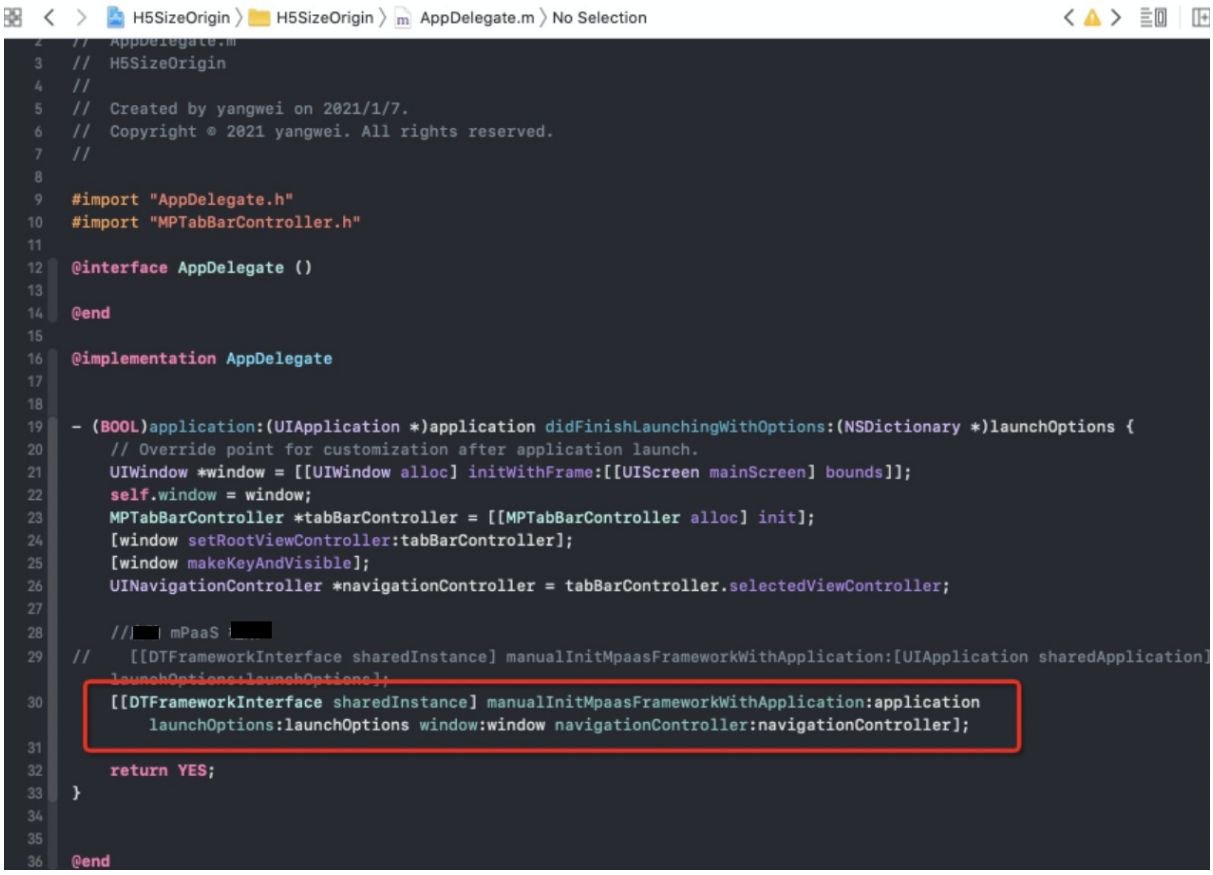
> **Note**
>
> The configuration of dynamic libraries has been cancelled in baselines of versions 10.1.68.15 (inclusive) and above, and no configuration is required.



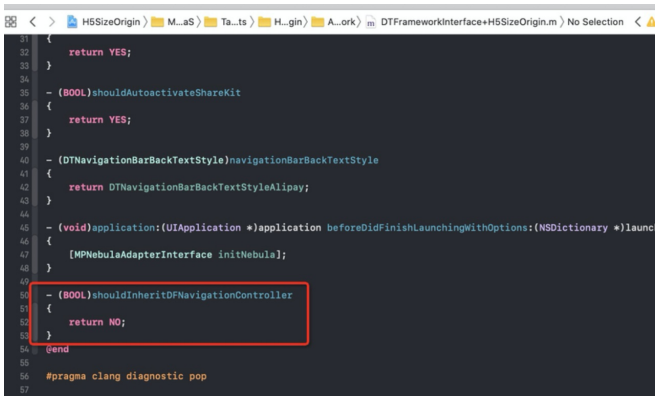
1.2 Non-framework host configuration (version 10.1.68.25 and above)

This section describes a simple method for non-framework hosted applications to initialize the mPaaS framework.

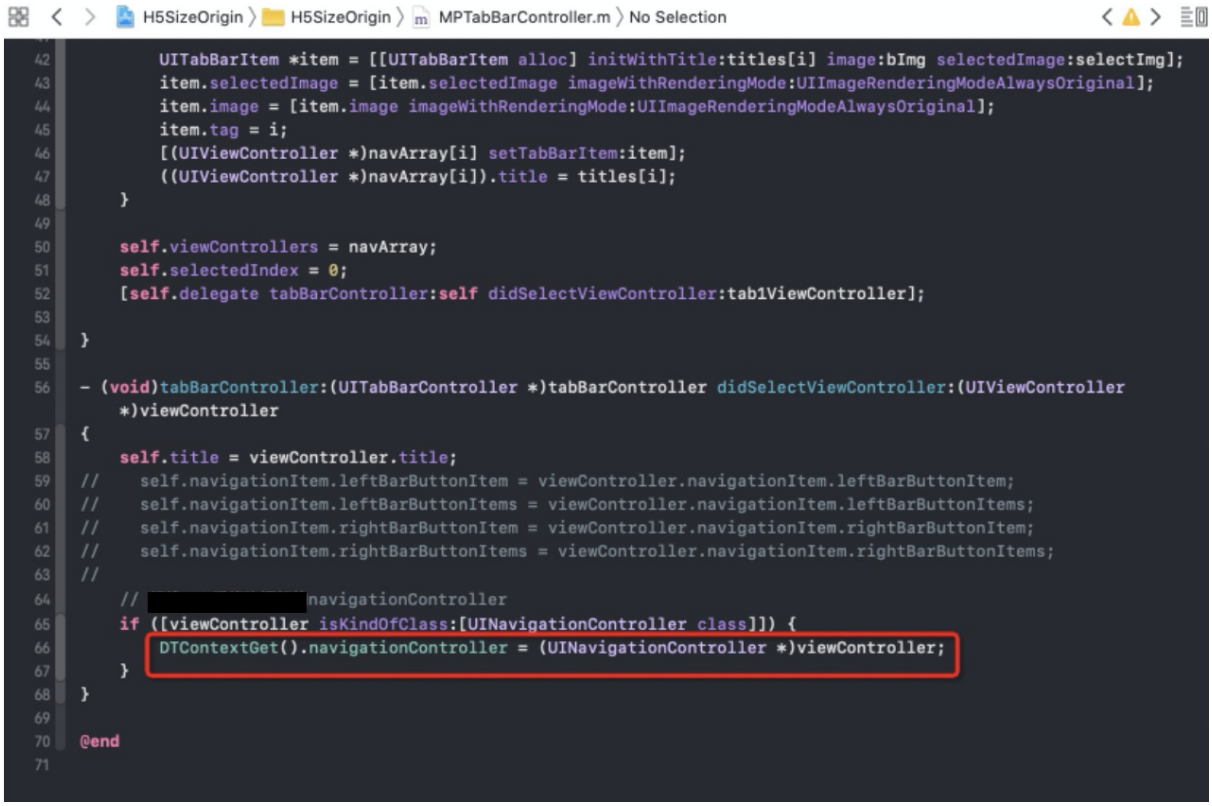
- Just call the following method after creating the application's `window` and `navigationController`. There is no need to create a `bootloader`, hide the frame `window`, etc.



- Support not inheriting `DFNavigationController`.



- If the App has multiple navigation bars and needs to open different offline packages in different navigation bars, you need to reset the container's navigation bar after switching the navigation bar.



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.

MPTinyAppDemo_pod 〉 MPTinyAppDemo_pod 〉 main.m 〉 No Selection

```
//
//  main.m
//  MPTinyAppDemo_pod
//
//  Created by yangwei on 2019/3/27.
//  Copyright © 2019 yangwei. All rights reserved.
//

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

int main(int argc, char * argv[]) {
    [MPAnalysisHelper enableCrashReporterService]; // USE MPAAS CRASH REPORTER
    @autoreleasepool {
        return UIApplicationMain(argc, argv, nil, NSStringFromClass([AppDelegate class]));
//        return UIApplicationMain(argc, argv, @"DFApplication", @"DFClientDelegate"); // NOW USE MPAAS FRAMEWORK
    }
}
```

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.

MPTinyAppDemp_plugin 〉 MPTinyAppDemp_plugin 〉 Sources 〉 AppDelegate.m 〉 -application:didFinishLaunchingWithOptions:

```
@implementation AppDelegate


+ (AppDelegate *)sharedInstance
{
    return [UIApplication sharedApplication].delegate;
}

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

    self.window = [[UIWindow alloc]initWithFrame:[UIScreen mainScreen].bounds];
    self.window.rootViewController = [[DFNavigationController alloc]
        initWithRootViewController:[[DemoHomeViewController alloc]init]];
    self.navigationController = self.window.rootViewController;
    [self.window makeKeyAndVisible];
    self.window.backgroundColor = [UIColor whiteColor];

    [[DTFrameworkInterface sharedInstance] manualInitMpaasFrameworkWithApplication:application
        launchOptions:launchOptions];

```

> **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.h 〉 No Selection

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

#import <APMobileFramework/APMobileFramework.h>

NS_ASSUME_NONNULL_BEGIN

@interface MPBootLoaderImpl : DTBootLoader

@end

NS_ASSUME_NONNULL_END

```

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>

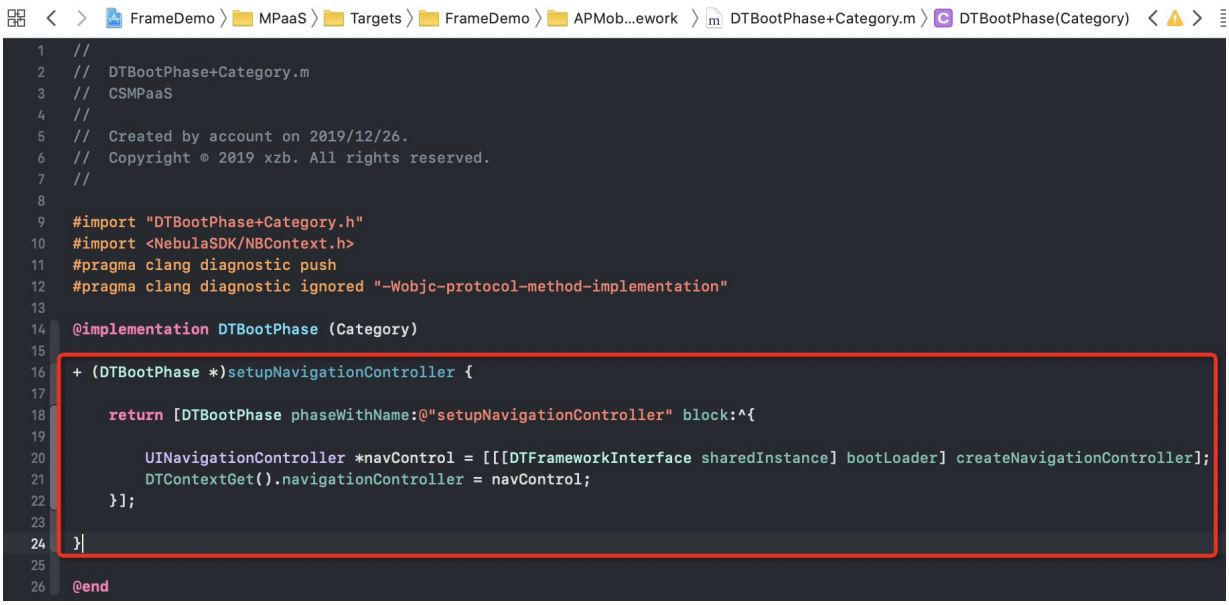
NS_ASSUME_NONNULL_BEGIN

@interface DTBootPhase (Category)

@end

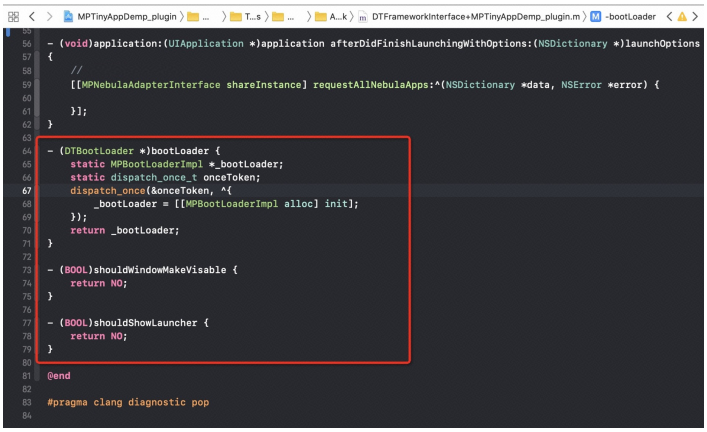
NS_ASSUME_NONNULL_END

```



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.



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 `example.com`.

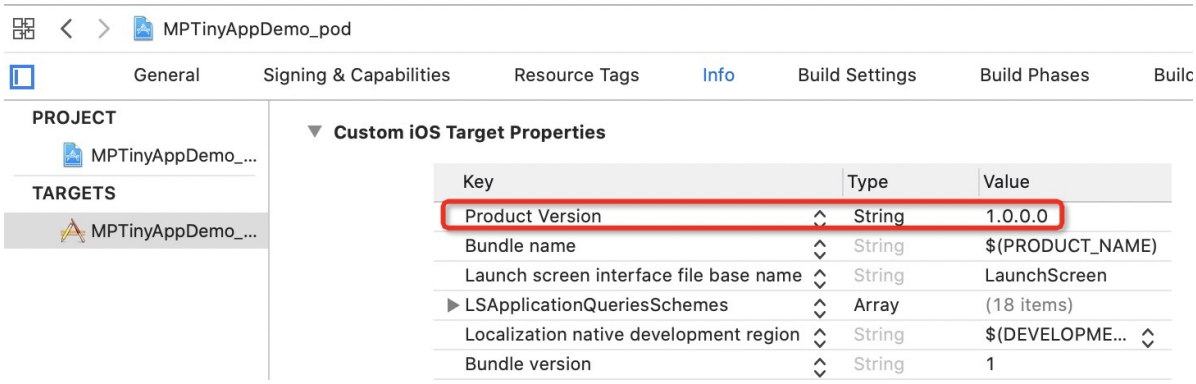
> **Note**
> Be sure to use a domain name that you registered yourself.

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** > **Mini Program Official Package Management** 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];
```

1.4.3.2. Advanced guide

1.4.3.2.1. Real-device preview and debugging of iOS Mini Program

This topic introduces the steps to implement real-device preview and debugging of mini programs in iOS clients.

> **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
```

1.4.3.2.2. 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.

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 ac
tion:@selector(onClickClose)];
    }
 }

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

1.4.3.2.3. Custom bidirectional 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:

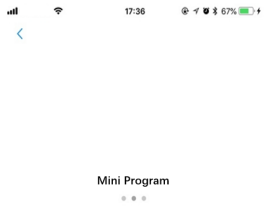
Parameter	Description
handlerName	The name of the event monitored by the Mini Program.
data	The parameters passed by the client to the Mini Program.
callback	The call back processing block after the Mini Program executes the event.

Unregister custom event

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

1.4.3.2.4. Customize startup loading page of iOS Mini Program

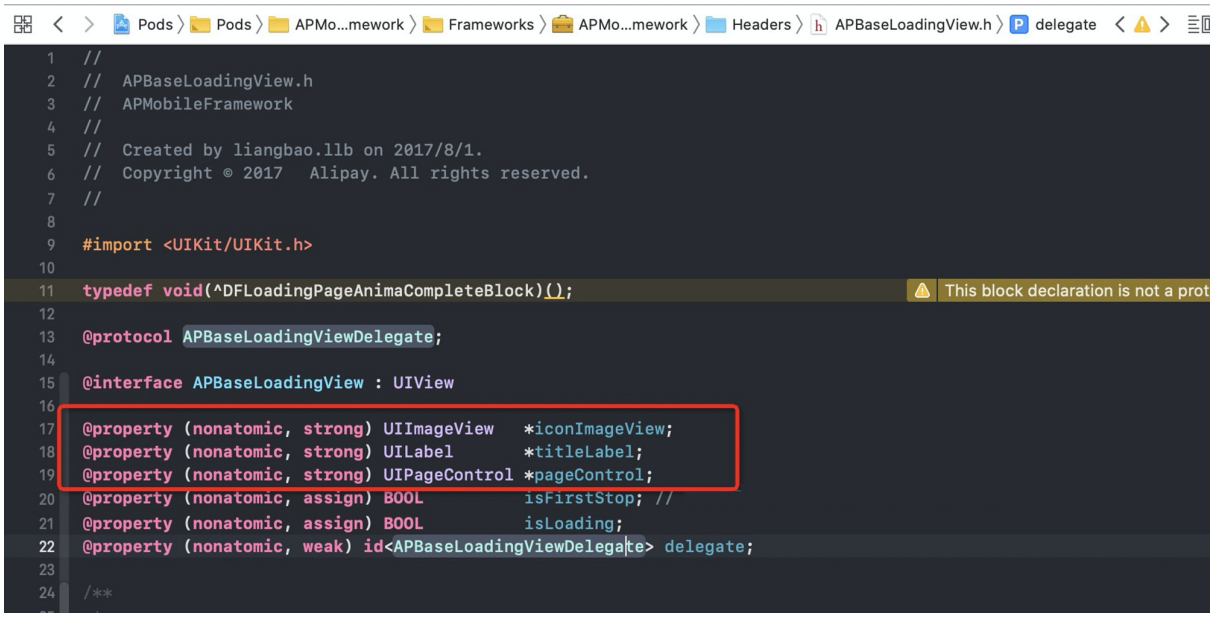
When the Mini Program is started, if the Mini Program has not been downloaded to the device, the Mini Program container will launch the loading page (as shown in the image below) to prompt the user to wait. After the Mini Program is installed on the device, the loading page will be closed and the page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

1.4.3.2.5. 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 similar to the following will appear.

"Network can't connect (-1009)"

This topic introduces how to customize the error 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.

   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>
```

1.4.3.2.6. iOS Mini Program supports custom View

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"} callback:^(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"
  ...
/>
```

> **Important**
> - 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
```

1.4.3.2.7. 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.

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

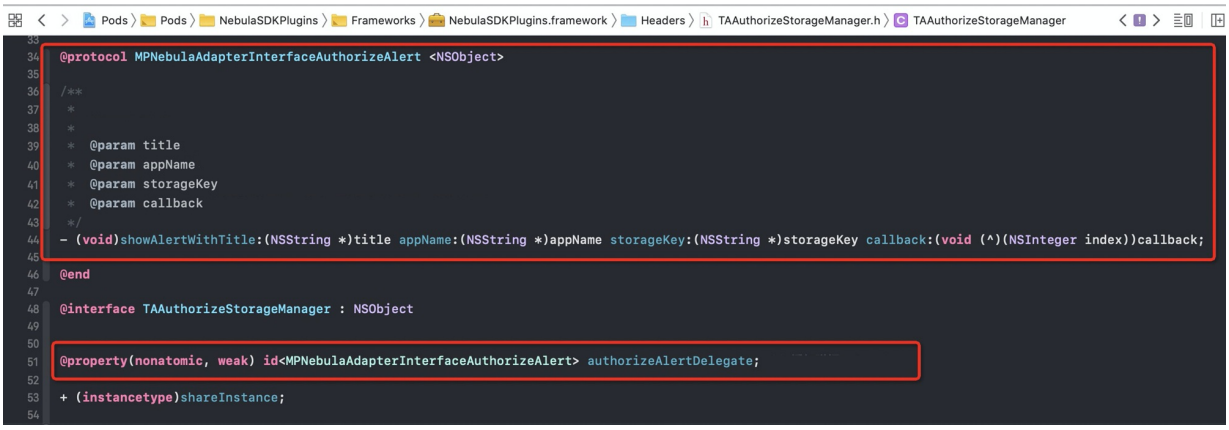
You can get the permission dictionary that the current Mini Program has requested:

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

```
46  @end
47
48  @interface TAAuthorizeStorageManager : NSObject
49
50
51  @property(nonatomic, weak) id<MPNebulaAdapterInterfaceAuthorizeAlert> authorizeAlertDelegate; //
52
53  + (instancetype)shareInstance;
54
55  /**
56   *
57   *
58   *
59   */
60  - (NSMutableDictionary *)authStatusDic4AppId:(NSString *)appid;
61
62  /**
```

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 ind
ex))callback
{
    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);
        }
    }

}
```

1.4.3.2.8. 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 Getting started.

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"] ofType:nil];
NSString *appPackagePath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPresetApps.bundle"] ofType:nil];
NSString *pluginsJsapisPath = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"DemoCustomPlugins.bundle/Poseidon-
UserDefine-Extra-Config.plist"] ofType:nil];
[MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:presetApplistPath customPresetAppPackagePath:appPackagePath
customPluginsJsapisPath: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.

   - 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"];
```

1.4.3.2.9. Add extended information for iOS Mini Program

This topic describes how to add extended information in the mini program console and IDE, and how to obtain extended information in iOS code.

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.

When using the mini program releasing function in the IDE, click **Settings** to enter the **Configure Mini Program** tab page to add extended information.

Obtain extension information

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

1.4.3.2.10. 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 Getting started.

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 included 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 mini program through the startup parameters

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

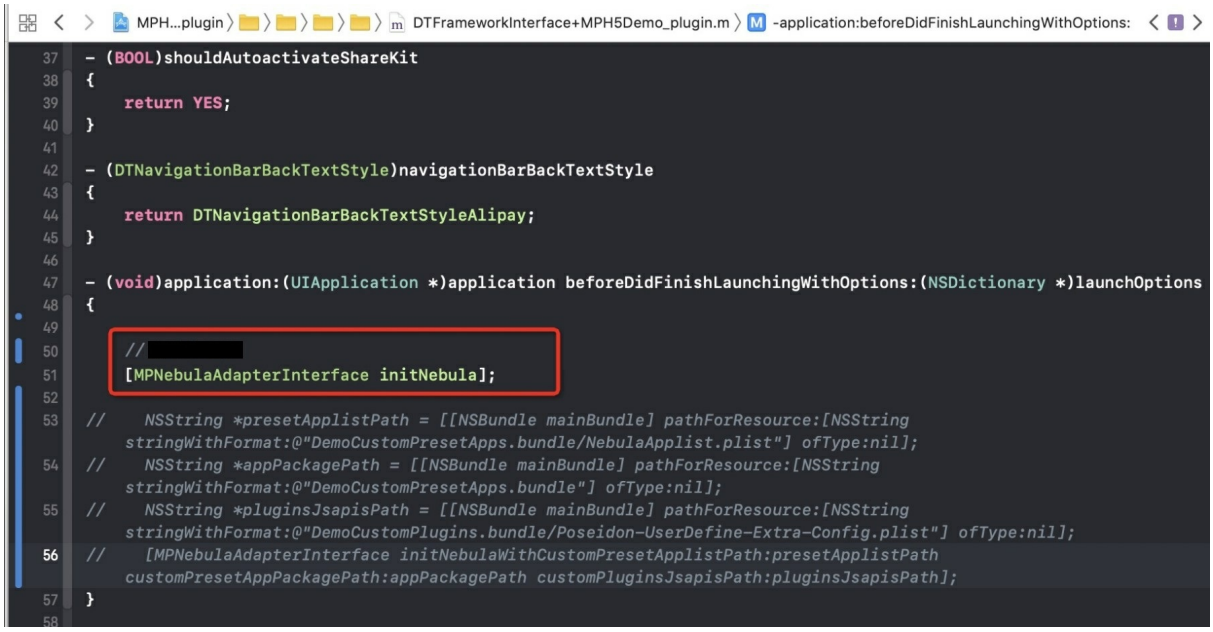
```
NSDictionary * dict = @{
    @"page" : @"/pages/demo3/index"
};
[MPNebulaAdapterInterface startTinyAppWithId:@"1234567891234567" params:dict];
```

1.4.3.3. 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.

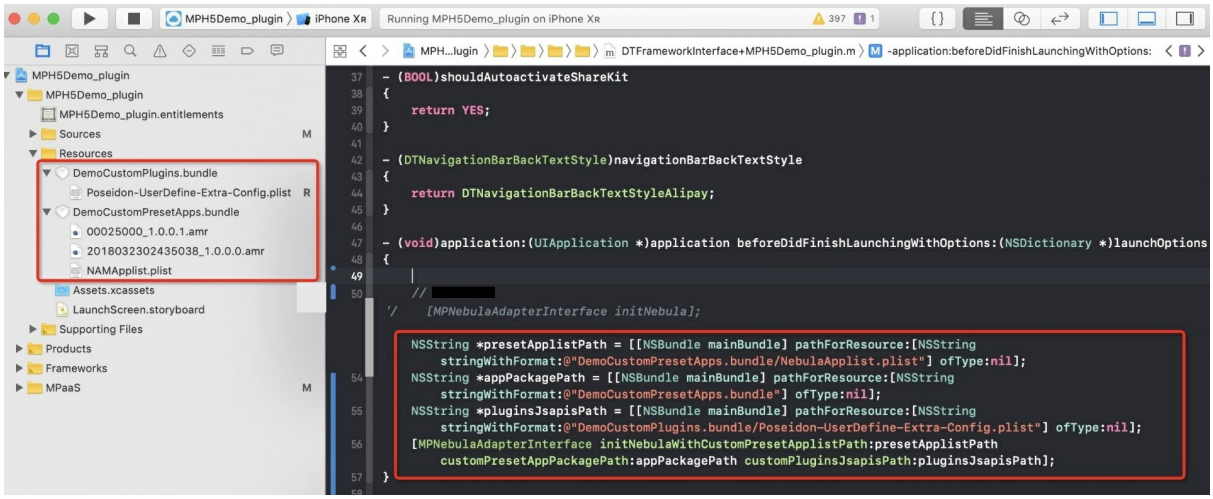
Initialization configuration

- 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`.



1.5. Develop Mini Program

1.5.1. Quick start

The steps to develop a Mini Program are as follows:

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

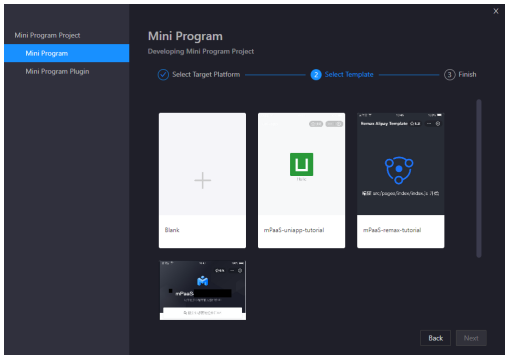
Download IDE

Download the Mini Program developer tool (IDE):

- Windows 7/8/10 (64-bit)
- macOS (x64, for most models)
- macOS (arm64, applicable to models with M1/M2 and other chips)

Create a Mini Program

1. After downloading and installing the Mini Program IDE, open the IDE, click **Mini Program** from the list on the left, and click **+** on the right to open the creation page, click **mPaaS** and **Select Templet**.



2. Set **Project Name** and **Project Path** in the project creation page, then click **Finish** to create the Mini Program.

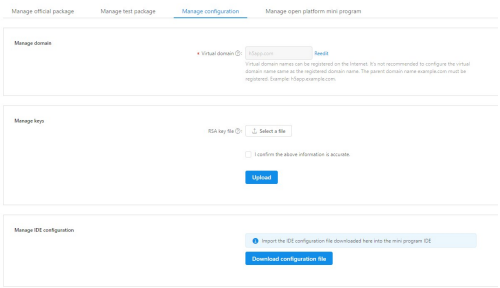
> **Important**
>
> - We recommend that you use a node version earlier than v18 for the uniapp template. Because uniapp templates may have compatibility issues with newer versions of node, it is recommended to use earlier node versions to avoid potential issues.
> - The steps of dependency installation wit YARN are as follows:
>
> a. Open the command line tool and go to the root directory where the project is located.
>
> b. Run the `yarn install` command.
>
> After you perform the preceding operations, the required dependencies are installed and the development environment is started. Then, you can develop and debug the uniapp project.

Download the configuration file

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

i. Log on to the mPaaS console. In the left-side navigation pane, choose **Mini Program** > **Mini Program Release** > **Manage Configuration**. On the page that appears, click **Manage IDE Configuration** and click **Download Configuration File** to download the IDE configuration file.

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



ii. After you click **Download Configuration File**, the **Download Configuration File** dialog box appears. You must enter a password in the **Dynamic Password** field. The password is going to be used to log on to the IDE.

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

Login to the Mini Program IDE

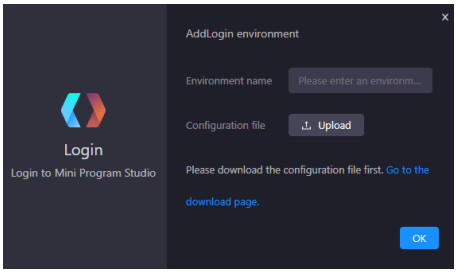
You can login to the Mini Program IDE in two ways:

- Dynamic password logon
- Aliyun AccessKey logon method

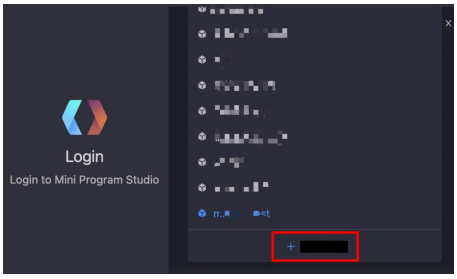
Dynamic password logon method

i. In the Mini Program IDE, click **Log On** in the upper-right corner.

ii. If there is no logon environment has been created, the Add Environment window appears. If a logon environment has been created, the Logon window appears.
   - If you are **creating a logon 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.

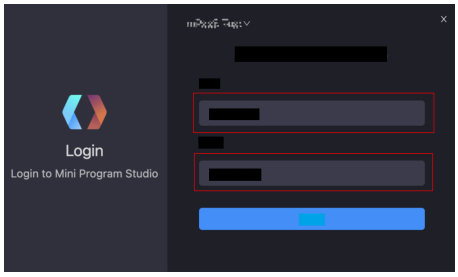


   - If you **have created a logon environment**, click the environment selection menu at the top of the window, choose **+ Add Environment** at the bottom of the menu, enter an environment name, and then upload the **Mini Program IDE configuration file** (`config.json` file) downloaded from the mPaaS console.



iii. Click **OK** to create a new logon environment.

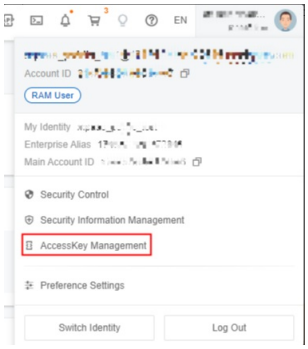
iv. After successfully adding the logon environment, in the log on window, enter the account and password to log on.
   - An account is the username used to log on to the Aliyun console.
   - The password is the **dynamic password** that you set when you download the configuration file.



Aliyun AccessKey logon method

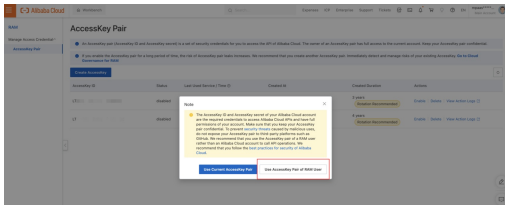
> **Important**
> If you use the Aliyun AccessKey, you must upgrade mPaaS Mini Program IDE to version 2.9 or later.

i. Move the pointer over the profile image in the upper-right corner of the Alibaba Cloud console and click **AccessKey Management** to enable Aliyun AccessKey.

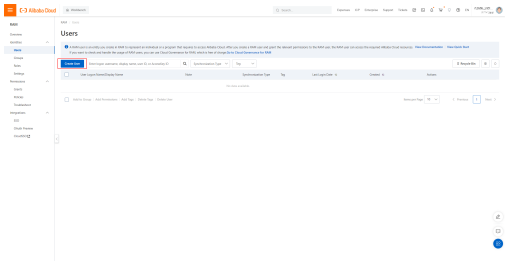


> **Important**
> - You can use the AccessKey pair of an Alibaba Cloud account or a RAM user to call API operations.
> - If you use the AccessKey pair of an Alibaba Cloud account to call API operations, you must create an AccessKey pair.
> - We recommend that you use the AccessKey pair of RAM user instead of an Alibaba Cloud account to call API operations because RAM user permissions are controllable and easy to manage.

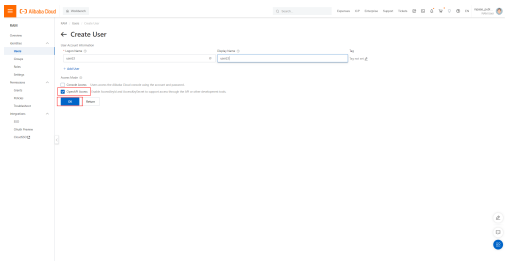
ii. Click **Use AccessKey Pair of RAM user**.



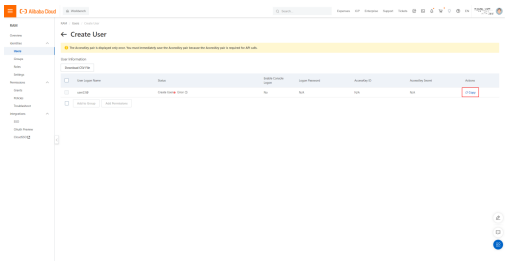
iii. Click **Create User**. The **Create User** page appears.



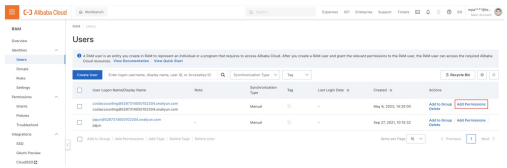
iv. Set **Logon Name** and **Display Name**, select **OpenAPI Access**, and then click **OK**.



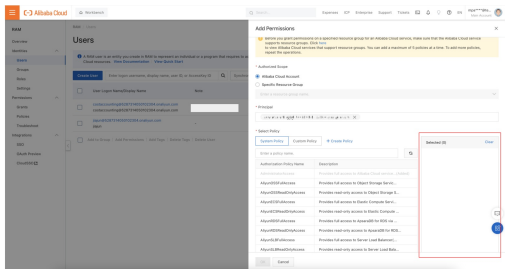
v. Copy and keep the generated AccessKey ID and AccessKey secret. If the AccessKey ID and AccessKey secret are not copied, you can generate them again.



vi. Click **Add Permissions** in the Actions column. The **Add Permissions** page appears.

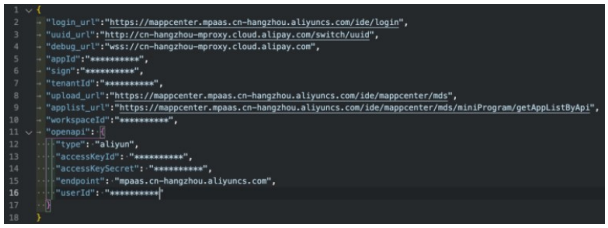


vii. In the Select Permissions search box, enter **MpaaSFullAccess**, and select a **AliyunMpaaSFullAccess**, then click **OK**. Add mPaaS permissions to the created user.

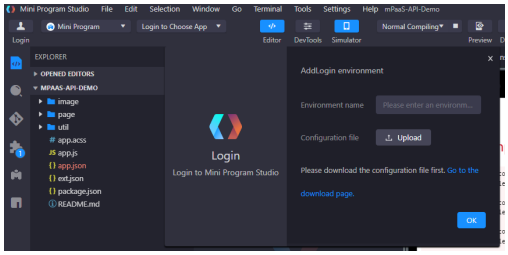


viii. Open the configuration file that you downloaded in the editor, enter the following content, and save the file:

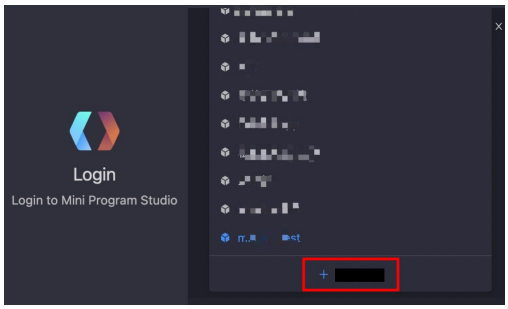
```
{
 // The content of the original configuration file,
  "openapi": {
    "type": "aliyun",
    "accessKeyId": "AccessKey ID generated in step 5",
    "accessKeySecret": "The AccessKey secret generated in step 5",
    "endpoint": "mpaas.cn-hangzhou.aliyuncs.com",
    "userId": "Username entered in Step 5"
  }
}
```



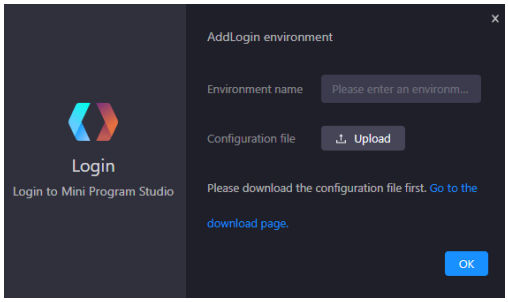
ix. In the mPaaS mini program IDE, click **Login** to open the **logon** page.



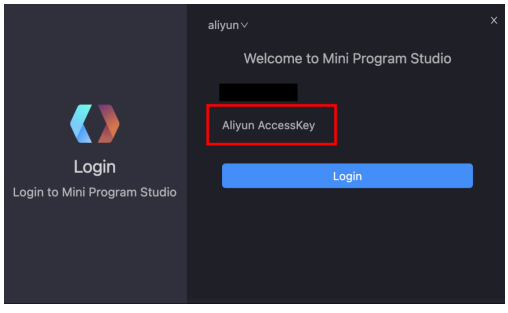
x. Click **+ Add Environment** to open the **Add Logon Environment** page.



xi. Enter an environment name in the **Environment Name** field, upload the **configuration file** configured in Step 8, and then click **OK**.

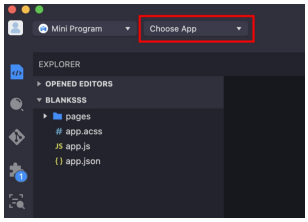


It shows **Aliyun AccessKey** in the Logon dialog box, then click **Login**.



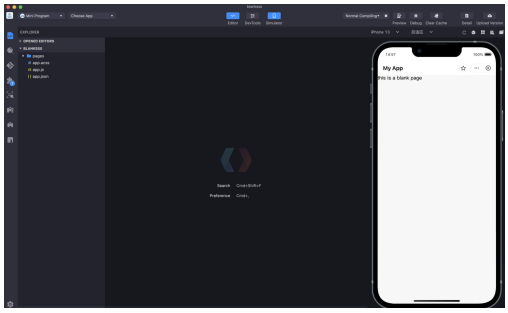
Select the Associated Mini Program

After you log on to the IDE, click **Choose App** in the upper-left corner and select the Mini Program that you created in the console from the drop-down list.



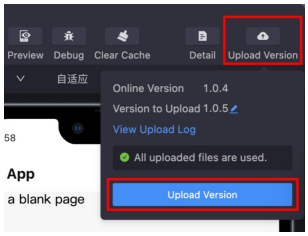
Edit the code

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



Upload Mini Program

After you edit the code, click **Upload** in the upper-right corner of the IDE to upload the Mini Program to the mPaaS console.



Release the Mini Program

Log on to the mPaaS console. In the left-side navigation pane, choose **Mini Program** > **Release Mii Program**. For more information, see Release a Mini Program package.

1.5.2. Advanced guide

1.5.2.1. 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.

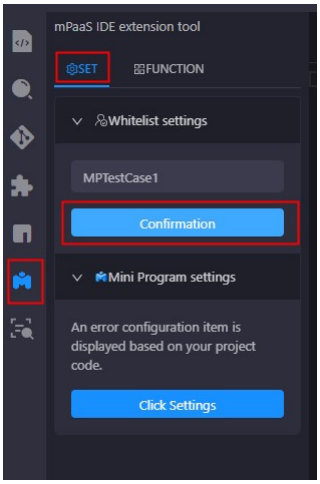
- Android
- iOS

1.5.2.2. 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.



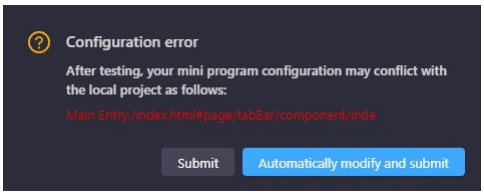
Settings

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.

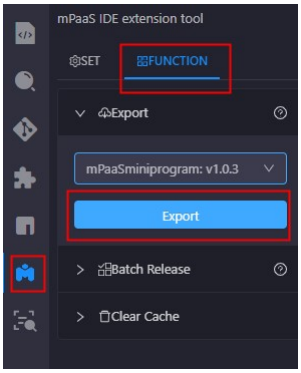


Features

Export

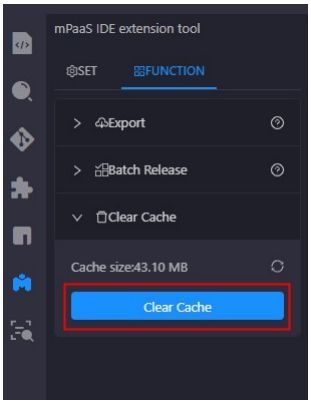
Click **Function** > **Export** in the toolbox, select the Mini Program version and click **Export**.

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.

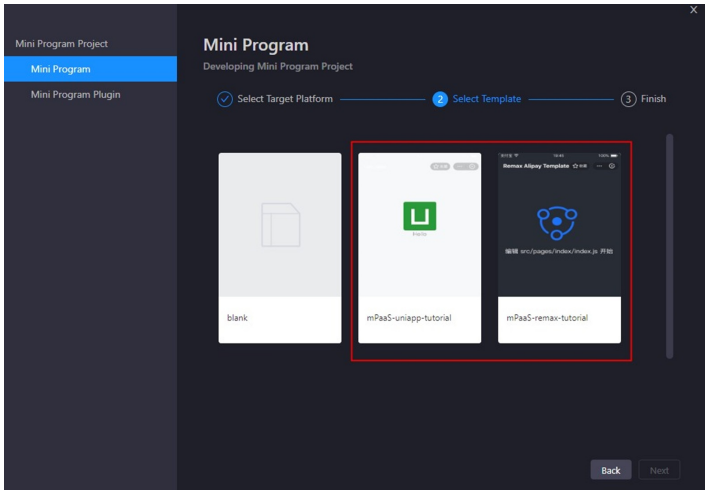


Multi-terminal development

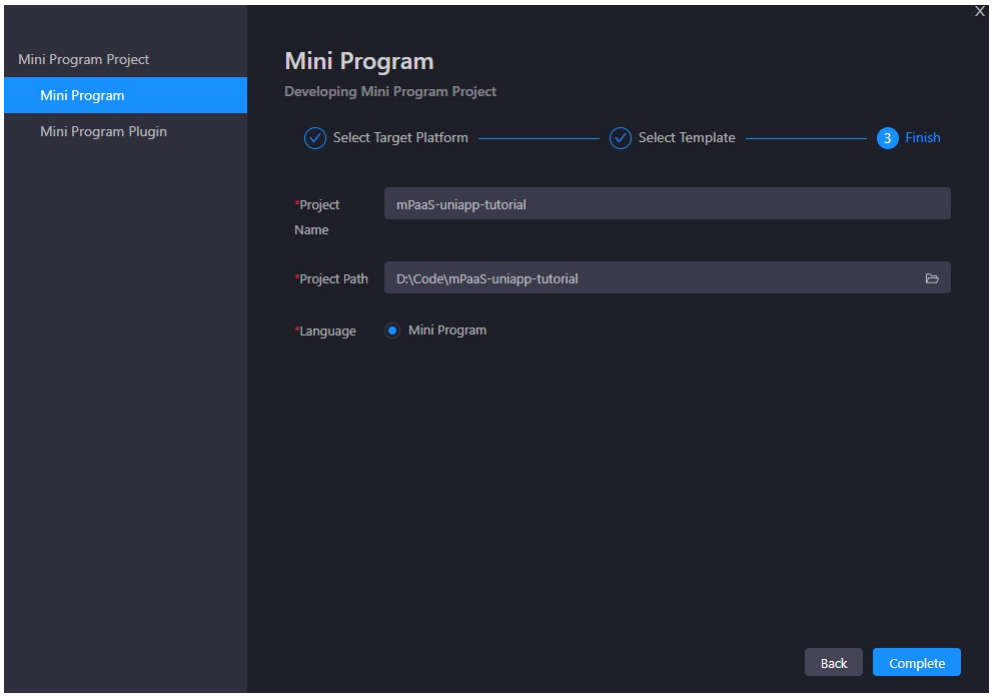
The Mini Program developed in the mPaaS Mini Program IDE can not only be put into the App developed using the mPaaS framework, but also can be used to evoke the WeChat Mini Program IDE for joint debugging through the mPaaS Mini Program IDE, or quickly build into an HTML5 application and realize real-device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

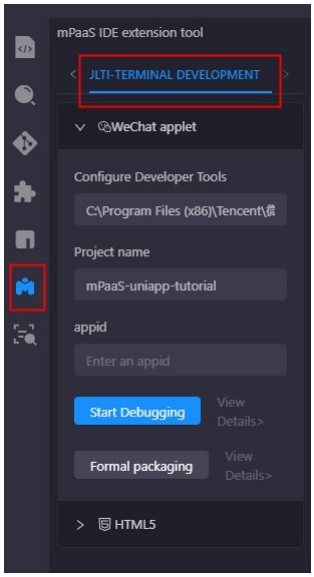


2. Enter the project name and project path, then click **Complete**.

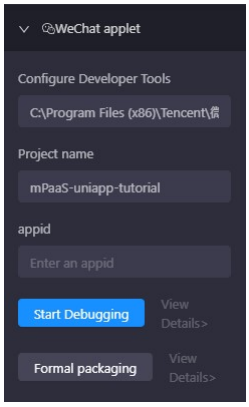


3. Download and install the WeChat Mini Program IDE. Log in and open the service port.
4. Click the mPaaS toolbox (

) 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.



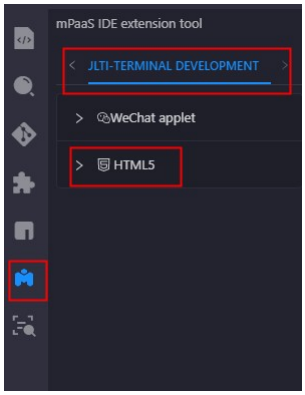
7. Real-device preview.

   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.

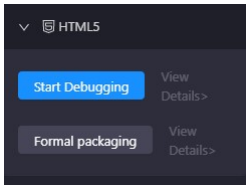
HTML5

1. Start the Mini Program IDE and select a project that you want to develop multi-terminal.
2. Click the mPaaS toolbox (

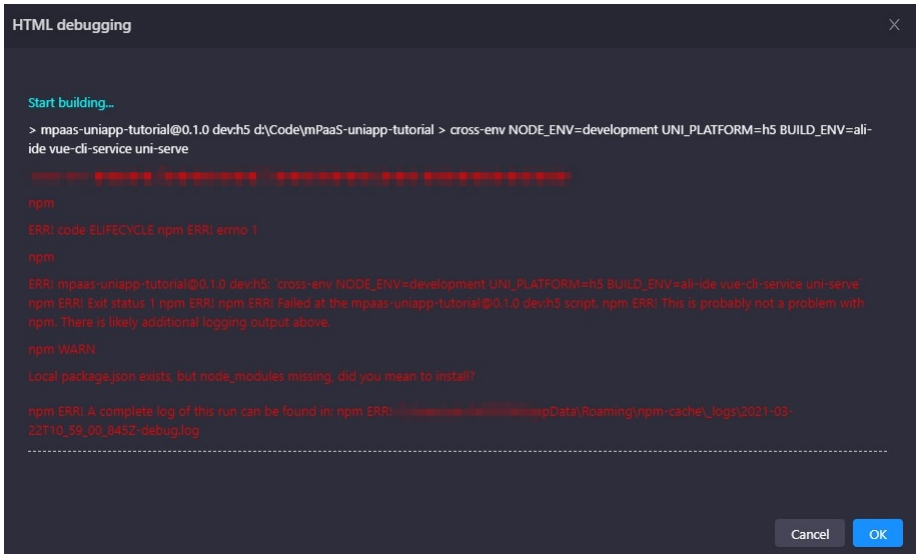
) on the left side of the IDE interface, and select the **Multi-terminal development** tab on the right side.



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.



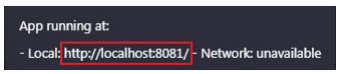
5. Preview.

   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.

     > **Note**
     > 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.



1.5.2.3. 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.

1.5.2.4. Performance listener for mini programs

Android

API

```
Mriver.setProxy(PrepareNotifyProxy.class, new PrepareNotifyProxy() {
            @Override
            public void notify(String s, PrepareStatus prepareStatus) {

            }

            @Override
            public void apmEvent(final String event, final String param1, final String param2, final String param3, final String param4) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        if (logs == null) {
                            logs = new StringBuilder();
                        }
                        logs.append(s).append(" ").append(s1).append(" ").append(s2).append(" ").append(s3).append("
").append(s4).append("\n");
                    }
                });

            }
});
```

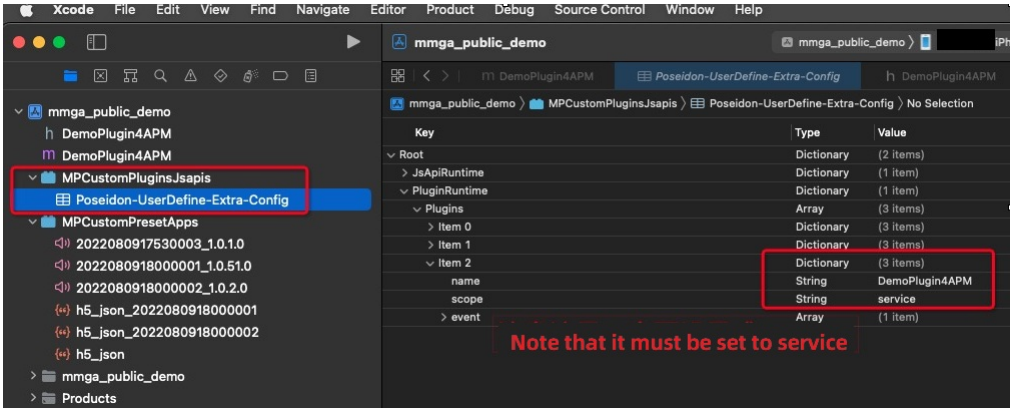
The events are as follows:

```
MINI_APP_PREPARE parameter errc! =1 indicates that the application is opened abnormally
MINI_PAGE_ABNORMAL white screen
The MINI_APP_REQUEST parameter step=fail indicates that the application pull package is abnormal
Restricted MINI_AL_NETWORK_PERMISSON_ERROR page access limited
MINI_AL_JSAPI_RESULT_ERROR jsapiName=httpRequest or request indicates that the request is abnormal, and other indicates that the JSAPI is
abnormal
Abnormal MINI_CUSTOM_JS_ERROR JS exception
MINI_AL_NETWORK_PERFORMANCE_ERROR resource request exception
The startup time of MiniAppStart. startCost indicates the time, in milliseconds
```

iOS

API

The method is to develop a plugin to intercept the corresponding system events. The Plist configuration is as follows:




```
// Plist initialization code:
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
//    [MPNebulaAdapterInterface initNebula];
    NSString* h5Json = [[NSBundle mainBundle] pathForResource:[NSString
stringWithFormat:@"MPCustomPresetApps.bundle/h5_json.json"]ofType:nil];
    NSString* amrBundle = [[NSBundle mainBundle] pathForResource:[NSString stringWithFormat:@"MPCustomPresetApps.bundle"] ofType:nil];
    NSString* jsPath = [NSBundle.mainBundle.bundlePath stringByAppendingPathComponent:@"MPCustomPluginsJsapis.bundle/Poseidon-UserDefine-
Extra-Config.plist"];
    [MPNebulaAdapterInterface initNebulaWithCustomPresetApplistPath:h5Json
                                        customPresetAppPackagePath:amrBundle
                                           customPluginsJsapisPath:jsPath];
}
```

The code is as follows:

```
// .h file
#import <AriverApp/RVAPluginBase.h>

NS_ASSUME_NONNULL_BEGIN

@interface DemoPlugin4APM : RVAPluginBase

@end

NS_ASSUME_NONNULL_END
```

```
// .m file
#import "DemoPlugin4APM.h"
#import <NebulaPoseidon/PSDMonitorEvent.h>

@implementation DemoPlugin4APM


- (void)pluginDidLoad
{
    self.scope = kPSDScope_Service;
    [self.target addEventListener:kEvent_Monitor_Log_Before withListener:self useCapture:NO];
    [super pluginDidLoad];
}

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

    if ([kEvent_Monitor_Log_Before isEqualToString:event.eventType]){

        PSDMonitorEvent *mEvent = (PSDMonitorEvent *)event;
        NSArray *params = [mEvent.params isKindOfClass:[NSArray class]]? mEvent.params : @[];
        NSString *bizType = [NSString stringWithFormat:@"%@", mEvent.bizType];
        NSString *seedId = [NSString stringWithFormat:@"%@", mEvent.seedId];
        if ([params count] != 4 || ![bizType length]) {
            return;
        }

        NSString *aplogstr = [NSString
stringWithFormat:@"%@\nbizType=%@,param1=%@,param2=%@,param4=%@",params[2],bizType,params[0],params[1],params[3]];
        NSLog(@"seed seedId:[%@]\nlog:[%@]", seedId, aplogstr);
        // Open Exception: Reproduce the path. Open the demo and click Open Exception H5_APP_PREPARE
        if ([seedId isEqualToString:@"H5_APP_PREPARE"]) {
            NSString *currentStep = [self mp_valueFromLogStr:params[2] key:@"step"];
            // If the step parameter is noexistForce, an open exception occurs
            if ([currentStep isEqualToString:@"noexistForce"]) {
                NSLog(@"seedId:[%@]\nlog:[%@]", seedId, aplogstr);
            }
        }
        // js exception: reproduce the path, open the demo and click "APM mini program-> js error"
        else if ([seedId isEqualToString:@"H5_CUSTOM_ERROR"]) {
            NSLog(@"seedId:[%@]\nlog:[%@]", seedId, aplogstr);
        }
        // White screen: Reproduce the path. After the network is disconnected (in flight mode, WiFi is disconnected), open the demo and click
"Open Exception H5_APP_PREPARE"
        else if ([seedId isEqualToString:@"H5_PAGE_ABNORMAL"]) {
            NSLog(@"Capture white screen:[%@]\nlog:[%@]", seedId, aplogstr);
        }
        // The pull package is abnormal. No path is available. The pull package can be reproduced only if the pull package interface is abnorma
l
        else if ([seedId isEqualToString:@"H5_APP_REQUEST"]) {
            NSString *currentStep = [self mp_valueFromLogStr:params[2] key:@"step"];
            NSLog(@"Pull package:[%@]\nlog[%@]:[%@]", seedId, currentStep, aplogstr);
            if ([currentStep containsString:@"fail"]) {
                NSLog(@"Package pulling exception:[%@]\nlog[%@]:[%@]", seedId, currentStep, aplogstr);
            }
        }
        // Page access limited H5 _AL_NETWORK_PERMISSON_ERROR
        // Reproduce path, open demo and click "APM mini program-> page access limited"
        else if ([seedId isEqualToString:@"H5_AL_PAGE_UNAUTHORIZED"] || [seedId isEqualToString:@"H5_AL_NETWORK_PERMISSON_ERROR"]) {
            NSLog(@"seedId:[%@]\nlog:[%@]", seedId, aplogstr);
        }
        // Request exception / JSAPI exception H5 _AL_JSAPI_RESULT_ERROR
        // reproduce the path, open demo and click "APM applet-> js api error"
        else if ([seedId isEqualToString:@"H5_AL_JSAPI_RESULT_ERROR"]) {
            NSLog(@"seedId:[%@]\nlog:[%@]", seedId, aplogstr);
        }
        // Resource request exception H5_AL_NETWORK_PERFORMANCE_ERROR
        // reproduce the path, network disconnection-> open demo and click "jump mini program-> point refresh"
        else if ([seedId isEqualToString:@"H5_AL_NETWORK_PERFORMANCE_ERROR"]) {
            NSLog(@"seedId:[%@]\nlog:[%@]", seedId, aplogstr);
        }
    }

    // The startup duration
    // Calculation rule: the amount of time that is consumed from AppStart to PageLoad. Unit: seconds.
    // The business value should only be calculated for the first callback, that is, the home page is loaded (the difference can be seen from
the log of h5WebVC.url below)
    if ([event.eventType isEqualToString:kEvent_Session_Create]) {
        // Remember the timestamp:
        CFTimeInterval tm=CACurrentMediaTime(); // CACurrentMediaTime() is based on the built-in clock, which can be measured more accurately a
nd atomically, and will not change due to external time changes (such as time zone changes, daylight saving time, second mutation, etc.), but
it is related to the uptime of the system, and the CACurrentMediaTime() will be reset after the system restarts.  CACurrentMediaTime() is
often used to test the efficiency of code.
        self.appStartTimestamp = tm;
```

```
        NSLog(@"kEvent_Session_Create: AppStart [%f]", tm);
    }

    if ([event.eventType isEqualToString:kEvent_Page_Load_Complete]) {
        PSDEvent *oldEvent = (PSDEvent *)event;
        H5WebViewController *h5WebVC = (H5WebViewController *)[oldEvent.context currentViewController];
        NSString* currentUrl = [h5WebVC.url absoluteString];
        NSLog(@"kEvent_Session_Create: page[%@]", currentUrl);
        if ([currentUrl containsString:@"#"]) { //
            // Timestamp:
            CFTimeInterval tm = CACurrentMediaTime();
            NSLog(@"kEvent_Session_Create: PageLoad [%f]", tm);
            CFTimeInterval appStartTime = tm - self.appStartTimestamp;
            NSLog(@"kEvent_Session_Create: AppStart-PageLoad=start time [%f]", appStartTime);
        }
    }
}

@end
```

Android and iOS event comparison list

Events	Android	iOS
Open Exception	MINI_APP_PREPARE	H5_APP_PREPARE
White screen	MINI_PAGE_ABNORMAL	H5_PAGE_ABNORMAL
Package pulling exception	MINI_APP_REQUEST	H5_APP_REQUEST
Limited page access	MINI_AL_NETWORK_PERMISSON_ERROR	H5_AL_NETWORK_PERMISSON_ERROR / H5_AL_PAGE_UNAUTHORIZED
Request exception /JSAPI exception	MINI_AL_JSAPI_RESULT_ERROR	H5_AL_JSAPI_RESULT_ERROR
js exception	MINI_CUSTOM_JS_ERROR	H5_CUSTOM_ERROR
Resource request exception	MINI_AL_NETWORK_PERFORMANCE_ERROR	H5_AL_NETWORK_PERFORMANCE_ERROR
Startup time	MiniAppStart	

Mini Program

my.onError(Function listener)

Overview

my.onError listens for applet error events.

Input parameters

Function listener

Parameter

View Example

Attribute	Category	Compatibility	Description
error	String	-	The description of the exception, which is generally the message field of the Error object.
stack	String	Base database: 2.7.4	The stack of exceptions, which is generally the stack field of the Error object.

Sample code

my.onError(Function listener)

```
Page({
 onLoad() {
 my.onError(this.errorHandler);
 },
 errorHandler(error, stack) {
 console.log('onError error', error);
 console.log('onError stack', stack);
 }
})
```

> **Note**
> - If you use my.onError to listen to errors, the onError method in app.js also listens to errors.
> - Use my.onError to listen for page errors. If the listener is enabled on multiple pages but not disabled, multiple listener events are triggered when the page errors are reported. We recommend that you call my.offError to disable the listener when the page is closed.

my.onUnhandledRejection(Function listener)

Overview

my.onUnhandledRejection listens for unhandled `Promise` rejection events.

Input parameters

Function listener

The callback function for the unhandled Promise reject event.

Parameter

Object res

View Example

Attribute	Category	Description
reason	any	The reason for the rejection. The received value of reject(), which is generally an Error object.
promise	Promise	The Promise object that was rejected. Note: Only supported on iOS

Sample code

my.onUnhandledRejection(Function listener)

```
Page({
 onLoad() {
 my.onUnhandledRejection(this.unhandledRejectionHandler);
 },
 unhandledRejectionHandler(res) {
 console.log('onUnhandledRejection reason', res.reason);
 console.log('onUnhandledRejection promise', res.promise);
 }
})
```

> **Note**
> - If the `unhandledrejection` event of `Promise` continues to be triggered within the callback function of my.onUnhandledRejection, the `unhandledrejection` event may be triggered cyclically. Please avoid it.
> - All `unhandledRejection` can be captured by this listener, but only `Error` types will trigger an alarm in the applet background.

1.5.2.5. Performance optimization suggestions

How it works

Different from traditional H5 applications, the mini program running architecture is divided into two parts: webview and worker. The webview is responsible for rendering, and the worker is responsible for storing data and executing business logic.

1. The communication between the webview and the worker is asynchronous. This means that when we call setData, our data is not rendered immediately, but needs to be transferred asynchronously from the worker to the webview.
2. When Data Transmission Service, it needs to be serialized as a string and then transmitted in evaluateJavascript way. The data size will affect the performance.

Optimize the first screen

The first screen has multiple definitions. Here, the first screen refers to the first meaningful rendering from a business perspective. For example, for a list page, the first screen is the first rendered content of the list.

Control mini program resource package size

When a user accesses an mini program, the client first downloads the mini program resource package from the Alibaba Cloud Content Delivery Network, so the size of the resource package affects the startup performance of the mini program.

Optimization suggestions

- Delete useless image resources in a timely manner, because all image resources are packaged by default.
- Control the image size and avoid using large images. We recommend that you upload large images from Alibaba Cloud Content Delivery Network channels.
- Clean up useless code in a timely manner.

Advance data requests to onLoad

- When the mini program runs, the onLoad lifecycle function of the page is triggered first, and then the page initial data (Page data) is passed from the worker to the webview for an initial rendering.
- The initial rendering of the page is complete and a notification is sent from the webview to the worker, triggering the onReady lifecycle function.

Some mini programs send requests in onReady, which delays the rendering of the first screen. We recommend that you advance data requests to onLoad.

Control the number of one-time rendering nodes on the first screen

After the service request is returned, setData is usually called to trigger page re-rendering. The execution process is as follows:

1. Data is passed from the worker to the webview.
2. The virtual DOM is constructed on the webview based on the passed data, compared with the previous differences (starting from the root node), and then rendered.

When the worker communicates with the webview, the data needs to be serialized and then the webview needs to execute the evaluateJavascript. Therefore, if the data transmitted at one time is too large, the rendering performance of the first screen will be affected.

In addition, if there are too many construction nodes and the hierarchy nesting is too deep on the webview, for example, some small program list pages render more than 100 list items at one time, and each list item has nested content, but in fact the whole screen may only display less than 10, which will lead to a long difference comparison time. At the same time, due to the first screen rendering, many DOM will be constructed at one time, affecting the first screen rendering performance.

Optimization suggestions

- The amount of setData data should not be too large to avoid passing an excessively long list at a time.
- Do not construct too many nodes at one time on the first screen. The server may request to transfer a large amount of data at one time. Do not setData at one time. You can setData part of the data first, and then wait for a period of time (such as 400ms, which requires service adjustment) before calling $spliceData to pass the other Data Transmission Service.

Optimize the setData logic

Any page change will trigger setData, and multiple setData may trigger the page to re-render at the same time. The following four interfaces will trigger the webview page to re-render.

- **Page.prototype.setData**: Trigger the entire page to do difference comparison.
- **Page.prototype.$spliceData**: Optimized for long lists to avoid passing the entire list each time and triggering the entire page for difference comparison.
- **Component.prototype.setData**: Differences are compared only from the corresponding component node.
- **Component.prototype.$spliceData**: Optimized for long lists to avoid passing the entire list each time, only starting from the corresponding component node for difference comparison.

Optimization suggestions

- Avoid frequently triggering setData or $spliceData, regardless of page level or component level. In the cases we analyzed, some pages have countdown logic, but some countdown triggers too frequently (ms-level triggers).
- If you need to frequently trigger re-rendering, do not use setData and $spliceData at the page level, encapsulate this block into a custom component, and then use setData or $spliceData at the component level to trigger the re-rendering of the component.
- When a long list of data triggers rendering, use $spliceData to append the data multiple times without passing the entire list.
- We recommend that you encapsulate complex pages into custom components to reduce setData at the page level.

Optimization cases

Set data recommend the specified path:

```
this.setData({
 'array[0]': 1,
 'obj.x':2,
});
```

The following usage is not recommend (although this.data is copied, its attributes are directly changed):

```
const array = this.data.array.concat();
array[0] = 1;
const obj={...this.data.obj};
obj.x=2;
this.setData({array,obj});
```

It is not recommend to directly change this.data (violating the principle of immutable data):

```
this.data.array[0]=1;
this.data.obj.x=2;
this.setData(this.data)
```

Long lists use $spliceData:

```
this.$spliceData({ 'a.b': [1, 0, 5, 6] })
```

Usage notes

Sometimes the business logic is encapsulated in a component. When the component UI needs to be re-rendered, you only need to call setData inside the component. However, sometimes you need to trigger component re-rendering from the page. For example, the onPageScroll event is listened to on the page. When the event is triggered, you need to notify the corresponding component to re-render. In this case, the following action is taken:

```
// 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 mount a component to the corresponding page in the didMount of the component. You can call setData at the component level on the page to only trigger the re-rendering of the component.

Use the key parameter

Use keys in for to improve performance.

> **Important**
> The key cannot be set on a block.

Sample code:

```
<view a:for="{{array}}" key="{{item.id}}"></view>
<block a:for="{{array}}"><view key="{{item.id}}"></view></block>
```

1.6. Mini Program Development Server API

1.6.1. Overview and preparation

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>
```

Environment Variable Configuration

Configuer environment variable MPAAS_AK_ENV and MPAAS_SK_ENV

- Linux and macOS system configuration methods execute the following commands:

```
export MPAAS_AK_ENV=access_key_id
export MPAAS_SK_ENV=access_key_secret
```

> **Note**
> `access_key_id` is replaced with the prepared AccessKey ID, and `access_key_secret` is replaced with the AccessKey Secret.

- Windows system configuration method

i. Create a new environment variable, add environment variables **MPAAS_AK_ENV** and **MPAAS_SK_ENV**, and write the prepared AccessKey ID and AccessKey Secret.

ii. Restart Windows system.

Code sample

```
It is strongly recommended not to save the AccessKey ID and AccessKey Secret in the project code, otherwise the AccessKey may be leaked, threa
tening the security of all resources under your account. // This example uses saving the AccessKey ID and AccessKey Secret in environment vari
ables as an example. You can also save it to the configuration file according to business needs.import com.aliyuncs.DefaultAcsClient;
import com.aliyuncs.IAcsClient;
import com.aliyuncs.mpaas.model.v20201028.QueryMcubeVhostRequest;
import com.aliyuncs.mpaas.model.v20201028.QueryMcubeVhostResponse;
import com.aliyuncs.profile.DefaultProfile;

public class MpaasApiDemo {

    /**
     * The corresponding APP ID on the mPaaS console
     */
    private static final String APP_ID = "ALIPUB40DXXXXXXX";

    /**
     * The corresponding workspace id on the mPaaS console
     */
    private static final String WORKSPACE_ID = "default";

    /**
     * The corresponding tenant id on the mPaaS console
     */
    private static final String TENANT_ID = "XVXXXXXF";

    /**
     * Region ID, the default is cn-hangzhou
     */
    private static final String REGION_ID = "cn-hangzhou";

    /**
     * Product name
     */
    private static final String PRODUCT = "mpaas";

    /**
     * The endpoint to call
     */
    private static final String END_POINT = "mpaas.cn-hangzhou.aliyuncs.com";

    public static void main(String[] args) {
        // Alibaba Cloud account AccessKey has access rights to all APIs. It is recommended that you use RAM users for API access or daily
operation and maintenance.
        // It is strongly recommended not to save the AccessKey ID and AccessKey Secret in the project code, otherwise the AccessKey may be
leaked, threatening the security of all resources under your account.
        // This example uses saving the AccessKey ID and AccessKey Secret in environment variables as an example. You can also save it to the
configuration file according to business needs.
        String accessKeyId = System.getenv("MPAAS_AK_ENV");
        String accessKeySecret = System.getenv("MPAAS_SK_ENV");

        DefaultProfile.addEndpoint(REGION_ID, PRODUCT, END_POINT);
        DefaultProfile profile = DefaultProfile.getProfile(REGION_ID, accessKeyId, accessKeySecret);
        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 (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

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.

1.6.2. APIs 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":"LTAI****************",
            "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":"kis**********"
        },
        "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.
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":"111111111111111111"
            },
            {
                "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.
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.
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.
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%3D99702121500000000&nbsv%3D1.0.1.7&nbsource%3Ddebug&nbprefer%3DYES
n%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":"111111111111111111",
                "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.
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.
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.
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.
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.
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","defaultResult":false},{"operator":"excludes","left":["REDMI NOTE 3","VIVO
X5M"],"right":"mobileModel","defaultResult":false},{"operator":"vLimitIn","exclusive":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.

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
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.
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.
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.

1.7. 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 attributes

New attributes of component cannot be implemented on earlier clients, but no error will be reported either. To downgrade the attributes, see the following code:

```
Page({
    data: {
        canIUse: my.canIUse('button.open-type.share')
    }
})
```

Base library version

Base library version	Corresponding baseline version
2.8.9	• Android: 10.2.3 • iOS: Mini Program New Container Baseline cp_change_15200851
1.14.1	10.1.68
1.14.1	10.1.60
1.9.0	10.1.32

Description of base library integration

Due to the iteration of baseline versions, different baseline versions require different ways to integrate the basic library. Before integrating the basic library, please confirm your baseline version.

iOS

10.2.3 baseline version

The new container baseline of Mini Program cp_change_15200851 supports the construction of Mini Program Basic Library 2.0.

Android

10.1.68.7 and later baseline versions

Set the Provider instance at startup. The sample code 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());
```

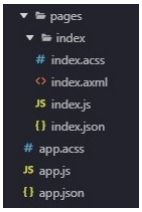
> **Note**
> - If the H5 public resource package is used in the client, please merge the relevant code of the public resource package into the instance class of the `H5AppCenterPresetProvider`.
> - For more information, Please refer to the Get code sample.

1.8. Frameworks

1.8.1. 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.

1.8.2. 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)
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 click 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:

```
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
```

> **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 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-attributes 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);
```

1.8.3. 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 update.'
    })
  },
  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**
> In case data is an object, changing it in the page affects different instances of the page.

Lifecycle method

- onLoad: Used on page loading. 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;
}
```

1.8.4. 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**
> 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**
> <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**
> 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 -->
```

1.8.5. 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.

1.8.6. 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**
  > 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;
}
```

1.8.7. Global configuration for mini program

1.8.7.1. About 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 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',
  },
});
```

1.8.7.2. 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	Required	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	Required	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.

> **Note**
> - The page reached by page jump (my.navigateTo) or page redirection (my.redirectTo) will not display the bottom tab bar even if it is a page defined in the tabBar configuration.
> - The first page of the tabBar must be the home page.

The following table describes the configuration items 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	Yes	The configurations of each tab in tabBar

The following table describes the detailed configurations of each tab.

Attribute	Type	Required	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 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"
      }
    ]
  }
}
```

1.8.7.3. 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.

1.8.7.4. 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.	

> **Note**
> - Do not perform page stack operations such as r`edirectTo` or `navigateTo` in `onShow`.
> - Do not call in `onLaunch` because the page has not been generated yet.

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
});
```

1.8.7.5. 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 subpage.

```
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.

1.8.8. Mini Program page

1.8.8.1. About Mini Program Page

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.

1.8.8.2. 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 shows a basic example:

```
{
  "optionMenu": {
    "icon": "https://img.alicdn.com/tps/i3/T1OjaVFl4dXXa.JOZB-114-114.png"
  },
  "titlePenetrate": true
}
```

1.8.8.3. 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.

1.8.8.4. 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.

1.8.8.5. Page registration

Page(object: Object)

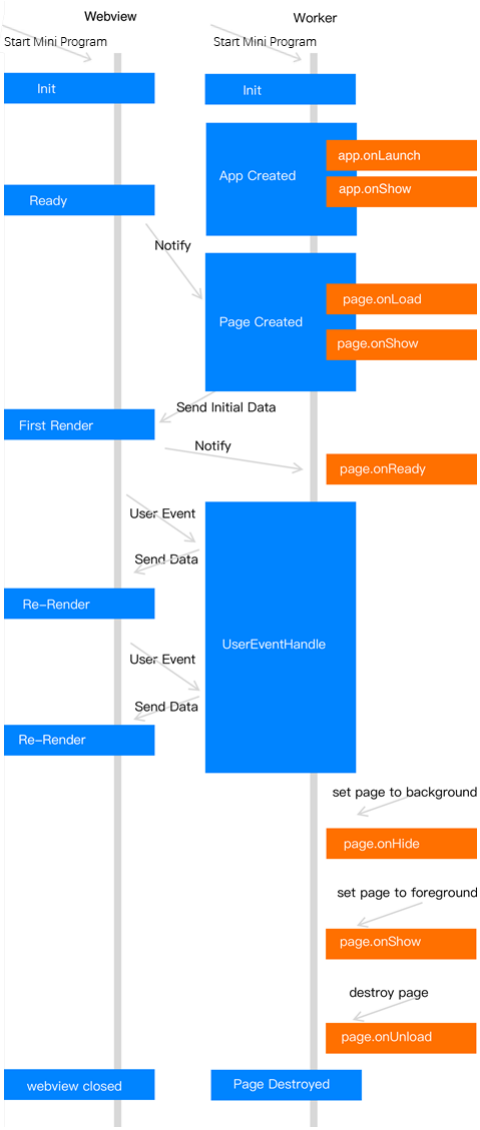
Define `Page()` in `.js` file of `/pages` directory, which is used for registering a Mini Program page and accept an object as an attribute, to specify information such as initial data, life cycle callback, and event processing of the page.

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.	-

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
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 manual, 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, t
he event 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 c
alled by the user.
      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.

1.8.8.6. 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.

1.8.9. AXML

1.8.9.1. About AXML

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,
    });
  },
});
```

1.8.9.2. 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.

Content

```
<view> {{ message }} </view>
```

```
Page({
  data: {
    message: 'Hello alipay!',
  },
});
```

Component attributes

Component attributes must be enclosed in double quotation marks (`""`).

```
<view id="item-{{id}}"> </view>
```

```
Page({
  data: {
    id: 0,
  },
});
```

Control attributes

Control 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}`.

1.8.9.3. 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 control 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.

1.8.9.4. 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.

> **Note**
> - If `a:key` is not provided, an error will be reported.
> - If you know that the list is static or you don't care about its order, you can ignore it.

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, allow users enter 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" />
```

1.8.9.5. 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)
  },
});
```

1.8.9.6. 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>
```

Incorrect 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 -->
```

1.8.10. SJS syntax reference

1.8.10.1. About SJS

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.

How to use

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	Required	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.

1.8.10.2. Variable

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
```

1.8.10.3. 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:

```
// 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';
```

1.8.10.4. 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.

1.8.10.5. 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)
```

1.8.10.6. Data type

SJS supports the following data types:

- **String**: String
- **Boolean**: Boolean
- **Number**: Number
- **Object**: Object
- **Function**: Function
- **Array**: Array
- **Date**: Date
- **Regexp**: Regular expression

Determine the data type

SJS provides `constructor` and `typeof` to determine 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);
```

1.8.10.7. 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.

1.8.10.8. esnext

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
```

1.8.11. 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.

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 starting with `.a-` and `.am-` are occupied for system components and cannot be used.
> - Attribute selectors are 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');
```

1.8.12. Event system

1.8.12.1. About event

- 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.

How to use

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

1.8.12.2. 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.

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.
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.

1.8.13. Custom components

1.8.13.1. About custom component

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.

1.8.13.2. 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>
```

1.8.13.3. Component configuration

Declare custom components 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 `../`.

1.8.13.4. Component template and style

Similar to pages, custom components can have their own AXML templates and ACSS styles.

AXML

AXML is a required 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 reference.

1.8.13.5. Component object

The parameter description of the component object is as follows.

Component constructor

The parameter description is as follows

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.

> **Note**
> - props is the attribute passed from the outside, which can specify the default attribute and cannot be modified in the internal code of the custom component.
> - The props attribute can be directly referenced in the axml of the custom component.
> - Events in the axml of the custom component can only be responded to by the methods in the methods of the js of the custom component. If you need to call the function passed by the parent component, you can call it through `this.props` in methods.

```
// /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 attributes

Attribute	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
```

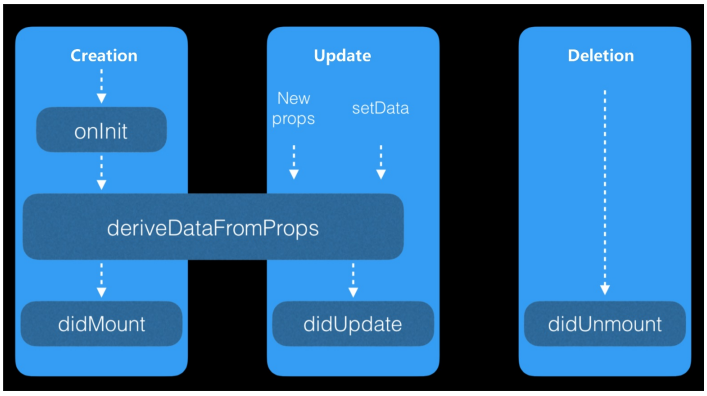
Component instance method

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.

1.8.13.6. 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.



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);
  },
});
```

> **Note**
> - Call `this.setData` inside a component will trigger didUpdate.
> - Call `this.setData` by an external caller will also trigger didUpdate.

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);
  },
});
```

1.8.13.7. 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',
  },
});
```

1.8.13.8. 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 oper
ation.
  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 using `ref` to bind `saveRef`, the `saveRef` method will be triggered when the component is initialized.
> - The parameter `ref` of the `saveRef` method is the custom component instance, which is passed to the `saveRef` method by the framework.
> - `ref` can also be used by 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>
```

1.8.13.9. Use custom component

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.

How to use

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 using a custom component, the attributes passed to the custom component can be obtained through `this.props` in the custom component, see props.
> - Custom components can only be used in the page's own AXML file and the component's own AXML file, and cannot be used through `import` or `include`.

Correct sample:

```
<!-- /pages/index/index.axml -->
<my-com />
```

Incorrect sample:

```
<!-- /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 "../".
```

1.8.13.10. Release 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 the developers to reuse and share.

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/'));
```

1.8.14. Performance optimization suggestions

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 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.
- 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 will affect 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 will affect 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:

Use the `$spliceData` method for long lists:

```
this.$spliceData({ 'a.b': [1, 0, 5, 6] })
```

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)
```

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.

> **Important**
> 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>
```

1.9. Basic Components

1.9.1. Component overview

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:

Attribute	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 series of property configurations, and each property value has a type requirement:

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.

1.9.2. Component FAQ

Interaction exception between keyboard and component

For the components that need to start **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.

1.9.3. View container

1.9.3.1. View

A view container, equivalent to the div label of Web or the View component of React Native.

Attributes

Attribute	Type	Default value	Description	Minimum version
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	-
hover-stop-propagation	Boolean	false	Whether to prevent the ancestor elements of the current element from appearing in the click state	1.10.0
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	-
onTransitionEnd	EventHandle	-	Trigger when the transition ends	1.8.0

onAnimationIteration	EventHandle	-	Trigger when a new animation process is started every time (not triggered for the first time)	1.8.0
onAnimationEnd	EventHandle	-	Trigger when the animation ends	1.8.0
onAppear	EventHandle	-	Trigger when the current element is visible	1.9.0
onDisappear	EventHandle	-	Trigger when the current element changes from visible to invisible	1.9.0
onFirstAppear	EventHandle	-	Trigger when the current element becomes visible for the first time	1.9.4

Note

The animation generated by using `my.createAnimation` is implemented through transition and only triggers `onTransitionEnd`; it does not trigger `onAnimationStart`, `onAnimationIteration`, or `onAnimationEnd`.

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>
```

1.9.3.2. Swiper

The slider view container.

Attributes

Attribute	Type	Default value	Description	Minimum version
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	-
active-class	String	-	The `class` for `swiper-item` when it is visible	1.13.7
changing-class	String	-	When `acceleration` is set to `{{true}}` and the scrolling process is in progress, the `class` is used when the middle screen is visible	1.13.7
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	-
vertical	Boolean	false	Whether the sliding direction is vertical	-
previous-margin	String	'0px'	Front margin, unit: px. 1.9.0 currently only supports horizontal direction	1.9.0
next-margin	String	'0px'	The back margin is in px. 1.9.0 currently only supports the horizontal direction	1.9.0
acceleration	Boolean	false	When it is turned on, multiple screens will slide continuously according to the sliding speed	1.13.7
disable-programmatic-animation	Boolean	false	Whether to disable the animation when the code changes trigger the swiper switch	1.13.7
onChange	EventHandle	-	It is triggered when `current` changes, `event.detail = {current, isChanging}`, where `isChanging` has a value only when `acceleration` is set to `{{true}}`. When sliding multiple screens continuously, `isChanging` is `true` when the `onChange` event is triggered on several screens in the middle, and `false` is returned on the last screen。	-
onTransition	EventHandle	-	The transition event is triggered when the position of `swiper-item` in `swiper` changes.	1.15.0
onAnimationEnd	EventHandle	-	When the animation ends, the `animationEnd` event is triggered, `event.detail = {current, source}`, where the source value can be `autoplay` and `touch`	1.15.0
disable-touch	Boolean	false	Whether to disable user touch operations	1.15.0

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
    })
  },
})
```

1.9.3.3. Scroll view

The scrollable view area.

Attributes

Attribute	Type	Default value	Description	Minimum version
class	String	-	External style name	-
style	String	-	Inline style name	-
scroll-x	Boolean	false	Allow horizontal scrolling	-
scroll-y	Boolean	false	Allow vertical scrolling	-
upper-threshold	Number	50	How far from the top/left (in px) does it take to trigger the scrolltoupper event	-
lower-threshold	Number	50	The distance from the bottom/right side (in px) when the scrolltolower event is triggered	-
scroll-top	Number	-	Set the vertical scroll bar position	-
scroll-left	Number	-	Set the horizontal scroll bar position	-
scroll-into-view	String	-	The value should be a child element ID, then scroll to that element and align the top of the element to the top of the scroll area	-
scroll-with-animation	Boolean	false	Use animated transitions when setting the scroll bar position	-

scroll-animation-duration	Number	-	When `scroll-with-animation` is set to `true`, you can set `scroll-animation-duration` to control the execution time of the animation in milliseconds	1.9.0
enable-back-to-top	Boolean	false	When you click the top status bar of iOS or double-click the title bar of Android, the scroll bar returns to the top, only supporting vertical direction	1.11.0
trap-scroll	Boolean	false	When scrolling vertically, when scrolling to the top or bottom, the page scrolling is forcibly prohibited, and only the scroll-view itself is scrolled	1.11.2
onScrollToUpper	EventHandle	-	Scrolling to the top/left will trigger the scrolltoupper event	-
onScrollToLower	EventHandle	-	Scrolling to the bottom/right will trigger the scrolltolower event	-
onScroll	EventHandle	-	Trigger when scrolling, `event.detail = {scrollLeft, scrollTop, scrollHeight, scrollWidth}`	-
onTouchStart	EventHandle	-	Touch action to start	1.15.0
onTouchMove	EventHandle	-	Move after touch	1.15.0
onTouchEnd	EventHandle	-	Touch action ends	1.15.0
onTouchCancel	EventHandle	-	Touch actions are interrupted by incoming call reminders and pop-up windows	1.15.0

Note

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-vie
w="{{toView}}" 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,
    });
  },
});
```

Tips

- `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`.

1.9.3.4. 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
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>
```

1.9.3.5. 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>
```

Cautions

- 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.

1.9.4. Basic content

1.9.4.1. Text

Text, only nesting is supported within components.

Attributes

Attribute	Type	Default value	Description	Minimum version
selectable	Boolean	false	Whether it is optional	-
space	String	-	Display consecutive spaces	-
decode	Boolean	false	Decode or not	-
number-of-lines	number	-	Multi-line ellipsis, the value must be greater than or equal to 1, and the performance is consistent with the CSS `-webkit-line-clamp` attribute.	-

Valid values for space

Value	Description
nbsp	Space size according to font settings
ensp	Half the size of a Chinese character space
emsp	Chinese character space size

Code example

```
<view class="page">
  <view class="text-view">
    <text>{{text}}</text>
  </view>
</view>
```

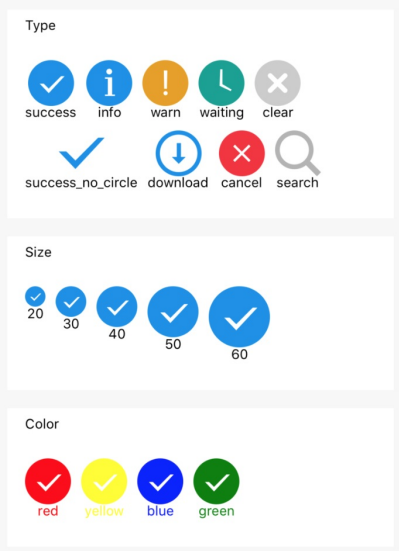
```
Page({
  data: {
    text: `Alipay is a large-scale lifestyle service platform, serving a wide variety of users ranging from middle-aged and elderly to
teenagers.
      There are not only self-operated official apps, but also integrated third-party apps, to provide users with rich choices.
      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:)
    `,
  },
})
```

1.9.4.2. Icon

API

Attribute	Type	Default value	Description	Minimum version
type	String	-	Icon type, valid values include: info, warn, waiting, cancel, download, search, clear, success, success_no_circle	loading (1.7.2)
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',
    ]
  }
})
```

1.9.4.3. Progressbar

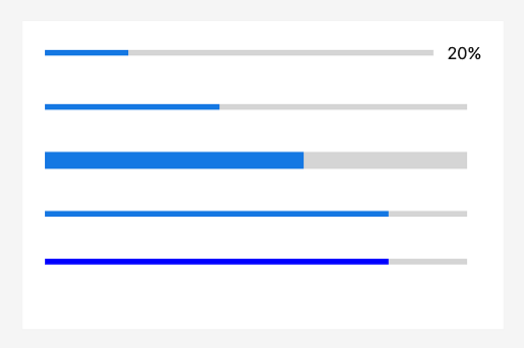
Progress bar.

Attributes

Attribute	Type	Default value	Description	Minimum version
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	-
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"/>
```

1.9.4.4. 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

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	-
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.

Display results	Description	Entity name
	Space	
<	Less-than sign	<

>	Greater-than sign	>
&	Ampersand	&
“	Quotation mark	"
‘	Apostrophe	'

Text node

Attribute	Type	Description	Required
type	String	Node type	Yes
text	String	The text of the word.	Yes

1.9.5. Form component

1.9.5.1. 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.	-
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.	-

open-type valid 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 valid 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

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>
```

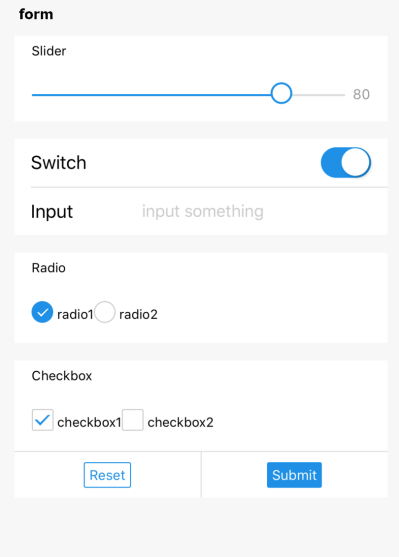
1.9.5.2. form

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.	-

Legend



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')
  }
})
```

1.9.5.3. 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.

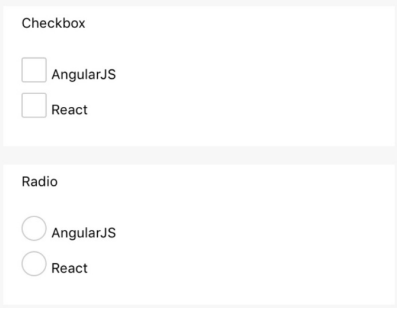
`for` has a higher priority than internal components. When multiple internal components exist, the first component is triggered by default.

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.	-

Legend



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>
```

1.9.5.4. 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`.	`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}.	-

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}}" />
```

1.9.5.5. textarea

This topic describes `textarea`, the multi-line input 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}.	-

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" />
```

1.9.5.6. radio

This topic describes the single-choice selector, that is, radio.

radio-group

The radio group.

Attribute	Type	Default value	Description	Minimum version

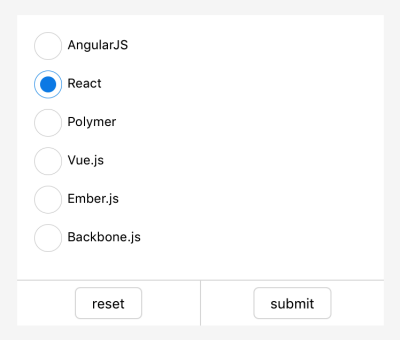
onChange	EventHandle	-	Trigger when the selected item is modified, where `event.detail = {value: The value of the selected radio item}`	-
name	String	-	The component name, which is used to submit the form to obtain data.	-

radio

The radio 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

Legend



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)
  }
})
```

1.9.5.7. checkbox

This topic describes the multi-choice selector, that is, checkbox.

checkbox-group

The multi-selector group.

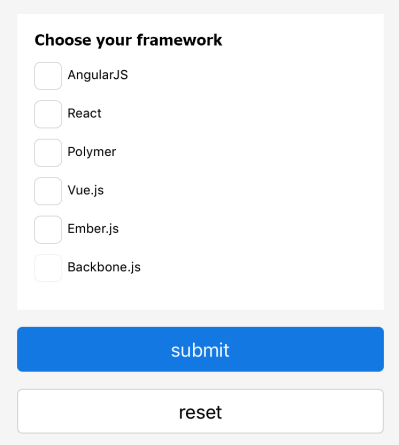
Attribute	Type	Default value	Description	Minimum version
name	String	-	The component name, which is used to submit the form to obtain data.	-
onChange	EventHandle	-	Trigger when the selected item is modified, where `detail = {value: The value of the selected checkbox item}`.	-

checkbox

The multi-selector 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

Legend



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>
```

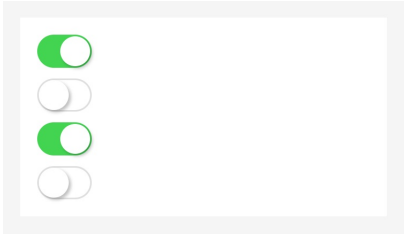
```
Page({
  data: {
    items: [
      {name: 'angular', value: 'AngularJS'},
      {name: 'react', value: 'React', checked: true},
      {name: 'polymer', value: 'Polymer'},
      {name: 'vue', value: 'Vue.js'},
      {name: 'ember', value: 'Ember.js'},
      {name: 'backbone', value: 'Backbone.js', disabled: true},
    ],
  },
  onChange(e) {
    my.alert({
      title: `You have selected the ${e.detail.value} framework`,
    });
  },
});
```

1.9.5.8. switch

This topic describes `switch`.

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

Legend



Code sample

```
<view class="page">
  <view class="switch-list">
    <view class="switch-item">
      <switch checked onChange="switchChange"/>
    </view>
  </view>
</view>
```

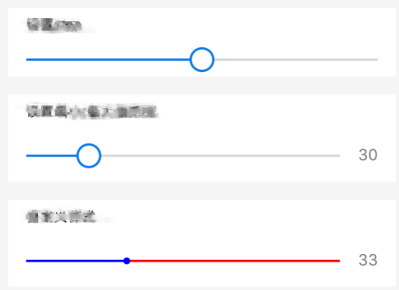
```
Page({
  switchChange (e){
    console.log('The switchChange event, and the value is: ', e.detail.value)
  },
})
```

1.9.5.9. 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

Legend



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)
})
```

1.9.5.10. 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>
```

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;
}
```

1.9.5.11. Bottom scroll selector

A scroll selector that pops up from the bottom.

Attributes

Attribute	Type	Defaults	Description	Minimum version
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.	
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	

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,
    });
  },
});
```

1.9.6. Navigator

This article is about page links (navigators).

Attribute	Type	Default value	Description	Minimum version
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	Description	Minimum version

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>
```

1.9.7. Media component

1.9.7.1. 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.
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`.

Legend

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.



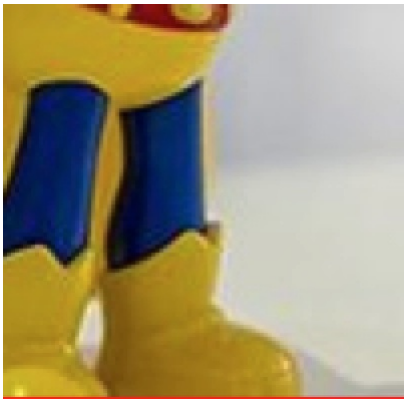
bottom left

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);
  }
})
```

1.9.7.2. video

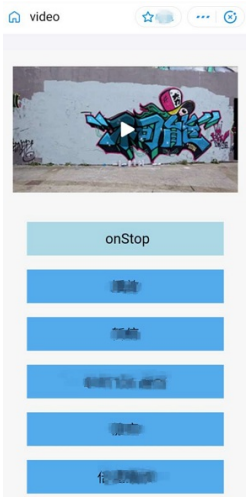
The video component is supported in base library 1.14.1 and later versions.

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.

> **Note**
> - CSS animations have no effect on the video component.
> - Customize the white padding that appears on both sides when watching videos in vertical screen:
>   - If the aspect ratio of the video is inconsistent with the aspect ratio of the video component, please adjust it through object-fit.
>   - If the actual aspect ratio of the poster is inconsistent with the aspect ratio of the container, please adjust it through 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.
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
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.

1.9.8. Canvas

Attribute

Attribute	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
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.

- 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)
  }
})
```

1.9.9. Map

This article introduces the map component.

Instructions

- The `map` component is a native component created by the client. The native component has the highest hierarchy, so no matter how much `z-index` is set for other components on the page, they cannot be above the native component.
- Do not use the `map` component in `scroll-view`.
- CSS animation is invalid for the map component.
- After reducing or enlarging the map scale, please reset the `scale` value of `data` in the `onRegionChange` function, otherwise the map scale will return to the size before zooming after dragging the map area and reloading it. For details, please refer to the `regionchange` function in the sample code. For basic library 1.14.0 and above, you can use the `default-scale` attribute instead of `scale` to optimize the code.

API

If you need to display multiple map components on the same page, you need to use different IDs.

Attribute	Type	Default value	Description	Supported versions
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	-
default-scale	Number	16	The default zoom level ranges from 5~8. If you only need to specify the initial scale, you can set default-scale instead of scale. When the user zooms in or out, there is no need to listen to `onRegionChange` to reset the scale.	Basic library 1.14.0
markers	Array		Marker	-
polyline	Array		Polyline	-
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.	-
include-points	Array	-	The field of view will be extended a small amount to include the passed coordinates. `[{ latitude: 30.279383, longitude: 120.131441,}]`	-
include-padding	Object	-	The field of view is displayed within the map padding. `{ left:0, right:0, top:0, bottom:0}`	-

ground-overlays	Array	-	Overlays, custom textures. `[{ // Upper right, lower left 'include-points':[{ latitude: 39.935029, longitude: 116.384377, }, { latitude: 39.939577, longitude: 116.388331, }], image:'/image/overlay.png', alpha: 0.25, zIndex:1}]`	-
tile-overlay	Object	-	Overlays, grid textures. `{ url:'http://xxx', type:0, // url type tileWidth:256, tileHeight:256, zIndex:1,}`	-
setting	Object	-	Setting. `{ // gesture gestureEnable: 1, // scale showScale: 1, // compass showCompass: 1, // slide down with both hands tiltGesturesEnabled: 1, // traffic condition display trafficEnabled: 0, // map POI information showMapText: 0, // Gaode map logo position logoPosition: { centerX: 150, centerY: 90 }}`	-
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	Description	Type	Required	Remarks	Minimum version
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		-
customCallout	Custom callout background; Currently only supports Gaode map style	Object	No	`{ "type": 2, "descList": [{ "desc": "Expected", "descColor": "#333333" }, { "desc": "5 minutes", "descColor": "#108EE9" }, { "desc": "Arrival", "descColor": "#333333" }], "isShow": 1}`	-
iconAppendStr	The marker image can come from View	String	No	When used together with iconPath, the image corresponding to iconPath and the string will be combined to generate an image as the marker icon.	-
iconAppendStrColor	The marker image can come from View, and the bottom describes the text color.	String	No	Default is: #33B276	-
fixedPoint	Based on screen position	Object	No	Based on screen position `{ //Number of pixels from the upper left corner of the map, Number originX:100, originY:100}`	-
markerLevel	The drawing level of the marker on the map	Number	No	The Z coordinate system is consistent with other overlays on the map.	-
label	Bubbles on marker	Object	No	The bubble on the marker. Multiple bubbles can be displayed on the map at the same time. Bind `onMarkerTap` . `{ content:"Hello Label", color:"#000000", fontSize:12, borderRadius:3, bgColor:"#ffffff", padding:5,}`	-
style	Customize marker styles	Object	No	Customize the marker style and content	-

polygon

Polygon is used to construct a polygonal object.

Attribute	Description	Type	Required	Remarks	Supported version
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.	-
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	Description	Type	Required	Remarks	Minimum version
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	-
iconPath	Texture address of the line	String	No	The image path under the project directory can be written as a relative path. If it starts with '/', it means it is relative to the root directory of the mini program.	-
iconWidth	The width when using texture	Number	No	-	-
zIndex	The Z coordinate of the overlay	Number	-	-	-
iconPath	Texture	String	-	The image path under the project directory can be written as a relative path. If it starts with '/', it means it is relative to the root directory of the applet. If there is an iconPath, color will be ignored. However, iconPath can be used in conjunction with colorList, so that the texture will float above the rainbow line. To avoid covering the rainbow line, the background color of the texture image can be set to transparent.	-
colorList	Rainbow Lines	Array	-	Rainbow line, segmented by points. For example, if `points` has 5 points, then `colorList` should pass 4 color values, and so on. If the number of `colorList` is less than 4, the color of the remaining lines is the same as the last color. `[ "#AAAAAA", "#BBBBBB"]`	-

circles

Circles is used to display circles on the map.

Attribute	Description	Type	Required	Remarks	Supproted version
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	Description	Type	Required	Remarks	Supproted version

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

The location of the control on the map, and the size of the control.

Attribute	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.

callout

Customize the bubble window above the marker point.

Attribute	Description	Type	Required	Remarks
content	Content	String	No	Default is null

customCallout

Customize callout background. Currently only supports Gaode map style.

Attribute	Description	Type	Required	Remarks
type	Style type.	Number	Yes	0 is black style, 1 is white style, 2 is background + text.
time	Time	String	Yes	Time value
descList	Describe the array.	Array	Yes	Describe array `{ "type": 0, "time": "3", "descList": [{ "desc": "Click to hail a taxi immediately", "descColor": "#ffffff" }], "isShow": 1}` isShow: 1 means show, 0 means don't show.

fixedPoint

Fixed point based on screen position.

Attribute	Description	Type	Required	Remarks
originX	Horizontal pixels	Number	Yes	The number of pixels from the top left corner of the map (starting from 0).
originY	Vertical pixels	Number	Yes	The number of pixels from the top left corner of the map (starting from 0).

The latitude and longitude of the map component must be set. If the latitude and longitude are not set, the default is the latitude and longitude of Beijing.

Marker illustrations

Marker style priority description

- `customCallout` , `callout` and `label` are mutually exclusive, and the priority order is: `label` > `customCallout` > `callout` .
- `style` and `icon` are mutually exclusive, and the priority order is: `style` > `iconAppendStr` ; `style` > `icon` .

style

```
{    type:1,    text1:"Style1",    icon1:'xxx',    icon2:'xxx'}
```

```
{    type:2,    text1:"Style2",    icon1:'xxx',    icon2:'xxx'}
```

```
{ type:3, icon:xxx, //optional text:xxx, //required color:xxx, //default value is #33B276 bgColor:xxx, //default value is #FFFFFF gravity:"left
/center/right", //default value is center fontType:"small/standard/large" //default value is standard}
```

customCallout

```
{    "type": 0,    "time": "3",    "descList": [{        "desc": "Click to hail a taxi now",        "descColor": "#ffffff"    }],    "isShow": 1
}
```

```
{    "type": 1,    "time": "3",    "descList": [{        "desc": "Click to hail a taxi now",        "descColor": "#333333"    }],    "isShow": 1
}
```

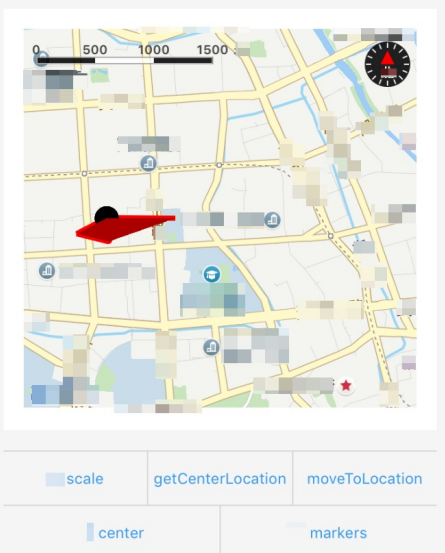
```
{ "type": 2, "descList": [{ "desc": "Expected", "descColor": "#333333" }, { "desc": "5 minutes", "descColor": "#108EE9" }, { "desc":
"Arrival", "descColor": "#333333" }], "isShow": 1}
```

isShow: 1 means show, 0 means don't show.

label

```
{    content:"Hello Label",    color:"#000000",    fontSize:16,    borderRadius:5,    bgColor:"#ffffff",    padding:12,}
```

- content: required
- color: optional, default value is "#000000"
- fontsize: optional, default value is 14
- borderRadius: optional, default value is 20
- bgColor: optional, default value is "#FFFFFF"
- padding: optional, default value is 10



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,
      }],
    });
  },
})
```

1.9.10. Open components

1.9.10.1. web-view

The `<web-view />` component bears HTML5 pages and automatically fills the entire Mini Program page. 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.

> **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.

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 }`

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.
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.
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()`.

1.10. Mini Program Extended Component antd-mini

1.10.1. Overview

The mini program extended component library is an important supplement to the basic component library, and is a set of open source UI component libraries developed based on the mini program custom component specifications for rapid reuse by mini program developers.

Installation

Run the following command to install a mini program extended component:

```
$ npm i antd-mini -S
```

Usage

Register a component in the JSON file of a page. For example, to register a tag component, you can use the following statement:

```
{
    "usingComponents": {
        "tag": "antd-mini/es/Tag/index"
    }
}
```

Call the component in the axml file:

```
<tag></tag>
```

1.10.2. Common

1.10.2.1. Button

A Button component is used to initiate an instant operation and mark a command or encapsulate a set of commands to respond to user taps and trigger the corresponding business logic.

Attribute

Attribute	Type	Required	Default value	Description
type	'default' | 'primary' | 'warn' | 'danger' | 'success' | 'light'	No	'default'	The button type. Valid values include: • default: auxiliary button. • primary: button in the brand color. • warn: warning button. • danger: danger button. • success: success button. • light: light button.
fill	'outline' | 'solid' | 'none'	No	'solid'	The style used for filling.
disabled	boolean	No	false	Whether to disable the component.
activeClassName	string	No	-	The class name used when the button is tapped.
subText	string	No	-	The auxiliary text, which is displayed on the second row.
inline	boolean	No	false	Whether to make the button an inline element that does not takes up the whole width of the parent container.
inlineSize	'small' | 'medium' | 'large' | 'x-large'	No	'medium'	The size of the inline button.
icon	string	No	-	The icon on the left of the button.
loading	boolean	No	false	Whether the button is being loaded. You cannot tap it if the button is being loaded.
loadingText	string	No	-	The text displayed when the button is being loaded.

htmlType	'button' | 'submit' | 'reset'	No	'button'	The native type of the button, which takes effect when you submit a form.
mode	string	No	-	When you use the button component with a form, you must set mode to form.
form	string	No	-	When you use this component with a form, you must set this property to the value of the form property of the form component.
className	string	No	-	The class name.

Event

Event name	Description	Event type
onTap	Callback fired when the button is tapped.	(e: Event) => void

Slot

Name	Description
icon	The icon slot.

Style class

Class name	Description
amd-button	The style of the entire button.
amd-button-content	The style of the button content.
amd-button-loading-container	The style of the loading area.
amd-button-loading-text	The text style of the loading area.
amd-button-loading	The style of the loading animation.
amd-button-wrap	The style of the loading area on the right.
amd-button-icon	The icon style.
amd-button-text	The style of the button text.
amd-button-subtext	The style of the subtitle.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Fill mode">
    <view class="btn-list btn-list-default">
      <button
        type="primary"
        fill="solid">
        solid
      </button>
      <button
        type="primary"
        fill="outline">
        outline
      </button>
      <button
        type="primary"
        fill="none">
        none
      </button>
    </view>
  </demo-block>
  <demo-block title="Semantic button">
    <view class="btn-list">
      <button
        a:for="{{list}}"
        type="{{item.type}}">
        {{item.type}}
      </button>
    </view>
  </demo-block>
  <demo-block title="Disabled status">
    <button
      disabled
      type="primary">
      Disabled status
    </button>
  </demo-block>
  <demo-block title="Loading status">
    <button
      loading
      type="primary"
      loadingText="Loading">
      Loading status
    </button>
  </demo-block>
  <demo-block title="With subtitle">
    <view class="btn-list">
      <button
        subText="subtitle"
        type="primary">
        Button
      </button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    list: [
      { type: 'default' },
      { type: 'primary' },
      { type: 'warn' },
      { type: 'danger' },
      { type: 'success' },
      { type: 'light' },
    ],
  },
});
```

The following shows an example of the code in the index.acss file:

```
.btn-list-default {
  margin-bottom: -24rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Button",
  "usingComponents": {
    "button": "antd-mini/es/Button/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "switch": "antd-mini/es/Switch/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Inline button

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Fill mode">
    <view class="container">
      <button
        a:for="{{fills}}"
        fill="{{item}}"
        type="primary"
        inline="{{true}}" >
        {{item}}
      </button>
    </view>
  </demo-block>
  <demo-block title="Semantic button">
    <view class="container">
      <button
        a:for="{{types}}"
        type="{{item}}"
        inline="{{true}}" >
        {{item}}
      </button>
    </view>
  </demo-block>
  <demo-block title="Button size">
    <view class="container">
      <button
        a:for="{{sizes}}"
        type="primary"
        inline="{{true}}"
        inlineSize="{{item}}">
        {{item}}
      </button>
    </view>
  </demo-block>
  <demo-block title="Disabled status">
    <view class="container">
      <button
        type="primary"
        inline="{{true}}"
        disabled="{{true}}">
        Disabled status
      </button>
    </view>
  </demo-block>
  <demo-block title="Loading status">
    <view class="container">
      <button
        type="primary"
        inline="{{true}}"
        loading="{{true}}"
        loadingText="Loading">
        Loading status
      </button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    types: ['default', 'primary', 'warn', 'danger', 'success', 'light'],
    sizes: ['small', 'medium', 'large', 'x-large'],
    fills: ['solid', 'outline', 'none'],
  },
});
```

The following shows an example of the code in the index.acss file:

```
.container button {
  margin-right: 12rpx;
  margin-bottom: 8rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Button",
  "usingComponents": {
    "button": "antd-mini/es/Button/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "switch": "antd-mini/es/Switch/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Custom icon

The following shows an example of the code in the index.axml file:

```
  <demo-block title="Custom icon">
    <view class="btn-list">
      <button
        type="primary"
        icon="AppOutline"
        onTap="handleTap" >
        Icon
      </button>

      <button
        type="primary"
        icon="{{url}}"
        onTap="handleTap" >
        Image
      </button>

      <button
        type="primary"
        onTap="handleTap" >
        <view slot="icon">※</view>Use icon slot
      </button>
    </view>
  </demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    url: 'https://gw.alipayobjects.com/mdn/rms_ce4c6f/afts/img/A*XMCgSYx3f50AAAAAAAAAAABkARQnAQ',
  },
  handleTap() {
    my.alert({
      title: 'tap',
    });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Button",
  "usingComponents": {
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Auxiliary button

The following shows an example of the code in the index.axml file:

```
<demo-block title="Assist button">
  <view class="container-1">
    <button type="default" fill="solid">Auxiliary operation</button>
    <button type="primary" fill="solid">Main operation</button>
  </view>

  <view class="container-2">
    <button type="default" fill="solid">Auxiliary operation</button>
    <button type="primary" fill="outline">Main operation</button>
  </view>

  <view class="container-3">
    <button type="default" inline="{{true}}">Auxiliary operation</button>
    <button type="primary" inline="{{true}}">Main operation</button>
  </view>

  <view class="container-4">
    <button type="default" fill="solid" inline="{{true}}">Auxiliary operation</button>
    <button type="primary" fill="outline" inline="{{true}}">Main operation</button>
  </view>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
});
```

The following shows an example of the code in the index.acss file:

```
.container-1,
.container-2 {
  display: flex;
  margin: 12px 0;
}

.container-3,
.container-4 {
  margin: 12px 0;
}

.container-1 .amd-button:first-child,
.container-2 .amd-button:first-child,
.container-3 .amd-button:first-child,
.container-4 .amd-button:first-child {
  margin-right: 6px;
}

.container-1 .amd-button:last-child,
.container-2 .amd-button:last-child,
.container-3 .amd-button:last-child,
.container-4 .amd-button:last-child {
  margin-right: 6px;
}

.amd-button {
  flex: 1;
}
```

The following shows an example of the code in the index.json file:

```
{
    "defaultTitle": "Button",
    "usingComponents": {
      "button": "antd-mini/es/Button/index",
      "demo-block": "../../components/DemoBlock/index"
    }
  }
```

1.10.2.2. Icon

An icon is a semantic vector graphic to metaphorically present the basic functionalities and provide users with correct, friendly, and clear guidance.

Attribute

Attribute	Type	Required	Default value	Description
type	string	Yes	""	The type of the icon.
size	'x-small' | 'small' | 'medium' | 'large' | 'x-large'	No	"medium"	The size of icon, x-small (16), small (32), medium (48), large (64), x-large (128).

color	string	No	-	The color of the icon, which is the same as the value of the color attribute in CSS.
fontSize	string	No	-	The icon size.
className	string	No	-	The class name.

Event

Event name	Description	Type
onTap	Callback fired when the icon is tapped.	`( e: Event ) => void`

Style class

Class name	Description
amd-icon	The style of the entire icon.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <search-bar placeholder="search" onInput="handleSearchChange" borderColor="#1677ff" />
  <view class="sum" a:if="{{!searchValue}}">
    There are currently
    <text>{{iconTypes.length}}</text>icon types
  </view>
  <view class="list-title">
    wireframe style
  </view>
  <view class="icon-list">
    <view class="icon-item" a:for="{{iconTypes}}" a:if="{{item.indexOf('Outline') > 1&&item.toUpperCase().indexOf(searchValue.toUpperCase())>-1
}}">
      <view class="icon-item-wrapper" onTap="handleClickIcon" data-info="{{item}}">
        <am-icon type="{{item}}" color="#1677ff" size="small" />
        <text class="icon-desc">{{item}}</text>
      </view>
    </view>
  </view>
  <view class="list-title">
    solid style
  </view>
  <view class="icon-list">
    <view class="icon-item" a:for="{{iconTypes}}" a:if="{{item.indexOf('Fill') > 1&&item.toUpperCase().indexOf(searchValue.toUpperCase())>-
1}}">
      <view class="icon-item-wrapper" onTap="handleClickIcon" data-info="{{item}}">
        <am-icon type="{{item}}" color="#1677ff" size="small" onTap="handleClickIcon" data-info="{{item}}" />
        <text class="icon-desc">{{item}}</text>
      </view>
    </view>
  </view>

</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    iconTypes: [
      'MinusOutline',
      'AlipayCircleFill',
      'CheckCircleFill',
      'FireFill',
      'FaceRecognitionOutline',
      'StarFill',
      'EyeInvisibleFill',
      'SmileFill',
      'FrownFill',
      'BankcardOutline',
      'HeartOutline',
      'EyeFill',
      'HeartFill',
      'DownFill',
      'CloseCircleFill',
      'VideoOutline',
      'CouponOutline',
```

```
    'ReceiptOutline',
    'AntOutline',
    'UserCircleOutline',
    'PayCircleOutline',
    'BillOutline',
    'PlayOutline',
    'PayOutline',
    'MoreOutline',
    'ShrinkOutline',
    'ArrowsAltOutline',
    'StarOutline',
    'CheckOutline',
    'DeleteOutline',
    'LinkOutline',
    'InformationCircleOutline',
    'GlobalOutline',
    'InformationCircleFill',
    'ExclamationCircleFill',
    'CheckCircleOutline',
    'CloseCircleOutline',
    'SetOutline',
    'QuestionCircleFill',
    'QuestionCircleOutline',
    'UpCircleOutline',
    'FrownOutline',
    'DownCircleOutline',
    'ExclamationCircleOutline',
    'MinusCircleOutline',
    'RedoOutline',
    'UndoOutline',
    'EyeInvisibleOutline',
    'ForbidFill',
    'PicturesOutline',
    'PictureOutline',
    'PictureWrongOutline',
    'EyeOutline',
    'AddCircleOutline',
    'ClockCircleFill',
    'ClockCircleOutline',
    'BellMuteOutline',
    'KeyOutline',
    'BellOutline',
    'SearchOutline',
    'CollectMoneyOutline',
    'UnorderedListOutline',
    'AppstoreOutline',
    'ExclamationTriangleOutline',
    'AddOutline',
    'ScanningOutline',
    'ScanCodeOutline',
    'ExclamationOutline',
    'CloseOutline',
    'ScanningFaceOutline',
    'LeftOutline',
    'DownOutline',
    'UpOutline',
    'RightOutline',
    'KoubeiOutline',
    'KoubeiFill',
    'AAOutline',
    'ArrowDownCircleOutline',
    'MovieOutline',
    'CompassOutline',
    'LoopOutline',
    'TextOutline',
    'TagOutline',
    'FlagOutline',
    'EnvironmentOutline',
    'CalendarOutline',
    'LocationFill',
    'PhoneFill',
    'PhonebookOutline',
    'SmileOutline',
    'UserAddOutline',
    'FileWrongOutline',
    'SoundMuteFill',
    'SoundMuteOutline',
    'LockOutline',
    'UnlockOutline',
    'EditSOutline',
    'UploadOutline',
    'SoundOutline',
    'DownlandOutline',
    'SendOutline',
    'FillinOutline',
    'AudioMutedOutline',
    'AudioOutline',
```

```
      'UserOutline',
      'UserContactOutline',
      'TeamOutline',
      'UserSetOutline',
      'FileOutline',
      'MailOutline',
      'TruckOutline',
      'MailOpenOutline',
      'ChatCheckOutline',
      'ChatAddOutline',
      'ChatWrongOutline',
      'PhonebookFill',
      'AddressBookFill',
      'CalculatorOutline',
      'PieOutline',
      'HandPayCircleOutline',
      'GiftOutline',
      'TransportQRcodeOutline',
      'FolderOutline',
      'AlipaySquareFill',
      'TravelOutline',
      'AppOutline',
      'HistogramOutline',
      'MailFill',
      'CameraOutline',
      'EditFill',
      'SystemQRcodeOutline',
      'LockFill',
      'AudioFill',
      'TeamFill',
      'FilterOutline',
      'EditSFill',
      'LikeOutline',
      'TextDeletionOutline',
      'StopOutline',
      'FingerdownOutline',
      'MessageFill',
      'LocationOutline',
      'ContentOutline',
      'ExclamationShieldFill',
      'ReceivePaymentOutline',
      'ExclamationShieldOutline',
      'AddSquareOutline',
      'CloseShieldOutline',
      'CheckShieldOutline',
      'CheckShieldFill',
      'ShopbagOutline',
      'MessageOutline',
    ],
    searchValue: '',
  },
  handleSearchChange(value) {
    this.setData({ searchValue: value });
  },
  handleClickIcon(e) {
    const { info } = e.target.dataset;

    top.postMessage({ iconType: info }, '*');

    my.showToast({
      content: `${info} Copied to clipboard`,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo {
  background: #fff;
}

.sum {
  padding: 24rpx;
}

.sum>text {
  color: red;
}

.list-title {
  color: red;
  font-size: 32rpx;
  font-weight: bold;
  padding: 24rpx;
}

.icon-list {
  display: flex;
  flex-wrap: wrap;
  padding: 24rpx;
  background: #fff;
}

.icon-item {
  display: flex;
  flex-direction: row;
  align-items: flex-start;
  justify-content: center;
  flex-wrap: wrap;
  flex: 0 33.33333%;
  height: 170rpx;
  padding: 16rpx;
  border: 1rpx solid #eee;
  box-sizing: border-box;
}

.icon-desc {
  flex: 0 100%;
  margin-top: 24rpx;
  font-size: 24rpx;
  text-align: center;
  word-wrap: break-word;
  word-break: break-all;
}

.icon-item-wrapper {
  display: flex;
  flex-direction: row;
  align-items: flex-start;
  justify-content: center;
  flex-wrap: wrap;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Icon",
  "usingComponents": {
    "am-icon": "antd-mini/es/Icon/index",
    "search-bar": "antd-mini/es/SearchBar/index"
  }
}
```

1.10.3. Navigation

1.10.3.1. Tabs

A Tabs component is used to navigate between content groups. The current content needs to be divided into groups of the same hierarchical structure for content switching and display, which is often used at the top of a form or list.

> **Important**
>
> - Currently, one page can only use one **Tabs** component.
> - In basic library 2.x, when the scrolling does not work due to the built-in scroll-view, we recommend using scroll-view to prevent bubbling of touch events such as catchTouchStart and catchTouchMove.
> - A display error occurs to the built-in popup if the transform style is used inside a Tabs component to implement carousel. We recommend that you do not internally nest a component with a popup. Alternatively, you can use fallback property to implement a simple carousel function. For more information, see the demo of fallback.

Attribute

Tabs

Attribute	Type	Required	Default value	Description
adjustHeight	string	No	'current'	Automatically specifies the height of the slider as the height of the entire container.
activeClass	string	No	''	The class used when swiper-item is visible.
animation	boolean	No	false	Whether an animation is used for transition.No
className	string	No	-	The class name.
easingFunction	string	No	'default'	Whether to switch to a slow-motion animation.
index	number	No	0	The index that is currently activated.
nextMargin	string	No	'0px'	The back margin, in px. In version 1.9.0, only the horizontal direction is supported.
plus	string | slot	No	-	The button in the upper right corner. This is a custom node.
previousMargin	string	No	'0px'	The front margin, in px. In version 1.9.0, only the horizontal direction is supported.
snapToEdge	boolean	No	false	When the number of swiper-item components is greater than or equal to 2, circular is disabled, and previous-margin or next-margin is enabled, you can specify whether this margin is applied to the first and last elements.
sticky	boolean	No	false	Whether to support keeping the tab pinned to the top of the screen.
swipeRatio	number	No	0.2	The maximum swipe distance that can be achieved by a swipe gesture to trigger the switching of swiper-item components. When the swipe distance exceeds the maximum value, the swiper-item components will be switched.
swipeSpeed	number	No	0.05	The swipe distance that can be achieved by a swipe gesture. The smaller the swipe distance, the smaller shift is required for swiper-item components.
swipeable	boolean	No	false	Whether to support switching tabs through a gesture.
title	slot-scope	No	-	The style of the custom tab title. This property can be used only when type is set to basis.
touchAngle	number	No	45	The angle at which an effective swipe gesture is performed. The value is calculated by using the coordinates of the touchstart event and the first touchmove event. The smaller the value, the higher demand for the accuracy of the swipe angle.
type	'basis' | 'capsule' | 'mixin'	No	'basis'	The type. Valid values include: • basis: basic. • capsule: capsule. • mixin: mixed.

TabItem

Attribute	Type	Required	Default value	Description
tab	{title: string; subTitle?: string; badge?: number; disabled?: boolean}[]	Yes	-	The content of each tab.
className	string	No	-	The class name.

Event

Event name	Description	Type
onChange	Callback fired when the panel is switched.	(index: number) => void
onAnimationEnd	The onAnimationEnd event of the internal swiper component (only valid for a basic library version later than 1.50.0).	(e: any) => void=> void
onTouchStart	The onTouchStart event for the internal swiper component (only valid for basic library version 2.x).	(e: any) => void
onTransition	The onTransition event of the internal swiper component (only valid for basic library version 2.x).	(e: any) => void

Slot

Tabs

Name	Description
plus	Extra content of a form item.

Slot-scope

Tabs

Name	Description
title	The style of the custom tab title.

Style class

Tabs

Class name	Description
amd-tabs	The style of the entire Tabs component.
amd-tabs-bar	The style of the title area on the top.
`amd-tabs-bar-active`	The style used when the title area on the top is activated.
`amd-tabs-bar-active:after`	The indicator style used when the title area on the top is activated.
amd-tabs-bar	The style of the title area on the top.
amd-tabs-bar-sticky	The style of the title area on the top when tabs are pinned to the top.
amd-tabs-bar-plus	The style of the label in the upper right corner.
amd-tabs-bar-fade	The fading style applied to the both sides.
amd-tabs-bar-fade-left	The fading style applied to the left side.
amd-tabs-bar-fade-right	The fading style applied to the right side.
amd-tabs-bar-scroll-view	The style of the internal ScrollView component.
amd-tabs-bar-wrap	The style of each title.
amd-tabs-bar-item	The style of each title.
amd-tabs-bar-basis	The style of each title when type is set to basis.
amd-tabs-bar-capsule	The style of each title when type is set to capsule.

amd-tabs-bar-capsule-title	The style of the internal text in each title when type is set to capsule.
amd-tabs-bar-capsule-badge	The style of the internal badge in each title when type is set to capsule.
amd-tabs-bar-mixin	The style of each title when type is set to mixin.
amd-tabs-bar-mixin-title	The style of the text in each title when type is set to mixin.
amd-tabs-bar-mixin-subtitle	The style of the subtitle in each title when type is set to mixin.
amd-tabs-bar-disabled	The style of a tab in disabled state.

TabItem

Class name	Description
amd-tabs-item	The style of a tab menu.
amd-tabs-item-pane	The style of the content panel of each tab.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <tabs
    index="{{index}}"
    type="{{type}}"
    animation="{{animation}}"
    swipeable="{{swipeable}}"
    sticky="{{sticky}}"
    onChange="handleChangeTab">
    <view a:if="{{plusSlot}}" slot="plus" >
      <icon
        size="small"
        type="AddOutline"
        className="plus-icon"
        onTap="handleClickIcon" />
    </view>
    <view
      a:if="{{titleSlot}}"
      slot-scope="prop"
      slot="title">
      {{prop.tab.title + ' slot'}}
    </view>
    <block a:for="{{tabs}}">
      <tab-content
        a:if="{{index === 1}}"
        tab="{{item}}">
        <view>
          {{item.title}}
            <view
              a:for="{{height}}">
              ........Adaptive height according to the content........
            </view>
        </view>
      </tab-content>

      <tab-content
        a:elif="{{index === 3}}"
        tab="{{item}}">
        <view style="height: 30vh">
          ........{{item.title}}........
          <text>The change after the height is set on the previous layer</text>
        </view>
      </tab-content>

      <tab-content tab="{{item}}" a:else badge="{{item.badge}}">
        <view style="height: 20vh">
          ........{{item.title}}........
        </view>
      </tab-content>

    </block>
  </tabs>
  <list radius className="actions">
    <list-item>
      number of options
```

```
      Number of options
      <radio-group
        slot="extra"
        class="radio-group"
        name="tabsNumber"
        onChange="handleChangeTabNum" >
        <label
          class="radio"
          a:for="{{tabsNumber}}">
          <radio
            value="{{item.name}}"
            checked="{{item.checked}}" />
          <text
            class="radio-text">
            {{item.value}}
          </text>
        </label>
      </radio-group>
    </list-item>
    <list-item>
      type
      <radio-group
        slot="extra"
        class="radio-group"
        name="tabsType"
        onChange="handleChangeType" >
        <label
          class="radio"
          a:for="{{tabsType}}">
          <radio
            value="{{item.name}}"
            checked="{{item.checked}}" />
          <text
            class="radio-text">
            {{item.value}}
          </text>
        </label>
      </radio-group>
    </list-item>
    <list-item>
      Content transition animation<switch checked="{{animation}}" controlled onChange="handleChangeAnimation" slot="extra"/>
    </list-item>
    <list-item>
      Support gesture switch<switch checked="{{swipeable}}" controlled onChange="handleChangeSwipeable" slot="extra"/>
    </list-item>
    <list-item brief="Sticky effect">
      Ceiling<switch checked="{{sticky}}" controlled onChange="handleChangeSticky" slot="extra"/>
    </list-item>
    <list-item brief="Custom tab display, priority is higher than configuration">
      title slot<switch checked="{{titleSlot}}" controlled onChange="handleChangeTitleSlot" slot="extra"/>
    </list-item>
    <list-item brief="upper right corner operation button">
      plus slot<switch checked="{{plusSlot}}" controlled onChange="handleChangePlusSlot" slot="extra"/>
    </list-item>
  </list>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    index: 0,
    height: 30,
    type: 'basis',
    animation: true,
    swipeable: true,
    titleSlot: false,
    plusSlot: true,
    sticky: false,
    tabs: [
      {
        title: 'option one',
        subTitle: 'description copy',
      },
      {
        title: 'option two',
        subTitle: 'description copy',
      },
      {
        title: 'option three',
        subTitle: 'description',
      },
    ],
    tabsType: [
      { name: 'basis', value: 'normal', checked: true },
      { name: 'capsule', value: 'capsule' },
      { name: 'mixin', value: 'with description' },
```

```
  ],
  tabsNumber: [
    { name: '1', value: 'one' },
    { name: '2', value: 'two' },
    { name: '3', value: 'three', checked: true },
    { name: '-1', value: 'many' },
  ],
},
handleChangeAnimation(checked) {
  this.setData({ animation: checked });
},
handleChangeSwipeable(checked) {
  this.setData({ swipeable: checked });
},
handleChangeSticky(checked) {
  this.setData({ sticky: checked });
},
handleChangeTitleSlot(checked) {
  this.setData({ titleSlot: checked });
},
handleChangePlusSlot(checked) {
  this.setData({ plusSlot: checked });
},
handleChangeType(e) {
  this.setData({
    type: e.detail.value,
  });
},
// The click callback of tabs
handleChangeTab(e) {
  this.setData({
    index: e,
  });
},
// The plus area click event in the upper right corner
handleClickIcon() {
  my.alert({
    title: 'slot="plus"',
    content: 'The icon of the custom slot was clicked',
  });
},
handleChangeTabNum(e) {
  if (e.detail.value === '1') {
    this.setData({
      tabs: [
        {
          title: 'option',
          subTitle: 'description copy',
          badge: 6,
        },
      ],
    });
  } else if (e.detail.value === '2') {
    this.setData({
      tabs: [
        {
          title: 'option',
          subTitle: 'description copy',
        },
        {
          title: 'option two',
          subTitle: 'description copy',
        },
      ],
    });
  } else if (e.detail.value === '3') {
    this.setData({
      tabs: [
        {
          title: 'option one',
          subTitle: 'description copy',
        },
        {
          title: 'option two',
          subTitle: 'description copy',
        },
        {
          title: 'option three',
          subTitle: 'description',
        },
      ],
    });
  } else {
    this.setData({
      tabs: [
        {
          title: 'option one',
```

```
            subTitle: 'description copy',
          },
          {
            title: 'option two',
            subTitle: 'description copy',
          },
          {
            title: 'option three',
            subTitle: 'description',
          },
          {
            title: '4 Tab',
            subTitle: 'description',
            showBadge: true,
            badge: 1,
          },
          {
            title: '5 Tab',
            subTitle: 'description',
            badge: 999,
          },
          {
            title: '3 Tab',
            subTitle: 'description',
          },
          {
            title: '4 Tab',
            subTitle: 'description',
          },
          {
            title: '151111 Tab',
            subTitle: 'description',
          },
          {
            title: '42345 Tab',
            subTitle: 'description',
          },
          {
            title: '1511116787 Tab',
            subTitle: 'description',
          },
          {
            title: '42452 Tab',
            subTitle: 'description',
          },
          {
            title: '15451111 Tab',
          },
          {
            title: '4234 Tab',
            subTitle: 'description',
          },
          {
            title: '11251111 Tab',
            subTitle: 'description',
          },
          {
            title: '44123 Tab',
          },
          {
            title: '1531111 Tab',
            subTitle: 'description',
          },
          {
            title: '41 Tab',
          },
          {
            title: '15111111 Tab',
          },
        ],
      });
    }
  },
});
```

The following shows an example of the code in the index.acss file:

```
.actions {
  margin-top: 24px;
}
.amd-tabs-item-pane view {
  background-color: #fff;
  padding: 24rpx;
}

.plus-icon {
  font-size: 48rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Tabs",
  "usingComponents": {
    "tabs": "antd-mini/es/Tabs/index",
    "tab-content": "antd-mini/es/Tabs/TabItem/index",
    "icon": "antd-mini/es/Icon/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "switch": "antd-mini/es/Switch/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

fallback

The following shows an example of the code in the index.axml file:

```
<tabs
  index="{{curIdx}}"
  type="{{type}}"
  fallback="{{true}}"
  sticky="{{sticky}}"
  onChange="changeTab">
  <block a:for="{{tabs}}">
    <tab-content
      style="{{curIdx===index ? '' : 'display:none'}}"
      tab="{{item}}">
      <view>
        {{item.title}}
          <view
            a:for="{{height}}">
            ........Adaptive height according to the content........
          </view>
      </view>
    </tab-content>
  </block>
</tabs>
```

The following shows an example of the code in the index.js file:

```
const component2 = my.canIUse('component2');

Page({
  data: {
    curIdx: 2,
    height: 30,
    type: 'basis',
    animation: false,
    swipeable: false,
    titleSlot: false,
    plusSlot: true,
    sticky: false,
    tabs: [
      {
        title: 'option one',
        subTitle: 'description copy',
        corner: true,
      },
      {
        title: 'option two',
        subTitle: 'description copy',
      },
      {
        title: 'option three',
        subTitle: 'description',
      },
    ],
    tabsType: [
      { name: 'basis', value: 'normal', checked: true },
      { name: 'capsule', value: 'capsule' },
      { name: 'mixin', value: 'with description' },
    ],
    tabsNumber: [
      { name: '1', value: 'one' },
      { name: '2', value: 'two' },
      { name: '3', value: 'three' },
      { name: '-1', value: 'many', checked: true },
    ],
    tabsAnimation: [
      { name: true, value: 'true' },
      { name: false, value: 'false', checked: true },
    ],
    tabsSwipeable: [
      { name: true, value: 'true' },
      { name: false, value: 'false', checked: true },
    ],
    tabsTitleSlotScope: [
      { name: true, value: 'true' },
      { name: false, value: 'false', checked: true },
    ],
    tabsSticky: [
      { name: true, value: 'true' },
      { name: false, value: 'false', checked: true },
    ],
    tabsPlusSlotScope: [
      { name: true, value: 'true', checked: true },
      { name: false, value: 'false' },
    ],
    canSwipe: true,
  },
  onLoad() {
    if (!component2) {
      this.setData({
        canSwipe: component2,
      });
    }
  },
  // The click callback of tabs
  changeTab(e) {
    this.setData({
      curIdx: e,
    });
  },
  // The plus area click event in the upper right corner
  iconClick() {
    my.alert({
      title: 'slot="plus"',
      content: 'The icon of the custom slot was clicked',
    });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
    "defaultTitle": "Tabs",
    "usingComponents": {
      "tabs": "antd-mini/es/Tabs/index",
      "tab-content": "antd-mini/es/Tabs/TabItem/index",
      "icon": "antd-mini/es/Icon/index"
    }
  }
```

1.10.3.2. VTabs

A VTabs component is a group of vertical tabs, which is used together with VTabItem.

> **Note**
> Currently, a page can use only one VTabs component.

Attribute

VTabs

Attribute	Type	Required	Default value	Description
index	number	No	0	The index that is currently activated.
className	string	No	-	The class name.

VTabsItem

Attribute	Type	Required	Default value	Description
tab	{title: string; disabled?: boolean, badge?: {type: 'dot' | 'number' | 'text', text: number | string }}[]	Yes	-	The content of each tab.
className	string	No	-	The class name.

Event

VTabs

Event name	Description	Type
onChange	Callback fired when the panel is switched.	`(index: number) => void`

Slot

VTabs

Name	Description	Type
title	The custom title style.	The slot-scope.
icon	The custom icon style.	The slot-scope.

Style class

VTabs

Class name	Description
amd-vtabs	The style of the entire component.
amd-vtabs-bar	The style of the tab bar on the left.
amd-vtabs-bar-scroll-view	The style of the tab bar on the left.
amd-vtabs-bar-item-wrap	The style of a tab on the left.

amd-vtabs-bar-item	The style of a tab on the left.
`amd-vtabs-bar-item-active`	The style of a tab in activated state.
amd-vtabs-bar-item-disabled	The style of a tab in disabled state.
amd-vtabs-bar-item-title	The style of the title in a tab.
amd-vtabs-bar-item-count	The style of the badge in a tab.
amd-vtabs-bar-item-icon	The style of the icon slot in a tab.
amd-vtabs-content	The style of the content area on the right.
amd-vtabs-content-slides	The style of a single content area on the right.

VTabItem

Class name	Description
amd-vtabs-item	The style of the entire component.

Code sample

The following examples show how to use VTabs:

The following shows an example of the code in the index.axml file:

```
<vtabs
  index="{{index}}"
  onChange="onChange">
  <view slot="title" slot-scope="prop">
    {{prop.tab.title}}
  </view>

  <view slot="icon" slot-scope="prop" a:if="{{prop.tab.title === 'Content 6'}}">
    <view class="badge"/>
  </view>

  <view slot="icon" slot-scope="prop" a:if="{{prop.tab.title === 'Content 7'}}">
    <view class="icon">
      <icon type="FireFill"/>
      </view>
  </view>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 1 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 1'}}}">
    <text>content of content 1</text>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 2 ? 'currentItem': ''}}"
    tab="{{{title: 'The content is too long', count: 23}}}">
    <text>content of content 2</text>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 3 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 3', disabled: true}}}">
    <view
      onTap="changeHeight"
      style="{{newHeight? `display: block;height: ${newHeight}vh`: ''}}">
      content of content 3
    </view>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 4 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 4', count: 999, disabled: true}}}">
    <text>content of content 4</text>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 5 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 5'}}}">
    <text>content of content 5</text>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 6 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 6', badge: { type: 'text', text: 'discount'}}}}">
    <text>content of content 6</text>
  </vtab-content>

  <vtab-content
    className="vtabItem {{getVtabIndex + 1 === 7 ? 'currentItem': ''}}"
    tab="{{{title: 'Content 7', badge:{type:'dot'}}}}">
    <text>content of content 7</text>
  </vtab-content>
</vtabs>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    index: 3,
    getVtabIndex: 0,
    newHeight: 100,
  },
  onLoad() {
    this.setData({
      getVtabIndex: this.data.index,
    });
  },
  onChange(idx) {
    this.setData({
      getVtabIndex: idx,
      index: idx,
    });
  },
  changeHeight() {
    this.setData({
      newHeight: this.data.newHeight + 20,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.vtabItem {
  transition: all 200ms linear;
  background-color: #fff;
}

.currentItem {
  color: #f00;
  background-color: #fff;
}

.vtabItem text {
  display: block;
  height: 100vh;
}
.badge {
  background: #ff411c;
  height: 9px;
  width: 9px;
  border-radius: 50%;
}
.badge-num {
  background: #ff411c;
  height: 14px;
  width: 14px;
  border-radius: 50%;
  color: white;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 9px;
}

.icon .amd-icon {
  font-size: 14px;
  color: #ff411c;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Vtabs",
  "usingComponents": {
    "vtabs": "antd-mini/es/VTabs/index",
    "vtab-content": "antd-mini/es/VTabs/VTabItem/index",
    "icon": "antd-mini/es/Icon/index"
  },
  "allowsBounceVertical": false
}
```

1.10.4. Information display

1.10.4.1. Avatar

An avatar is a representation of a user or an object. It can display features of a person or an object more intuitively.

Attribute

Attribute	Type	Required	Default value	Description
size	'x-small' | 'small' | 'medium' | 'large'	No	'medium'	x-small: 80 x 80 small: 88 x 88 medium: 104 x 104 large: 120 x 120
src	string	No	-	The source URL of an avatar. The default image is a built-in image in gray.
name	string	No	-	The information on the first row.
desc	string	No	-	The additional information on the second row. When the name property does not exist, this property is not displayed. When the value of size is x-small, this property is not displayed.
className	string	No	-	The class name.

Style class

Class name	Description
amd-avatar	The style of the entire avatar.
amd-avatar-src	The image style.
amd-avatar-content	The style of the avatar description.
amd-avatar-name	The name style.
amd-avatar-desc	The desc style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Basic usage - Four sizes">
    <view class="demo-list">
      <view class="list-item" a:for="{{images}}">
        <avatar src="{{item}}" />
      </view>
    </view>
  </demo-block>
  <demo-block title="placeholder avatar">
    <avatar/>
  </demo-block>
  <demo-block title="different size">
    <view class="demo-list">
      <view class="list-item" a:for="{{sizes}}">
        <avatar size="{{item}}" src="{{images[0]}}" />
      </view>
    </view>
  </demo-block>
  <demo-block title="Use with lists" padding="0">
    <list-item>
      <avatar name="Novlee Spicer" desc="Deserunt dolor ea eaque eos" src="{{images[0]}}" />
    </list-item>
    <list-item>
      <avatar desc="When there is only desc, the name is not displayed" src="{{images[0]}}" />
    </list-item>
    <list-item>
      <avatar name="When size=x-small, desc is not displayed" desc="summary information" size="x-small" src="{{images[0]}}" />
    </list-item>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    sizes: ['x-small', 'small', 'medium', 'large'],
    images: [
      'https://images.example.com/photo-1548532928-b34e3be62fc6?ixlib=rb-
1.2.1&q=80&fm=jpg&crop=faces&fit=crop&h=200&w=200&ixid=eyJhcHBfaWQiOjE3Nzg0fQ',
      'https://images.example.com/photo-1493666438817-866a91353ca9?ixlib=rb-
0.3.5&q=80&fm=jpg&crop=faces&fit=crop&h=200&w=200&s=b616b2c5b373a80ffc9636ba24f7a4a9',
      'https://images.example.com/photo-1542624937-8d1e9f53c1b9?ixlib=rb-
1.2.1&q=80&fm=jpg&crop=faces&fit=crop&h=200&w=200&ixid=eyJhcHBfaWQiOjE3Nzg0fQ',
      'https://images.example.com/photo-1546967191-fdfb13ed6b1e?ixlib=rb-
1.2.1&q=80&fm=jpg&crop=faces&fit=crop&h=200&w=200&ixid=eyJhcHBfaWQiOjE3Nzg0fQ',
    ],
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo-list {
  display: flex;
  align-items: flex-start;
}
.list-item {
  margin-right: 24rpx;
  text-align: left;
}

.list-item .size-text {
  padding-top: 12rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Avatar",
  "usingComponents": {
    "avatar": "antd-mini/es/Avatar/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.4.2. Collapse

A collapse component is a content area that can be collapsed or expanded. It can be used to group or hide complex areas to keep the page clean.

> **Important**
> - Accordin is a special kind of Collapse, which allows only one panel to be expanded at a time.
> - The Collapse component must share the same unique `uid` with the CollapseItem component.

Attribute

Collapse

Attribute	Type	Required	Default value	Description
name	string[]	No	[]	The index that is currently activated.
accordion	boolean	No	-	Whether the component is in accordin mode, in which only one panel is expanded.
uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Collapse components. The uid must the same as that of the internal CollapseItem component.
className	string	No	-	The class name.

CollapseItem

Attribute	Type	Required	Default value	Description
title	string	No	-	The title.
name	string	Yes	-	The name, which must be unique.

uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Collapse components. The uid must the same as that of the external CollapseItem component.
className	string	No	-	The class name.

Event

Collapse

Event name	Description	Type
onChange	This event obtains the panel that is currently being expanded when expanding or collapsing panels.	`( value : string[]) => void`

Slot

Slot name	Description
title	The title slot of the CollapseItem component. When the title property exists, the slot does not take effect.

Style class

Collapse

Class name	Description
amd-avatar	The entire style of the component.
amd-avatar-src	The image style.
amd-avatar-content	The style of the avatar description.
amd-avatar-name	The name style.
amd-avatar-desc	The desc style.

CollapseItem

Class name	Description
amd-avatar	The entire style of the component.
amd-avatar-src	The image style.
amd-avatar-content	The style of the avatar description.
amd-avatar-name	The name style.
amd-avatar-desc	The desc style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage" padding="0">
    <collapse
      name="{{['item-0']}}"
      onChange="onChange"
      uid="collapse-0"
      accordion="{{false}}">
      <collapse-item
        title="First item"
        uid="collapse-0"
        name="item-0">
        Pariatur dolore commodo commodo elit adipisicing sunt adipisicing ex duis labore nisi sunt. Magna ut minim deserunt. Sunt velit occaeca
t incididunt aliqua. Dolore officia voluptate aute reprehenderit anim excepteur elit.
      </collapse-item>

      <collapse-item
        name="item-1"
        title="Second item"
        uid="collapse-0">
        Dolor reprehenderit cillum aliqua qui id Lorem elit anim do minim mollit. Commodo id cupidatat est tempor anim. Fugiat ipsum dolor nost
rud officia mollit. Aliquip aliqua pariatur tempor excepteur commodo non et adipisicing magna ex nostrud dolore cillum exercitation enim. In s
unt velit laboris ullamco et in reprehenderit sit excepteur aute in dolor. Sunt minim incididunt consectetur laborum sint fugiat voluptate
sunt culpa fugiat duis. Ad consectetur ad aliquip aute labore magna commodo est cupidatat.
      </collapse-item>

      <collapse-item
        title="Third item"
        name="item-2"
        uid="collapse-0">
        Ad ut ullamco exercitation do excepteur ipsum ipsum consectetur nulla fugiat est et. Occaecat ullamco nulla mollit cupidatat dolore nul
la minim cillum proident laboris mollit. Veniam consectetur esse consectetur. Fugiat in laborum anim.
      </collapse-item>

    </collapse>
  </demo-block>
  <demo-block title="Accordion mode"  padding="0">
    <collapse
      name="{{['item-0']}}"
      onChange="onChange"
      uid="collapse-1"
      accordion>
      <collapse-item
        title="First item"
        uid="collapse-1"
        name="item-0">
        Accordion mode can only expand one at the same time
      </collapse-item>

      <collapse-item
        name="item-1"
        title="Second item"
        uid="collapse-1">
        Accordion mode can only expand one at the same time
      </collapse-item>

      <collapse-item
        title="Third item"
        name="item-2"
        uid="collapse-1">
        Accordion mode can only expand one at the same time
      </collapse-item>
    </collapse>
  </demo-block>

</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {

  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Collapse",
  "usingComponents": {
    "collapse": "antd-mini/es/Collapse/index",
    "collapse-item": "antd-mini/es/Collapse/CollapseItem/index",
    "icon": "antd-mini/es/Icon/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Custom mode

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Disabled status" padding="0">
    <collapse
      className="demo-collapse"
      uid="collapse-1"
      accordion="{{false}}">
      <collapse-item
        title="First item"
        uid="collapse-1"
        name="item-0">
        Here is the content of the first item
      </collapse-item>

      <collapse-item
        name="item-1"
        uid="collapse-1"
        disabled="{{true}}"
        title="Second item">
        Here is the content of the second item
      </collapse-item>

      <collapse-item
        name="item-2"
        uid="collapse-1"
        disabled="{{true}}"
        title="Third item">
        Here is the content of the third item
      </collapse-item>
    </collapse>
  </demo-block>
  <demo-block title="Custom icon" padding="0">
    <collapse
      className="demo-collapse"
      uid="collapse-2"
      accordion="{{false}}">
      <collapse-item
        uid="collapse-2"
        name="item-0">
        <view slot="title">
          <icon type="FireFill" size="small"/>
          <text style="color: red; padding-left:12px">title slot</text>
        </view>
        Custom icon
      </collapse-item>

      <collapse-item
        name="item-1"
        uid="collapse-2"
        expandIcon="AddOutline"
        closeIcon="MinusOutline"
        brief="Auxiliary information">
        <view slot="title">
          <text style="color: red; padding-right:8px">title slot</text>
          <icon type="FireFill" size="small"/>
        </view>
        Custom icon
      </collapse-item>
    </collapse>
  </demo-block>
  <demo-block title="Controlled mode" padding="0">
    <collapse
      className="demo-collapse"
      name="{{name}}"
      onChange="handleChange"
      uid="collapse-0"
      accordion="{{false}}">
      <collapse-item
        title="Adaptive height"
        uid="collapse-0"
        name="item-0">
        <view class="item-content">
          <view>Content area</view>
        </view>
      </collapse-item>

      <collapse-item
        name="item-1"
        uid="collapse-0">
        <view slot="title">
          <text style="color: red;">title slot</text>
        </view>
        <view class="item-content content2">
          <view>Content area</view>
```

```
        </view>
      </collapse-item>

      <collapse-item
        title="title"
        name="item-2"
        uid="collapse-0">
        <view class="item-content content3">
          <view>Content area</view>
        </view>
      </collapse-item>

      <collapse-item
        title="title"
        name="item-3"
        uid="collapse-0">
        <view class="item-content content3">
          <view>Content area</view>
        </view>
      </collapse-item>
    </collapse>
    <button onTap="handleControl">Randomly expand an item</button>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    name: ['item-1'],
  },
  handleChange(e) {
    this.setData({ name: e });
    console.log(e);
  },
  handleControl() {
    const getRandom = () => {
      const random = Math.random();
      return random < 0.25 ? 0 : random < 0.5 ? 1 : random < 0.75 ? 2 : 3;
    };
    const { name } = this.data;
    let newName = [];
    if (name.length === 1) {
      let randonIndex;
      // eslint-disable-next-line no-constant-condition
      while (true) {
        randonIndex = getRandom();
        if (randonIndex !== Number(name[0].substring(5))) {
          break;
        }
      }
      newName = [`item-${randonIndex}`];
    } else {
      newName = [`item-${getRandom()}`];
    }
    this.setData({
      name: newName,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.item-content {
  font-size: 34rpx;
  color: #333;
  line-height: 48rpx;
  background-color: white;
}
.amd-icon {
  font-size: 22px;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Collapse: Custom",
  "usingComponents": {
    "collapse": "antd-mini/es/Collapse/index",
    "collapse-item": "antd-mini/es/Collapse/CollapseItem/index",
    "icon": "antd-mini/es/Icon/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.4.3. Container

A universal card container can contain a variety of content, such as texts, lists, pictures, paragraphs, to make it easier for users to browse.

Attribute

Attribute	Type	Required	Default value	Description
title	string	No	-	The title.
image	string	No	-	The thumbnail URL.
icon	string	No	-	The icon on the right.
className	string	No	-	The class name.

Event

Event name	Description	Type
onIconTap	Callback fired when a user taps the icon in the upper right corner.	`() => void`

Slot

Slot name	Description
title	The title slot of the component. When the title property exists, the slot does not take effect.

Style class

Class name	Description
amd-container	The style of the entire container.
amd-container-header	The entire style of the container header.
amd-container-header-image	The image style of the container header.
amd-container-header-title	The image style of the container header.
amd-container-header-icon	The icon style of the container header.
amd-container-content	The style of the container content.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo-container">
  <container
    title="Basic usage"
    image="{{imageUrl}}"
    icon="SetOutline"
    onIconTap="onIconTap"
  >
    <view class="demo-container-container">
      This is container custom content
    </view>
  </container>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    imageUrl: 'https://gw.alipayobjects.com/mdn/rms_226d75/afts/img/A*06fDQa9nxDkAAAAAAAAAAAAAARQnAQ'
  },
  onIconTap() {
    my.alert({
      title: 'icon onTap',
      content: 'You clicked the top right icon! '
    })
  }
});
```

The following shows an example of the code in the index.acss file:

```
.demo-container {
  padding-top: 24rpx;
}

.demo-container-container {
  padding: 24rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Container",
  "usingComponents": {
    "container": "antd-mini/es/Container/index"
  }
}
```

1.10.4.4. FloatPanel

A FloatPanel component is a content panel for users to flexibly scroll and browse content.

- The initial height of the panel is 18% of the default window height. When you scroll up, the height of the panel is raised to 35% and if you continue to scroll up, the height reaches 95% of the default window height.
- When you scroll down, the height is changed back to 35% and then to 18% if you scroll down further.
- You can scroll the panel content area when the panel height reaches the highest.
- You must use a basic library version later than 2.7.7.

Attribute

Attribute	Type	Required	Default value	Description
minHeight	number	No	0.18	The minimum height of the panel. The unit is the percentage of the page height.
middleHeight	number	No	0.35	The second highest height of the panel. The unit is the percentage of the page height.
maxHeight	number	No	0.9	The maximum height of the panel. The unit is the percentage of the page height.
lowerThreshold	number	No	50	The distance to the bottom of the content area that can fire the callback of onContentToBottom.
withMask	boolean	No	true	Whether to enable the mask layer. The default value is no.
className	string	No	-	The root node of the component.

Event

Event name	Description	Type
onContentToBottom	Scrolls down to bottom of the content area to load data.	`() => void`

Slot

Name	Description
header	The header slot.

content	The slot for swiping.
footer	The footer slot.

Style class

Class name	Description
amd-swiper-box	The root node of the component.
amd-swiper-arrow-wrapper	The indicator style.
amd-swiper-header	The style of the header area.
amd-swiper-scroll-view	The scroll-view style of the content area.
amd-swiper-footer	The style of the footer area.
amd-swiper-background	The style of the mask layer.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="map">

</view>
<view class="buttonWrapper">
 <switch class="button" onChange="handleToggleMask" inlineSize="small" inline />
 <text>mask</text>
</view>

<float-panel
className="wrapper"
maxHeight="{{0.9}}"
withMask="{{withMask}}"
>
 <view slot="header" class="title">
  <text>
   Header title
  </text>
 </view>
 <view class="content" slot="content">
  <list radius="{{false}}">
   <list-item class="noLine" a:for="{{isvList1}}">{{index}}</list-item>
  </list>
 </view>
 <view slot="footer" class="footer">
  Footer content
 </view>
</float-panel>
```

The following shows an example of the code in the index.js file:

```
Page({
    data: {
        isvList1: new Array(20).fill(0),
        withMask: false,
        button1Text: 'Close the mask',
    },
    handleToggleMask () {
        this.setData({
            button1Text: !this.data.withMask ? 'Close the mask' : 'Open the mask',
            withMask: !this.data.withMask
        })
    }
})
```

The following shows an example of the code in the index.acss file:

```
.title {
    background: #FFF;
    border-radius: 16rpx 16rpx 0 0;
    font-family: PingFangSC-Medium;
    font-size: 36rpx;
    color: #333;
    text-align: center;
    border-bottom: 1rpx solid #EEE;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 98rpx;
    box-sizing: border-box;
}

.content {
    background: #FFF;
    display: flex;
    flex-direction: column;
    justify-content: center;
    padding: 24rpx;
}

.item {
    width: 100%;
    height: 160rpx;
    border-bottom: 1px solid grey;
    display: flex;
}
.item .left {
    width: 200rpx;
    display: flex;
    align-items: center;
}

.item .right {
  flex: 1;
  text-align: right;
  display: flex;
  align-items: center;
  justify-content: flex-end;
}

.footer {
    padding-bottom: constant(safe-area-inset-bottom);
    padding-bottom: env(safe-area-inset-bottom);
    border-radius: 0;
    height: 128rpx;
    background: #FFF;
    display: flex;
    align-items: center;
    justify-content: center;
    box-sizing: border-box;
}

.wrapper {
}

.wrapper .amd-swiper-section {
  background-color: #fff;
}

.buttonWrapper {
    display: flex;
    position: fixed;
    align-items: center;
    justify-content: center;
    top: 0;
    z-index: 9999;
}

.button {
    margin: 20rpx;
}

.map {
    width: 100vw;
    height: 100vh;
    background-image: url("https://gw.alipayobjects.com/mdn/rms_186a6d/afts/img/A*Z1x_QYGRR1kAAAAAAAAAAAAAAARQnAQ");
}
```

The following shows an example of the code in the index.json file:

```
{
    "usingComponents":{
        "float-panel": "antd-mini/es/FloatPanel/index",
        "list": "antd-mini/es/List/index",
        "list-item": "antd-mini/es/List/ListItem/index",
        "switch": "antd-mini/es/Switch/index"
    },
    "allowsBounceVertical": "NO"
}
```

Event listening

The following shows an example of the code in the index.axml file:

```
<view class="map" />

<float-panel
    className="wrapper"
    maxHeight="{{0.9}}"
    onScroll="handleScrolllStatus"
    onContentToBottom="handleContentScrollToLower"
    ref="saveRef"
>
    <view slot="header" class="title">
        <text>
            The head area, through the onScroll event callback to monitor the current height of the panel is<text style="color: red">{{pos1}}</
text>
        </text>
    </view>
    <view class="content" slot="content">
        <list radius="{{false}}">
            <list-item class="noLine" a:for="{{isvList2}}">{{index}}</list-item>
        </list>
        <loading text="Loading" color="#1677ff" a:if="{{showLoading}}" />
    </view>
    <view slot="footer" class="footer">
        Bottom content
    </view>
</float-panel>
```

The following shows an example of the code in the index.js file:

```
Page({
    data: {
        isvList2: new Array(10).fill(0),
        pos1: 'minimum height',
        showLoading: true
    },
    handleContentScrollToLower () {
        setTimeout(() => {
            this.setData({
                isvList2: new Array(20).fill(0),
                showLoading: false
            })
        }, 2000)
    },
    handleScrolllStatus (pos) {
        this.setData({ pos1: pos === 'MAX' ? 'maximum height' : pos === 'MIDDLE' ? 'second maximum height' : 'minimum height' })
    },
    saveRef (ref) {
        this.panel = ref
    },
})
```

The following shows an example of the code in the index.acss file:

```
.title {
    background: #FFF;
    border-radius: 16rpx 16rpx 0 0;
    font-family: PingFangSC-Medium;
    font-size: 36rpx;
    color: #333;
    text-align: center;
    border-bottom: 1rpx solid #EEE;
    display: flex;
    align-items: center;
    justify-content: center;
    height: 98rpx;
    box-sizing: border-box;
}

.content {
    background: #FFF;
    display: flex;
    flex-direction: column;
    justify-content: center;
    padding: 24rpx;
}

.item {
    width: 100%;
    height: 160rpx;
    border-bottom: 1px solid grey;
    display: flex;
}
.item .left {
    width: 200rpx;
    display: flex;
    align-items: center;
}

.item .right {
  flex: 1;
  text-align: right;
  display: flex;
  align-items: center;
  justify-content: flex-end;
}

.footer {
    padding-bottom: constant(safe-area-inset-bottom);
    padding-bottom: env(safe-area-inset-bottom);
    border-radius: 0;
    height: 128rpx;
    background: #FFF;
    display: flex;
    align-items: center;
    justify-content: center;
    box-sizing: border-box;
}

.wrapper {
}

.wrapper .amd-swiper-section {
  background-color: #fff;
}

.buttonWrapper {
    display: flex;
    position: fixed;
    align-items: center;
    justify-content: center;
    top: 0;
    z-index: 9999;
}

.button {
    margin: 20rpx;
}

.map {
    width: 100vw;
    height: 100vh;
    background-image: url("https://gw.alipayobjects.com/mdn/rms_186a6d/afts/img/A*Z1x_QYGRR1kAAAAAAAAAAAAAAARQnAQ");
}
```

The following shows an example of the code in the index.json file:

```
{
    "usingComponents":{
        "float-panel": "antd-mini/es/FloatPanel/index",
        "loading": "antd-mini/es/Loading/index",
        "list": "antd-mini/es/List/index",
        "list-item": "antd-mini/es/List/ListItem/index"
    },
    "allowsBounceVertical": "NO"
}
```

1.10.4.5. List

A List is a generic list that carries text, lists, images, and paragraphs in a clean and efficient manner.

Attribute

List

Attribute	Type	Required	Default value	Description
radius	boolean	No	false	Whether the list contains rounded corners.
header	string	No	-	The description of the header.
footer	string	No	-	The description of the footer.
className	string	No	-	The class name.

ListItem

Attribute	Type	Required	Default value	Description
image	string	No	-	The image on the left.
imageSize	'small' | 'medium' | 'large'	No	-	The image size.
arrow	'right' | 'up' | 'down'	No	-	The direction to which the arrow is pointing. If this property is not passed in, there is no arrow.
title	string	No	-	The title information.
brief	string	No	-	The information on the second row.
extra	string	No	-	Extra content on the right.
extraBrief	string	No	-	The help information on the right.
disabled	boolean	No	false	Whether to disable the component.
last	boolean	No	false	Whether to display the underline.
className	string	No	-	Class name

Slot

Name	Description
header	The content slot of the header.
footer	The content slot of the footer.

Name	Description
brief	The introduction content slot below.

extra	The content slot on the right.
image	The content slot on the left.

Event

Event name	Description	Type
onTap	Callback fired when the icon is tapped.	`( e: Event ) => void`

Style class

Class name	Description
amd-list	The style of the entire list.
amd-list-header	The style of the header.
amd-list-body	The internal content style.
amd-list-footer	The internal content style.

Class name	Description
amd-list-item	The style of an entire list item.
amd-list-item-line	The content style.
amd-list-item-content	The content style except for extra and brief.
amd-list-item-content-main	The main content style.
amd-list-item-image	The style of the image on the left.
amd-list-item-brief	The brief style.
amd-list-item-extra	The extra style.
amd-list-item-arrow	The style of the arrow on the right.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <list header="Basic usage"  radius="{{radius}}">
    <list-item>1</list-item>
    <list-item>2</list-item>
    <list-item>3</list-item>
  </list>
  <list header="Clickable list"  radius="{{radius}}">
    <list-item image="UnorderedListOutline" arrow="right" onTap="handleTap" data-info="Bill">Bill</list-item>
    <list-item image="PayCircleOutline" arrow="right" onTap="handleTap" data-info="Total assets">Total assets</list-item>
    <list-item image="SetOutline" arrow="right" onTap="handleTap" data-info="Settings">Settings</list-item>
  </list>
  <list
    radius="{{radius}}"
    header="Complex layout">
    <list-item>
      Rounded corners
      <switch slot="extra" checked="{{radius}}" onChange="handleSetRadius"/>
    </list-item>
    <list-item
      extraBrief="not enabled"
      arrow="right">
        Large font size mode
    </list-item>
    <list-item
      brief="Manage authorized products and devices"
      arrow="{{item.arrow}}">
      Authorized management
    </list-item>
    <list-item
      title="Title"
      brief="Description information"
      image="AlipaySquareFill"
      extra="Secondary information"
      imageSize="large"
      extraBrief="Secondary auxiliary information"
      arrow="right">
      Three line list
    </list-item>
  </list>
  <list
    radius="{{radius}}"
    header="Disabled status">
    <list-item disabled image="UnorderedListOutline" arrow="right" data-info="Bill">Bill</list-item>
    <list-item disabled image="PayCircleOutline" arrow="right" data-info="Total assets">Total assets</list-item>
  </list>
  <white-space />
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    radius: false,
    list: [
      {
        info: 'The first list-item was clicked',
        image:
          'https://gw.alipayobjects.com/mdn/rms_ce4c6f/afts/img/A*XMCgSYx3f50AAAAAAAAAAABkARQnAQ',
        arrow: 'right',
        content: 'The first list-item',
      },
      {
        info: 'The second list-item was clicked',
        image: 'AlipaySquareFill',
        arrow: 'right',
        content: 'The second list-item',
      },
      {
        info: 'The third list-item is clicked',
        image: '',
        arrow: 'right',
        content: 'The third list-item',
      },
    ],
  },
  handleTap(e) {
    my.alert({
      title: 'onTap',
      content: e.currentTarget.dataset.info,
    });
  },
  handleSetRadius(checked) {
    this.setData({
      radius: checked,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.customImage {
  display: flex;
  justify-content: center;
  align-items: flex-end;
  width: 100%;
  height: 100%;
  overflow: hidden;
  font-size: 12rpx;
  color: red;
  background-color: #e5e5e5;
  border-radius: 12rpx;
  box-sizing: border-box;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "List",
  "usingComponents": {
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "white-space": "../../components/WhiteSpace/index",
    "switch": "antd-mini/es/Switch/index"
  }
}
```

1.10.4.6. Steps

A Steps component is a navigation bar that instructs users to complete a task. When a task is complex or needs to be completed in order, you can divide it into a series of steps to simplify the task.

Attribute

Steps

Attribute	Type	Required	Default value	Description
index	number	No	0	The current progress.
direction	'horizontal' | 'vertical'	No	'horizontal'	The direction.

uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Steps components. The uid is the same as that of the internal StepItem component.
className	string	No	-	The class name.

Attribute	Type	Required	Default value	Description
index	number	Yes	-	This is required for a mini-program. It is used to mark the current step number, which must increment in order.
title	string | slot	Yes	-	The title.
desc	string | slot	No	-	Extra information.
fail	boolean	No	false	Whether it is a failed step.
icon	string | slot	No	-	The icon. A default icon exists for both the portrait and landscape orientations.
activeIcon	string | slot	No	-	The icon of an activated step. A default icon exists for both the portrait and landscape orientations.
failIcon	string | slot	No	-	The icon of a failed step. A default icon exists for both the portrait and landscape orientations.
uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Steps components. The uid is the same as that of the external StepItem component.
className	string	No	-	The class name.

Slot

Name	Description
title	The title slot.
desc	The extra content slot.
icon	The default icon slot.
activeIcon	The slot of an activated step icon.
failIcon	The slot of a failed step icon.

Style class

Class name	Description
amd-steps	The style of the entire component.
amd-steps-horizontal	The style of the entire component in horizontal orientation.
amd-steps-vertical	The style of the entire component in vertical orientation.

Class name	Description
amd-steps-item	The style of an entire step.
amd-steps-item-horizontal	The style of a step in horizontal orientation.
amd-steps-item-vertical	The style of a step in vertical orientation.

amd-steps-item-line	The style of a step.
amd-steps-item-line-fail	The style of a failed step.
amd-steps-item-icon	The icon style.
amd-steps-item-text	The style of the text area.
amd-steps-item-title	The title style.
amd-steps-item-desc	The style of the additional description.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Horizontal step bar">
    <steps
      index="{{1}}"
      direction="horizontal"
      uid="steps-1">
      <step-item
        index="{{0}}"
        title="The first step"
        desc="Description"
        uid="steps-1"
      />

      <step-item
        index="{{1}}"
        desc="Description"
        uid="steps-1">
        <view slot="title">The second step</view>
      </step-item>

      <step-item
        index="{{2}}"
        title="The third step"
        uid="steps-1">
        <view slot="desc">Description</view>
      </step-item>
    </steps>
  </demo-block>
  <demo-block title="Horizontal step bar failed">
    <steps
      index="{{2}}"
      uid="steps-2">
      <step-item
        index="{{0}}"
        title="The first step"
        uid="steps-2"/>
      <step-item
        index="{{1}}"
        title="The second step"
        uid="steps-2">
      </step-item>

      <step-item
        fail
        index="{{2}}"
        title="The third step"
        uid="steps-2">
      </step-item>

      <step-item
        index="{{3}}"
        title="The fourth step"
        uid="steps-2">
      </step-item>
    </steps>
  </demo-block>
  <demo-block title="Vertical step bar">
    <steps
      direction="vertical"
      index="{{2}}"
      uid="steps-3">
      <step-item
        index="{{0}}"
        title="The first step"
```

```
      uid="steps-3"/>
    <step-item
      index="{{1}}"
      title="The second step"
      uid="steps-3">
    </step-item>

    <step-item
      index="{{2}}"
      title="The third step"
      uid="steps-3">
    </step-item>

    <step-item
      index="{{3}}"
      title="The fourth step"
      uid="steps-3">
    </step-item>
  </steps>
</demo-block>
<demo-block title="Vertical step bar failed">
  <steps
    direction="vertical"
    index="{{2}}"
    uid="steps-4">
    <step-item
      index="{{0}}"
      title="The first step"
      uid="steps-4"/>
    <step-item
      index="{{1}}"
      title="The second step"
      uid="steps-4">
    </step-item>

    <step-item
      fail
      index="{{2}}"
      title="The third step"
      uid="steps-4">
    </step-item>

    <step-item
      index="{{3}}"
      title="The fourth step"
      uid="steps-4">
    </step-item>
  </steps>
</demo-block>

<demo-block title="Custom icon">
  <steps
    index="{{2}}"
    direction="horizontal"
    uid="steps-5">
    <step-item
      index="{{0}}"
      title="The first step"
      desc="content of desc part"
      uid="steps-5">
        <icon slot="activeIcon" type="FireFill" className="steps-icon"/>
    </step-item>

    <step-item
      index="{{1}}"
      desc="The second step"
      fail="{{true}}"
      uid="steps-5">
      <icon slot="failIcon" type="CloseCircleFill" className="steps-icon"/>
      <view slot="title">title slot</view>
    </step-item>

    <step-item
      index="{{2}}"
      title="The third step"
      uid="steps-5">
      <view slot="desc"> desc slot </view>
      <icon slot="failIcon" class="steps-icon"/>
    </step-item>

    <step-item
      index="{{3}}"
      title="The fourth step"
      uid="steps-5">
      <icon slot="icon" type="AAOutline" className="steps-icon"/>
    </step-item>
  </steps>
```

```
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    activeIndex: 1,
    failIndex: false,
    showNumberSteps: true,
    direction: 'vertical',
    activeIcon1: 'CheckCircleFill',
    activeIcon2:
      'https://gw.alipayobjects.com/mdn/rms_ce4c6f/afts/img/A*XMCgSYx3f50AAAAAAAAAAABkARQnAQ',
  },
  handleNextStep() {
    this.setData({
      activeIndex: this.data.activeIndex + 1,
    });
  },
  handlePreStep() {
    this.setData({
      activeIndex: this.data.activeIndex - 1,
    });
  },
  handleSetFailIndex() {
    this.setData({
      failIndex: !this.data.failIndex,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo-btn-container {
  display: flex;
  justify-content: space-between;
  margin: 20px;
}

.demo-btn {
  width: 47%;
}
.full-btn {
  width: 100%;
}
.steps-icon {
  width: 18px;
  height: 18px;
  font-size: 18px !important;
  background-color: white;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Steps",
  "usingComponents": {
    "steps": "antd-mini/es/Steps/index",
    "step-item": "antd-mini/es/Steps/StepItem/index",
    "icon": "antd-mini/es/Icon/index",
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.4.7. SwipeAction

A SwipeAction component is an extended function of a list. It is used to display hidden functional menus by swiping.

Attribute

Attribute	Type	Required	Default value	Description
autoClose	boolean	No	false	Whether to automatically collapse when the button is tapped.
disabled	boolean	No	false	Whether to disable the component.

left	{ text: string, type: 'default' | 'primary' | 'danger'; className: string } []	No	-	Whether to expand the left operation area when you swipe right.
right	{ text: string, type: 'default' | 'primary' | 'danger'; className: string }[]	No	-	Whether to expand the right operation area when you swipe left.
className	string	No	-	The class name.

Event

Event name	Description	Type	Remarks
onLeftButtonTap	Callback fired when the button on the left is tapped.	`(index: number, text: string, type: string, extraInfo?: unknown, dateSet: Record<string, any>) => void`	The nth button from left to right.
onRightButtonTap	Callback fired when the button on the right is tapped.	`(index: number, text: string, type: string, extraInfo?: unknown, dateSet: Record<string, any>) => void`	The nth button from right to left.

Style class

Class name	Description
amd-swipe-action	The style of the entire component.
amd-swipe-action-closeSwipe	The style applied to a closed component.
amd-swipe-action-wrap	The style of the content wrapping the component.
amd-swipe-action-left	The style of the button area on the left.
amd-swipe-action-right	The style of the button area on the right.
amd-swipe-action-btn	The button style.
amd-swipe-action-btn-text	The button text style.
amd-swipe-action-content	The style of the surface layer.
amd-swipe-action-item	The content style of the surface layer.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <list radius="{{true}}" header="Single swipe action component (disappears when delete)">
    <swipe-action
        a:for="{{singleListDelete}}"
        a:key="id"
        data-index="{{index}}"
        autoClose="{{item.autoClose}}"
        right="{{item.right}}"
        speed="{{item.speed}}"
        extraInfo="{{{sequence:index+1,supportClear:item.supportClear}}}"
        onRightButtonTap="onSingleRightItemClickDelete"
        >
        <list-item
          arrow="right"
          data-index="{{index}}"
          data-content="{{item.content}}"
          onTap="onItemClick">
          {{item.content}}
        </list-item>
    </swipe-action>
  </list>
  <list radius="{{true}}" header="Single swipe action component (disappears when delete)">
    <swipe-action
        a:for="{{singleList}}"
        a:key="id"
        data-index="{{index}}"
        autoClose="{{item.autoClose}}"
        right="{{item.right}}"
        speed="{{item.speed}}"
        extraInfo="{{{sequence:index+1,supportClear:item.supportClear}}}"
        onRightButtonTap="onSingleRightItemClick"
        >
        <list-item
          arrow="right"
          data-index="{{index}}"
          data-content="{{item.content}}"
          onTap="onItemClick">
           {{item.content}}
        </list-item>
    </swipe-action>
  </list>
  <list radius="{{true}}" header="Swipe action components list">
    <swipe-action
      a:for="{{multiList}}"
      a:key="id"
      data-index="{{index}}"
      autoClose="{{item.autoClose}}"
      right="{{item.right}}"
      left="{{item.left}}"
      speed="{{item.speed}}"
      extraInfo="{{{sequence:index+1,supportClear:item.supportClear}}}"
      onRightButtonTap="onMultiRightItemClick"
      onLeftButtonTap="onMultiLeftItemClick">
      <list-item
        arrow="right"
        data-index="{{index}}"
        data-content="{{item.content}}"
        onTap="onItemClick">
          {{item.content}}
      </list-item>
    </swipe-action>
  </list>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    multiList: [{
      left: [
        {
          type: 'default',
          text: 'Add on',
        }, {
          type: 'primary',
          text: 'Cancel favorite',
        },
        {
          type: 'danger',
          text: 'Delete',
        }],
      content: 'Only the left button + delete disappears',
      autoClose: false,
      speed: 100,
      supportClear: true,
      id: 1,
```

```
  }, {
    right: [{
      type: 'default',
      text: 'Add one',
    }, {
      type: 'primary',
      text: 'Cancel favorite',
    }, {
      type: 'danger',
      text: 'Delete',
    }],
    content: 'Only the right button + delete disappears',
    autoClose: false,
    speed: 20,
    supportClear: true,
    id: 2,
  }, {
    right: [{
      type: 'danger',
      text: 'Delete',
    }],
    left: [{
      type: 'primary',
      text: 'Cancel favorite',
    }],
    content: 'Left and right button each + delete disappears',
    autoClose: true,
    speed: 15,
    supportClear: true,
    id: 3,
  }, {
    right: [{
      type: 'primary',
      text: 'Cancel favorite',
    }, {
      type: 'danger',
      text: 'Delete',
    }],
    left: [{
      type: 'primary',
      text: 'Cancel favorite',
    }, {
      type: 'danger',
      text: 'Delete',
    }],
    content: 'Two left and right buttons each + delete does not disappear + does not automatically recover',
    autoClose: false,
    speed: 10,
    supportClear: false,
    id: 4,
  }, {
    right: [
      {
        type: 'default',
        text: 'Do not disturb',
      }, {
        type: 'primary',
        text: 'Cancel favorite',
      }, {
        type: 'danger',
        text: 'Delete',
      },
      {
        type: 'danger',
        text: 'Clear',
      },
    ],
    left: [
      {
        type: 'default',
        text: 'Do not disturb',
      }, {
        type: 'primary',
        text: 'Favorite',
      }, {
        type: 'danger',
        text: 'Delete',
      }],
    content: 'Three left and right buttons each + delete does not disappear + does not automatically recover',
    autoClose: true,
    speed: 1,
    supportClear: false,
    id: 5,
  }],
  singleListDelete: [{
    right: [{
      type: 'default',
```

```
        text: 'Add one',
      }, {
        type: 'primary',
        text: 'Cancel favorite',
      }, {
        type: 'danger',
        text: 'Delete',
      }],
      content: 'Only the right button + delete disappears',
      autoClose: false,
      speed: 20,
      supportClear: true,
      id: 6,
    }],
    singleList: [{
      right: [{
        type: 'default',
        text: 'Add one',
      }, {
        type: 'primary',
        text: 'Cancel favorite',
      }, {
        type: 'danger',
        text: 'Delete',
      }],
      content: 'Only the right button + delete disappears',
      autoClose: false,
      speed: 20,
      supportClear: true,
      id: 7,
    }],
  },
  onItemClick(e) {
    my.alert({
      content: `dada__${e.currentTarget.dataset.content}`,
    });
  },
  onMultiRightItemClick(btnIndex, btnText, btnType, extraInfo) {
    const { sequence, supportClear } = extraInfo;
    my.confirm({
      title: 'Kind tips',
      content: `Confirm${btnText}?`,
      confirmButtonText: btnText,
      cancelButtonText: 'Cancel',
      success: (result) => {
        if (!result.confirm) return;
        if (btnType === 'danger' && sequence && supportClear) {
          const newList = this.data.multiList.filter((item, index) => index !== sequence - 1);
          this.setData({
            multiList: newList,
          });
        }
      },
    });
  },
  onMultiLeftItemClick(btnIndex, btnText, btnType, extraInfo) {
    const { sequence, supportClear } = extraInfo;
    my.confirm({
      title: 'Kind tips',
      content: `Confirm${btnText}?`,
      confirmButtonText: btnText,
      cancelButtonText: 'Cancel',
      success: (result) => {
        if (!result.confirm) return;
        if (btnType === 'danger' && sequence && supportClear) {
          const newList = this.data.multiList.filter((item, index) => index !== sequence - 1);
          this.setData({
            multiList: newList,
          }, () => {
            my.alert({
              title: `${btnText}Succeed`,
            });
          });
        }
      },
    });
  },
  onSingleRightItemClickDelete(btnIndex, btnText, btnType, extraInfo) {
    const { supportClear } = extraInfo;
    my.confirm({
      title: 'Kind tips',
      content: `Confirm${btnText}?`,
      confirmButtonText: btnText,
      cancelButtonText: 'Cancel',
      success: (result) => {
        if (!result.confirm) return;
        if (btnType === 'danger' && supportClear) {
```

```
          this.setData({
            singleListDelete: [],
          }, () => {
            my.alert({
              title: `${btnText}succeed`,
            });
          });
        }
      },
    });
  },
  onSingleRightItemClick(btnIndex, btnText) {
    my.confirm({
      title: 'Kind tips',
      content: `Confirm${btnText}?`,
      confirmButtonText: btnText,
      cancelButtonText: 'Cancel',
      success: (result) => {
        if (!result.confirm) return;
        my.alert({
          title: `${btnText}succeed`,
        });
      },
    });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Swipe-Action",
  "usingComponents": {
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "swipe-action": "antd-mini/es/SwipeAction/index"
  }
}
```

Component instantiation method

The following shows an example of the code in the index.axml file:

```
<demo-block title="Component instance methods">
  <swipe-action
    data-index="{{index}}"
    right="{{[{
        type: 'default',
        text: 'Add one',
      }, {
        type: 'primary',
        text: 'Cancel favorite',
      }, {
        type: 'danger',
        text: 'delete',
      }]}}"
    onRightButtonTap="onSingleRightItemClickDelete"
    onGetRef="getRef"
    >
    <list-item
      arrow="right"
      data-index="{{index}}"
      onTap="onItemClick">
      Component instance methods
    </list-item>
  </swipe-action>
  <button onTap="resetPosition" style="margin-top:24rpx;">
    Control close
  </button>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  getRef(ins) {
    this.reset = ins.setItemPosition;
  },
  resetPosition() {
    this.reset(0);
  },
});
```

The following shows an example of the code in the index.json file:

```
{
    "defaultTitle": "Swipe-Action",
    "usingComponents": {
      "list-item": "antd-mini/es/List/ListItem/index",
      "swipe-action": "antd-mini/es/SwipeAction/index",
      "demo-block": "../../components/DemoBlock/index"
    }
  }
```

1.10.4.8. Tag

A Tag component is used for marking object attributes and dimensions and classifying objects.

Attribute

Attribute	Type	Required	Default value	Description
type	'outline' | 'fill' | 'fill-light'	No	'fill'	Type outline: display outlines. fill: fill in with dark colors. fill-light: fill in with light colors.
color	'primary' | 'success' | 'warn' | 'danger'	No	'primary'	The tag color. Valid values include: primary: blue. success: green. warn: yellow. danger: red.
icon	string	No	-	The icon.
className	string	No	-	The class name.

Slot

Name	Description
icon	The icon slot.

Style class

Class name	Description
amd-tag	The style of the entire tag.
amd-tag-outline	The style of the outline.
amd-tag-fill	The style used when filling in with dark colors.
amd-tag-fill-light	The style used when filling in with light colors.
amd-tag-primary	The basic style.
amd-tag-success	The style of a success tag.
amd-tag-warn	The style of a warning tag.
amd-tag-danger	The style of a danger tag.
amd-tag-icon-container	The style of an icon area.
amd-tag-content	The content style of the default slot.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic">
    <tag>Tag</tag>
  </demo-block>
  <demo-block title="Semantic label">
    <view class="tag-list">
      <tag className="myTag">default</tag>
      <tag className="myTag" color="success">success</tag>
      <tag className="myTag" color="warn">warn</tag>
      <tag className="myTag" color="danger">danger</tag>
    </view>
  </demo-block>
  <demo-block title="Fill mode">
    <view class="tag-list">
      <tag className="myTag" type="fill">fill</tag>
      <tag className="myTag" type="outline">outline</tag>
      <tag className="myTag" type="fill-light">fill-light</tag>
    </view>
  </demo-block>
  <demo-block title="Custom">
    <view class="tag-list">
      <tag className="myTag" type="fill-light" icon="AlipayCircleFill">tag</tag>
      <tag className="myTag" type="fill-light" color="success" icon="AlipayCircleFill">tag</tag>
      <tag className="myTag" type="fill-light" color="warn" icon="AlipayCircleFill">tag</tag>
      <tag className="myTag" type="fill-light" color="danger" icon="AlipayCircleFill">tag</tag>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    image:
      'https://gw.alipayobjects.com/mdn/rms_ce4c6f/afts/img/A*XMCgSYx3f50AAAAAAAAAAABkARQnAQ',
  },
});
```

The following shows an example of the code in the index.acss file:

```
.myTag {
  margin-right: 16rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Tag",
  "usingComponents": {
    "tag": "antd-mini/es/Tag/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5. Information input

1.10.5.1. Checkbox

A Checkbox component offers a group of options from which an end user can select multiple options. You can use this component independently to switch between two states. A checkbox is similar to a switch. The difference between a checkbox and a switch is that turning on or off the switch directly triggers the status change, however, the checkbox is used for marking a status and needs to be used with the submission operation.

Attribute

Attribute	Type	Default value	Description
checked	boolean	false	Whether a checkbox is selected.
disabled	boolean	false	Whether to disable a checkbox.
color	string	false	The color of the selected checkbox, which is the same as the value in the CSS code.
value	string	-	The value carried by a checkbox. This value is used when you submit the native form.
icon	string	-	The custom icon to indicate that a checkbox is not selected. The value can be an Icon or a path to an image.

checkedIcon	string	-	The custom icon to indicate that a checkbox is selected. The value can be an Icon or a path to an image.
disabledIcon	string	-	The custom icon to indicate that a checkbox is in disabled state. The value can be an Icon or a path to an image.
disabledCheckedIcon	string	-	The custom icon to indicate that a checkbox is selected and in disabled state. The value can be an Icon or a path to an image.
id	string	-	The ID of the form element.
name	string	-	The name of the form element.
className	string	-	The class name.

Event

Event name	Description	Type
onChange	Callback fired when the status of the selected checkbox is changed.	`(checked: boolean) => void`

Style class

Class name	Description
amd-checkbox	The style of a checkbox.
amd-checkbox-disabled	The style of a checkbox in disabled state.
amd-checkbox-checked	The style of a selected checkbox.
amd-checkbox-base	The style of an original checkbox.
amd-checkbox-fake	The style of a checkbox that is not selected.
amd-checkbox-fake-custom	The style of a custom icon.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Basic usage" padding="0">
    <list>
      <list-item className="demo-item">
        <label>
          <checkbox/>
        </label>
      </list-item>
      <list-item className="demo-item">
        <label>
          <checkbox />
          <text>Checkbox with description</text>
        </label>
      </list-item>
      <list-item className="demo-item">
        <label>
          <checkbox color="#00b578" checked/>
          <text>Specify the color</text>
        </label>
      </list-item>
    </list>
  </demo-block>
  <demo-block title="Default selected" padding="0">
    <list-item className="demo-item">
      <label>
        <checkbox checked />
        <text>Default selected</text>
      </label>
    </list-item>
  </demo-block>
  <demo-block title="Disabled status" padding="0">
    <list-item className="demo-item">
      <label>
        <checkbox disabled checked/>
        <text>Disabled status</text>
      </label>
    </list-item>
  </demo-block>
  <demo-block title="Custom icon" padding="0">
    <list-item className="demo-item demo-item-icon">
      <label>
        <checkbox icon="SmileOutline" checkedIcon="SmileFill"/>
        <text>Custom icon (Icon)</text>
      </label>
    </list-item>
    <list-item className="demo-item demo-item-image">
      <label>
        <checkbox color="transparent" checked
checkedIcon="https://gw.alipayobjects.com/mdn/rms_ffbcbf/afts/img/A*2oqcRL38fWwAAAAAAAAAAAAAARQnAQ"/>
        <text>Custom icon (picture)</text>
      </label>
    </list-item>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    checked: false,
  },
  handleChange(v) {
    my.showToast({
      content: `The current checkbox is: ${v ? 'selected' : 'unselected'} status. `,
      duration: 1000,
    });
  },
  handleChangeControlledValue() {
    this.setData({ checked: !this.data.checked });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo-item label {
  display: flex;
  align-items: center;
  line-height: 1;
}
label > text {
  padding-left: 12rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "CheckBox",
  "usingComponents": {
    "checkbox": "antd-mini/es/Checkbox/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.5.2. CheckboxGroup

A CheckboxGroup component is used to group together a set of CheckboxItem components.

> **Important**
> - CheckboxItem components must be used together with the CheckboxGroup component. If you want to use CheckboxItem components independently, use Checkbox.
> - The CheckboxGroup component must share the same uid with its inside CheckboxItem components, and the uid must be unique globally.
> - When you use CheckboxGroup as a form component with `Form` / `FormItem`, you must set the `CheckboxGroup` property of `mode` to `form`.

Attribute

CheckboxGroup

Attribute	Description	Type	Default value
value	The value of the CheckboxGroup component. This property determines whether to select checkboxes.	string[]	[]
radius	Whether the component has a rounded corner.	boolean	false
position	The layout of the component.	'horizontal' | 'vertical'	'vertical'
uid	A globally unique uid must be passed in when the page contains multiple CheckboxGroup components. In addition, the uid must be the same as that used by the internal CheckboxItem components.	string	-
header	The description of the header.	string	-
footer	The description of the footer.	string	-
disabled	Whether to disable the component.	boolean	false
mode	The mode.	'normal' | 'form'	'normal'
className	The class name.	string	-

CheckboxItem

Property	Description	Type	Default value
checked	Whether a checkbox is selected.	boolean	false
disabled	Whether to disable a checkbox.	boolean	false
color	The color of a checkbox, which is the same as the value in the CSS code.	string	-
value	The value carried with a checkbox. This value is used when you submit the native form or use the CheckboxGroup component.	string	-
icon	The custom icon to indicate that a checkbox is not selected. The value can be an Icon or a path to an image.	string	-

checkedIcon	The custom icon to indicate that a checkbox is selected. The value can be an Icon or a path to an image.	string	-
disabledIcon	The custom icon to indicate that a checkbox is in disabled state. The value can be an Icon or a path to an image.	string	-
disabledCheckedIcon	The custom icon to indicate that a checkbox is selected and in disabled state. The value can be an Icon or a path to an image.	string	-
uid	A globally unique uid must be passed in when the page contains multiple CheckboxGroup components. In addition, the uid must be the same as that used by the external CheckboxItem components.	string	-
id	The ID of the form element.	string	-
name	The name of the form element.	string	-
className	The class name.	string	-

Event

Event name	Description	Type
onChange	The function is triggered when the selection status is changed.	`(value) => {}`

Slot

Name	Description
header	The content slot of the header.
footer	The content slot of the footer.

Style class

Class name	Description
amd-checkbox-group	The style of the entire CheckboxGroup component.
amd-list-header	The style of the header content.
amd-list-body	The style of the internal content.
amd-list-footer	The style of the footer content.

Class name	Description
amd-checkbox-item	The style of the entire CheckboxItem component.
amd-checkbox	The style of the entire original checkbox.
amd-checkbox-base	The style of the original checkbox.
amd-checkbox-fake	The style of the checkbox that is not selected.
amd-checkbox-checked	The style of the selected checkbox.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<demo-block title="Basic">
  <checkbox-group
    uid="group1"
    onChange="handleChange"
  >
    <checkbox-item
      a:for="{{list}}"
      value="{{item.value}}"
      uid="group1">
        {{item.label}}
    </checkbox-item>
  </checkbox-group>
</demo-block>


<demo-block title="Horizontal layout">
  <checkbox-group
    uid="group4"
    position="horizontal"
    onChange="handleChange"
  >
  <checkbox-item
    a:for="{{list}}"
    value="{{item.value}}"
    uid="group4">
      {{item.label}}
  </checkbox-item>
  </checkbox-group>
</demo-block>

<demo-block title="Part disabled">
  <checkbox-group
    uid="group2"
    value="{{['orange','banner']}}"
  >
    <checkbox-item
      a:for="{{list}}"
      value="{{item.value}}"
      disabled="{{index===1}}"
      uid="group2">
        {{item.label}}
    </checkbox-item>
  </checkbox-group>
</demo-block>

<demo-block title="Whole group disabled">
  <checkbox-group
    uid="group3"
    value="{{['orange','banner']}}"
    disabled="{{true}}">

    <checkbox-item
      a:for="{{list}}"
      value="{{item.value}}"
      uid="group3">
        {{item.label}}
    </checkbox-item>
  </checkbox-group>
</demo-block>

<demo-block title="Custom icon">
  <checkbox-group
    uid="group5"
    onChange="handleChange"
  >
    <checkbox-item
      a:for="{{list}}"
      value="{{item.value}}"
      icon="SmileOutline" checkedIcon="SmileFill"
      uid="group5">
        {{item.label}}
    </checkbox-item>

  </checkbox-group>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: ['orange'],
    list: [
      { value: 'apple', label: 'apple' },
      { value: 'orange', label: 'orange' },
      { value: 'banana', label: 'banana' },
    ],
  },

  handleChange(value) {
    console.log('onChange', value);
  },
});
```

The following shows an example of the code in the index.acss file:

```
.btns {
  display: flex;
  padding: 0 24rpx 24rpx;
  justify-content: space-between;
}

.btns button {
  flex: 1;
  margin-right: 12rpx;
}

.btns button~button {
  margin-right: 0;
  margin-left: 12rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "CheckBoxGroup",
  "usingComponents": {
    "demo-block": "../../components/DemoBlock/index",
    "checkbox-group": "antd-mini/es/CheckboxGroup/index",
    "checkbox-item": "antd-mini/es/CheckboxGroup/CheckboxItem/index"
  }
}
```

1.10.5.3. Checklist

A Checklist component shows a checklist,

- from which an end user can select one or multiple items.
- For a selectable checklist, at least one item must be selected so that an end user can know that this checklist can be selected.

Attribute

Attribute	Description	Type	Required	Default value
value	The selected data.	string | number (string |)[]	No	-
options	ChecklistItem for configuring options in each column	ChecklistItem[]	No	[]
multiple	Whether to support multiple selections.	boolean	No	false
className	The class name.	string	No	-

ChecklistItem

Attribute	Description	Type	Required	Default value
title	The title.	string	Yes	-
value	The value.	string | value	Yes	-
image	The image.	string	No	-
description	The description.	string	No	-

disabled	Whether to disable the component.	boolean	Class name	false
readOnly	Whether the checklist is read-only.	boolean	The class name.	false

Event

Event name	Description	Type
onChange	Callback fired when the selected item is changed.	`(value:string|number|[], column: ChecklistItem)=>void`

Slot

Name	Description	Type
content	The custom style of CheckListItem.	The slot-scope for receiving selected item parameters.
icon	The custom selected icon.	-

Style class

Class name	Description
amd-checklist	The style of a selectable checklist.
amd-checklist-item-content	The content style of a selectable checklist.
amd-checklist-item-text	The text style of a selectable checklist.
amd-checklist-item-image	The image style of a selectable checklist.
amd-checklist-item-text-description	The style of the text description of a selectable checklist.
amd-checklist-item-check-icon	The style of a selected icon of a selectable checklist.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<demo-block title="Simple layout - single choice" padding="0">
  <checklist
    value="{{1}}"
    options="{{options_1}}"
    onChange="onChange"
  />
</demo-block>

<demo-block title="Complex layout - multiple choice"  padding="0">
  <checklist
    value="{{value}}"
    options="{{options_2}}"
    multiple
    onChange="onChange"
  />
</demo-block>
<demo-block title="Disabled status" padding="0">
  <checklist
    value="{{[2]}}"
    options="{{options_3}}"
    multiple
    onChange="onChange"
  />
</demo-block>

<demo-block title="Read-only status" padding="0">
  <checklist
    value="{{[2]}}"
    options="{{options_4}}"
    multiple
    onChange="onChange"
  />
</demo-block>
<demo-block title="Custom check icon && component content" padding="0">
  <checklist
    value="{{[2]}}"
    options="{{options_3}}"
    multiple
    onChange="onChange"
  >
    <view slot="icon">
      <icon color='red' type="LikeOutline" size="x-small" class="demo-checklist-checked-icon"/>
    </view>
    <view slot="content" slot-scope="props">
      title: {{props.item.title}}
    </view>
  </checklist>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: [1, 2],
    options_1: [
      {
        value: 1,
        title: 'Tickable list item 1'
      },
      {
        value: 2,
        title: 'Tickable list item 2'
      },
      {
        value: 3,
        title: 'Tickable list item 3'
      }
    ],
    options_2: [
      {
        value: 1,
        image: 'https://gw.alipayobjects.com/mdn/rms_226d75/afts/img/A*5m0ZQYhxhjEAAAAAAAAAAAAAARQnAQ',
        description: "Here is the description information",
        title: 'Tickable list item 1'
      },
      {
        value: 2,
        image: 'https://gw.alipayobjects.com/mdn/rms_226d75/afts/img/A*5m0ZQYhxhjEAAAAAAAAAAAAAARQnAQ',
        description: "Here is the description information",
        title: 'Tickable list item 2'
      },
      {
        value: 3,
        image: 'https://gw.alipayobjects.com/mdn/rms_226d75/afts/img/A*5m0ZQYhxhjEAAAAAAAAAAAAAARQnAQ',
        description: "Here is the description information",
        title: 'Tickable list item 3'
      }
    ],
    options_3: [
      {
        value: 1,
        title: 'Tickable list item 1'
      },
      {
        value: 2,
        title: 'Disabled list item 2',
        disabled: true
      },
      {
        value: 3,
        title: 'Tickable list item 3'
      }
    ],
    options_4: [
      {
        value: 1,
        title: 'Tickable list item 1'
      },
      {
        value: 2,
        title: 'Read-only list item 2',
        readOnly: true
      },
      {
        value: 3,
        title: 'Tickable list item 2'
      }
    ]
  },
  onChange(v) {
    console.log('The current selected value is: ', v)
  }
})
```

The following shows an example of the code in the index.acss file:

```
.demo-checklist-label {
  font-size: 32rpx;
  color: #999;
  padding: 36rpx 0 16rpx 16rpx;
}

.demo-checklist-checked-icon {
  font-size: 36rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Checklist",
  "usingComponents": {
    "checklist": "antd-mini/es/Checklist/index",
    "icon": "antd-mini/es/Icon/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.4. Filter

A Filter is a menu panel that pops up downward. It is used to filter, sort, and change the scope or order of content display on the current page. It must be used with a FilterItem component.

Attribute

Filter

Attribute	Type	Required	Default value	Description
uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Filters. The uid must be the same as that of the internal FilterItem component.
className	string	No	-	The class name.

FilterItem

Property	Type	Required	Default value	Description
type	'default' | 'multiple'	No	'default'	The type. default: single selection. multiple: multiple selections.
value	any	No	-	The value of each item. Only the controlled mode of this component is supported.
items	{value: string; text: string; subText: string}[]	No	-	This property takes effect when type is set to default or multiple.
placeholder	string	No	-	The text that is displayed when the value of an item is empty.
uid	string	No	-	A globally unique uid must be passed in when the page contains multiple Filters. The uid must be the same as that of the external FilterItem component.
className	string	No	-	The class name.

Event

FilterItem

Event name	Description	Type
onChange	Callback fired when the selected item is changed.	`(changedFields: Record<string, any>, allFields: Record<string, any>) => void`
onOpen	Callback fired when the panel for selection is opened.	`() => void`

Style class

Class name	Description
amd-filter	The style of the entire filter.
amd-filter-bar	The style of the tab bar.
amd-filter-bar-text	The style of the title of the tab bar.

amd-filter-bar-text-icon	The icon style of the tab bar.
amd-filter-items	The style of the panel for selection.

Class name	Description
amd-filter-item	The entire style.
amd-filter-item-content	The content style.
amd-filter-item-content-wrap	The style of the panel area for selection.
amd-filter-item-btns	The style of the button area.
amd-filter-item-btns-button	The style of the reset/confirmation button.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Two filters" padding="0">
    <filter
      uid="filter-1"
      className="filter-1">
      <filter-item
        uid="filter-1"
        placeholder="Filter item one"
        value="3"
        items="{{items}}"
        type="default"
        ref="ref"
        onChange="changeSelect"
        onOpen="onOpen"/>

      <filter-item
        uid="filter-1"
        placeholder="Filter item two"
        value="{{value}}"
        items="{{items1}}"
        type="multiple"
        onChange="changeSelect"
        onOpen="onOpen"
        />
    </filter>
  </demo-block>
  <demo-block title="Three filters" padding="0">
    <filter
      uid="filter-2"
      className="filter-2">
      <filter-item
        uid="filter-2"
        placeholder="Filter item one"
        value="3"
        items="{{items}}"
        type="default"
        ref="ref"
        onChange="changeSelect"
        onOpen="onOpen"/>

      <filter-item
        uid="filter-2"
        placeholder="Filter item two"
        value="{{value}}"
        items="{{items1}}"
        type="multiple"
        onChange="changeSelect"
        onOpen="onOpen"
        />

      <filter-item
        uid="filter-2"
        placeholder="The Filter item is too long"
        value=""
        items="{{items}}"
        onChange="changeSelect"
        onOpen="onOpen"/>
```

```
  </filter>
</demo-block>
<demo-block title="Four filters" padding="0">
  <filter
    uid="filter-3"
    className="filter-3">
    <filter-item
      uid="filter-3"
      placeholder="Filter item one"
      value="3"
      items="{{items}}"
      type="default"
      ref="ref"
      onChange="changeSelect"
      onOpen="onOpen"/>

    <filter-item
      uid="filter-3"
      placeholder="Filter item two"
      value="{{value}}"
      items="{{items1}}"
      type="multiple"
      onChange="changeSelect"
      onOpen="onOpen"
      />

      <filter-item
      uid="filter-3"
      placeholder="Filter item three"
      value="3"
      items="{{items}}"
      type="default"
      ref="ref"
      onChange="changeSelect"
      onOpen="onOpen"/>

    <filter-item
      uid="filter-3"
      placeholder="Filter item four"
      value="{{value}}"
      items="{{items1}}"
      type="multiple"
      onChange="changeSelect"
      onOpen="onOpen"
      />

  </filter>
</demo-block>
<demo-block title="Slot: title and icon" padding="0">
  <filter
    uid="filter-4"
    className="filter-4">

    <view
      slot-scope="item"
      slot="title">
      {{item.title === 'Filter item one' ? 'Filter item 1' : 'Filter item 2'}}
    </view>

    <view slot="icon" slot-scope="item">
      <icon
        type="DownOutline"
        className="amd-filter-bar-text-icon {{item.active ? 'amd-filter-bar-text-icon-up' : ''}}" />
    </view>

    <filter-item
      uid="filter-4"
      placeholder="Filter item one"
      value="3"
      items="{{items}}"
      type="default"
      ref="ref"
      onChange="changeSelect"
      onOpen="onOpen"/>

    <filter-item
      uid="filter-4"
      placeholder="Filter item two"
      value="{{value}}"
      items="{{items1}}"
      type="multiple"
      onChange="changeSelect"
      onOpen="onOpen"
      />

  </filter>
</demo-block>
```

```
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: ['3', '5'],
    items: [
      {
        text: 'option one',
        value: '1',
      },
      {
        text: 'option two',
        value: '2',
      },
      {
        text: 'option three',
        value: '3',
      },
      {
        text: 'option four',
        value: '4',
      },
      {
        text: 'option five',
        value: '5',
      },
    ],
    items1: new Array(100).fill(0).map((_, idx) => {
      return {
        text: `option${idx + 1}`,
        value: `${idx + 1}`,
      };
    }),
  },
  changeSelect(v) {
    if (v.length > 0) {
      my.alert({
        content: `Currently selected ${v}`,
      });
    } else {
      my.showToast({
        content: 'None selected',
      });
    }
  },
  formatValue(fv) {
    return `${fv}`;
  },
  onOpen() {
    my.alert({
      title: 'tab opened',
    });
  },
  onTap() {
    this.ins.changeSelect('1');
  },
  ref(ins) {
    this.ins = ins;
  },
});
```

The following shows an example of the code in the index.acss file:

```
.filter-1 .amd-filter-item {
  z-index: 2;
}

.filter-2 .amd-filter-item {
  z-index: 2;
}

.filter-3 .amd-filter-item {
  z-index: 2;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Filter",
  "usingComponents": {
    "filter": "antd-mini/es/Filter/index",
    "filter-item": "antd-mini/es/Filter/FilterItem/index",
    "icon": "antd-mini/es/Icon/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.5. Input

Entering content in an input box through the keyboard is the most basic form of field packaging. Generally, an input box is used to collect information on a form page. Two types of input boxes are provided: text box and selection box.

When you use Input as a form with a Form/FormItem component, you must set mode to form.

Attribute

Attribute	Type	Required	Default value	Description
label	string | slot	No	-	The label.
controlled	boolean	No	false	Whether the input box is controlled.
type	'text' | 'number' | 'idcard' | 'digit' | 'numberpad' | 'digitpad' | 'idcardpad'	No	'text'	The input box type.
password	boolean	No	false	Whether the entered content is of password type.
placeholder	string	No	-	The placeholder.
placeholderClass	string	No	-	The style class of the placeholder.
placeholderStyle	string	No	-	The style of the placeholder. You can set intervals.
maxLength	number	No	140	The maximum length.
confirmType	'done' | 'go' | 'next' | 'search' | 'send'	No	"done"	Sets the text on the button on the right corner of the keyboard. Valid values include done (display "done"); go (display "go"), next (display "next"), search (display "next"), and send (display "send"). The text displayed varies depending on the platform. **Important** This property takes effect only when type is set to text.
confirmHold	boolean	No	false	Whether to keep the keyboard in a non collapsed state when tapping the button in the lower right corner of the keyboard.
cursor	number	No	-	The position of the cursor when the focus is set.
selectionStart	number	No	-1	The cursor start position of the selected text. This property must be used together with the selection-end property.
selectionEnd	number	No	-1	The cursor end position of the started text. This property must be used together with the selection-start property.
randomNumber	boolean	No	false	Whether the numeric keypad is arranged randomly when type is set to number, digit, or idcard.
enableNative	boolean	No	-	Whether to enable Native rendering.
layer	'horizontal' | 'vertical'	No	'horizontal'	The orientation of the input box.
inputCls	string	No	-	The style class of the input box.

labelCls	string	No	-	The style class of the label area.
value	string	No	-	The value of the input box.
clear	boolean	No	true	Whether to display the clear icon.
autoFocus	boolean	No	false	Whether to enable auto-focus. This property may not take effect on iOS devices.
ref	React.Ref	No	-	The instance for operating the form. Currently, two methods focus and blur are provided.
id	string	No	-	The ID of the form element.
name	string	No	-	The name of the form element.
disabled	boolean	No	false	Whether to disable the component.
mode	'noraml' | 'form'	No	normal	When you use this component with a From/FormItem component, you must set this property to form.
className	string	No	-	The class name.

Event

Event name	Description	Type
onConfirm	Callback fired when the confirm event of the keyboard is received.	`(v: string) => void`
onClear	Callback fired when the entered content is cleared.	`(v: string) => void`
onFocus	Callback fired when the focus is set on the input box.	`(v: string) => void`
onBlur	Callback fired when the input box loses the focus.	`(v: string) => void`
onChange	Callback fired when a user enters content.	`(v: string) => void`

Style class

Class name	Description
amd-input-item	The entire style.
amd-input-item-line	The entire style.
amd-input-item-layer	The style of the content area on the left.
amd-input-item-layer-vertical	The style of the content area on the left.
amd-input-item-layer-normal	The style of the content area on the left.
amd-input-item-label	The style of the label.
amd-input-item-content	The style of the input component.
amd-input-item-clear	The style of the clear icon area.
amd-input-item-clear-icon	The style of the clear icon.
amd-input-item-extra	The style of the extra area.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <input-item placeholder="Please enter the content" clear="{{false}}" type="text" onChange="handleItemChange" onFocus="handleItemFocus"
onBlur="handleItemBlur" onConfirm="handleItemConfirm" >
    </input-item>
  </demo-block>
  <demo-block title="with clear button">
    <input-item placeholder="Please enter the content" clear="{{true}}" type="text" onChange="handleItemChange" onClear="handleItemClear">
    </input-item>
  </demo-block>
  <demo-block title="Disabled status">
    <input-item placeholder="Disabled input box" disabled="{{true}}" clear="{{true}}" type="text" onChange="handleItemChange"
onClear="handleItemClear">
    </input-item>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
  },
  handleItemChange(e) {
    // eslint-disable-next-line no-console
    console.log('onItemChange:', e);
  },
  handleItemFocus(v) {
    // eslint-disable-next-line no-console
    console.log('focus:', v);
  },
  handleItemBlur(v) {
    // eslint-disable-next-line no-console
    console.log('blur:', v);
  },
  handleItemConfirm(v) {
    // eslint-disable-next-line no-console
    console.log('confirm:', v);
  },
  handleItemClear() {
    // eslint-disable-next-line no-console
    console.log('onItemClear');
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demoList {
  margin-top: 12px;
  margin-bottom: 12px;
  padding: 0 12px
}
.amd-input-item-title {
  margin-top: 12px;
  font-family: PingFangSC-Regular;
  font-size: 17px;
  color: #666666;
  margin-bottom: 8px;
  text-align: center;
}
.amd-list-item-line {
  position: relative;
  -webkit-box-flex: 1;
  -webkit-flex: 1;
  flex: 1;
  display: -webkit-box;
  display: -webkit-flex;
  display: flex;
  -webkit-box-align: center;
  -webkit-align-items: center;
  align-items: center;
  -webkit-align-self: stretch;
  align-self: stretch;
  max-width: 100%;
  overflow: hidden;
  padding: 0px 0px;
}

.amd-list-input-item-arrow {
  height: 15px;
  width: 7.5px;
}
.amd-list-input-item-phone {
  height: 18px;
  width: 16px;
}

.amd-list-input-item-code {
  font-family: PingFangSC-Regular;
  font-size: 17px;
  color: #1677ff;
  text-align: center;
}
.amd-list-input-item-line {
  color: #EEEEEE;
  margin: 0 12px 0 10px;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "InputItem",
  "usingComponents": {
    "input-item": "antd-mini/es/InputItem/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.6. Picker

A Picker component displays a scrollable list of one or more option sets, offering consistent experiences for iOS and Android users compared with the native Picker component. When there are less than 5 options, we recommend that you directly tile the options and use radio buttons.

Attribute

```
type PickerColumnItem = string | number | {
  label: string
  value: string|number
}
```

Attribute	Type	Required	Description	Default value
value	PickerColumnItem | (PickerColumnItem)[]	No	The selected data.	-
data	PickerColumnItem array	Yes	The picker data for configuring options of each column.	[]

placeholder	string	No	The prompt message.	-
disabled	boolean	No	Whether to disable the component.	false
title	string	No	The title of the popup dialog.	-
okText	string	No	The text on the confirmation button.	'Confirm'
dismissText	string	No	The text on the cancellation button.	'Cancel'
maskStyle	string	No	The style of the mask layer.	-
maskClass	string	No	The class name of the mask layer.	-
indicatorStyle	string	No	The style of the selected item.	-
indicatorClass	string	No	The class name of the selected item.	-
className	string	No	The class name.	-

Event

Event name	Description	Type
onOk	Callback fired when the confirmation button is tapped.	`(value: PickerColumnItem, column: PickerColumnItem ) => void`
onDismiss	Callback fired when the cancellation button is tapped.	`() => void`
onChange	Callback fired when the selected item is changed.	`(value: PickerColumnItem, column: PickerColumnItem) => void`
onFormat	The text display format of the selected value.	`(value: PickerColumnItem, column: PickerColumnItem) => string`
onTriggerPicker	Callback fired when the status of the popup dialog is changed (from hidden to visible or vice versa).	`(visible:boolean) => void`

Slot

Name	Description	Type
default	The tag name of the text area.	The slot-scope, which receives the selected value parameter.
title	The name of the title in the popup dialog.	-

Style class

Class name	Description
amd-picker	The style of the text display area.
amd-picker-placeholder	The placeholder style.
amd-picker-popup	The style of the entire popup dialog.
amd-picker-header	The style of the header area in the popup dialog.
amd-picker-header-item	The text style of the header area in the popup dialog.
amd-picker-content	The style of the selection area.

amd-picker-content-item	The style of a single option in the selection area.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<demo-block title="Basic usage" padding="0">
  <list-item>
    Select city
    <picker
      slot="extra"
      onDismiss="handleCancelPicker"
      onOk="handleOk"
      value="{{value}}"
      placeholder="Please select"
      title="Please select"
      onChange="handleChange"
      data="{{cityList}}">
    </picker>
  </list-item>
</demo-block>

<demo-block title="Object usage"  padding="0">
<list-item>
  Select date
  <picker
    slot="extra"
    onDismiss="handleCancelPicker"
    onOk="handleOk"
    placeholder="Please select"
    title="Please select"
    onChange="handleChange"
    data="{{weekList}}">
  </picker>
</list-item>
</demo-block>

<demo-block title="Multi-column complex type data" padding="0">
<list-item>
  Please select time
  <picker
    slot="extra"
    placeholder="Please select"
    value="{{['Tues', 'pm']}}"
    title="Please select"
    onOk="handleOk"
    onFormat="formatTime"
    data="{{columns}}">
  </picker>
</list-item>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: 'Shanghai',
    cityList: ['Beijing', 'Shanghai', 'Shenzhen', 'Guangzhou', 'Nanjing', 'Wuhan', 'Wuxi', 'Suzhou'],
    weekList: [
      { label: 'Monday', value: 'Mon' },
      { label: 'Tuesday', value: 'Tues' },
      { label: 'Wednesday', value: 'Wed' },
      { label: 'Thursday', value: 'Thur' },
      { label: 'Friday', value: 'Fri' },
    ],
    columns: [
      [
        { label: 'Monday', value: 'Mon' },
        { label: 'Tuesday', value: 'Tues' },
        { label: 'Wednesday', value: 'Wed' },
        { label: 'Thursday', value: 'Thur' },
        { label: 'Friday', value: 'Fri' },
      ],
      [
        { label: 'Morning', value: 'am' },
        { label: 'Afternoon', value: 'pm' },
      ],
    ],
  },

  handleCancelPicker() {
    my.showToast({
      content: 'Cancel the operation and close the picker',
    });
  },

  handleOk(value, column) {
    console.log('onOk value', value, 'onOk  column', column);
  },

  handleChange(value, column) {
    console.log('onChange value', value, 'onChange  column', column);
  },

  formatTime(value, column) {
    return column.map(c => c && c.label).join('')
  },
});
```

The following shows an example of the code in the index.acss file:

```
.pickerTips {
  padding: 24rpx;
}

.pickerTips text {
  color: #1677ff;
}

.pickerItem {
  display: inline-block;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Picker",
  "usingComponents": {
    "picker": "antd-mini/es/Picker/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.7. RadioGroup

A RadioGroup component is a UI component that contains a set of radio buttons and allows an end user to make a single selection from the set. The selected item is specified by a value. All radio buttons are visible by default to make it easy for end users to choose. Therefore, you cannot provide too many radio buttons.

Attribute

RadioGroup

Attribute	Description	Type	Default value

value	The value of RadioGroup, which determines whether a radio button is selected.	string	-
radius	Whether the component has rounded corners.	boolean	false
position	The layout of the radio group.	'horizontal' | 'vertical'	'vertical'
uid	A globally unique uid must be passed in when the page contains multiple RadioGroup components. The uid must be the same as that of the internal RadioItem components.	string	-
header	The description of the header.	string	-
footer	The description of the footer.	string	-
disabled	Whether to disable the component.	boolean	false
className	The class name.	string	-

RadioItem

Attribute	Description	Type	Default value
value	The value of the radio button. This property takes effect when you submit a native form or use RadioGroup.	any	-
color	The selected color, which is the same as the value of the color property in the CSS code.	string	false
disabled	Whether to disable the component.	boolean	false
icon	The custom icon that is not selected. The value can be an Icon or a path to an image.	string	-
checkedIcon	The custom icon that is selected. The value can be an Icon or a path to an image.	string	-
disabledIcon	The custom icon that is disabled. The value can be an Icon or a path to an image.	string	-
disabledCheckedIcon	The custom icon that is selected and in disabled state. The value can be an Icon or a path to an image.	string	-
uid	A globally unique uid must be passed in when the page contains multiple RadioGroup components. The uid must be the same as that of the external RadioItem components.	string	-
className	The class name.	string	-

Event

RadioGroup

Event name	Description	Type
onChange	Callback fired when a radio button is selected.	`(value) => void`

Slot

RadioGroup

Slot name	Description
header	The description of the header.
footer	The description of the footer.

Style class

RadioGroup

Class name	Description
amd-radio-group	The style of the entire radio group.
amd-radio-group-header	The style of the header.
amd-radio-group-body	The style of the radio group body.
amd-radio-group-footer	The style of the footer.

RadioItem

Class name	Description
amd-radio-item-wrap	The style used for wrapping a radio button.
amd-radio-item-base	The style of a radio button.
amd-radio-item-fake	The style of a selected radio button.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <radio-group
      value="option 2"
      uid="basic"
      onChange="handleChange">
      <radio-item a:for="{{list}}" value="{{item.value}}" uid="basic">{{item.label}}</radio-item >
    </radio-group>
  </demo-block>
  <demo-block title="horizontal layout">
    <radio-group
      value="option 2"
      uid="horizontal"
      position="horizontal"
      onChange="handleChange">
      <radio-item a:for="{{list}}" value="{{item.value}}" uid="horizontal">{{item.label}}</radio-item >
    </radio-group>
  </demo-block>
  <demo-block title="contains disabled items">
    <radio-group
      uid="basic1"
      value="banner"
      onChange="handleChange">
      <radio-item a:for="{{list}}" value="{{item.value}}" uid="basic1" disabled="{{index===1}}">{{item.label}}</radio-item >
    </radio-group>
  </demo-block>
  <demo-block title="overall disabled">
    <radio-group
      uid="basic2"
      value="banner"
      disabled>
      <radio-item a:for="{{list}}" value="{{item.value}}" uid="basic2">{{item.label}}</radio-item >
    </radio-group>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    list: [
      { value: 'apple', label: 'apple' },
      { value: 'orange', label: 'orange' },
      { value: 'banana', label: 'banana' },
    ],
  },
  handleChange(e) {
    console.log(e);
  },
});
```

The following shows an example of the code in the index.acss file:

```
.btns {
  display: flex;
  padding: 0 24rpx 24rpx;
  justify-content: space-between;
}

.btns button {
  flex: 1;
  margin-right: 12rpx;
}

.btns button ~button {
  margin-right: 0;
  margin-left: 12rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "RadioGroup",
  "usingComponents": {
    "radio-group": "antd-mini/es/RadioGroup/index",
    "radio-item": "antd-mini/es/RadioGroup/RadioItem/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.8. SearchBar

A SearchBar component is an input box component in a search scenario. It can narrow down the scope of an information pool and easily and quickly obtain the target information. Sometimes, a SearchBar input box may flicker. Therefore, you need to use enableNative to avoid flicking. When you use your own handwriting to enter text into a SearchBar input box on some Android phones, the text will be entered continuously. In this case, set controlled to false.

Attribute

Attribute	Type	Required	Default value	Description
value	string	No	-	The value in the search bar.
autoFocus	boolean	No	false	Whether to enable auto-focus. This property may not take effect on iOS devices.
bizIconType	string	No	'AudioFill'	The icon of auxiliary icon.
cancelText	string	No	"Cancel"	The UI text for the Cancel button.
className	string	No	-	The class name.
controlled	boolean	No	false	Whether the search bar is controlled.
disabled	boolean	No	false	Whether to disable the component.
enableNative	boolean	No	false	Whether to enable Native rendering.
id	string		No	The ID of the form element.
maxLength	number	No	-	The maximum length.
name	string	No	-	The name of the form element.
placeholder	string	No	-	The prompt message.
showBizIcon	boolean	No	false	Whether to display the auxiliary icon.
showCancelButton	boolean	No	false	Whether to display the Cancel button.
showVoice	boolean	No	false	Whether to display the voice icon.

type	'text' | 'number' | 'idcard' | 'digit' | 'numberpad' | 'digitpad' | 'idcardpad'	No	'text'	The type of the search bar.

Event

Event name	Description	Type
onChange	Callback fired when the form is updated.	`(v: any) => void`
onBlur	Callback fired when the search bar loses its focus.	`(v: string) => void`
onBizIconTap	Callback fired when the voice icon is tapped.	`() => void`
onCancel	Callback fired when the Cancel button is tapped.	`(v: string) => void`
onClear	Callback fired when the Delete button is tapped.	`(v: string) => void`
onFocus	Callback fired when the focus is set on the search bar.	`(v: string) => void`
onInput	Callback fired when a user enters text into the input box.	`(v: string) => void`
onSubmit	Callback fired when a user submits text.	`(v: string) => void`
onVoiceTap	Callback fired when the voice icon is tapped.	`() => void`

Style class

Class name	Description
amd-search-bar	The style of the entire search bar.
amd-search-bar-focus	The style of the entire search bar when the focus is set on the search bar.
amd-search-bar-input	The style of the internal input box.
amd-search-bar-input-focus	The style of the input box when the focus is set on the input box.
amd-search-bar-synthetic	The style of the search icon.
amd-search-bar-synthetic-icon	The style of the search icon.
amd-search-bar-value	The style of the input component.
amd-search-bar-clear-icon	The style of the clear icon.
amd-search-bar-biz-icon	The style of an extra icon.
amd-search-bar-cancel-container	The style of the Cancel button.
amd-search-bar-cancel-container-show	The style of the Cancel button.
amd-search-bar-cancel	The style of the Cancel button.

CSS variable

CSS variable name	Description
--am-color-brand-1	The color of the cursor for entering text.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage" background="#f5f5f5" padding="0">
    <search-bar
      placeholder="Please enter the content"
      value="{{basicValue}}"
      onInput="handleBasicInput"
      onClear="handleBasicClear"/>
  </demo-block>
  <demo-block title="The cancel button is always displayed" background="#f5f5f5" padding="0">
    <search-bar
      placeholder="Please enter the content"
      showCancelButton
      value="{{withCancelValue}}"
      onInput="handleWithCancelInput"
      onClear="handleBasicClear"
      onCancel="handleCancelWithCancel"/>
  </demo-block>
  <demo-block title="Display the cancel button after getting focus" background="#f5f5f5" padding="0">
    <search-bar
      placeholder="Please enter the content"
      value="{{focusWithCancelValue}}"
      showCancelButton="{{focusWithCancelFocus}}"
      onInput="handleFocusWithCancelInput"
      onClear="handleFocusWithCancelClear"
      onCancel="handleFocusCancelWithCancel"
      onBlur="handleFocusCancelWithBlur"
      onFocus="handleFocusCancelWithFocus"/>
  </demo-block>
  <demo-block title="extra Voice icon" background="#f5f5f5" padding="0">
    <search-bar
      placeholder="Please enter the content"
      showBizIcon
      value="{{voiceValue}}"
      onInput="handleVoiceInput"
      onClear="handleVoiceClear"
      onBizIconTap="handleTapVoice"/>
  </demo-block>
  <demo-block title="Support for evoking the numeric keypad" background="#f5f5f5" padding="0">
    <search-bar
      placeholder="Please enter the content"
      value="{{numberValue}}"
      type="number"
      onInput="handleNumberInput"
      onClear="handleNumberClear"/>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: '',
    showVoice: false,
    showBizIcon: false,
    basicValue: '',
    withCancelValue: '',
    voiceValue: '',
    numberValue: '',
    focusWithCancelValue: '',
    focusWithCancelFocus: false,
  },
  handleBasicInput(value) {
    this.setData({ basicValue: value });
  },
  handleBasicClear() {
    this.setData({ basicValue: '' });
  },
  handleWithCancelInput(value) {
    this.setData({ withCancelValue: value });
  },
  handleWithCancelClear() {
    this.setData({ withCancelValue: '' });
  },
  handleCancelWithCancel() {
    this.setData({ withCancelValue: '' });
    my.showToast({ content: 'click cancel', duration: 1000 });
  },
  handleVoiceInput(value) {
    this.setData({ voiceValue: value });
  },
  handleVoiceClear() {
    this.setData({ voiceValue: '' });
  },
  handleTapVoice() {
    my.showToast({ content: 'click voice', duration: 1000 });
  },
  handleFocusWithCancelInput(value) {
    this.setData({ focusWithCancelValue: value });
  },
  handleFocusWithCancelClear() {
    this.setData({ focusWithCancelValue: '' });
  },
  handleFocusCancelWithCancel() {
    this.setData({ focusWithCancelValue: '' });
    my.showToast({ content: 'click cancel', duration: 1000 });
  },
  handleFocusCancelWithFocus() {
    this.setData({ focusWithCancelFocus: true });
  },
  handleFocusCancelWithBlur() {
    this.setData({ focusWithCancelFocus: false });
  },
  handleNumberInput(value) {
    this.setData({ numberValue: value });
  },
  handleNumberClear() {
    this.setData({ numberValue: '' });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "SearchBar",
  "usingComponents": {
    "search-bar": "antd-mini/es/SearchBar/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.9. Selector

A Selector component provides multiple items for users to choose. Generally, it is used as a form component in a filter or form. When you use it with a Form/FormItem component, you must set mode to form.

Attribute

```
type SelectorItem = {
  text: string;
  value: string|number;
  subText?: srting;
  disabled?: boolean;
}
```

Attribute	Type	Required	Default value	Description
value	string | number | string[] | number[]	No	-	The value of a selected item. This property equals the value of an item specified by the items property.
items	SelectorItem[]	Yes	-	Items available for selection.
activeItemClassName	string	No	-	The class name of an activated item.
multiple	boolean	No	false	Whether multiple selections are allowed. The current status (single selection or multiple selections) will be displayed if you include the selector component in the tab bar.
title	string	No	''	The title of the tab bar.
desc	string	No	''	The description of the tab bar.
id	string	No	-	The ID of the form element.
name	string	No	-	The name of the form element.
disabled	boolean	No	false	Whether to disable this component.
mode	'noraml' | 'form'	No	'normal'	When you use this component together with a From/FormItem component, you must set mode to from.
className	string	No	-	The class name.

Event

Event name	Description	Type
onChange	Callback fired when the selected value is changed.	`(v: string | string[], selectedItem: SelectItem | SelectItem[] ) => void`

Style class

Class name	Description
amd-selector	The style of the entire selector.
amd-selector-disabled	The style of the entire selector in disabled mode.
amd-selector-content	The style of the selector content.
amd-selector-item	The style of a selector item.
amd-selector-item-active	The style of a selector item in active state.
amd-selector-item-disabled	The style of a selector item in disabled state.
amd-selector-item-text	The text style.
amd-selector-item-subtext	The subtext style.
amd-selector-item-badge-active	The style of the badge in active state.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<selector
  title="Single choice"
  value="11"
  items="{{items1}}"/>

<selector
  title="Single choice"
  value="2"
  desc="Option with subtitle"
  items="{{items2}}"/>

<selector
  title="Multiple choice"
  value="{{['1','2']}}"
  items="{{items1}}"
  multiple="{{true}}"/>

<selector
  title="All disabled"
  value="{{['1','2']}}"
  items="{{items1}}"
  disabled="{{true}}"
  multiple="{{true}}"/>

<selector
  title="Part disabled"
  value="{{['1','2']}}"
  items="{{items3}}"
  multiple="{{true}}"/>

<selector
  title="Change the value"
  value="{{value}}"
  items="{{items1}}"
  onChange="handleChange"/>

<view class="valusBox">
  <button
    type="danger-ghost"
    inline="{{true}}"
    inlineSize="larger"
    onTap="handleChangeValue"
    data-value="11">
    Change the selected value to: 3
  </button>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    items1: [
      { text: 'Option one', value: '1' },
      { text: 'Option two', value: '2' },
      { text: 'Option three', value: '11' }],
    items2: [
      { text: 'Option one', subText: 'Subtitle one', value: '1' },
      { text: 'Option two', subText: 'Subtitle two', value: '2' },
      { text: 'Option three', subText: 'Subtitle three', value: '3' }],
    items3: [
      { text: 'Option one', subText: 'Subtitle one', value: '1' },
      { text: 'Option two', subText: 'Subtitle two', value: '2', disabled: true },
      { text: 'Option three', subText: 'Subtitle three', value: '3' }],
    items: [
      {
        text: 'Option one',
        value: '1',
      },
      {
        text: 'Option two',
        subText: 'Description copy 2',
        value: '2',
      },
      {
        text: 'Option three',
        disabled: true,
        value: '3',
      },
      {
        text: 'Option four',
        subText: 'Description copy 4',
        disabled: true,
        value: '4',
      },
      {
        text: 'Option five',
        subText: 'Description copy 5',
        value: '5',
      },
    ],
    value: '1',
  },
  handleChangeValue(e) {
    const { value } = e.currentTarget.dataset;
    this.setData({
      value,
    });
  },
  handleChange(e) {
    this.setData({
      value: e,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.btns,
.valusBox {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
  padding: 24rpx;
  background-color: #fff;
}
.btns .amd-button {
  flex: 1;
  margin: 0 12rpx;
}
.valusBox {
  flex-wrap: wrap;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Selector",
  "usingComponents": {
    "selector": "antd-mini/es/Selector/index",
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.10. Stepper

A Stepper component is a two-segment UI control used to incrementally increase or decrease a numeric value. No prompt is displayed if you enter the maximum or minimum value into a stepper. However, when the stepper loses its focus and you enter a value larger than the maximum value or smaller than the minimum value, the system will automatically auto-fill a numeric value as the maximum or minimum value. When you use Stepper as a form component with a Form/FormItem component, you must set mode to form.

Attribute

Attribute	Type	Required	Default value	Description
controlled	boolean	No	false	Whether the stepper is controlled.
value	number	No	-	The value of the input box. The property takes effect when the form is submitted.
type	'number' | 'digit'	No	-	The type of the input box. The property takes effect when the form is submitted.
min	number	No	-	The minimum value.
max	number	No	-	The maximum value.
step	number	No	1	The value you want to increment or decrement by every time.
inputWidth	string	No	-	The width of the input box.
precision	number	No	-	The precision for calculation. Several decimals are reserved.
autoFocus	boolean	No	false	Whether to enable auto-focus. This property may not take effect on iOS devices.
id	string	No	-	The ID of the form element.
name	string	No	-	The name of the form element.
disabled	boolean	No	false	Whether to disable the component.
mode	'noraml' | 'form'	No	'normal'	You must set this property to form when you use a stepper component with a From/FormItem component.
className	string	No	-	The class name.

Event

Event name	Description	Type
onFocus	Callback fired when the focus is set on the stepper.	`( e: number ) => void`
onBlur	Callback fired when the stepper loses its focus.	`( e: number ) => void`
onChange	Callback fired when data changes.	`( e: number, dataSet: Record<string, any> ) => void`

Style class

Class name	Description
amd-stepper	The style of the entire stepper.
amd-stepper-disabled	The style of the entire stepper in disabled mode.
amd-stepper-handler	The style of the +/- icon area.
amd-stepper-handler-up	The style of the + icon area.

amd-stepper-handler-up	The style of the - icon area.
amd-stepper-handler-disabled	The style of the +/- icon area in disabled mode.
amd-stepper-handler-up-inner	The style of the +/- icon.
amd-stepper-input-wrap	The style of the input box area.
amd-stepper-input	The style of the input box.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <stepper
      step="{{1}}"
      value="{{0}}"
      inputWidth="60px"/>
  </demo-block>
  <demo-block title="Controlled component">
    <stepper
      data-a="a"
      controlled
      step="{{1}}"
      value="{{value}}"
      inputWidth="60px"
      onChange="handleChange" />
  </demo-block>
  <demo-block title="Step size setting">
    <stepper
      step="{{0.01}}"
      value="{{0}}"/>
  </demo-block>
  <demo-block title="Limit input range">
    <stepper
      min="{{0}}"
      max="{{10}}"
      step="{{1}}"
      value="{{0}}"/>
  </demo-block>
  <demo-block title="Disabled status">
    <stepper
      value="{{0}}"
      disabled/>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: 0,
  },
  handleChange(value, dataSet) {
    this.setData({ value });
    console.log(dataSet)
  },
  handleAddValue() {
    this.setData({ value: this.data.value + 1 });
  },
  handleMinusValue() {
    this.setData({ value: this.data.value - 1 });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.actions {
  display: flex;
  justify-content: space-between;
  padding-top: 24rpx;
}
.actions .amd-button {
  width: 47%;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Stepper",
  "usingComponents": {
    "stepper": "antd-mini/es/Stepper/index",
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.5.11. Switch

A switch is a component which controls the ON and OFF states. It provides a consistent experience for iOS and Android users compared with native switch components. The difference between a checkbox and a switch is that turning on or off the switch directly triggers the status change, however, the checkbox is used for marking a status and needs to be used with the submission operation.

Attribute

Attribute	Type	Required	Default value	Description
checked	boolean	No	-	Whether to select the switch.
disabled	boolean	No	false	Whether to disable the component.
loading	boolean	No	false	Whether to load the status of the switch.
color	string	No	#1677ff	# color of the background when the switch is turned on.
checkedText	string	No	-	The content displayed when the switch is turned on.
uncheckedText	string	No	-	The content displayed when the switch is turned off.
size	'medium' | 'small' | 'x-small'	No	medium	The size of the switch.
controlled	boolean	No	false	Whether the switch is controlled.
mode	'normal' |'form'	No	'normal'	When you use a switch with a From/FormItem component, you must set mode to from.
className	string	No	-	The class name.

Event

Event name	Description	Type
onChange	Callback fired when the switch is tapped.	`( checked: boolean ) => void`

Slot

Name	Description
checked	The content slot when the switch is turned on.
unchecked	The content slot when the switch is turned off.

Style class

Class name	Description
amd-switch	The style of the entire switch.
amd-switch-checked	The style of the switch that is turned on.
amd-switch-disabled	The style of the switch in disabled mode.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <switch
      checked="{{false}}"
      onChange="handleChange"
    />
  </demo-block>
  <demo-block title="Have default value">
    <switch
      checked="{{true}}"/>
  </demo-block>
  <demo-block title="Text and icons">
    <switch
      checkedText="Open"
      uncheckedText="Close"/>
    <switch>
      <am-icon type="CheckOutline" slot="checked" size="x-small"/>
      <am-icon type="CloseOutline" slot="unchecked"  size="x-small"/>
    </switch>
  </demo-block>
  <demo-block title="Custom color">
    <switch
      checked="{{true}}"
      color="#00b578"/>
  </demo-block>
  <demo-block title="Disabled status">
    <switch
      checked="{{true}}"
      disabled="{{true}}"/>
    <switch
      disabled="{{true}}"/>
  </demo-block>
  <demo-block title="Loading status">
    <switch
      loading="{{true}}"/>
  </demo-block>

  <demo-block title="size small">
    <switch  size="small" />
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    value: false,
  },
  handleChange(checked) {
    console.log('change checked', checked)
    my.alert({
      title: `The current switch is ${checked ? 'on' : 'off'} status. `,
    });
  },

});
```

The following shows an example of the code in the index.acss file:

```
.amd-switch {
  margin-right: 16rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Switch",
  "usingComponents": {
    "switch": "antd-mini/es/Switch/index",
    "demo-block": "../../components/DemoBlock/index",
    "am-icon": "antd-mini/es/Icon/index"
  }
}
```

1.10.5.12. Terms

When you use Terms as a form component with a Form/FormItem component, you must set mode to form.

Attribute

Attribute	Type	Required	Default value	Description
fixed	boolean	No	false	Whether to fix the component at the bottom.
hasCheckbox	boolean	No	true	Whether a checkbox is required.
disabled	boolean	No	false	Whether to disable the component.
mainButtonText	string	Yes	'Agree'	The text on the main button.
addonButtonText	string	No	-	The text on the auxiliary button.
className	string	No	-	The class name.

Event

Event name	Description	Type
onChange	Callback fired when the checkbox is tapped.	`( v : boolean ) => void`
onMainBtnTap	Callback fired when the main button is tapped.	`() => void`
onSubBtnTap	Callback fired when the auxiliary button is tapped.	`() => void`

Slot

Name	Description
The default slot.	The UI text with an agreement, for example, agree to User Agreement.

Style class

The class name	Description
amd-terms	The style of the entire component.
amd-terms-fixed	The style of the entire component.
amd-terms-term	The style of the term area.
amd-terms-term-checkbox	The style of the checkbox.
amd-terms-content	The style of the content area.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title=" Contains checkbox, no suction bottom">
    <terms
      hasCheckbox="{{true}}"
      isDesc="{{true}}"
      fixed="{{false}}"
      mainButtonText="Agree to authorize"
      addonButtonText="Cancel"
      onChange="handleSelectTerm"
      onMainBtnTap="handleTapMainBtn"
    >
      <view>
        Agree<text style="color: #1677ff">《User Authorization Agreement》</text>
      </view>
    </terms>
  </demo-block>
  <demo-block title="Does not contain checkbox, no suction bottom">
    <terms
      hasCheckbox="{{false}}"
      isDesc="{{true}}"
      fixed="{{false}}"
      mainButtonText="Agree to authorize"
      addonButtonText="Cancel"
      onMainBtnTap="handleTapMainBtn"
      onSubBtnTap="handleTapSubBtn">
      <view>
        Agree<text style="color: #1677ff">《User Authorization Agreement》</text>
      </view>
    </terms>
  </demo-block>
  <view class="fixed-title">Suction bottom status</view>
  <terms
    hasCheckbox="{{false}}"
    isDesc="{{true}}"
    fixed="{{true}}"
    mainButtonText="Agree to authorize"
    addonButtonText="Cancel"
    onMainBtnTap="handleTapMainBtn"
    onSubBtnTap="handleTapSubBtn">
    <view>
      Agree<text style="color: #1677ff">《User Authorization Agreement》</text>
    </view>
  </terms>

</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {},
  handleSelectTerm(checked) {
    my.showToast({
      content: `The current selected status is: ${checked}`,
    });
  },
  handleTapMainBtn() {
    my.alert({
      content: 'Agree to authorize',
    });
  },
  handleTapSubBtn() {
    my.alert({
      content: 'Cancel',
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.fixed-title {
  position: fixed;
  bottom: 188px;
  padding: 24rpx 16rpx 12rpx;
  color: #969696;
  font-size: 28rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Terms",
  "usingComponents": {
    "terms": "antd-mini/es/Terms/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.6. Feedback

1.10.6.1. Dialog

A dialog informs users about critical information or operation feedback and offers a few options for them to make decisions. When requiring users to interact with the app without jumping to a new page and interrupting the workflow, you can use Modal to create a new floating layer over the current page to get user feedback or display information.

Attribute

Attribute	Type	Required	Default value	Description
direction	'vertical' | 'horizontal'	No	'vertical'	The direction in which the buttons are arranged.
title	string	No	-	The title.
content	string	No	-	The content.
visible	boolean	Yes	false	Whether the dialog is visible and in controlled mode.
duration	number	No	300	The duration of the animation used for transition. The unit is milliseconds.
maskClosable	boolean	No	true	Whether to close the mask layer upon taps.
disableScroll	boolean	No	true	Whether to disable scrolling of the page when the dialog box is displayed.
animation	boolean	No	true	Whether to enable the animation for transition.
zIndex	number	No	998	The layer level of the dialog.
className	string	No	-	The class name.

Event

Event name	Description	Type
onButtonTap	Callback fired when an internal button of the Modal component is tapped.	`(index: number ) => void`
onClose	Callback fired when the dialog is closed.	`() => void`

Slot

Name	Description
default	The pop-up content.

Style class

Class name	Description
amd-dialog	The style of the entire dialog.
amd-dialog-vertical	The style of the entire dialog.
amd-dialog-horizontal	The style of the entire dialog.

amd-dialog-content	The style of the entire content.
amd-dialog-content-title	The title style.
amd-dialog-content-content	The content style.
amd-dialog-content-button-container	The style of the button area.
amd-dialog-content-button-container-vertical	The style of the button area.
amd-dialog-content-button-container-horizontal	The style of the button area.
amd-dialog-content-button	The button style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <dialog
    content="Example 1"
    buttonText="{{['I know']}}"
    visible="{{isNoBtnShow}}"
    onButtonTap="handleClose"
    onClose="handleClose">
  </dialog>
  <dialog
    content="Example 1"
    buttonText="{{['I know']}}"
    visible="{{isMaskClosableShow}}"
    onButtonTap="handleClose"
    onClose="handleClose"
    maskClosable>
  </dialog>
  <dialog
    title="Vertical"
    content="Title content"
    buttonText="{{['Main operation for long copy', 'More','Cancel']}}"
    visible="{{isVerticalShow}}"
    direction="vertical"
    maskClosable="{{true}}"
    onClose="handleClose"
    onButtonTap="handleButtonTap">
  </dialog>

  <dialog
    title="Horizontal"
    content="Title content"
    buttonText="{{['Auxiliary operation', 'Main operation']}}"
    visible="{{isHoriShow}}"
    maskClosable="{{true}}"
    direction="horizontal"
    onClose="handleClose"
    onButtonTap="handleButtonTap"/>

  <dialog
    title="Custom component"
    content="Title content"
    buttonText="{{['Auxiliary operation', 'Main operation']}}"
    visible="{{isCusDialogShow}}"
    maskClosable="{{true}}"
    direction="horizontal"
    onClose="handleClose"
    onButtonTap="handleClose">
    <view class="input-container">
      <input-item placeholder="message a friend"></input-item>
    </view>
  </dialog>

  <dialog
    title="With big picture"
    content="Title content"
    buttonText="{{['Auxiliary operation', 'Main operation']}}"
    imageSize="x-large"
    image="{{url}}"
    visible="{{isLImgDialogShow}}"
    maskClosable="{{true}}"
    direction="horizontal"
    onClose="handleClose"
    onButtonTap="handleClose"/>
  <demo-block title="Basic usage">
    <view class="btn-list">
      <button  onTap="handleOpenNoBtn">The simplest small dialog</button>
      <button  onTap="handleOpenMaskClosable">Click mask to close</button>
    </view>
  </demo-block>
  <demo-block title="Operation button">
    <view class="btn-list">
      <button  onTap="handleOpenVertical">Vertical</button>
      <button  onTap="handleOpenHori">Horizontal</button>
    </view>
  </demo-block>
  <demo-block title="Content area">
    <view class="btn-list">
      <button  onTap="handleOpenCus">Custom content area</button>
      <button  onTap="handleOpenLImg">With picture</button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    isNoBtnShow: false,
    isMaskClosableShow: false,
    isVerticalShow: false,
    isHoriShow: false,
    isLImgDialogShow: false,
    isCusDialogShow: false,
    url: 'https://gw.alipayobjects.com/zos/rmsportal/yFeFExbGpDxvDYnKHcrs.png',
  },
  handleClose() {
    this.setData({
      isNoBtnShow: false,
      isMaskClosableShow: false,
      isVerticalShow: false,
      isHoriShow: false,
      isLImgDialogShow: false,
      isCusDialogShow: false,
    });
  },
  handleButtonTap(index) {
    my.alert({
      title: `Clicked the${index + 1}button`,
      complete: () => {
        this.handleClose();
      },
    });
  },
  handleOpenNoBtn() {
    this.setData({ isNoBtnShow: true });
  },
  handleOpenMaskClosable() {
    this.setData({ isMaskClosableShow: true });
  },
  handleOpenVertical() {
    this.setData({
      isVerticalShow: true,
    });
  },
  handleOpenHori() {
    this.setData({
      isHoriShow: true,
    });
  },
  handleOpenLImg() {
    this.setData({
      isLImgDialogShow: true,
    });
  },
  handleOpenCus() {
    this.setData({
      isCusDialogShow: true,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.deleteBtn {
  color: #f93a4a;
  font-weight: bolder;
}

.cancelBtn {
  color: #ccc;
}

.buttonBold,
.modalButtonBold .am-modal-footer {
  font-weight: bold;
}

.space {
  margin-top: 10px;
}

.slide {
  margin-top: 10px;
  padding-bottom: 10px;
}

.amd-input-item{
  border-color: #ccc;
}

.input-container{
  border: 1px solid #e5e5e5;
  padding: 8px 12px;
}

.a-input-content{
  font-size: 15px;
}

.a-input-placeholder{
  font-size:15px !important;
}
```

The following shows an example of the code in the index.json file:

```
{
  "usingComponents": {
    "dialog": "antd-mini/es/Dialog/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.6.2. Loading

A Loading component is used to inform an end user that partial content or the page is being loaded.

Attribute

Attribute	Type	Default value	Description
className	string	-	The class name.
color	string	'#999'	The color used for loading. When type is set to 'spin', only hexadecimal color codes are supported, such as '#fff'.
delay	number	-	Delay in displaying the loading status, in millisecond. Note that the change of this property does not take effect in real. This property takes effect only when type is set to 'spin'.
height	string	'200rpx'	The height of the loading icon. If you do not pass in this property, the default value is the same as the size property. This property takes effect when type is set to 'mini'.
loading	boolean	true	Whether the content is being loaded. This property takes effect when type is set to 'spin'.
miniSize	string	'200rpx'	The width of the loading icon. This property takes effect when type is set to 'mini'.

size	'x-large' | 'large' | 'medium' | 'small'	"medium"	The size of the loading icon. This property takes effect when type is set to 'spin'.
text	string	-	The loading text. This property takes effect when type is set to 'spin'.
theme	'dark' | 'light'	"dark"	The color. dark indicates a dark color while light indicates a light color. This property takes effect when type is set to 'spin'.
type	'spin' | 'mini'	'spin'	The loading style type.

Slot

Name	Description
indicator	The custom indicator of loading. This property takes effect when type is set to 'spin'.
text	The custom text. This property takes effect when type is set to 'spin'.

Style class

The following style classes are only used for a specific type:

Class name	Description
amd-loading	The style of the entire component.
amd-loading-spin-container	The style of the external container.
amd-loading-spin	The style of the internal container.
amd-loading-spin-dark	The style of the dark mode.
amd-loading-spin-light	The style of the light mode.
amd-loading-spin-icon	The external style of the loading icon.
amd-loading-spin-icon-indicator	The external style of the custom loading icon.
amd-loading-spin-icon-small	The style of the loading icon.
amd-loading-spin-icon-medium	The style of the loading icon.
amd-loading-spin-icon-large	The style of the loading icon.
amd-loading-spin-icon-x-large	The style of the loading icon.
amd-loading-spin-text	The style of the loading text.
amd-loading-spin-text-dark	The style of the loading text in dark mode.
amd-loading-spin-text-light	The style of the loading text in light mode.
amd-loading-mini-container	The style of the external container.
amd-loading-mini	The style of the external container.
amd-loading-mini-item	The style of all items that need to be loaded.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block title="Basic usage" padding="0">
    <loading type="mini"></loading>
  </demo-block>
  <demo-block title="Theme color loading"  padding="0">
    <loading type="mini" color="#1677ff"></loading>
  </demo-block>
  <demo-block title="spin">
    <loading type="spin" size="x-large" text="x-large" />
    <loading type="spin" size="large" text="large" />
    <loading type="spin" size="medium" text="medium" />
    <loading type="spin" size="small" text="small" />
  </demo-block>
  <demo-block title="light color spin" background="rgba(0, 0, 0, 0.7)">
    <loading size="x-large" text="x-large" theme="light" />
    <loading size="large" text="large" theme="light" />
    <loading size="medium" text="medium" theme="light" />
    <loading size="small" text="small" theme="light" />
  </demo-block>
  <demo-block title="Custom color">
    <loading size="x-large" text="x-large" color="#ff0000" />
    <loading size="large" text="large" color="#ff0000" />
    <loading size="medium" text="medium" color="#ff0000" />
    <loading size="small" text="small" color="#ff0000" />
  </demo-block>
  <demo-block title="Appears after a delay of 3000ms">
    <loading  delay="{{3000}}" />
  </demo-block>
  <demo-block title="Custom icon">
    <loading text="Custom icon">
      <am-icon slot="indicator" type="HeartFill" />
    </loading>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({ });
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Loading",
  "usingComponents": {
    "loading": "antd-mini/es/Loading/index",
    "am-icon": "antd-mini/es/Icon/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.6.3. Mask

A Mask is a background layer in a dark color. It is frequently used as the background layer of a modal window to set the visual focus on the modal window itself.

Attribute

Attribute	Type	Default value	Description
maskZindex	string	-	The level of the mask layer.
type	'product' | 'market'	'product'	Type
show	boolean	-	Whether to display the mask layer.
fixMaskFull	boolean	false	Whether to display the mask layer in full screen through fixing. The default value is false (not fixing).
className	string	-	The class name.

Event

Event name	Description	Type
onMaskTap	The event is triggered upon taps on the mask.	`(v: Record<string, any>) => void`

CSS variable

CSS variable name
--am-mask-backgroundColor
--am-mask-market-backgroundColor

Style class

Class name
amd-mask
amd-mask__m
amd-mask__fix

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<mask type="{{type}}" show="{{show}}" maskZindex="{{maskZindex}}" onMaskTap="handleClickMask" />
<demo-block title="Basic usage">
    <view class="btn-list">
        <button data-type="product" onTap="handleClickBtn">Display the mask layer of product type</button>
        <button data-type="market" onTap="handleClickBtn">Display the mask layer of market type</button>
    </view>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    type: 'market',
    maskZindex: 10,
    show: false,
  },
  handleClickMask() {
    this.setData({ show: false });
  },
  handleClickBtn(e) {
    const { type } = e.target.dataset;
    this.setData({ type, show: true });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Mask",
  "usingComponents": {
    "mask": "antd-mini/es/Mask/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.6.4. Modal

A modal window is used when you want to give a clear and obvious warning or notification to an end user for what he is doing. The user must perform a specific operation on the window in order to proceed.

Attribute

Attribute	Type	Required	Default value	Description
title	string	No	-	The title.
content	string	Yes	-	The content.
image	string	No	-	The thumbnail.

imageSize	'medium' | 'large' | 'x-large'	No	'medium'	The size of the thumbnail.
visible	boolean	Yes	false	Whether the modal window is visible or in controlled mode.
duration	number	No	-	The duration of the animation for transition, in milliseconds.
mainButtonText	string	No	'Main operation'	The main button.
addonButtonText	string	No	'Auxiliary operation'	The auxiliary button, which is the second button.
maskClosable	boolean	No	true	Whether to close the mask layer upon taps.
disableScroll	boolean	No	true	Whether to disable scrolling of the page when the modal window is being displayed.
animation	boolean	No	true	Whether to enable the animation for transition.
zIndex	number	No	998	The layer level of the modal window.
className	string	No	-	The class name.

Event

Event name	Description	Type
onButtonTap	Callback fired when an internal button of the modal window is tapped.	`(type: 'main' | 'addon' ) => void`
onClose	Callback fired when the close icon is tapped.	`() => void`

Slot

Name	Description
default	The modal window content.

Style class

Class name	Description
amd-modal	The style of the entire modal window.
amd-modal-content	The content style of the main modal window.
amd-modal-content-image	The image style of the modal window.
amd-modal-content-image-medium	The image style of the modal window.
amd-modal-content-image-large	The image style of the modal window.
amd-modal-content-title	The title style of the modal window.
amd-modal-content-content	The content style of the modal window.
amd-modal-buttons-container	The style of the button area on the modal window.
amd-modal-buttons-addon	The style of the auxiliary button.
amd-modal-close	The style of the close icon.

Code sample

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <modal
    visible="{{isBaseModalShow}}"
    content="Example 1"
    mainButtonText="I know"
    addonButtonText=""
    maskClosable="{{false}}"
    onClose="closeBaseModal"
    onButtonTap="closeBaseModal">
  </modal>

  <modal
    visible="{{isCloseableModalShow}}"
    content="Example 1"
    mainButtonText="I know"
    addonButtonText=""
    maskClosable
    onClose="closeCloseableModal"
    onButtonTap="closeCloseableModal">
  </modal>
  <modal
    visible="{{isCustomBtnModalShow}}"
    content="Example 1"
    mainButtonText="Online reading"
    addonButtonText="Download the file"
    maskClosable="{{false}}"
    onClose="closeCustomBtnModal"
    onButtonTap="handleButtonTap">
  </modal>
  <modal
    visible="{{isMainBtnModalShow}}"
    content="Example 1"
    mainButtonText="Online reading"
    addonButtonText=""
    maskClosable="{{false}}"
    onClose="closeMainBtnModal"
    onButtonTap="handleButtonTap">
  </modal>

  <modal
    title="Kind tips"
    content="Please select the range"
    visible="{{isCustomModalShow}}"
    onClose="closeCustomModal"
    mainButtonText="Confirm"
    addonButtonText="Cancel"
    maskClosable="{{true}}"
    onButtonTap="closeCustomModal">
      <slider value="5" step="5" />
  </modal>

  <modal
    title="Pop-up window with picture"
    imageSize="x-large"
    image="{{url}}"
    visible="{{isLImgModalShow}}"
    content="The description prompts the user for a solution."
    mainButtonText="Main button"
    addonButtonText="Auxiliary button"
    onClose="closeLImgModal"
    onButtonTap="closeLImgModal">
  </modal>
  <demo-block title="Basic usage">
    <view class="btn-list">
      <button onTap="openBaseModal">The simplest popup</button>
      <button onTap="openCloseableModal">Click mask to close</button>
    </view>
  </demo-block>
  <demo-block title="Operation button">
    <view class="btn-list">
      <button onTap="openCustomBtnModal">Custom button</button>
      <button onTap="openMainBtnModal">Only main button</button>
    </view>
  </demo-block>
  <demo-block title="Operation area">
    <view class="btn-list">
      <button  onTap="openCustomModal">Custom operation area</button>
      <button  onTap="openLImgModal">Pop-up window with picture</button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    isBaseModalShow: false,
    isCloseableModalShow: false,
    isCustomBtnModalShow: false,
    isMainBtnModalShow: false,
    isCustomModalShow: false,
    isLImgModalShow: false,
    url: 'https://gw.alipayobjects.com/zos/rmsportal/yFeFExbGpDxvDYnKHcrs.png',
  },
  openBaseModal() {
    this.commonShow('isBaseModalShow');
  },
  closeBaseModal() {
    this.commonHide('isBaseModalShow');
  },
  openCloseableModal() {
    this.commonShow('isCloseableModalShow');
  },
  closeCloseableModal() {
    this.commonHide('isCloseableModalShow');
  },
  openCustomBtnModal() {
    this.commonShow('isCustomBtnModalShow');
  },
  closeCustomBtnModal() {
    this.commonHide('isCustomBtnModalShow');
  },
  openMainBtnModal() {
    this.commonShow('isMainBtnModalShow');
  },
  closeMainBtnModal() {
    this.commonHide('isMainBtnModalShow');
  },
  openCustomModal() {
    this.commonShow('isCustomModalShow');
  },
  closeCustomModal() {
    this.commonHide('isCustomModalShow');
  },
  openLImgModal() {
    this.commonShow('isLImgModalShow');
  },
  closeLImgModal() {
    this.commonHide('isLImgModalShow');
  },
  handleButtonTap(type) {
    my.alert({
      title: `Clicked${type === 'main' ? 'Main button' : 'Auxiliary button'}`,
    });
  },
  commonShow(prop) {
    this.setData({
      [prop]: true,
    });
  },
  commonHide(prop) {
    this.setData({
      [prop]: false,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.deleteBtn {
  color: #f93a4a;
  font-weight: bolder;
}
.cancelBtn {
  color: #ccc;
}
.buttonBold,
.modalButtonBold .am-modal-footer {
  font-weight: bold;
}
.space {
  margin-top: 10px;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Modal",
  "usingComponents": {
    "modal": "antd-mini/es/Modal/index",
    "button": "antd-mini/es/Button/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.6.5. Popover

A Popover component is a pop-up box that appears when a user taps on an element. It can only be opened by an icon on the navigation bar and generally used for implementing functions that are less used.

Attribute

Popover

Attribute	Type	Required	Default value	Description
visible	boolean	No	false	Whether the popover is visible.
mode	'dark' | 'light'	No	'dark'	The display mode of the popover.
placement	'top' | 'top-right' | 'top-left' | 'bottom' | 'bottom-left' | 'bottom-right' | 'left' | 'left-top' | 'left-bottom' | 'right' | 'right-top' | 'right-bottom'	No	'bottom-right'	The direction.
className	string	No	-	The class name.
mask	boolean	No	false	Whether to display the mask layer.
maskClosable	boolean	No	true	Whether to close the mask layer upon taps.
fixMaskFull	boolean	No	false	This property is used to fix the problem that the mask layer cannot be fully displayed due to the transform property.

PopoverItem

Attribute	Type	Required	Default value	Description
icon	string	No	-	The icon type.
className	string	No	-	The class name.

Event

Popover

Event name	Description	Type
onVisibleChange	Callback fired when the popover state is being switched from hidden to visible or vise versa.	`( visible: boolean, mode: 'component' | 'mask' ) => void`

PopoverItem

Event name	Description	Type
onTap	Callback fired when the popover is tapped.	`() => void`

Slot

Popover

Name	Description
items	The tooltip slot. You can use PopoverItem to render the list.

PopoverItem

Name	Description
icon	The icon slot.

Style class

Popover

Class name	Description
amd-popover	The style of the entire popover.
amd-popover-container	The style of the main content.
amd-popover-content	The content style.
amd-popover-arrow	The arrow style.
amd-popover-inner	The internal content style.
amd-popover-inner-pseudo	The style of the entire tooltip content.

Class name	Description
amd-popover-item	The style of the entire popover.
amd-popover-item-icon	The icon style.
amd-popover-item-icon-item	The icon style.
amd-popover-item-text	The text style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <view class="wrap dark">
      <popover
        placement="bottom"
        visible="{{showDark}}"
        onVisibleChange="handleDarkVisibleChange">
        <button inlineSize="large" inline>Click me</button>
        <view slot="items" class="tooltip">Hello World</view>
      </popover>
    </view>
  </demo-block>
  <demo-block title="Light bubble">
    <view class="wrap">
      <popover
        placement="right"
        mode="light"
        visible="{{showLight}}"
        onVisibleChange="handleLightVisibleChange">
        <button inlineSize="large" inline>Click me</button>
        <view slot="items" class="tooltip">Hello World</view>
      </popover>
    </view>
  </demo-block>
  <demo-block title="Bubble position" className="multi-demo">
    <view class="multi-wrap">
      <popover
        placement="{{placement}}"
        visible="{{show}}"
        mask="{{showMask}}"
        onVisibleChange="handleVisibleChange">
        <view class="multi">
          <view>Click{{show ? 'Hide' : 'Display'}}</view>
          <view>
            {{placement}}
          </view>
        </view>
        <view slot="items" class="tooltip">
          <view>Popover</view>
          <view>
            Content
          </view>
        </view>
      </popover>
    </view>
    <view class="demo-btn-container">
      <button
        class="demo-btn"
        onTap="handleNextPosition">
        Next position
      </button>
      <button
        class="demo-btn"
        onTap="handleToggleMask">
        {{showMask?'Hide':'Display'}}Mask
      </button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
const placement = [
  'top',
  'top-right',
  'top-left',
  'bottom',
  'bottom-left',
  'bottom-right',
  'left',
  'left-top',
  'left-bottom',
  'right',
  'right-top',
  'right-bottom',
];
Page({
  data: {
    placement: placement[0],
    show: true,
    showMask: false,
    showLight: true,
    showDark: true,
  },
  handleLightVisibleChange(e) {
    this.setData({
      showLight: e,
    });
  },
  handleDarkVisibleChange(e) {
    this.setData({
      showDark: e,
    });
  },
  handleNextPosition() {
    let index = placement.indexOf(this.data.placement);
    index = index >= placement.length - 1 ? 0 : index + 1;
    this.setData({
      show: true,
      placement: placement[index],
    });
  },
  handleVisibleChange(visible, mode) {
    this.setData({
      show: visible,
    });
    if (mode === 'mask') {
      my.showToast({ content: 'Click mask to close', duration: 2000 });
    }
  },
  handleToggleMask() {
    this.setData({
      showMask: !this.data.showMask,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.wrap {
  display: flex;
}
.wrap.dark {
  padding-bottom: 100rpx;
}
.wrap.dark .tooltip {
  color: #fff;
}
.wrap .tooltip {
  white-space: nowrap;
  font-size: 24rpx;
  padding: 16rpx;
}
.multi-demo .demo-block-content {
  background: transparent;
}

.multi-wrap {
  height: 600rpx;
  display: flex;
  align-items: center;
  justify-content: center;
}
.multi-wrap .multi {
  background: #aaa;
  border-radius: 12rpx;
  height: 200rpx;
  width: 200rpx;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  color: #fff;
}
.multi-wrap .tooltip {
  color: #fff;
  font-size: 24rpx;
  padding: 16rpx;
}
.demo-btn-container {
  display: flex;
  justify-content: space-around;
}

.demo-btn {
  width: 45%;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "PopoverBase",
  "usingComponents": {
    "popover": "antd-mini/es/Popover/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

Used with the PopoverItem component

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Dark bubble menu">
    <view class="wrap">
      <popover
        placement="bottom-left"
        visible="{{showDark}}"
        mask
        onVisibleChange="handleDarkVisibleChange">
        <button inline inlineSize="large">Click me</button>
        <block slot="items">
          <popover-item
            onTap="handleTapItem"
            icon="ScanningOutline"
            data-type="showDark"
            data-name="Scan">
            Scan
          </popover-item>

          <popover-item
            onTap="handleTapItem"
            icon="ReceivePaymentOutline"
            data-type="showDark"
```

```
          data-name="pay/receive money">
          pay/receive money
        </popover-item>

        <popover-item
          onTap="handleTapItem"
          icon="TransportQRcodeOutline"
          data-type="showDark"
          data-name="Passenger code">
          Passenger code
        </popover-item>
        <popover-item
          onTap="handleTapItem"
          data-type="showDark"
          data-name="icon slot">
          <image
            slot="icon"
            mode="scaleToFill"
            src="{{url}}"
            style="width: 100%;height: 100%;"/>
          icon slot
        </popover-item>
      </block>
    </popover>
  </view>
</demo-block>
<demo-block title="Light bubble menu">
  <view class="wrap">
    <popover
      placement="bottom-left"
      visible="{{showLight}}"
      mode="light"
      mask
      onVisibleChange="handleLightVisibleChange">
      <button inline inlineSize="large">Click</button>
      <block slot="items">
        <popover-item
          onTap="handleTapItem"
          icon="ScanningOutline"
          data-type="showLight"
          data-name="Scan">
          Scan
        </popover-item>

        <popover-item
          onTap="handleTapItem"
          icon="ReceivePaymentOutline"
          data-type="showLight"
          data-name="pay/receive money">
          pay/receive money
        </popover-item>

          <popover-item
            onTap="handleTapItem"
            icon="TransportQRcodeOutline"
            data-type="showLight"
            data-name="passenger code">
            passenger code
        </popover-item>
        <popover-item
          onTap="handleTapItem"
          data-type="showLight"
          data-name="icon slot">
          <image
            slot="icon"
            mode="scaleToFill"
            src="{{url}}"
            style="width: 100%;height: 100%;"/>
          icon slot
        </popover-item>
      </block>
    </popover>
  </view>
</demo-block>
<demo-block title="bubble menu without icon">
  <view class="wrap">
    <popover
      placement="bottom-left"
      visible="{{showNoIcon}}"
      mask
      onVisibleChange="handleNoIconVisibleChange">
      <button inline inlineSize="large">Click</button>
      <block slot="items">
        <popover-item
          onTap="handleTapItem"
          data-type="showNoIcon"
          data-name="Scan">
```

```
            Scan
          </popover-item>

          <popover-item
            onTap="handleTapItem"
            data-type="showNoIcon"
            data-name="pay/receive money">
            ay/receive money
          </popover-item>

          <popover-item
            onTap="handleTapItem"
            data-type="showNoIcon"
            data-name="passenger code">
            passenger code
          </popover-item>
        </block>
      </popover>
    </view>
  </demo-block>

</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    showLight: false,
    showDark: false,
    showNoIcon: false,
    url: 'https://gw.alipayobjects.com/mdn/rms_ce4c6f/afts/img/A*XMCgSYx3f50AAAAAAAAAAABkARQnAQ',
  },
  handleLightVisibleChange(e, mode) {
    this.setData({
      showLight: e,
    });
    if (mode === 'mask') {
      my.showToast({ content: 'click mask to close', duration: 2000 });
    }
  },
  handleDarkVisibleChange(e, mode) {
    this.setData({
      showDark: e,
    });
    if (mode === 'mask') {
      my.showToast({ content: 'click mask to close', duration: 2000 });
    }
  },
  handleNoIconVisibleChange(e, mode) {
    this.setData({
      showNoIcon: e,
    });
    if (mode === 'mask') {
      my.showToast({ content: 'click mask to close', duration: 2000 });
    }
  },
  handleTapItem(e) {
    this.setData({
      [e.target.dataset.type]: false,
    });
    my.showToast({ content: `Clicked${e.target.dataset.name}` });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.wrap {
  display: flex;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Popover: combine PopoverItem component",
  "usingComponents": {
    "popover": "antd-mini/es/Popover/index",
    "popover-item": "antd-mini/es/Popover/PopoverItem/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.6.6. Popup

A Popup component is a custom content area floated or displayed on the screen. It is used to display popup content or prompt messages and allow you to select, enter, or switch content. You can use multiple popups on the same page for overlay display. The onClose function is not triggered when maskClosable is set to false.

Attribute

Attribute	Type	Required	Default	Description
visible	boolean	No	false	Whether to display the component.
maskClosable	boolean	No	false	Whether the mask layer can be closed upon taps.
showCloseIcon	boolean	No	false	Whether to display the close icon.
disableScroll	boolean	No	true	Whether to disable scrolling of the page when a popup is displayed.
animation	boolean	No	true	Whether to enable the animation for transition.
duration	number	No	300	The animation duration in milliseconds.
position	'center' | 'top' | 'bottom' | 'left' | 'right'	No	'center'	The layout of the popup.
zIndex	number	No	998	The layer level of the popup.
className	string	No	-	The class name.

Event

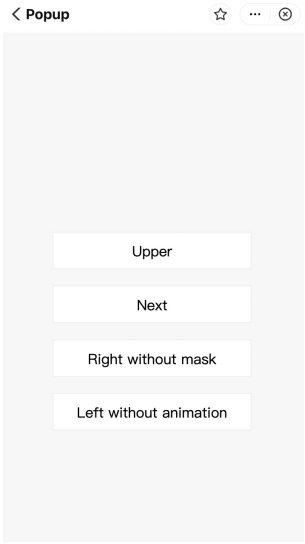
Event name	Description	Type
onClose	Callback fired when the popup is closed.	`( visible: boolean ) => void`

Style class

Class name	Description
amd-popup	The style of the entire component.
amd-popup-mask	The style of the popup mask layer.
amd-popup-disable-scroll	The style of the popup with scrolling disabled.
amd-popup-animation	The style of the popup with the animation turned on.
amd-popup-content	The content style.
amd-popup-top	The style of the popup on the top.
amd-popup-bottom	The style of the popup on the bottom.
amd-popup-left	The style of the popup on the left.
amd-popup-right	The style of the popup on the right.
amd-popup-center	The style of the popup on the center.
amd-popup-close-container	The style of the area around the close icon.
amd-popup-close-container	The style of the close icon.

Code sample

Basic usage



The following shows an example of the code in the index.axml file:

```
<view>
  <popup visible="{{basicShow}}" maskClosable="{{maskClosable}}" position="{{position}}" animation="{{animation}}" onClose="handlePopupClose"
    showCloseIcon="{{showCloseIcon}}">
  </popup>
  <popup visible="{{showCenterDisableScoll}}" maskClosable="{{maskClosable}}" position="center" animation="{{animation}}"
    onClose="handlePopupClose" showCloseIcon="{{showCloseIcon}}">
    <scroll-view scroll-y="{{true}}" class="box center" disable-lower-scroll="out-of-bounds" disable-upper-scroll="out-of-bounds">
      <view class="centerContent"> try to scroll</view>
    </scroll-view>
  </popup>

  <popup visible="{{showCenterScoll}}" maskClosable="{{maskClosable}}" position="center" animation="{{animation}}" onClose="handlePopupClose"
    showCloseIcon="{{showCloseIcon}}" disableScroll="{{false}}">
    <scroll-view scroll-y="{{true}}" class="box center">
      <view class="centerContent"> try to scroll</view>
    </scroll-view>
  </popup>
  <demo-block title="Popup position">
    <view class="btn-list">
      <button data-position="top" onTap="handleShowBasic">Top popup</button>
      <button data-position="bottom" onTap="handleShowBasic">Bottom popup</button>
      <button data-position="left" onTap="handleShowBasic">Left popup</button>
      <button data-position="right" onTap="handleShowBasic">Right popup</button>
      <button data-position="center" onTap="handleShowBasic">Middle popup</button>
    </view>
  </demo-block>
  <list>
    <list-item>
      Display close button
      <switch slot="extra" checked="{{showCloseIcon}}" controlled onChange="handleChangeShowCloseIcon" />
    </list-item>
    <list-item>
      Clickable mask to close
      <switch slot="extra" checked="{{maskClosable}}" controlled onChange="handleChangeMaskClosable" />
    </list-item>
    <list-item>
      Turn on transition animation
      <switch slot="extra" checked="{{animation}}" controlled onChange="handleChangeAnimation" />
    </list-item>
  </list>
  <demo-block title=" Set disableScroll">
    <view class="btn-list">
      <button onTap="handleShowDisableScroll">Popup inner scrolling - disable page scrolling</button>
      <button onTap="handleShowScroll">Popup internal scrolling - enable page scrolling</button>
    </view>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    position: '',
    basicShow: false,
    maskClosable: true,
    showCloseIcon: true,
    animation: true,
    showCenterScoll: false,
    showCenterDisableScoll: false,
  },
  handlePopupClose() {
    this.setData({
      basicShow: false,
      showCenterScoll: false,
      showCenterDisableScoll: false,
    });
  },
  handleShowBasic(e) {
    const { position } = e.target.dataset;
    this.setData({
      position,
      basicShow: true,
    });
  },
  handleShowDisableScroll() {
    this.setData({
      showCenterDisableScoll: true,
    });
  },
  handleShowScroll() {
    this.setData({
      showCenterScoll: true,
    });
  },
  handleChangeMaskClosable(checked) {
    const { showCloseIcon } = this.data;
    if (!showCloseIcon && !checked) {
      return my.alert({
        content: 'Hiding the close button and the mask close event at the same time will not close the popup layer',
      });
    }
    this.setData({ maskClosable: checked });
  },
  handleChangeShowCloseIcon(checked) {
    const { maskClosable } = this.data;
    if (!maskClosable && !checked) {
      return my.alert({
        content: 'Hiding the close button and the mask close event at the same time will not close the popup layer',
      });
    }
    this.setData({ showCloseIcon: checked });
  },
  handleChangeAnimation(checked) {
    this.setData({ animation: checked });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.box {
  display: flex;
  justify-content: center;
  align-items: center;
}

.box.center {
  display: block;
  height: 200px;
}

.centerContent {
  width: 60%;
  height: 500px;
  margin: 24rpx auto;
  padding: 24rpx;
  background-color: #ccc;
  box-sizing: border-box;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Popup",
  "usingComponents": {
    "popup": "antd-mini/es/Popup/index",
    "demo-block": "../../components/DemoBlock/index",
    "button": "antd-mini/es/Button/index",
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "switch": "antd-mini/es/Switch/index"
  }
}
```

1.10.6.7. Result

A Result component provides feedback on the results of the previous operation. It is used when you need to inform an end user of the complex processing result of an important operation.

Attribute

Attribute	Type	Required	Default value	Description
type	'success' | 'danger' | 'info' | 'warn' | 'wait'	No	-	The built-in type. success: success. danger: error / danger. info: information warn: warning. wait: wait for processing.
image	string | slot	No	-	A custom image. This property does not take effect if you configured the type property.
title	string | slot	No	-	The main text.
message	string | slot	No	-	The message.
buttons	{text: string; type: 'default' | 'primary' | 'warn' | 'danger' | 'success' | 'light'}[]	No	-	The button type.
className	string	No	-	The class name.

Event

Event name	Description	Type
onButtonTap	Callback fired when the popup is closed.	`(idx: number) => void`

Slot

Name	Description
title	The title.
message	The description.
image	The icon.

Style class

The class name.	Description
amd-result	The style of the entire component.
amd-result-main	The style of the content display area.
amd-result-image	The style of the icon area.
amd-result-buttons	The style of the button area.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <demo-block padding="0" title="Success status">
    <result title="Operation succeed" message="The details of the content can be divided into lines, it is recommended not to exceed two
lines" type="success"/>
  </demo-block>
  <demo-block padding="0" title="Waiting status">
    <result title="pending" message="The details of the content can be divided into lines, it is recommended not to exceed two lines"
type="wait"/>
  </demo-block>
  <demo-block padding="0" title="Prompt status">
    <result title="information prompt" message="The details of the content can be divided into lines, it is recommended not to exceed two
lines" type="info"/>
  </demo-block>
  <demo-block padding="0" title="Prompt status">
    <result title="warning prompt" message="The details of the content can be divided into lines, it is recommended not to exceed two lines" t
ype="warn"/>
  </demo-block>
  <demo-block padding="0" title="Failure status">
    <result title="could not complete the operation" message="The details of the content can be divided into lines, it is recommended not to e
xceed two lines" type="danger"/>
  </demo-block>
  <demo-block padding="0" title="Custom icon">
    <result
      buttons="{{buttons}}"
      image="https://gw.alipayobjects.com/mdn/miniProgram_mendian/afts/img/A*wiFYTo5I0m8AAAAAAAAAABjAQAAAQ/original"
      onButtonTap="handleTabBtn">
      <view slot="title">Title slot</view>
      <view slot="message">Description slot</view>
    </result>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    buttons: [
      {
        text: 'Main operation',
        type: 'primary',
      },
      {
        text: 'Auxiliary operation',
        type: 'default',
      },
    ],
  },
  handleTabBtn(e) {
    my.alert({
      content: `The currently clicked button is ${e + 1}: ${this.data.buttons[e].text}`,
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.amd-result-main {
  margin-bottom: 0;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Result",
  "usingComponents": {
    "result": "antd-mini/es/Result/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.6.8. Toast

A Toast component displays the result of an operation, and can appear automatically without user operation. A Toast component can show a maximum of 24 characters (at most two lines). If the message is too long, it will be truncated.

Attribute

Attribute	Type	Required	Default value	Description

type	string	No	-	The type of the toast. A corresponding icon will be displayed. The value includes success, fail, warning, and loading.
content	string	Yes	-	The content to be displayed by the toast.
icon	Icon type	No	-	The toast icon.
image	string	No	-	The toast icon.
duration	number	No	2000 ms	The duration in which the toast is being displayed on the page. When duration is set to 0, the toast will not be automatically closed.
visible	boolean	Yes	false	Whether to hide the toast.
showMask	boolean	No	false	Whether to display the mask layer.
maskCloseable	boolean	No	false	Whether to close the mask layer upon taps.
className	string	No	-	The class name.

Event

Event name	Description	Type
onClose	Callback fired when the toast is closed.	`(e) => void`

Style class

Class name	Description
amd-toast-wrapper	The style of the entire toast.
amd-toastWithIcon-wrapper	The style of the toast with an icon.
amd-toast-icon	The icon style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<button onTap="handleShowToast" data-index="1" class="button" inline="{{true}}" inlineSize="{{medium}}">
  text only toast
</button>

<button onTap="handleShowToast" data-index="4" class="button" inline="{{true}}" inlineSize="{{medium}}">
  text only toast, the copy is too long
</button>

<button onTap="handleShowToast" data-index="2" class="button" inline="{{true}}" inlineSize="{{medium}}">
  Custom Icon
</button>

<button onTap="handleShowToast" data-index="5" class="button" inline="{{true}}" inlineSize="{{medium}}">
  Custom picture
</button>

<button onTap="handleShowToast" data-index="3" class="button" inline="{{true}}" inlineSize="{{medium}}">
  The prompt lasts for 3s
</button>

<button onTap="handleShowToast" data-index="6" class="button" inline="{{true}}" inlineSize="{{medium}}">
  type = loading
</button>

<toast
  content="{{content1}}"
  visible="{{toast1Show}}"
  duration="{{0}}"
  showMask="{{true}}"
  maskCloseable="{{true}}"
  onClose="handleCloseToast"
  data-index="1"
```

```
  class="toastWrapper"
/>

<toast
  content="{{content4}}"
  visible="{{toast4Show}}"
  data-index="4"
  duration="{{0}}"
  showMask="{{true}}"
  maskCloseable="{{true}}"
  onClose="handleCloseToast"
  class="toastWrapper"
/>

<toast
  content="{{content5}}"
  icon="{{icon2}}"
  visible="{{toast2Show}}"
  duration="{{0}}"
  showMask="{{true}}"
  maskCloseable="{{true}}"
  onClose="handleCloseToast"
  data-index="2"
  class="toastWrapper"
/>

<toast
  content="{{content2}}"
  image="{{image1}}"
  visible="{{toast5Show}}"
  duration="{{0}}"
  showMask="{{true}}"
  maskCloseable="{{true}}"
  onClose="handleCloseToast"
  data-index="5"
  class="toastWrapper"/>

<toast
  content="{{content3}}"
  visible="{{toast3Show}}"
  duration="{{3000}}"
  class="toastWrapper"
  data-index="3"
  onClose="handleCloseToast"
  showMask="{{true}}"
  maskCloseable="{{true}}"
/>

<toast
  content="{{content2}}"
  visible="{{toast6Show}}"
  class="toastWrapper"
  data-index="6"
  type="loading"
  onClose="handleCloseToast"
  showMask="{{true}}"
  duration="{{0}}"
  maskCloseable="{{true}}"
/>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    toastShow: false,
    content1: "The longest copy exceeds 2 lines and can display up to 24 characters.",
    content2: "Loading",
    content3: "The prompt lasts for 3s",
    content4: "This is a very long copy----------------------------------------------------------------------",
    content5: "Welcome to use the new version",
    toast1Show: false,
    toast2Show: false,
    toast3Show: false,
    toast4Show: false,
    toast5Show: false,
    toast6Show: false,
    icon2: "LikeOutline",
    image1: 'https://gw.alipayobjects.com/mdn/rms_5118be/afts/img/A*4NPGQ66arP0AAAAAAAAAAAAAARQnAQ'
  },

  handleShowToast(e) {
    const { index }  = e.target.dataset;
    this.setData({
      toast1Show: false,
      toast2Show: false,
      toast3Show: false,
      toast4Show: false,
      toast5Show: false,
      toast6Show: false,
    })

    this.setData({
      [`toast${index}Show`]: true
    })

    console.log(this.data)
  },

  handleCloseToast(e) {
    const { index }  = e.target.dataset;
    this.setData({
      [`toast${index}Show`]: false
    })
  }
});
```

The following shows an example of the code in the index.acss file:

```
.toastWrapper {

}

.image {
  width: 80rpx;
  height: 80rpx;
}

.button {
  margin-left: 12px;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Toast",
  "usingComponents": {
    "toast": "antd-mini/es/Toast/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.7. Guide and Toast

1.10.7.1. Badge

A badge is displayed with a red dot, a numeric number, or text to tell users the number of pending transactions or updates. A badge appear in the upper right corner with a numeric number, a red dot, or text with eye-catching appeal, typically displaying unread messages count, new functions, or new services.

Attribute

Attribute	Type	Required	Default value	Description

type	'dot' | 'text' | 'bubble' | 'number'	No	'dot'	The badge type. Valid values include: 'dot': red dot. 'number': numeric type (If the count exceeds 99, the count will be automatically changed into ...) 'text': text bubble. 'bubble': bubble state with an arrow.
text	string | number	No	-	The red dot content. Only a red dot is displayed when the value is empty. The value can be a numeric number or text. If the number exceeds 99, it is automatically changed to `...`
placement	'top-left' | 'top-right'	No	'top-right'	The orientation relative to children. left-top: left upper corner. top-right: right upper corner.
stroke	boolean	No	false	Whether the component has strokes.
iconType	string	No	-	The custom icon.
bgColor	string	No	-	The custom background color, which is the same as the value in the CSS code.
className	string	No	-	The class name.

Style class

Class name	Description
amd-badge	The style of the entire component.
amd-badge-inner-text	The internal text style.
amd-badge-text	The text style.
amd-badge-dot	The style of the red dot.
amd-badge-icon	The icon style.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Basic usage">
    <view class="badge-list">
      <badge
        type="dot"
        position="top-right">
        <view class="box"/>
      </badge>
      <badge
        type="text"
        text="New"
        position="top-right">
        <view class="box"/>
      </badge>
      <badge
        type="number"
        text="{{1}}"
        position="top-right">
      <view class="box"/>
      </badge>
        <badge
          type="number"
          text="{{100}}"
          position="top-right">
        <view class="box"/>
      </badge>
      <badge
        type="bubble"
        text="bubble type"
        position="top-right">
        <view class="box"/>
```

```
      </badge>
    </view>
  </demo-block>
  <demo-block title="Bordered">
    <view class="badge-list">
      <badge
        type="dot"
        stroke
        position="top-right">
        <view class="box dark"/>
      </badge>
      <badge
        type="text"
        text="New"
        stroke
        position="top-right">
        <view class="box dark"/>
      </badge>
      <badge
        type="number"
        text="{{1}}"
        stroke
        position="top-right">
      <view class="box dark"/>
      </badge>
        <badge
          type="number"
          text="{{100}}"
          stroke
          position="top-right">
        <view class="box dark"/>
      </badge>
      <badge
        type="bubble"
        text="bubble type"
        stroke
        position="top-right">
        <view class="box dark"/>
      </badge>
    </view>
  </demo-block>
  <demo-block title="Custom colors and offsets">
    <view class="badge-list">
      <badge
        type="dot"
        position="top-left">
        <view class="box"/>
      </badge>
      <badge
        type="dot"
        position="top-right">
        <view class="box"/>
      </badge>
      <badge
        type="dot"
        bgColor="green"
        position="top-right">
        <view class="box"/>
      </badge>
    </view>
  </demo-block>
  <demo-block title="Custom icon">
    <badge
      type="text"
      iconType="GlobalOutline"
      position="top-right">
      <view class="box"/>
    </badge>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {},
});
```

The following shows an example of the code in the index.acss file:

```
.badge-list {
  display: flex;
}
.badge-list .amd-badge {
  margin-right: 32rpx;
}
.box {
  background: #eee;
  width: 40px;
  height: 40px;
  display: block;
  border-radius: 8px;
}

.box.dark {
  background: #666666;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Badge",
  "usingComponents": {
    "badge": "antd-mini/es/Badge/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.7.2. NoticeBar

A NoticeBar component displays a group of notifications on the current page. It is quite eye-catching.

Attribute

Attribute	Type	Required	Default value	Description
mode	'link' | 'closeable'	No	-	The notification type. 'link' indicates that each row of notifications is clickable. 'closeable' indicates that you can tap x to close notifications. If this property is not set, there is no icon on the right.
actions	string[]	No	-	The action. At most two actions can be set. When the actions property exists, the mode property does not take effect.
showIcon	boolean	No	false	Whether to display the icon on the left.
enableMarquee	boolean	No	false	Whether to enable a scroll animation.
loop	boolean	No	false	Whether to loop the scroll animation. This property takes effect when enableMarquee is set to true.
type	'default' | 'info' | 'error' | 'primary'	No	'default'	The notification type. Valid values include 'default': orange; 'info': gray: 'error': red; 'primary': blue.
className	string	No	-	The class name.

Event

Event name	Description	Type
onTap	Callback fired when the icon (arrow or x) on the right of the notice bar is tapped.	`() => void`
onActionTap	Callback fired when the text on the action area on the right is tapped.	`(index: number) => void`

Style class

Class name	Description
amd-notice-bar	The style of the entire notice bar.
amd-notice-bar-default	The style of the entire notice bar.

amd-notice-bar-danger	The style of the entire notice bar.
amd-notice-bar-primary	The style of the entire notice bar.
amd-notice-bar-transparent	The style of the entire notice bar.
amd-notice-bar-content	The style of the internal area.
amd-notice-bar-scroll-left	The style of the shadow gradient on the left.
amd-notice-bar-scroll-right	The style of the shadow gradient on the right.
amd-notice-bar-marquee	The style of the text display area.
amd-notice-bar-operation	The style of the operation area on the right.
amd-notice-bar-operation-icon	The style of the icon on the right action area.
amd-notice-bar-operation-text	The text style of the operation area on the right.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view>
  <demo-block title="Notice bar Semantics " padding="0">
    <notice
      type="default"
      showIcon>
      default
    </notice>
    <white-space/>
    <notice
      type="error"
      showIcon>
      error
    </notice>
    <white-space/>
    <notice
      type="info"
      showIcon>
      info
    </notice>
    <white-space/>
    <notice
      type="primary"
      showIcon>
      primary
    </notice>
  </demo-block>
  <demo-block title="Closeable" padding="0">
    <notice
      showIcon
      onTap="handleClose"
      mode="closeable">
      This notification can be turned off
    </notice>
  </demo-block>
  <demo-block title="extra long scrolling" padding="0">
    <block a:for="{{typeList}}">
      <notice
        type="{{item}}"
        showIcon="{{true}}"
        enableMarquee="{{true}}"
        loop="{{true}}"
        onTap="handleTapLink"
        mode="link">
        Turn on circular scrolling when the text overflows. If the text is not enough, continue to add text to make up the number.
      </notice>
      <white-space/>
    </block>
  </demo-block>
  <demo-block title="Custom" padding="0">
    <notice
      showIcon
      actions="{{actionsText2}}"
      mode="link"
      onTap="handleTapLink"
      onActionTap="handleTapAction">
      Custom right button
    </notice>
  </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    actionsText2: ['No more prompt', 'see details'],
    typeList: ['default', 'error', 'info', 'primary'],
  },
  handleTapAction(e) {
    my.alert({
      title: `The element ${e} in actions is currently clicked. `,
    });
  },
  handleTapLink() {
    my.alert({
      title: 'link type was clicked ',
    });
  },
  handleClose() {
    my.alert({
      title: 'click to close',
    });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Notice",
  "usingComponents": {
    "notice": "antd-mini/es/NoticeBar/index",
    "white-space": "../../components/WhiteSpace/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

1.10.7.3. Tips

A Tips component is a popup that displays extra information related to an element.

Attribute

Attribute	Type	Required	Default value	Description
image	string	No	-	The URL of the image to be used.
title	string	Yes	-	The prompt message.
arrowPosition	'top-left' | 'top-center' | 'top-right' | 'left' | 'right' | 'bottom-left' | 'bottom-center' | 'bottom-right'	No	-	The arrow position. If this property is not passed in, no arrow is used.
buttonText	string	No	-	The button text.
showClose	boolean	No	false	Whether a close button is displayed.
buttonPosition	'right' | 'bottom'	No	'right'	The position of the text button. The default position is right.
className	string	No	-	The class name.

Event

Event name	Description	Type
onButtonTap	Callback fired when the button is tapped.	`() => void`

Style class

Class name	Description
amd-tips	The style of the entire component.
amd-tips-wrap	The internal style of the component.
amd-tips-wrap-pseudo	The style of the pseudo content area.
amd-tips-wrap-pseudo-content	The style of the pseudo content area.
amd-tips-arrow	The arrow style.
amd-tips-wrap-real	The style of the real content area.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <view class="display-area">
    <tips
      title="{{title}}"
      buttonText="{{buttonText}}"
      buttonPosition="{{buttonPosition}}"
      image="{{showImage ? image : ''}}"
      showClose="{{showClose}}"
      arrowPosition="{{tipsArrow[arrowPosition].name}}"
      onButtonTap="onButtonTap"/>
  </view>
  <list radius>
    <list-item>
      Display picture<switch slot="extra" checked="{{showImage}}" onChange="handleChangeShowImage" controlled />
    </list-item>
    <list-item>
      Display close icon<switch slot="extra" checked="{{showClose}}" onChange="handleChangeShowClose" controlled />
    </list-item>
    <list-item>
      Title
      <input-item value="{{title}}" onChange="handleChangeTitle" controlled slot="extra" clear="{{false}}"/>
    </list-item>
    <list-item>
      Button copywriting
      <input-item value="{{buttonText}}" onChange="handleChangeButtonText" controlled slot="extra" clear="{{false}}"/>
    </list-item>
    <radio-group header="arrow position" uid="arrowPosition" value="{{arrowPosition}}" controlled onChange="handleChangeArrowPosition">
      <radio-item a:for="{{tipsArrow}}" value="{{item.value}}" uid="arrowPosition">{{item.name||'no arrow'}}</radio-item>
    </radio-group>
    <radio-group header="button position" uid="buttonPosition" onChange="handleChangeButtonPosition" value="{{buttonPosition}}" controlled>
      <radio-item value="right" uid="buttonPosition">right</radio-item>
      <radio-item value="bottom" uid="buttonPosition">bottom</radio-item>
    </radio-group>
  </list>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    image:
      'https://gw.alipayobjects.com/zos/rmsportal/AzRAgQXlnNbEwQRvEwiu.png',
    tipsArrow: [
      { name: '', value: '0' },
      { name: 'top-left', value: '1' },
      { name: 'top-center', value: '2' },
      { name: 'top-right', value: '3' },
      { name: 'left', value: '4' },
      { name: 'right', value: '5' },
      { name: 'bottom-left', value: '6' },
      { name: 'bottom-center', value: '7' },
      { name: 'bottom-right', value: '8' },
    ],
    title: 'This is a prompt box',
    buttonText: 'operation button',
    showImage: true,
    showClose: true,
    arrowPosition: '2',
    buttonPosition: 'right',
  },
  handleChangeShowImage(checked) {
    this.setData({ showImage: checked });
  },
  handleChangeShowClose(checked) {
    this.setData({ showClose: checked });
  },
  handleChangeTitle(value) {
    this.setData({ title: value });
  },
  handleChangeButtonText(value) {
    this.setData({ buttonText: value });
  },
  handleChangeButtonPosition(value) {
    this.setData({ buttonPosition: value });
  },
  handleChangeArrowPosition(value) {
    this.setData({ arrowPosition: value });
  },
  handleClose() {
    my.alert({
      title: 'Close tab was clicked',
      content: 'Tips component closed',
    });
  },
  handleTapBtn() {
    my.alert({
      title: 'The button was clicked',
      content: 'The onButtonTap in the page was clicked',
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.display-area {
  min-height: 100px;
  padding: 40rpx 12px 12px;
  box-sizing: border-box;
  overflow: hidden;
}

.demo {
  padding-bottom: 40rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Tips",
  "usingComponents": {
    "list": "antd-mini/es/List/index",
    "list-item": "antd-mini/es/List/ListItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "tips": "antd-mini/es/Tips/index",
    "switch": "antd-mini/es/Switch/index",
    "radio-group": "antd-mini/es/RadioGroup/index",
    "radio-item": "antd-mini/es/RadioGroup/RadioItem/index"
  }
}
```

Slot

The following shows an example of the code in the index.axml file:

```
<tips arrowPosition="bottom-center" showClose="{{false}}">
    Default slot
</tips>
```

The following shows an example of the code in the index.js file:

```
Page({});
```

The following shows an example of the code in the index.json file:

```
{
    "defaultTitle": "TipsSlot",
    "usingComponents": {
      "tips": "antd-mini/es/Tips/index"
    }
  }
```

Closure component

The following shows an example of the code in the index.axml file:

```
<view class="demo">

    <tips title="Controlled shutdown" showClose="{{true}}" onClose="onClose" visible="{{visible}}" />

    <button class="btn" type="primary" onTap="onTap">
        {{visible === true ? 'Close' : 'Open'}}
    </button>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    visible: true,
  },
  onTap() {
    this.setData({
      visible: !this.data.visible,
    });
  },
  onClose() {
    my.alert({
      content: 'Component shutdown',
    });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.btn {
  margin: 24rpx 0;
  display: block;
}
.demo {
  display: flex;
  flex-direction: column;
  padding: 24rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "usingComponents": {
    "button": "antd-mini/es/Button",
    "tips": "antd-mini/es/Tips"
  }
}
```

1.10.8. Experimental components

1.10.8.1. Form

A Form component is a high performance control with data domain management functionalities. It includes data entry, verification, and corresponding styles and is used for creating an entity or collecting information.

Important

- To use a Form component in a mini-program project, you need to enable `Component2`.
- When you use a Form component with the `a:for` command, we recommend that you specify `key`. Otherwise, an error may occur.
- The value of form in the Form tag must be unique and the same as that in the internal FormItem tag. The value of name in the internal FormItem tag must be unique.
- When you use this component together with the form component from the component library, the value of `mode` of the form component must be `form`.

Attribute

Form

Attribute	Type	Required	Default value	Description
form	string	Yes	-	The uid of the form.
initialValues	Record<sring, any>	No	-	The initial value of the form.
position	'horizontal' | 'vertical'	No	'horizontal'	The layout of the form.
requiredMarkStyle	'asterisk' | 'text-required' | 'text-optional'	No	'asterisk'	The styling used to mark a required or an optional field. Asterisk: Use a red asterisk to mark a required field. Text-required: Use the word "Required" to mark a required field. Text-optional: Use the word "Optional" to mark an optional field.
className	string	No	-	The class name.

FromGroup

Attribute	Type	Required	Default value	Description
header	string	No	-	The name of the FormGroup component.
radius	boolean	No	false	Whether the FormGroup component has a round corner.
className	string	No	-	The class name.

FormItem

Attribute	Type	Required	Default value	Description
form	string	Yes	default	The uid of the form.
name	string	Yes	default	The uid of the field.
label	string	No	-	The field name.
position	'horizontal' | 'vertical'	No	'horizontal'	The layout of the FormItem component, whose priority is higher than that of position in Form.
arrow	boolean	No	false	The arrow on the right of the FormItem component.
required	boolean	No	false	Whether a label is required. This property must be set if you want to display a label.
help	string	No	-	The description of a label.
className	string	No	""	The class name.

Event

Form

Event name	Description	Type
onValuesChange	Callback fired when a field is updated.	`( changedFields: Record<string, any>, allFields: Record<string, any> ) => void`
onFinish	Callback fired when the form is submitted.	`( changedFields: Record<string, any>, allFields: Record<string, any> ) => void`

Slot

FormGroup

Name	Description
header	The header.

FormItem

Name	Description
extra	Extra content about a form item.

Instantiation methods

Form

Event name	Description	Type
getComponentIns	Obtains a component example. The value is the same as the default return value of ref.	() => Component
setFieldsValue	Sets the value of a form field.	(formName: `string` , fieldsVals: `Record<string, any>`) => void

Description

Form

Class name	Description
amd-form	The style of the entire form.
amd-form-footer	The style of the footer.

FormGroup

Class name	Description
amd-form-group-header	The style of the header.
amd-form-group-radius	The style of the rounded corner.

FormItem

Class name	Description
amd-form-item-label	The style of a label.
amd-form-item-field	The style of a field.
amd-form-item-extra	The style of the extra content.
amd-form-item-arrow	The style of an arrow.

Code sample

Basic usage

The following shows an example of the code in the index.axml file:

```
<view class="demo">
    <demo-block title="Horizontal layout form" padding="0">
        <form
            form="{{form}}"
            initialValues="{{initialValues}}"
            onFinish="handleSubmit"
            onValuesChange="handleValuesChange">
            <form-item
                label="Name"
                name="account"
                required
                help="Explanation of the name"
                form="{{form}}">
                <input-item mode="form" placeholder="Please enter the name"/>
            </form-item>
            <form-item
                label="Address"
                name="address"
                form="{{form}}">
                <input-item mode="form" password placeholder="Please enter the address"/>
            </form-item>
            <form-item
                label="Quantity"
                name="quantity"
                form="{{form}}">
                <stepper step="{{1}}"  min="{{0}}" mode="form"/>
            </form-item>
            <form-item
                label="Home delivery"
                name="needDelivery"
                form="{{form}}">
                <switch mode="form"/>
            </form-item>
            <button
                slot="footer"
                type="primary"
                mode="form"
                form="{{form}}"
                htmlType="submit">Submit
            </button>
        </form>
    </demo-block>
    <demo-block title="Vertical layout form" padding="0">
        <form
            form="{{formV}}"
            initialValues="{{initialValues}}"
            onFinish="handleSubmit"
            onValuesChange="handleValuesChange">
            <form-item
                label="Name"
                position="vertical"
                name="account"
                required
                form="{{formV}}">
                <input-item mode="form" placeholder="Please enter the name"/>
            </form-item>
            <form-item
                label="Address"
                position="vertical"
                name="address"
                form="{{formV}}">
                <input-item mode="form" password placeholder="Please enter the address"/>
            </form-item>
            <form-item
                label="Quantity"
                position="vertical"
                name="quantity"
                form="{{formV}}">
                <stepper step="{{1}}"  min="{{0}}" mode="form"/>
            </form-item>
            <form-item
                label="Home delivery"
                position="vertical"
                name="needDelivery"
                form="{{formV}}">
                <switch mode="form"/>
            </form-item>
            <button
                slot="footer"
                type="primary"
                mode="form"
                form="{{formV}}"
                htmlType="submit">Submit
            </button>
        </form>
```

```
    </demo-block>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    form: 'form',
    formV: 'formV',
    initialValues: {
      quantity: 1,
      needDelivery: false,
    },
  },
  handleValuesChange(value, values) {
    console.log(value, values);
  },
  handleSubmit(e) {
    my.alert({ title: 'Submit', content: JSON.stringify(e) });
  },
});
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index",
    "switch": "antd-mini/es/Switch/index",
    "stepper": "antd-mini/es/Stepper/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Combined form component

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <form
    form="{{form}}"
    onFinish="handleSubmit"
    initialValues="{{initialValues}}"
    onValuesChange="handleValuesChange">
    <form-item
      required
      label="input"
      name="input"
      form="{{form}}">
      <input-item mode="form" placeholder="Please enter"/>
    </form-item>
    <form-item
      required
      label="password"
      name="password"
      rules="{{[{ pattern: /\d{6}/, message: 'The password must be 6 digits' },]}}"
      form="{{form}}">
      <input-item mode="form" placeholder="Please enter" password/>
    </form-item>
    <form-item
      label="stepper"
      name="stepper"
      form="{{form}}">
      <stepper
        mode="form"
        min="{{0}}"
        max="{{100}}"
        step="{{1}}"/>
    </form-item>
    <form-item
      label="switch"
      name="switch"
      form="{{form}}">
      <switch
        mode="form"
      />
    </form-item>
    <form-item
      label="picker"
      labelWidth="110px"
      name="picker"
      form="{{form}}">
      <picker
```

```
        pickerholder="Please select"
        data="{{pickerList}}"
        mode="form"
        onFormat="formatValue"
        onDismiss="cancelPicker"
      >
        <view slot="title">
          <text style="color: red;">Picker</text> selector</view>
      </picker>
    </form-item>
    <form-item
      label="selector"
      name="selector"
      position="vertical"
      form="{{form}}">
      <selector
        items="{{selectorItems}}"
        multiple
        mode="form" />
    </form-item>
    <form-item
      label="radio"
      required
      name="radio"
      position="vertical"
      form="{{form}}">
      <radio-group mode="form" position="horizontal">
        <radio-item value="a1">choice one</radio-item>
        <radio-item value="a2">choice two</radio-item>
        <radio-item value="a3">choice three</radio-item>
      </radio-group>
    </form-item>
    <form-item
      label="checkboxGroup"
      required
      name="checkboxGroup"
      position="vertical"
      form="{{form}}">
      <checkbox-group mode="form" position="horizontal">
        <checkbox-item value="a1">checkbox 1</checkbox-item>
        <checkbox-item value="a2">checkbox 2</checkbox-item>
        <checkbox-item value="a3">checkbox 3</checkbox-item>
      </checkbox-group>
    </form-item>
    <button
      slot="footer"
      type="primary"
      mode="form"
      form="{{form}}"
      htmlType="submit">Submit
    </button>
  </form>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    form: 'form',
    initialValues: {
      stepper: 20,
      switch: false,
      picker: ['2012', '12', 12],
    },
    selectorItems: [
      {
        text: 'option 1',
        value: '1',
      },
      {
        text: 'option 2',
        value: '2',
      },
      {
        text: 'option 3',
        value: '3',
      },
      {
        text: 'option 4',
        value: '4',
      },
      {
        text: 'option 5',
        value: '5',
      },
    ],
    pickerList: [
      [
        '2011',
        '2012',
        '2013',
        '2014',
        '2015',
        '2016',
        '2017',
        '2018',
        '2019',
        '2020',
        '2021',
        '2022',
      ],
      ['01', '02', '03', '04', '05', '06', '07', '08', '09', '10', '11', '12'],
      [
        1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
        21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31,
      ],
    ],
  },
  formatValue(v) {
    return v && v.join('/') || '';
  },
  handleValuesChange(value, values) {
    console.log(value, values);
  },
  handleSubmit(e) {
    my.alert({ title: 'Submit', content: JSON.stringify(e) });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo {
  padding: 24rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index",
    "checkbox": "antd-mini/es/Checkbox/index",
    "stepper": "antd-mini/es/Stepper/index",
    "picker": "antd-mini/es/Picker/index",
    "checkbox-group": "antd-mini/es/CheckboxGroup/index",
    "checkbox-item": "antd-mini/es/CheckboxGroup/CheckboxItem/index",
    "radio-group": "antd-mini/es/RadioGroup/index",
    "radio-item": "antd-mini/es/RadioGroup/RadioItem/index",
    "switch": "antd-mini/es/Switch/index",
    "selector": "antd-mini/es/Selector/index",
    "form-group": "antd-mini/es/Form/FormGroup/index"
  }
}
```

Form grouping

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <form form="{{form}}">
    <form-group radius header="Basic information">
      <form-item
        required
        label="Name"
        name="name"
        form="{{form}}">
        <input-item mode="form" placeholder="Please enter the name"/>
      </form-item>
      <form-item
        required
        label="Address"
        name="address"
        form="{{form}}">
        <input-item mode="form" placeholder="Please enter the address"/>
      </form-item>
    </form-group>
    <form-group radius header="Contact details">
      <form-item
        required
        label="Mobile phone number"
        name="phone"
        form="{{form}}">
        <input-item mode="form" placeholder="Please enter the mobile phone number"/>
      </form-item>
      <form-item
        required
        label="Email"
        name="email"
        form="{{form}}">
        <input-item mode="form" placeholder="Please enter the email"/>
      </form-item>
    </form-group>
  </form>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    form: 'form',
  },
  handleValuesChange(value, values) {
    console.log(value, values);
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo {
  padding: 24rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "form-group": "antd-mini/es/Form/FormGroup/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index",
    "switch": "antd-mini/es/Switch/index",
    "checkbox": "antd-mini/es/Checkbox/index",
    "radio-group": "antd-mini/es/RadioGroup/index",
    "radio-item": "antd-mini/es/RadioGroup/RadioItem/index",
    "stepper": "antd-mini/es/Stepper/index",
    "selector": "antd-mini/es/Selector/index"
  }
}
```

Dynamic form

The following shows an example of the code in the index.axml file:

```
<view class="demo">
    <form
        form="{{form}}"
        initialValues="{{initialValues}}"
        onFinish="handleSubmit"
        onValuesChange="handleValuesChange">
        <form-item
            label="Account"
            name="account"
            required
            form="{{form}}">
            <input-item mode="form" placeholder="Please enter the account"/>
        </form-item>
        <form-item
            label="Login method"
            name="type"
            position="vertical"
            form="{{form}}">
            <radio-group mode="form">
                <radio-item value="password">Password</radio-item>
                <radio-item value="code">Verification code</radio-item>
            </radio-group>
        </form-item>
        <form-item
            a:if="{{values.type==='password'}}"
            label="Password"
            required
            name="password"
            form="{{form}}">
            <input-item mode="form" password placeholder="Please enter the password"/>
        </form-item>
        <form-item
            a:if="{{values.type==='code'}}"
            label="Verification code"
            required
            name="code"
            form="{{form}}">
            <input-item mode="form" password placeholder="Please enter the verification code"/>
        </form-item>
        <button
            slot="footer"
            type="primary"
            mode="form"
            form="{{form}}"
            htmlType="submit">Login
        </button>
    </form>
</view>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    form: 'form',
    initialValues: {
      type: 'password',
    },
    values: {
      type: 'password',
    },
  },
  handleValuesChange(value, values) {
    console.log(value, values);
    this.setData({ values });
  },
  handleSubmit(e) {
    my.alert({ title: 'Submit', content: JSON.stringify(e) });
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo {

}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index",
    "radio-group": "antd-mini/es/RadioGroup/index",
    "radio-item": "antd-mini/es/RadioGroup/RadioItem/index"
  }
}
```

Display style of required fields

The following shows an example of the code in the index.axml file:

```
<view class="tip">
    Form supports three mandatory and optional display styles
</view>
<demo-block title="Asterisk" padding="0">
    <form form="{{form1}}" requiredMarkStyle="asterisk">
        <form-item
            position="vertical"
            label="Name"
            name="name"
            required
            form="{{form1}}">
            <input-item placeholder="Please enter the name"/>
        </form-item>
        <form-item
            position="vertical"
            label="Address"
            name="address"
            help="Detailed address"
            form="{{form1}}">
            <input-item placeholder="Please enter the address"/>
        </form-item>
    </form>
</demo-block>
<demo-block title="Text-Required" padding="0">
    <form form="{{form2}}" requiredMarkStyle="text-required">
        <form-item
            position="vertical"
            label="Name"
            name="name"
            required
            form="{{form2}}">
            <input-item placeholder="Please enter the name"/>
        </form-item>
        <form-item
            position="vertical"
            label="Address"
            name="address"
            help="Detailed address"
            form="{{form2}}">
            <input-item placeholder="Please enter the address"/>
        </form-item>
    </form>
</demo-block>
<demo-block title="Text-Optional" padding="0">
    <form form="{{form3}}" requiredMarkStyle="text-optional">
        <form-item
            position="vertical"
            label="Name"
            name="name"
            required
            form="{{form3}}">
            <input-item placeholder="Please enter the name"/>
        </form-item>
        <form-item
            position="vertical"
            label="Address"
            name="address"
            help="Detailed address"
            form="{{form3}}">
            <input-item placeholder="Please enter the address"/>
        </form-item>
    </form>
</demo-block>
```

The following shows an example of the code in the index.js file:

```
Page({
  data: {
    form1: 'form1',
    form2: 'form2',
    form3: 'form3',
  },
});
```

The following shows an example of the code in the index.acss file:

```
.tip {
  background: #fff;
  padding: 24rpx;
  font-size: 28rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "demo-block": "../../components/DemoBlock/index"
  }
}
```

Usage of the instantiation methods

The following shows an example of the code in the index.axml file:

```
<view class="demo">
  <form
      form="{{form}}"
      ref="getForm"
      onFinish="handleSubmit"
      initialValues="{{initialValues}}"
      onValuesChange="handleValuesChange">
      <form-item
          label="Account"
          name="account"
          form="{{form}}">
          <input-item mode="form" placeholder="Please enter the account"/>
      </form-item>
      <form-item
          label="Password"
          name="password"
          form="{{form}}">
          <input-item mode="form" password placeholder="Please enter the password"/>
      </form-item>
      <view slot="footer">
        <button
            slot="footer"
            type="primary"
            mode="form"
            form="{{form}}"
            htmlType="submit">Login
        </button>
      </view>
  </form>
  <button onTap="handleSetValue" className="btn">Reset</button>
</view>
```

The following shows an example of the code in the index.js file:

```
const initialValues = {
  account: '',
  password: '',
};
Page({
  data: {
    form: 'form',
    initialValues,
  },
  handleValuesChange(value, values) {
    console.log(value, values);
  },
  handleSubmit(e) {
    my.alert({ title: 'Submit', content: JSON.stringify(e) });
  },
  getForm(ref) {
    this.formRef = ref;
  },
  handleSetValue() {
    this.formRef.setFieldsValue(this.data.form, initialValues);
  },
});
```

The following shows an example of the code in the index.acss file:

```
.demo {
}
.btn {
  margin: 0 12px;
}
```

The following shows an example of the code in the index.json file:

```
{
  "defaultTitle": "Form",
  "usingComponents": {
    "form": "antd-mini/es/Form/index",
    "form-item": "antd-mini/es/Form/FormItem/index",
    "input-item": "antd-mini/es/InputItem/index",
    "button": "antd-mini/es/Button/index"
  }
}
```

1.10.8.2. SafeArea

A SafeArea component is a component in the safe area of the notch screen.

Attribute

Attribute	Type	Required	Default value	Description
position	'bottom' | 'top' | 'both'	No	'both'	The location of the safe area.
className	string	No	''	The class name.

Style class

Class name	Description
amd-safe-area	The style of the entire safe area.
amd-safe-area-top	The style of the safe area on the top.
amd-safe-area-bottom	The style of the safe area on the bottom.

Code sample

The following shows an example of the code in the index.axml file:

```
<safe-area className="container">
I celebrate myself, and sing myself,

And what I assume you shall assume,

For every atom belonging to me as good belongs to you.

I loafe and invite my soul,

I lean and loafe at my ease observing a spear of summer grass.

My tongue, every atom of my blood, form’d from this soil, this air,

Born here of parents born here from parents the same, and their parents the same,

I, now thirty-seven years old in perfect health begin,

Hoping to cease not till death.

Creeds and schools in abeyance,

Retiring back a while sufficed at what they are, but never forgotten,

I harbor for good or bad, I permit to speak at every hazard,

Nature without check with original energy.
</safe-area>
```

The following shows an example of the code in the index.js file:

```
Page({});
```

The following shows an example of the code in the index.acss file:

```
.container {
  line-height: 48rpx;
}
```

The following shows an example of the code in the index.json file:

```
{
    "defaultTitle": "SafeArea",
    "usingComponents": {
      "safe-area": "antd-mini/es/SafeArea/index"
    },
    "transparentTitle": "always"
  }
```

1.11. API

1.11.1. Overview

The mPaaS framework provides developers with more JSAPI and OpenAPI capabilities, and also provide users with a variety of convenient services through Mini Program.

Description

The API starting with `my.on` is used to monitor system events and receives a `callback` function as a parameter. When the event is triggered, the `callback` function will be called. The `callback` function can be passed to the corresponding API starting with `my.off` to cancel the monitoring relationship. If you call the API starting with `my.off` directly, all monitoring relationships will be canceled. 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 result of the interface call. The callback result is generally an object unless otherwise specified. If there is an `error` / `errorMessage`, it means the call 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);
})
```

1.11.2. 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.

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.
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.
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.
Return value	You can bind `WebViewContext` with a web-view component and implement some features through `webviewId`.

1.11.3. UI

1.11.3.1. 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
        })
      }
    })
  }
});
```

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?

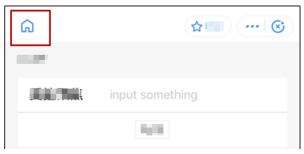
A: This color is by default and cannot be set.

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.

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.

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;
}
```

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 the menu page within the mini program capsule button be customized?

  A: Currently, the menu page in the mini program capsule button does not support custom modification. For details, please refer to Right-side control section.

- **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.

1.11.3.2. tabBar

How to use 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.

> **Note**
> - The page reached through page jump (`my.navigateTo`) or page redirection (`my.redirectTo`), even if it is a page defined in the tabBar configuration, will not display the bottom tab bar.
> - The first page of the `tabBar` must be the home page.

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.

Configurations of item

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.

> **Important**
> 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.

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.
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.

> **Important**
> 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.

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.

> **Important**
> 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.

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.

> **Important**
> 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.

> **Important**
> 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.

> **Important**
> 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.

1.11.3.3. Route

my.switchTab

This interface is used to jump to the specified tab page (tabBar) and close all other non-tab pages.

> **Note**
> - If the mini program is a multi-tab application, that is, the bottom bar of the client window can switch pages, then the tab configuration item can be used to specify the presentation of the tab bar and the corresponding page displayed when the tab is switched.
> - The page reached by page jump (my.navigateTo) or page redirection (my.redirectTo) will not display the bottom tab bar, even if it is a page defined in the tab configuration.
> - The first page of the tab must be the home page.

For more information, see Route related FAQs.

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.

Sample code

```
my.reLaunch({
  url: '/page/index'
})
```

Parameters

The parameters are of Object type, with attributes as follows:

Attribute	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.

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:

Attribute	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.

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:

Attribute	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.
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.

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:

Attribute	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.

1.11.3.4. 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 success

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 success

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.
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
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”. **Important** 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.
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}`
        });
      },
    });
  },
});
```

1.11.3.5. Pull down refresh

This API is supported in mPaaS 10.1.32 and later versions.

onPullDownRefresh

> **Note**
> 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.

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

> **Note**
> 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.

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.

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()
```

1.11.3.6. Contact

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
    });
  },
});
```

1.11.3.7. 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.
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.

Custom city list: 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.
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 success

> **Note**
> 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.
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 of failure

Name	Type	Description
error	String	The error code.
errorMessage	String	The error description.

Return values of success

Name	Type	Description
locatedCityName	String	The name of the currently located city.

Error code

Error code	Description	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)}`,
        })
      },
    })
  },
})
```

1.11.3.8. 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 than1.1.1 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 than1.1.1 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.
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 of 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;
}
```

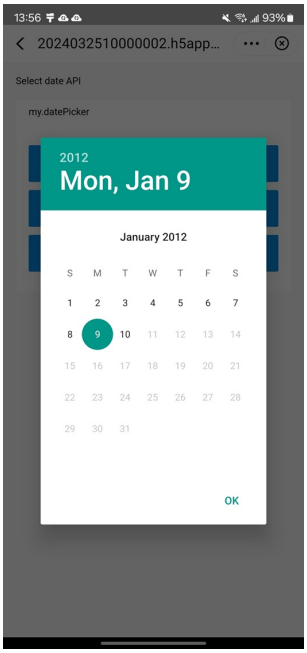
Note

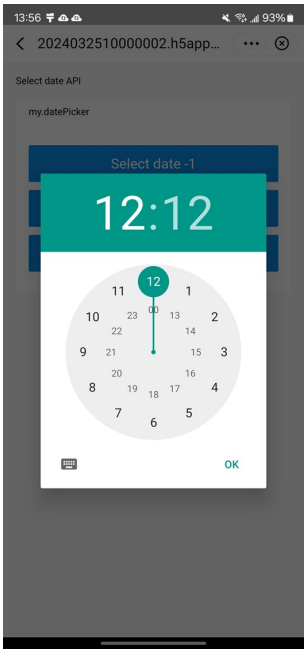
For iOS users who use the baseline version V10.1.68.35 and later, the latest style of time picker can be set by creating the `AUImplDatePicker` class and overriding the `userNewYearDateAndTime` method to return the `YES` method.

```
@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:





1.11.3.9. 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.

> **Important**
> The `export` method will clear the previous animation operation after it is called.

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
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

Add the following code to the `axml` file:

```
<view animation="{{animationInfo}}" style="background:yellow;height:100rpx;width:100rpx"></view>
```

Add the following code to the `js` file:

```
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()
    });
  }
})
```

1.11.3.10. 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	Minimum version of the base library
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.	-
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	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 `<number>`	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.
complete	Function	No	Callback function upon call completion.

Callback parameters of 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	Minimum version of the base library
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

- **Code example 1**

```
const ctx = my.createCanvasContext('awesomeCanvas');
ctx.setFillStyle('blue');
ctx.fillRect(20, 20, 180, 80);
ctx.draw();
ctx.fillRect(60, 60, 250, 120);
// Retain the last drawing result
ctx.draw(true);
```

- **Code example 2**

```
//.js
const ctx = my.createCanvasContext('awesomeCanvas')
ctx.setFillStyle('blue')
ctx.fillRect(20, 20, 180, 80)
ctx.draw()
ctx.fillRect(60, 60, 250, 120)
// // Do not retain the last drawing result
ctx.draw(false)
```

1.11.3.11. 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();
    }
  },
});
```

1.11.3.12. Scroll

my.pageScrollTo

This interface is used to scroll to the target position of the page.

> **Note**
> `my.pageScrollTo` is supported by mPaaS 10.1.32 and above.

- The priority of scrollTop is higher than selector.
- When using my.pageScrollTo to jump to the top of the mini program, the scrollTop value must be set to greater than 0 to achieve the jump.

Parameters description

Attribute	Type	Default value	Required	Description	Minimumu version

scrollTop	Number	-	No	Scroll to the target position of the page, in px. When using my.pageScrollTo to jump to the top of the mini program, the scrollTop value must be set to greater than 0 to achieve the jump.	-
duration	Number	0	No	The duration of the scroll animation, in ms.	1.20.0
selector	String	-	No	Selector.	1.20.0
success	Function	-	No	The callback function of successful interface call.	-
fail	Function	-	No	Callback function when the interface call fails.	-
complete	Function	-	No	The callback function for when the interface call ends (will be executed whether the call succeeds or fails).	-

Selector syntax

When the `selector` parameter is passed in, the framework will execute `document.querySelector(selector)` to select the target node.

Code example

```
<!-- API-DEMO page/API/page-scroll-to/page-scroll-to.axml-->
<view class="page">
  <view class="page-description">Page Scrolling API</view>
  <view class="page-section">
    <view class="page-section-title">
      my.pageScrollTo
    </view>
    <view class="page-section-demo">
      <input type="text" placeholder="key" name="key" value="{{scrollTop}}" onInput="scrollTopChange"></input>
    </view>
    <view class="page-section-btns">
      <view onTap="scrollTo">Page Scrolling</view>
    </view>
  </view>
  <view style="height:1000px"/>
</view>
```

```
// API-DEMO page/API/page-scroll-to/page-scroll-to.js
Page({
  data: {
    scrollTop: 0,
  },
  scrollTopChange(e) {
    this.setData({
      scrollTop: e.detail.value,
    });
  },
  onPageScroll({ scrollTop }) {
    console.log('onPageScroll', scrollTop);
  },
  scrollTo() {
    my.pageScrollTo({
      scrollTop: parseInt(this.data.scrollTop),
            duration: 300,
    });
  },
});
```

1.11.3.13. 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.
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.

> **Important**
> `exec` must be called after `Page onReady`.

1.11.3.14. 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 option selector component does not support interactions among level 2 data.

Component 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 of 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.

1.11.3.15. Multilevel select

my.multiLevelSelect(Object)

> **Note**
> The multilevel selection interface is only supported in mPaaS 10.1.32 and later versions.

Multilevel 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": "multilevel selector"
}
```

```
<!-- API-DEMO page/API/multi-level-select/multi-level-select.axml-->
<view class="page">
  <view class="page-description">multilevel 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">multilevel selector</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/multi-level-select/multi-level-select.js
Page({
  openMultiLevelSelect() {
    my.multiLevelSelect({
        title: 'multilevel selector',//Title for multilevel 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: "Longxiangqiao Subdistrict"
                        }
                    ]
                }
            ]//sub-level data list
        }],//data list
        success:(res)=>{
            my.alert({title:JSON.stringify(res)})
        }
    });
  }
})
```

1.11.3.16. Set background color

my.setBackgroundColor

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

Dynamically set window background color.

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

> **Note**
> 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.

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.
})
```

1.11.3.17. Set page pulldown

Set whether to support pulldown on the page (supported by default on Mini Program pages).

my.setCanPullDown

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

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
})
```

1.11.3.18. Settings

my.setOptionMenu

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

This interface is used to configure additional icons on the `optionMenu` navigation bar. Clicking any of these icons will trigger `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

- Due to the 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',
});
```

1.11.4. Multimedia

1.11.4.1. Image

my.chooseImage

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

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.

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).

Return value of success

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/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.previewImage

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

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.

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/TB1h9xxIFXXXXbKXXXXXXXXXXXX.jpg'
      ],
    });
  },
  saveImage() {
    my.saveImage({
      url: 'https://img.alicdn.com/tps/TB1sXGYIFXXXXc5XpXXXXXXXXXX.jpg',
      showActionSheet: true,
      success: () => {
        my.alert({
          title: 'Saved',
        });
      },
    });
  }
});
```

my.saveImage

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

This interface is used to save Web images to your phone's photo album.

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.
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/TB1sXGYIFXXXXc5XpXXXXXXXXX.jpg',
      showActionSheet: true,
      success: () => {
        my.alert({
          title: 'Saved',
        });
      },
    });
  }
});
```

my.compressImage

> **Note**
>
> This interface is supported by basic library version 1.4.0 and above, also supported by mPaaS version 10.1.60 and above. Lower versions need to be compatible. For more information, see Compatibility process.

This interface is used to compress the images.

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).

Return value of success

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

> **Note**
> This interface is supported by basic library version 1.4.0 and above, also supported by mPaaS version 10.1.32 and above. Lower versions need to be compatible. For more information, see Compatibility process.

This interface is used to get image information.

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.
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed).

Return value success

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))
      }
    })
```

1.11.4.2. 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 version 1.14.1 or later versions

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();
  },

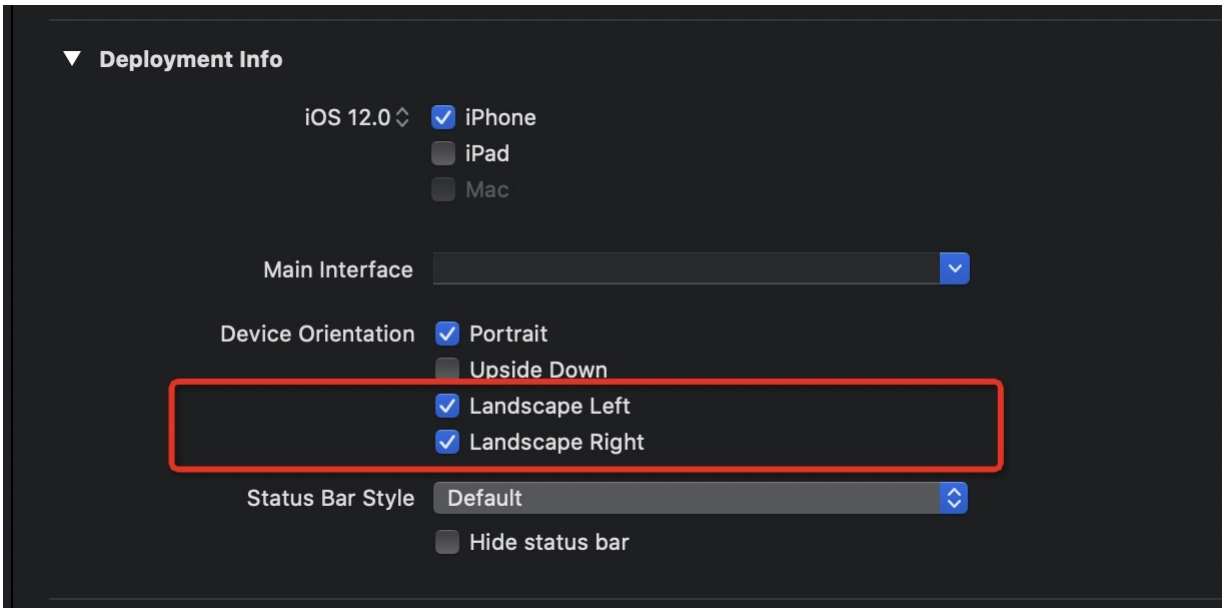
  mute() {
    this.videoContext.mute(false);
  },


});
```

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.
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.



1.11.5. Cache

The caching mechanism is local caching based on appId and userId, so you need to use the `MPLogger.setUserId(String userId);` method to set the whitelist ID before using the cache.

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 corresponds to.

> **Note**
> - mPaaS 10.1.32 and later versions support this interface.
> - This is an asynchronous interface.
> - The storage of the embedded webview is supported to be isolated from the storage of the mini program, and the data stored with the specified key in the embedded webview will not overwrite the data corresponding to the same key of the mini program itself.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is a synchronous interface.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is an asynchronous interface.
> - Supports the isolation of embedded webview internal cache and mini program cache. Obtaining the cache of the specified key of the embedded webview will not return the cached data under the same key of the mini program at the same time.

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)

Return value of success

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**
> - mPaaS 10.1.32 and later versions support this interface.
> - This is an asynchronous interface.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is an asynchronous interface.
> - When removing the stored data of the embedded webview, the stored data of the current mini program will not be removed.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is a synchronous interface.

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**
> - mPaaS 10.1.32 and later versions support this interface.
> - This is an asynchronous interface.
> - When clearing the storage of the embedded webview, the storage data of the current mini program itself will not be cleared at the same time.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is a synchronous interface.

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is an asynchronous interface.
> - Obtaining information about the current storage in the embedded webview will not obtain information about the storage of the current mini program.

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)

Return value success

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**
> - This interface is supported by mPaaS 10.1.32 and above.
> - This is a synchronous interface.

Return value success

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.

1.11.6. File

my.saveFile

> **Note**
> - This interface is supported by basic library version 1.13.0 and above, also supported by mPaaS version 10.1.32 and above.
> - Lower versions need to be compatible. For more information, see compatibility process.

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 of 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.

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 of success

Parameter	Type	Description
size	Number	File size
digest	String	Digest result.

Sample code

```
my.getFileInfo({
    apFilePath:'https://resource/apm1953bb093ebd2834530196f50a4413a87.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 of 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 of 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
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')
          }
        })
        }
    });
```

1.11.7. 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**
> - Basic library 1.11.0 and above support this interface and can use `my.canIUse('request')` for compatibility processing. For more details, please refer to Mini Program Base Library.
> - This interface is supported by mPaaS 10.1.60 and above.
> - Interface `my.httpRequest` will be deprecated, please use interface `my.request` instead.
> - The default value of `my.request` 's request header is **`{'content-type': 'application/json'}`** ,not `{'content-type': 'application/x-www-form-urlencoded'}` . In addition, the key and value in the request header object must be of type String.

For more information, see my.request FAQ.

Instructions for use:

- You need to turn on the **Permission control switch** of the mini program 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.

Parameter

Name	Type	Required	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.

Return value of success

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

> **Note**
> 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`).

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)

Return value success

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

> **Note**
> This API is supported in mPaaS 10.1.32 and later versions.

Download the file resource to local position.

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)

Return value success

Name	Type	Description
apFilePath	String	Temporary storage location of files

Error code

Error code	Description	Solution
12	Download failed	It is recommended to check the network and server
13	No permission	It is recommended to check permissions
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

> **Note**
> 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.

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

Note

This API is supported in mPaaS 10.1.60 and later versions.

Listen to the WebSocket connection establishment event.

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

> **Note**
> 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

> **Note**
> This API is supported in mPaaS 10.1.60 and later versions.

Listen to WebSocket errors.

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

> **Note**
> 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

> **Note**
> 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.

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

> **Note**
> 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

> **Note**
> 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

> **Note**
> This API is supported in mPaaS 10.1.60 and later versions.

Close the WebSocket connection.

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

> **Note**
> This API is supported in mPaaS 10.1.60 and later versions.

Listen to WebSocket shutdown.

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 callbacks 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 callbacks 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 callbacks 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 the background
      success: (res) => {
        my.showToast({
          content: 'success', // Text content
        });
      },
      fail:()=>{
        my.showToast({
          content: 'fail', // Text content
        });
      }
    });
  },
```

my.offSocketClose

> **Note**
> 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);
```

my.rpc(Object)

Mini program framework dedicated rpc interface, and mini program gateway interface.

Parameter description

Name	Type	Required	Description
operationType	String	Yes	Gateway operationtype.
requestData	Array	No	Gateway requests data.
success	Function	No	Invoke the successful callback function.
fail	Function	No	Callback function that failed to be called.
complete	Function	No	The callback function for the end of the call (both the call succeeds and fails).

Code sample

The case is for reference only, it is recommended to use your own address for testing.

```
my.rpc({
      operationType: 'com.xx.xx',
      requestData: [{
        "param1":"",
        "param2":0
      }],
      success: res => {
      },
      fail: res => {
      }
});
```

1.11.8. Device

1.11.8.1. Can I use

my.canIUse(String)

> **Note**
> 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 program.

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.

1.11.8.2. 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.

1.11.8.3. System info

my.getSystemInfo

> **Note**
> This API is supported in mPaaS 10.1.32 and later versions.

This interface is used for system information.

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)

Return value success

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
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	Return value of model
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

> **Note**
> 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(),
    });
  },
})
```

1.11.8.4. Network status

my.getNetworkType

> **Note**
> This API is supported in mPaaS 10.1.32 and later versions.

This API is used to get the current network status.

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)

Return value success

Name	Type	Description
networkAvailable	Boolean	Whether the network is available
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)

> **Note**
> 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

> **Note**
> This API is supported in mPaaS 10.1.32 and later versions.

Cancel monitoring the 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);
```

1.11.8.5. Clipboard

This interface is used to get the clipboard data.

my.getClipboard

> **Note**
> mPaaS 10.1.32 and later versions support this interface.

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
complete	Function	No	Callback function for ended call (executed regardless of whether the call is successful or failed)

Return value success

Name	Type	Description
text	String	Clipboard data

Sample code

```
// API-DEMO page/API/clipboard/clipboard.json
{
  "defaultTitle": "Clipboard"
}
```

```
<!-- API-DEMO page/API/clipboard/clipboard.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">setClipboard</view>
    <view class="page-section-demo">
      <input onInput="handleInput" value="{{text}}" />
      <button class="clipboard-button" type="primary" size="mini" onTap="handleCopy">copy</button>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">getClipboard</view>
    <view class="page-section-demo">
      <input onInput="bindInput" value="{{copy}}" disabled />
      <button class="clipboard-button" type="default" size="mini" onTap="handlePaste">paste</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/clipboard/clipboard.js
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handleInput(e) {
    this.setData({
      text: e.detail.value,
    });
  },

  handleCopy() {
    my.setClipboard({
      text: this.data.text,
    });
  },

  handlePaste() {
    my.getClipboard({
      success: ({ text }) => {
        this.setData({ copy: text });
      },
    });
  },
});
```

```
/* API-DEMO page/API/clipboard/clipboard.acss */
.clipboard-button {
  margin-left: 5px;
}
```

my.setClipboard

> **Note**
> mPaaS 10.1.32 and later versions support this interface.

This interface is used to set the clipboard data.

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

```
// API-DEMO page/API/clipboard/clipboard.json
{
  "defaultTitle": "Clipboard"
}
```

```
<!-- API-DEMO page/API/clipboard/clipboard.axml-->
<view class="page">
  <view class="page-section">
    <view class="page-section-title">setClipboard</view>
    <view class="page-section-demo">
      <input onInput="handleInput" value="{{text}}" />
      <button class="clipboard-button" type="primary" size="mini" onTap="handleCopy">copy</button>
    </view>
  </view>
  <view class="page-section">
    <view class="page-section-title">getClipboard</view>
    <view class="page-section-demo">
      <input onInput="bindInput" value="{{copy}}" disabled />
      <button class="clipboard-button" type="default" size="mini" onTap="handlePaste">paste</button>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/clipboard/clipboard.js
Page({
  data: {
    text: '3.1415926',
    copy: '',
  },

  handleInput(e) {
    this.setData({
      text: e.detail.value,
    });
  },

  handleCopy() {
    my.setClipboard({
      text: this.data.text,
    });
  },

  handlePaste() {
    my.getClipboard({
      success: ({ text }) => {
        this.setData({ copy: text });
      },
    });
  },
});
```

```
/* API-DEMO page/API/clipboard/clipboard.acss */
.clipboard-button {
  margin-left: 5px;
}
```

1.11.8.6. Shake

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

```
// API-DEMO page/API/watch-shake/watch-shake.json
{
    "defaultTitle": "Shake"
}
```

```
<!-- API-DEMO page/API/watch-shake/watch-shake.axml-->
<view class="page">
  <button type="primary" onTap="watchShake">
   Bind Shake and click the Shake button to see the effect
  </button>
</view>
```

```
Page({
  watchShake() {
    my.watchShake({
      success: function() {
        console.log('Moved up')
        my.alert({ title:'Moved up o.o'});
      }
    });
  },
});
```

1.11.8.7. Vibrate

my.vibrate(OBJECT)

> **Note**
> 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)

> **Note**
> 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)

> **Note**
> 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'
      });
    }
  }
});
```

1.11.8.8. 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);
```

1.11.8.9. 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 return 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);
```

1.11.8.10. 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);
```

1.11.8.11. 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' });
  },
});
```

1.11.8.12. User captures screen event

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)

> **Note**
> mPaaS 10.1.32 and later versions support this interface.

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. You can use my.offUserCaptureScreen() to cancel monitoring.

Sample code

```
<!-- API-DEMO page/API/user-capture-screen/user-capture-screen.axml-->
<view class="page">
  <view class="page-description">User screenshot event API</view>
  <view class="page-section">
    <view class="page-section-title">my.onUserCaptureScreen</view>
    <view class="page-section-demo">
      <view>Current status：{{ condition ? "Monitoring is already enabled" : 'Monitoring has been cancelled' }}</view>
      <view a:if="{{condition}}">
        <button type="primary" onTap="offUserCaptureScreen">Cancel monitoring screen events</button>
      </view>
      <view a:else>
        <button type="primary" onTap="onUserCaptureScreen">Enable monitoring of screen events</button>
      </view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/user-capture-screen/user-capture-screen.js
Page({
  data: {
    condition: false,
  },
  onReady() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Receive screenshots from users',
      });
    });
  },
  offUserCaptureScreen() {
    my.offUserCaptureScreen();
    this.setData({
      condition: false,
    });
  },
  onUserCaptureScreen() {
    my.onUserCaptureScreen(() => {
      my.alert({
        content: 'Receive screenshots from users'
      });
    });
    this.setData({
      condition: true,
    });
  },
});
```

my.offUserCaptureScreen()

> **Note**
> mPaaS 10.1.32 and later versions support this interface.

This interface is used to cancel the listening of screen capture events. It usually needs to be used with `my.onUserCaptureScreen`.

Sample code

```
<!-- API-DEMO page/API/user-capture-screen/user-capture-screen.axml-->
<view class="page">
  <view class="page-description">User screenshot event API</view>
  <view class="page-section">
    <view class="page-section-title">my.onUserCaptureScreen</view>
    <view class="page-section-demo">
      <view>Current status：{{ condition ? "Monitoring is already enabled" : 'Monitoring has been cancelled' }}</view>
      <view a:if="{{condition}}">
        <button type="primary" onTap="offUserCaptureScreen">Cancel monitoring screen events</button>
      </view>
      <view a:else>
        <button type="primary" onTap="onUserCaptureScreen">Enable monitoring of screen events</button>
      </view>
    </view>
  </view>
</view>
```

Whether to pass callback value

- If callback value is not passed, all event callbacks will be removed. The code example is as follows:

```
my.offUserCaptureScreen();
```

- Pass the callback value and only remove the corresponding callback event. The code example is as follows:

```
my.offUserCaptureScreen(this.callback);
```

1.11.8.13. Screen brightness

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}`
        });
      }
    })
  }
});
```

1.11.8.14. Add phone 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.

The user can choose to write the form into the phone system's address book by **creating a new contact** or **adding to an existing contact**.

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" />
```

```
    <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
directories
      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'
      });
    }
  }
});
```

```
});
```

1.11.8.15. Scan codes

Call the scan function.

my.scan

Call the scan function.

Input parameter

Name	Type	Required	Description
scanType	String	No	Scan code recognition type, the default value is `['qrCode', 'barCode']`.
hideAlbum	Boolean	No	Whether to hide the album (do not allow selecting pictures from the album, only scanning from the camera), the default value is false.
success	Function	No	Callback function for successful call
fail	Function	No	Callback function for failed called
complete	Function	No	Callback function when the call ends (will be executed whether the call succeeds or fails)

Return value success

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	Solution
10	User canceled	This is a normal user interaction flow branch and does not require special processing.
11	Operation failed	The specific cause needs to be checked on the client to assist in troubleshooting.

Code example

```
// API-DEMO page/API/scan-code/scan-code.json
{
    "defaultTitle": "Scan"
}
```

```
<!-- API-DEMO page/API/scan-code/scan-code.axml-->
<view class="page">
  <view class="page-section">
    <form onSubmit="scanCode">
      <view>
        <button type="primary" onTap="scan">Scan code</button>
      </view>
    </form>
  </view>
</view>
```

```
// API-DEMO page/API/scan-code/scan-code.js
Page({
  scan() {
    my.scan({
      scanType: ['qrCode','barCode'],
      success: (res) => {
        my.alert({ title: res.code });
      },
    });
  }
})
```

FAQ

Q: After scanning the mini program experience code, why is the page still loading?

A: It is recommended to check the domain name whitelist configured in the background. The whitelist must be configured if there is a network request on the homepage.

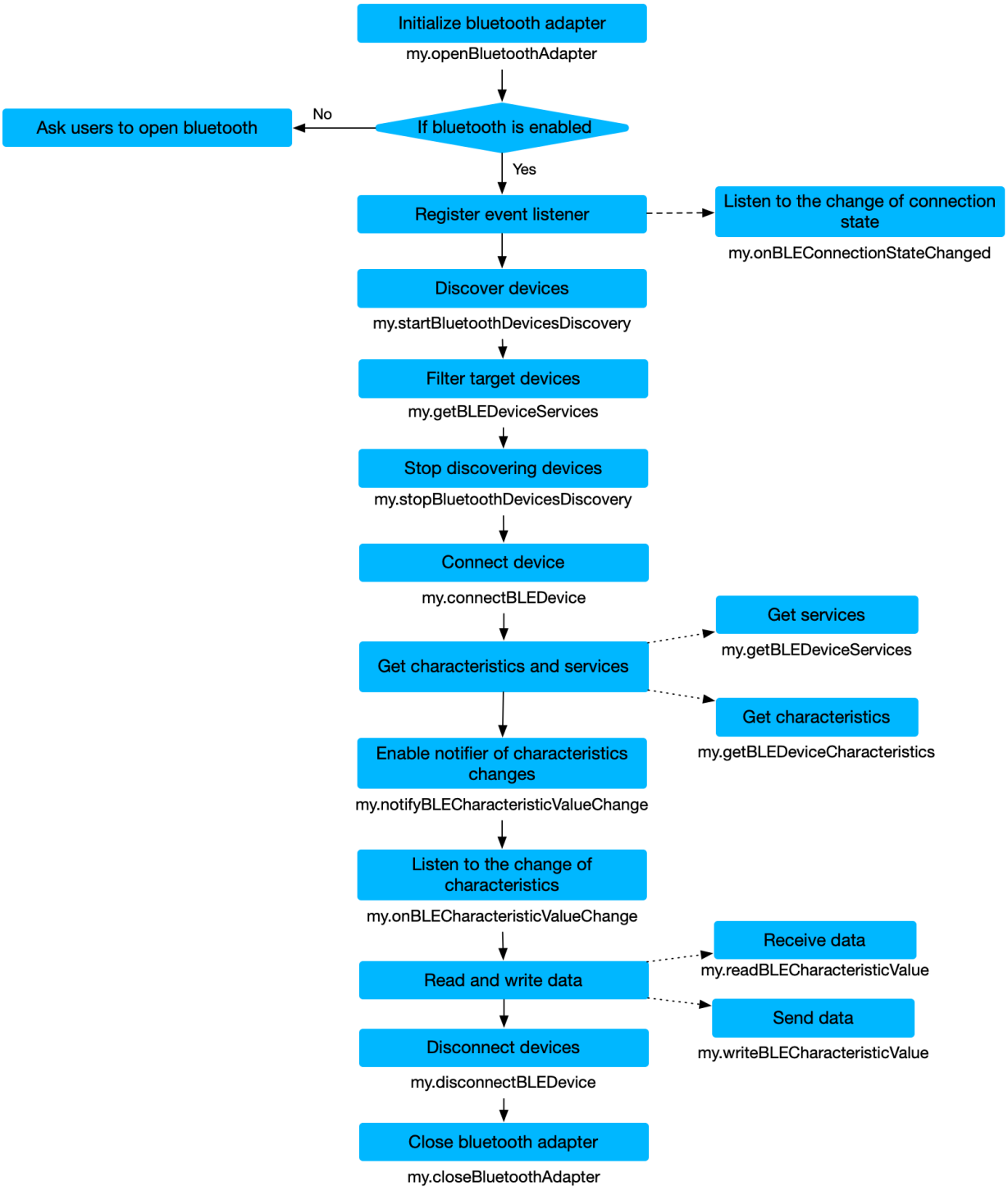
1.11.8.16. Bluetooth API overview

Version requirement

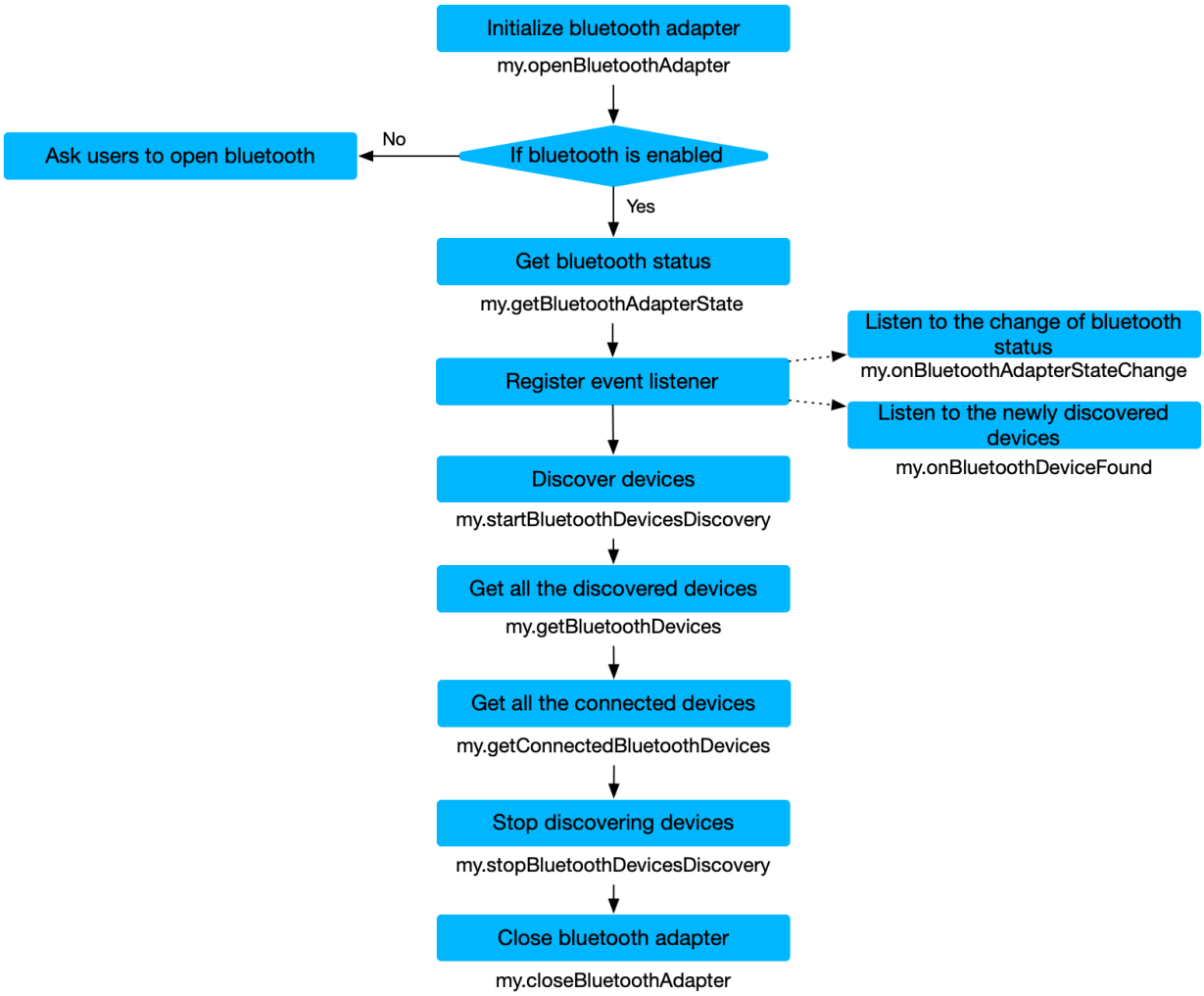
Bluetooth	mPaaS version	Android/iOS
Bluetooth Low Energy (BLE)	mPaaS 10.1.60 or later versions	• Android: 5.0 and later versions • iOS: No requirement
Traditional Bluetooth	mPaaS 10.1.60 or later versions	-

Process flow

BLE



Traditional Bluetooth



Bluetooth API

Bluetooth low energy

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.
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)=>{
  }
});
```

1.11.8.17. 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.

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. **Important** 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 of success

Name	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: 'FFE7'
```

```
    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 characteris
tics are 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 b
y attribute 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();
  },
});
```

Bug & Tip

- Bug: If the user's Bluetooth switch is not turned on or the phone does not support Bluetooth, calling `my.openBluetoothAdapter` will return an error. For the error code, see the Bluetooth API error codes list; at this time, the Mini Program Bluetooth module has been initialized, and the change of the phone's Bluetooth state can be monitored through `my.onBluetoothAdapterStateChange`.
- Tip: Before calling the `my.openBluetoothAdapter` API, call other Mini Program Bluetooth module related APIs, and the API will return an error.
  - Error code: 10000
  - Error description: Bluetooth adapter is not initialized
  - Solution: Please call `my.openBluetoothAdapter`

my.closeBluetoothAdapter

This interface is used to close the Bluetooth module in the mini program.

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)=>{
  }
});
```

Bug & Tip

- Tip: Calling this method will disconnect all established Bluetooth connections and release system resources.
- Tip: It is recommended to call this method when ending the Bluetooth process of the mini program, and call it in pairs with `my.openBluetoothAdapter`.
- Tip: Calling `my.closeBluetoothAdapter` to release resources is an asynchronous operation. It is not recommended to use `my.closeBluetoothAdapter` and `my.openBluetoothAdapter` as an exception handling process (equivalent to closing and opening again, reinitializing, low efficiency, and prone to thread synchronization problems).

my.getBluetoothAdapterState

Obtains the status of the Bluetooth module in the mini program.

Parameters

Name	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 of success

Name	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.

Parameters

Name	Type	Required	Description
services	Array	No	The UUID list of the Bluetooth device's master service.
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)=>{
  }
});
```

Bug & Tip

- Tip: 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.

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.

Parameters

Name	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 of success

Name	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)=>{
  }
});
```

Bug & Tip

- Tip: The simulator may not be able to obtain advertisData and RSSI. Please use a physical device for debugging.
- Tip: 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.

Parameters

Name	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)=>{
  }
});
```

Bug & Tip

- Tip: 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.
- Tip: 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.

Parameters

Name	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)=>{
  }
});
```

Bug & Tip

- Tip: 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.
- Tip: If the specified Bluetooth device is connected, you'll be returned with a success response if the connection is repeated.
- Tip: 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.

Parameters

Name	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)=>{
  }
});
```

Bug & Tip

- Tip: 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.
- Tip: 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 list for details. It is recommended to perform the reconnection operation.
- Tip: 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.

Parameters

Name	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.
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: 'ffffe',
  success: (res) => {
      console.log(res)
  },
  fail:(res) => {
  },
  complete: (res)=>{
  }
});
```

Bug & Tip

- Tip: The characteristic of the device must support “write”, otherwise the interface cannot be successfully called. Please see the `properties` of the characteristic for details.
- Tip: The binary data written needs to be hex encoded.
- Tip: 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.

Parameters

Name	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 of success

Name	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)=>{
  }
});
```

Bug & Tip

- Tip: The characteristic of the device must support “read”, otherwise the interface cannot be successfully called. Please see the `properties` of the characteristic for details.
- Tip: Multiple parallel reading and writing calls might cause failures.
- Tip: 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 list.

- 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.

Parameters

Name	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`.
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)=>{
  }
});
```

Bug & Tip

- Tip: The characteristic of the device must support “notify/indicate”, otherwise the interface cannot be successfully called.
- Tip: The “notify” feature must be enabled to listen to the `characteristicValueChange` event of the device.
- Tip: After the subscription operation is successful, the device needs to actively update the characteristic value to trigger `my.onBLECharacteristicValueChange`.
- Tip: 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.

Parameters

Name	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 of 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.
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"
    }]
}
```

Bug & Tip

- Tip: After the connection is established, you should execute `my.getBLEDeviceServices` and `my.getBLEDeviceCharacteristics` first, and then perform data interaction with the Bluetooth device.
- Tip: 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.

Parameters

Name	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.
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 of 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)=>{
  }
});
```

Bug & Tip

- Tip: After the connection is established, you should execute `my.getBLEDeviceServices` and `my.getBLEDeviceCharacteristics` first, and then perform data interaction with the Bluetooth device.
- Tip: 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.

Parameters

Name	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);
  },
})
```

Bug & Tip

- Tip: The simulator may not be able to obtain `advertisData` and RSSI. Please use a physical device for debugging.
- Tip: 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`).
- Tip: 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);
```

Bug & Tip

- Tip: 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.

Parameters

Name	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Name	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);
  },
})
```

Bug & Tip

- Tip: 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.

Parameters

Name	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.

Parameters

Name	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Name	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 characteris
tics are 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 b
y attribute 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();
  },
});
```

Bug & Tip

- Tip: 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);
```

Bug & Tip

- Tip: 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.

Parameters

Name	Type	Required	Description
callback	Function	Yes	The callback function that is triggered when a specified event occurs.

Return values of callback

Name	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({
```

```
  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 characteris
tics are 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 b
y attribute 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);
```

Bug & Tip

- Tip: 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.

1.11.8.18. Bluetooth API error codes list

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 `deviceId` and make sure peripheral broadcast of the target device is enabled.
10003	Connection failed	Check the `deviceId` and make sure peripheral broadcast of the target device is enabled.
10004	Service not found	Check the `serviceId` and make sure the service is available for target devices.
10005	Characteristic not found	Use a correct `characteristicId` 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 `serviceId` and `characteristicId`.
10011	Invalid device ID, or the device ID is empty	Use a correct `deviceId`.
10012	Invalid service ID, or the service ID is empty	Use a correct `serviceId`.
10013	Invalid characteristic ID, or the characteristic ID is empty	Use a correct `characteristicId`.
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.

1.11.8.19. Bluetooth API FAQ

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.

1.11.9. 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 of success

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' +
     '3ouBXXXsfm2ekey8PpQxfXaj81lhM9t8rJlXXXXXXs8Qp7Q5/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();
    }
}
```

1.11.10. 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	Minimum 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 supported 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. Th
e 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 {

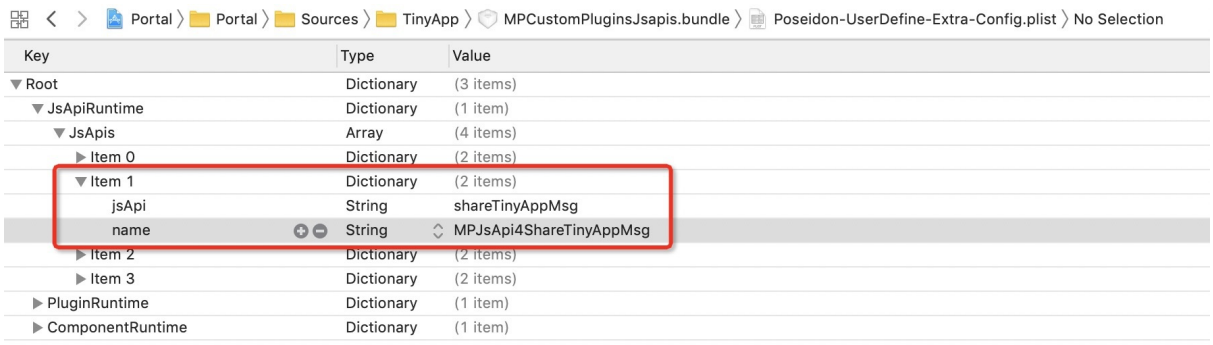
    ...

    [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,
kAPSKChannelQQZone, 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
```

1.11.11. The current version type of the Mini Program

my.getRunScene

> **Note**
> - The basic library version 1.10.0 and above supports this interface. Lower versions need to be compatible. For details, see the Mini Program Base Library.
> - mPaaS version 1.10.60 and above supports this interface.

This interface is used to obtain the current running version of the Mini program.

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 when the call ends (will be executed whether the call succeeds or fails)

Return value of success

Name	Type	Description
envVersion	String	The version of the mini program currently running. Enumeration type: develop (development version), trial (trial version), release (release version), gray (grayscale 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}`
    });
  },
})
```

1.11.12. Custom analysis

my.reportAnalytics

> **Note**
> This interface is supported by mPaaS 10.1.32 and above.

Customize the reporting interface for analysis data. Note before use:

- You need to create a new event in **Mobile Analysis Service** > **Custom Analysis** > **Custom Configuration** in the mPaaS console in advance and configure the event name and attributes. For more information, see Configure custom analysis.
- `my.reportAnalytics` only counts the usage data of the Mini Programs that have been released and launched.

Parameter

Name	Type	Required	Description
eventName	String	Yes	Custom event name, which you need to apply for
data	Object	Yes	Data reported

data description

data is of Object type, with the following attributes:

Attribute	Type	Required	Description

key	String	Yes	The name of the field in the configuration
value	Any	Yes	The data to be reported

Code example

```
<!-- API-DEMO page/API/report-analytics/report-analytics.axml-->
<view class="page">
  <view class="page-description">Custom analysis API</view>
  <view class="page-section">
    <view class="page-section-title">my.reportAnalytics</view>
    <view class="page-section-demo">
      <view class="report" onTap="reportAnalytics">Custom analysis</view>
    </view>
  </view>
</view>
```

```
// API-DEMO page/API/report-analytics/report-analytics.js
Page({
  reportAnalytics() {
    my.reportAnalytics('demo_click', {});
    my.alert({
      content: 'The data has been successfully reported. Please check it in the data analysis of the mini program management background',
    });
  },
});
```

```
/* API-DEMO page/API/report-analytics/report-analytics.acss */
.report {
    width: 100%;
    background: #108ee9;
    color: #fff;
    border: 1px solid #108ee9;
    height: 47px;
    line-height: 47px;
    border-radius: 5px;
    text-align: center;
    font-size: 18px;
}
```

1.11.13. Custom API

If the existing Mini program APIs cannot meet your requirements, you can extend customize APIs by yourself.

Custom APIs

- Custom JSAPIs for Android
- Custom JSAPIs for iOS

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.

Unregister custom event

If you no longer need a custom event, see Unregister custom event.

1.11.14. Mini Program jumping

my.navigateToMiniProgram (Object)

> **Note**
> 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)

> **Note**
> 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 Mini 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))
    }
});
```

1.11.15. Webview component control

You can provide Mini Program the ability to send messages to `web-view` by creating `webviewContext`.

> **Note**
> - This function is supported by the basic library version 1.8.0 and above. Lower versions need to be compatible. For operation, refer to the description of the Mini Program Base Library.
> - This function is supported by mPaaS version 10.1.32 and above.

my.createWebViewContext(webviewId)

This interface is used to provide the ability to send messages from the Mini Program to the `web-view` by creating a `webviewContext`. Create and return the `web-view` context `webViewContext` object.

Parameters

Name	Type	Required	Description
webviewId	String	Yes	The ID attribute corresponding to the `web-view` to be created.

Return value

A webViewContext object.

The `webViewContext` is bound to a `web-view` component through the `webviewId`, through which some functions can be implemented.

The methods of the `webViewContext` object are as follows:

Method	Parameter	Description	Compatibility
postMessage	Object	The Mini Program sends messages to the `web-view` component. With the help of `my.postMessage` provided in `web-view.js`, two-way communication between the Mini Program and the `web-view` webpage can be achieved.	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.

1.11.16. Application-level events

1.11.16.1. my.onAppShow

Dscription

my.onAppShow

Listen for the foreground event of the Mini Program.

Calling this interface is actually the callback of dynamically registering app.js registering Mini Program lifecycle event, and the interface of cancelling listening is my.offAppShow.

Input parameter

Function listener

Parameter

Object res

Attribute	Type	Description
path	String	The path (code package path) to start the Mini Program.
scene	Number	The scenario value that starts the Mini Program.
query	Object	The query parameters used to start the Mini Program.
referrerInfo	Object	The source message. **Important** This field may be undefined on the base library of earlier version. Therefore, When accessing its properties, please perform null check first, such as: `options.referrerInfo && options.referrerInfo.appId`.
apiCategory	string	The API category.

referrerInfo

Attribute	Type	Description
appId	String	The source Mini Program.
extraData	Object	The data that source Mini Program passed through `xtraData` input parameter of `my.navigateToMiniProgram`.

apiCategory

Enumerator	Description	Compatibility
default	The default type.	-
embedded	Returns when the Mini Program is opened in half-screen mode.	Basic library: 2.7.22 +

Sample code

my.onAppShow(Function listener)

```
const onAppShowHandler = (res) => {
  console.log('onAppShow:', res)
};

my.onAppShow(onAppShowHandler);
```

1.11.16.2. my.offAppShow

Description

my.offAppShow

Remove the listener function for the foreground event of the Mini Program.

Input parameter

Function listener

The callback function for the foreground event of the Mini Program.

Sample code

my.offAppShow(Function listener)

```
const onAppShowHandler = (res) => {
    console.log('onAppShow:', res)
};

// Call the API at an appropriate time and add a listener function.
my.onAppShow(onAppShowHandler)

// Call the API at an appropriate time and remove the listener function.
my.offAppShow(onAppShowHandler)
```

1.11.16.3. my.onAppHide

Description

my.onAppHide

Listen for background event of Mini Program.

Calling this interface is actually the callback of dynamically registering app.js registering Mini Program lifecycle event, and the interface of cancelling listening is my.offAppHide.

Input parameter

Function listener

The callback function for the background event of the Mini Program.

Sample code

my.onAppHide(Function listener)

```
const onAppHideHandler = () => {
 console.log ('How to switch the listener to the background')
};

my.onAppHide(onAppHideHandler);
```

1.11.16.4. my.offAppHide

Description

my.offAppHide

Remove the listener function for background event of Mini Program.

Input parameter

Function listener

The callback function for the background event of the Mini Program.

Sample code

my.offAppHide(Function listener)

```
const onAppHideHandler = () => {
 console.log ('How to switch the listener to the background')
};

// Call the API at an appropriate time and add a listener function.
my.onAppHide(onAppHideHandler)


// Call the API at an appropriate time and remove the listener function.
my.offAppHide(onAppHideHandler);
```

1.11.16.5. my.onPageNotFound

Description

my.onPageNotFound

Listen for the event that the page to be opened by the Mini Program does not exist.

It only responds to the page not found event when the Mini Program is cold started or hot started, and does not support to deal with the failures of the Route.

If the listener is not registered with the `my.onPageNotFound`, when the page does not exist, the `page not found` prompt page will be displayed.

Input parameter

Function listener

The listener function for the event that the page to be opened by the Mini Program does not exist.

Parameter

Object res

Attribute	Type	Description
path	String	The path (code package path) of the page does not exist.
query	Object	Open the query parameter of the page that does not exist.
isEntryPage	Boolean	Specifie whether it is the first page of this startup. For example, enter the Mini Program from the sharing, the first page is the sharing page configured by the developer.

Sample code

my.onPageNotFound(Function listener)

```
//app.js

const handlePageNotFound = (res) => {
 my.redirectTo({
 url: 'pages/...'
 }); // For a tabbar page, use my.switchTab.
};

my.onPageNotFound(handlePageNotFound)

App({
 onLaunch() {

 }
})
```

> **Note**
>
> - Developers can perform page redirection in the callback, but they must be processed synchronously in the callback. Asynchronous processing (for example, `setTimeout` asynchronous execution) is invalid.
> - The page to be redirected in the callback must be a page that has been loaded with resources. If the page is a split page or a plugin page that has not been loaded, an error will be reported when running, and the redirection cannot be completed.

1.11.16.6. my.offPageNotFound

Description

my.offPageNotFound

Remove the listener function for the event that the page to be opened by the Mini Program does not exist.

Input parameter

Function listener

The listener function for the event that the page to be opened by the Mini Program does not exist.

Sample code

my.offPageNotFound(Function listener)

```
//app.js
const handlePageNotFound = (res) => {
 my.redirectTo({
 url: 'pages/...'
 }); // For a tabbar page, use my.switchTab.
};

my.onPageNotFound(handlePageNotFound)

// Call the API at an appropriate time and remove the listener function.
my.offPageNotFound(handlePageNotFound)

App({
 onLaunch() {

 }
})
```

1.11.16.7. my.onUnhandledRejection

Description

my.onUnhandledRejection

Listen for unhandled `Promise` rejection event.

Input parameter

Function listener

The callback function for the unhandled `Promise` reject event.

Parameter

Object res

Attribute	Type	Description
reason	any	The reason for the rejection. The received value of the `reject()` is generally a `Error` object.
promise	Promise	The rejected `Promise` object. **Note** Android supports this field starting with Base Library 2.9.7.

Sample code

my.onUnhandledRejection(Function listener)

```
Page({
 onLoad() {
 my.onUnhandledRejection(this.unhandledRejectionHandler);
 },
 unhandledRejectionHandler(res) {
 console.log('onUnhandledRejection reason', res.reason);
 console.log('onUnhandledRejection promise', res.promise);
 }
})
```

Note

- If the `unhandledrejection` event of `Promise` continues to be triggered within the callback function of `my.onUnhandledRejection`, it may cause the `unhandledrejection` event to be triggered in a loop. Please be careful to avoid it.
- All `unhandledRejection` can be captured by this listener, but only the `Error` type will trigger an alarm in the backstage of the Mini Program.

1.11.16.8. my.offUnhandledRejection

Description

my.offUnhandledRejection

Remove listener function for unhandled `Promise` reject event.

Input parameter

Function listener

The callback function for the unhandled `Promise` reject event.

Sample code

my.offUnhandledRejection(Function listener)

```
Page({
 onLoad() {
 my.onUnhandledRejection(this.unhandledRejectionHandler);
 },
 unhandledRejectionHandler(res) {
 console.log('onUnhandledRejection reason', res.reason);
 console.log('onUnhandledRejection promise', res.promise);
 },
 offUnhandledRejection() {
 my.offUnhandledRejection(this.unhandledRejectionHandler);
 }
})
```

1.11.16.9. my.onError

Description

my.onError

Listen for Mini Program error event. Currently, only JS execution errors are supported. The trigger timing and parameters are the same as app.js registering Mini Program.

Input parameter

Function listener

Parameter

Attribute	Type	Compatibility	Description
error	String	-	The description of the exception, which is generally the `message` field of the `Error` object.
stack	String	Base database: 2.7.4	The exception stack, which is generally the `stack` field of the `Error` object.

Sample code

my.onError(Function listener)

```
Page({
 onLoad() {
 my.onError(this.errorHandler);
 },
 errorHandler(error, stack) {
 console.log('onError error', error);
 console.log('onError stack', stack);
 }
})
```

> **Note**
>
> - If you use `my.onError` to listen for errors, `onError` methods in the `app.js` will also listen for errors.
> - Use `my.onError` to listen for page errors. If listener is enabled on multiple pages and not closed, multiple listen events will be triggered when the page reports error. Therefore, it is recommended to call `my.offError` to close the listener function when the page is closed.

1.11.16.10. my.offError

Description

my.offError

Remove the listener function for the Mini Program error event.

Input parameter

Function listener

The callback function for Mini Program error to be removed.

Sample code

my.offError(Function listener)

```
Page({
 onLoad() {
 my.onError(this.errorHandler);
 },
 errorHandler(res) {
 console.log('onError error', res.error);
 console.log('onError stack', res.stack);
 },
 onUnload() {
 my.offError(this.errorHandler)
 }
})
```

1.11.16.11. my.onComponentError

Description

my.onComponentError listens for the `error` event of the JS code inside the Mini Program's custom component. When the JS code inside the custom component reports an error, in addition to executing the `my.onComponentError` callback, the my.onError callback will also be triggered.

Input parameter

Function listener

Parameter

Attribute	Node type	Description
error	Error	The exception object.
method	String	The custom component method where the exception occurred.
component	Component	The custom component instance where the exception occurred.

Sample code

my.onComponentError(Function listener)

```
Page({
 onLoad() {
 my.onComponentError(this.errorHandler);
 },

 errorHandler(error, method, component) {
 console.log('onComponentError res', error);
 console.log('onComponentError stack', method);
 console.log('onComponentError component', component);
 }
})
```

1.11.16.12. my.offComponentError

Description

my.offComponentError

Remove the `error` event listener function of the JS code inside the Mini Program custom component.

Input parameter

Function listener

The callback function for the JS code inside the custom component reports an error when running.

Parameter

Attribute	Node type	Description
error	Error	The exception object.
method	String	The custom component method where the exception occurred.
component	Component	The custom component instance where the exception occurred.

Sample code

my.offComponentError(Function listener)

```
Page({
 onLoad() {
 my.onComponentError(this.errorHandler);
 },

 errorHandler(error, method, component) {
 console.log('onComponentError res', error);
 console.log('onComponentError stack', method);
 console.log('onComponentError component', component);
 },

 // Stop listening to the events.
 offComponentError() {
 my.offComponentError(this.errorHandler)
 }
})
```

1.12. Design Guide

1.12.1. Principles

1.12.1.1. Clarity

When designing an mPaaS Mini Program, it needs to be simple and easy to use to attract more users. The more users there are, the wider the market and the greater the profits.

A single page focus on a single task

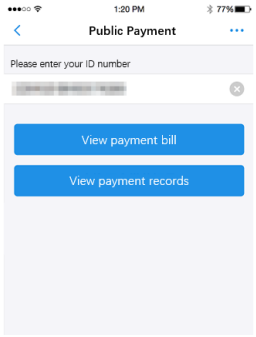
The information that can be displayed on a page of an Mini Program is very limited. Since the use environment of mobile phones is very unstable and is often affected by various factors, such as people's behavior (such as walking, riding, etc.), network signals, etc., this further limits the amount of information on the page. It is best to highlight one key point on a page so that users can quickly understand and complete tasks. Avoid other interference factors that are not related to the user's decision and operation on the page.

Most mPaaS Mini Program services are task-oriented, aiming to help users achieve a certain task goal, such as transferring money, paying bills, etc. This principle is particularly important in task-oriented pages, because we hope that users can focus on and quickly complete the current task.

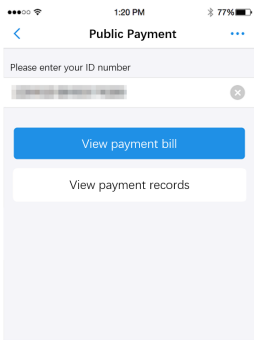
Example

If multiple main action buttons are set in one single page and the primary and secondary operations are not distinguished, the user may feel confused and doesn't know how to choose.

Incorrect example:



Correct example:



Delete and hide appropriately

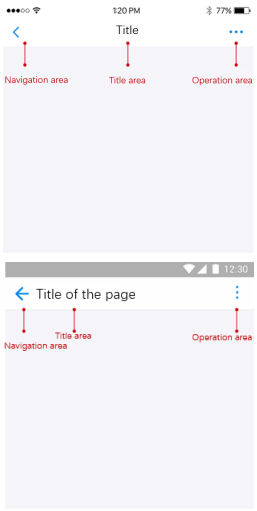
People's ability to process information, learn rules, and memorize details is limited. In practice, people may also face various interruptions and distractions, which further limits their cognitive ability. Like a roadblock, too many minor details in the interface will increase the cognitive burden on 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 Program, 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 Program, without adding learning cost or changing habits when switching between pages.



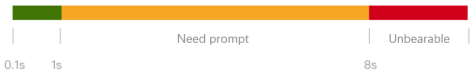
1.12.1.2. Efficiency

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.

To design an efficient and thoughtful product, you need to follow these rules:

One second wait

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.



Divert attention

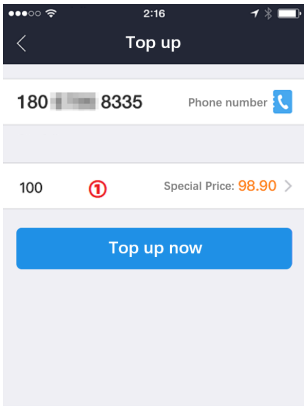
Diverting attention is a typical way to reduce the impact of waiting. In practice, the strategy of diverting attention is very common. For example, some restaurants provide free snacks and leisure services to divert customers' attention when they are waiting to eat, allowing customers to enjoy queuing and reduce anxiety during waiting.

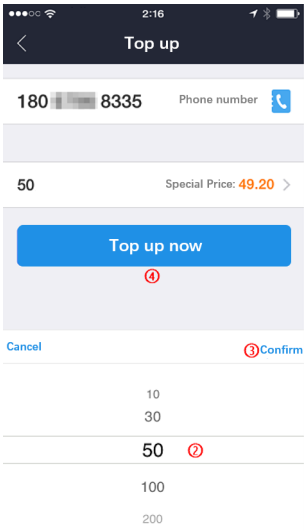
This approach also works in application design.

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 Program 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.

For 9.2 version 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 Program 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.

1.12.1.3. Security

mPaaS Mini Program accounts contain not only users' private informationbut also a large amount of funds. Therefore, account security is very important. To eliminate users' 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 desensitization

Many applications and services require users to enter or display private information, such as account passwords, mobile phone numbers, ID numbers, bank card numbers, etc. When displaying and entering this information, desensitize it and hide part of the information instead of exposing it completely. In this way, users will feel that users' information security is guaranteed and have a sense of security.

Input prompt

When users are required to enter sensitive information, clearly inform them of the purpose of the information to eliminate users' 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 users' trust and eliminate users' concerns.

Operation controllable

Always remember that users control the app, not the app. Don't make decisions for users.

Good design should be trustworthy and easy to believe. When users are asked to perform an action, try to help them understand why this action is necessary. Each step needs to rely on honest and clear statements to build trust. Through gradual accumulation, the user's sense of trust is gradually improved.

The foundation of building trust is to let users take control.

Second confirmation

To make users feel that they are taking control, the application should use familiar and predictable interactive elements. At the same time, when users perform destructive operations, provide a second confirmation to avoid irreparable losses.

Let the user decide

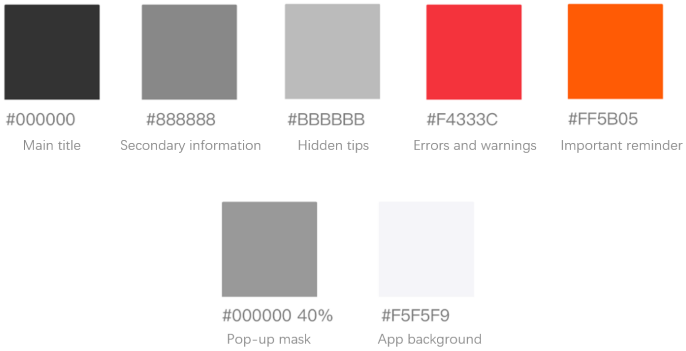
You can provide suggestions for user behavior and warn about actions that may have serious consequences, but don't make decisions for users. Good design will find a balance between letting users take the lead and avoiding unwanted results.

Users are accustomed to controlling their app experience. Some users even begin to pursue customized experiences and apps that suit their needs. So, leave room for user choice, even for the smallest options, such as whether to turn on the notification system.

1.12.2. Visual guidelines

1.12.2.1. Color

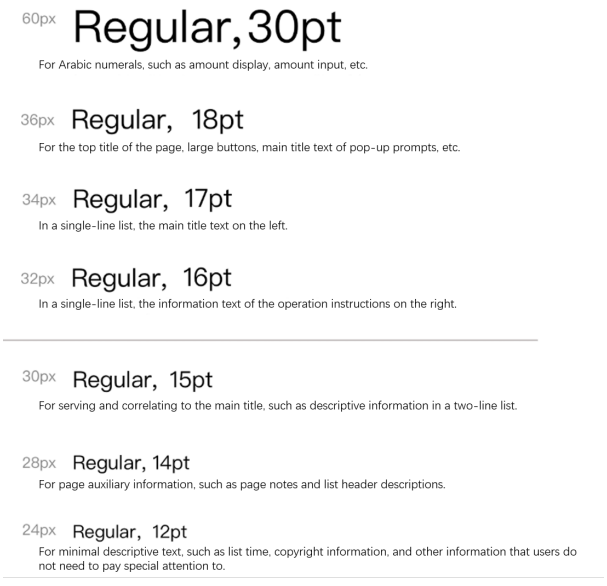
The mPaaS Mini Program has complete color usage specifications. The following figure illustrates the basic colors used for main text content, background, etc.



1.12.2.2. Font

To ensure the universality of the mPaaS applet, PingFang SC is preferred as the Chinese font to take into account both the web version and the mobile version.

The font size and usage scenario specifications are as follows:



1.12.2.3. Icon

Icons are an important part of the graphical interface. They are highly condensed, quickly conveyed, and easy to remember. To make it easier for users to identify icon information and achieve consistency in icon design, mPaaS Mini Program provide a unified style of icon design principles.

Design principle

The shape needs to be distinct and the information needs to be simplified; the icon needs to be designed with geometric shapes and symmetrical designs.

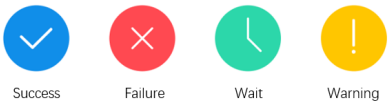
Design principle for system icon

Use rounded corners extensively to avoid abrupt and jagged edges. Take the icon at 36 x 36px as an example, the line structure is 3px and the outer corner radius is 4px.



Status icon

The status icons are designed with distinct colors, so the status of the result page is clear at a glance.



1.12.3. Component guidelines

1.12.3.1. Navigation

Navigation is the most critical factor to ensure that users do not get lost when browsing and jumping around on a web page. Navigation needs to tell users where they are currently, where they can go, how to get back, etc. mPaaS Mini Program provides navigation components such as navigation bar, segmented control, and tab bar.

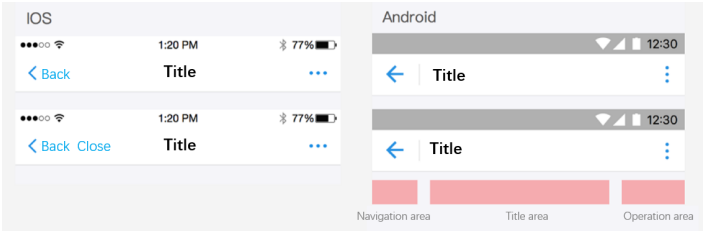
Navigation bar

Your page needs to be embedded in the navigation framework of the mPaaS Mini Program before being displayed to the user. The mPaaS Mini Program navigation bar is directly inherited from the client, and you do not need to and cannot customize the content in it. However, you need to define the jump relationship between each page so that the navigation system can work in a reasonable way.

The mPaaS Mini Program navigation bar includes navigation area, title area and operation area.

- **Navigation area**

The navigation area controls the program page process. The display is different on iOS and Android clients.



In the navigation area, there is usually only one operation, which is to return to the previous interface. You can only define its content. When the user enters a page with more than three levels, the Back and **Close** buttons are displayed at the same time; click **Close** to leave the current page and return to the mPaaS Mini Program.

- **Title area**

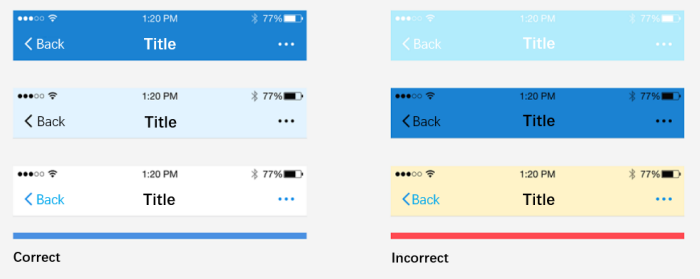
You can define a title for each page. The text display area of the title is fixed, and it will be omitted if it exceeds the length. If it is not displayed, the name of your service or application will be displayed.

- **Operation area**

The default action area is the **More** icon, which will bring up the action menu. You can also customize the action area, defining up to two action buttons with icons or one action button with text (two words).

- **Custom color**

The mPaaS Mini Program navigation bar supports basic background color customization. The selected color needs to be harmoniously matched with the three sets of main navigation bar icons provided by the mPaaS Mini Program while meeting the availability requirements. The title color will automatically adjust to white or black following the background color. It is recommended to refer to the following color selection effects and color selection scheme examples:

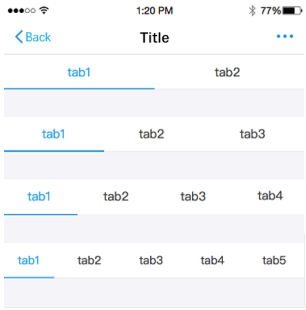


Segmented control

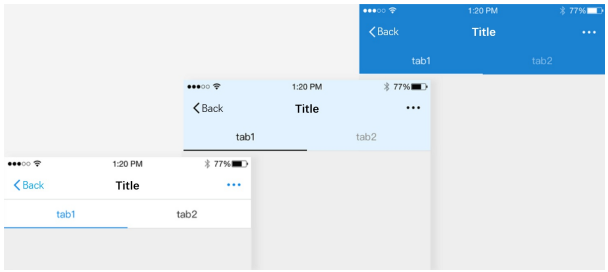
Segmented controls usually appear at the top of a page or in the middle of a page, allowing users to quickly switch between different contents on the page.

Design principles:

- Highlight the current option to let users know where they are. Set up to 5 options, and options beyond that can be displayed by sliding.



- The color of the top segment control can be customized. When choosing a custom color, be sure to ensure the usability, visibility, and operability of the paging bar label.



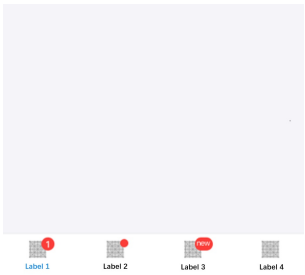
Wrong example:



Tab bar

The tab bar is used to organize information architecture in a hierarchical page. It can make your application's information hierarchy flatter, which helps users explore more content.

The tab bar is located at the bottom of the page, allowing users to quickly switch between different pages.



Design principles:

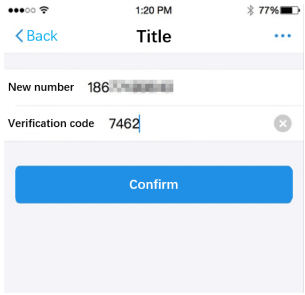
- You can set up to 5 tabs. When the number of tabs exceeds 5, the last one will be labeled **More**. More navigation options will be displayed in the **More** tab.
- The icons and text of the tab bar can be customized, but the color cannot be changed. Only the standard link color provided by the mPaaS Mini Program can be used.
- When switching between pages, it must be done on the current page instead of opening a new page.

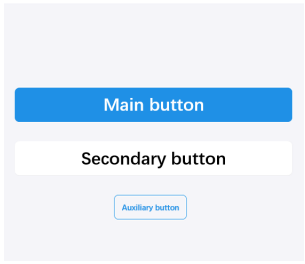
1.12.3.2. Input information

In the interaction between user and application, users often need to input, edit, and delete some information. Diversified and tailored input components

Button

Button is used to start an instant operation and submit a set of input data in the form.





Definition and principle

As the main action point on the page, the button guides users to perform the main operation. The action button should be eye-catching, attractive to users, and provide feedback after the user clicks.

There are three kinds of buttons: main button, secondary button, and auxiliary button.

- Main button: Only one button appears in the picture, indicating it is the most important user conversion point.
- Secondary button: There can be multiple secondary buttons in a page as supplementary operations for the current scene.
- Auxiliary button: An operation button located on the right side of the list, which guides users to click on the list more strongly in the form of a button.

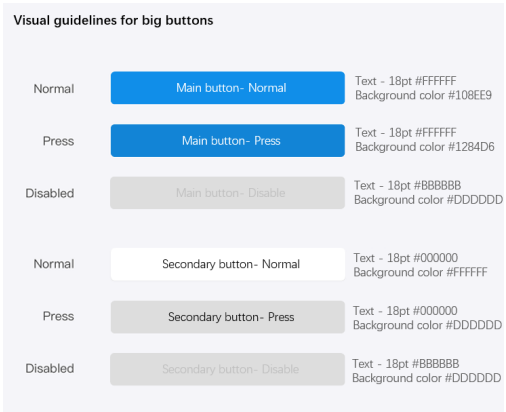
Visual style

Big button

The main purpose of big buttons is to encourage users to take actions. The usage specifications of big buttons are as follows:

- The button text needs to be centered in all directions.
- The button height is fixed at 94px (47pt) and the corner radius is 10px (5pt).

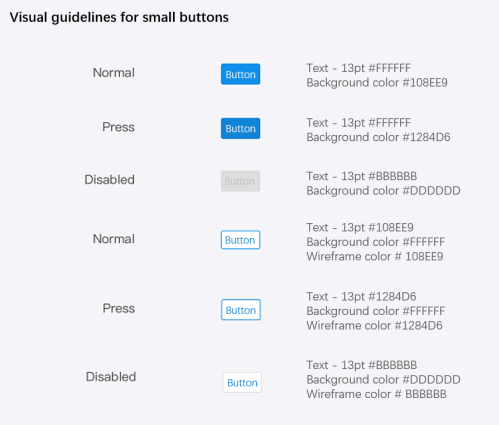
> **Note**
> The main button can only appear once in a page.



Small button

Small buttons are used to operate/select certain content or options on a page and can be used repeatedly. The usage specifications of small buttons are as follows:

- 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 corner radius is 6px (3pt).
- The distance between the text and the border in the button is 30px (15pt). If there is not enough space for the text, the button width can be extended horizontally.



Example

The button and page content should be displayed together to make sense.

Checkbox

The multi-select control allows users to select multiple elements at the same time.

Definition and principle

The multi-select control usually appears in the list that needs to be edited. When the user completes the selection, the selected elements are edited uniformly. There are two states of multi-select control: selected and unselected.

Text input box

The text input box is the simplest input component, which allows users to enter a single line of data through components such as keyboards and selectors.

Definition and principle

Single-line input boxes have a limit on the length of information input, usually up to 15 characters. You can also limit the type of information that can be entered in the input box, such as: Chinese, English, numbers, email addresses, etc.

When different types of input boxes are activated, the corresponding types of keyboards need to pop up: text keyboard, English keyboard, numeric keyboard, email keyboard, etc. This helps improve the user's input efficiency.

The input box is generally composed of three parts: **label area**, **input area**, and **auxiliary operation area**. The **label area** has a word limit, up to 4 words; the **input area** generally has a hidden prompt; the **auxiliary operation area** can be placed with auxiliary input operation buttons, or more detailed input description buttons.

If the input data is sensitive information, it should be desensitized, such as passwords, mobile phone numbers, etc.

Visual style

Labels, icons, and auxiliary input buttons together form a variety of input boxes.

Example

Call up the corresponding system keyboard according to the input data type. Both iOS and Android systems have prepared keyboards for different types of information input, which helps improve user input efficiency and thus improve user experience.

Selector

The selector provides a set of preset data, allowing users to complete input or setting by selection.

Definition and principle

The selector is triggered by clicking an input box on the page. When the selector appears, a semi-transparent mask should be added to the page to allow users to focus on the selector's selection.

The data in the selector should have a certain logical relationship and meet the user's expectations. Because the selector may not be able to display all the data at once, the user needs to slide to select. A logical order that meets the user's expectations can help the user quickly find the desired selection.

The selector can set a combination of multiple columns of data, up to four columns, but the text in the longest column cannot exceed the width limit.

Date selector

The time selector allows users to quickly select a time, from year, month, day to hour, minute, and second.

Radio button

Radio controls usually appear on the right side of a list, and a check mark appears to indicate the currently selected option.

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 find what they want among a large amount of information.

Definition and principle

The search bar is usually located below the navigation bar. When you click to activate it, it will call up the system keyboard. Click the **Cancel** button to exit the activation state.

If the input box is displayed by default, you can provide a gray hint text (such as a keyword) to help the user input data. Below the search bar, useful labels are provided for users to complete the input by clicking them, such as : recently searched content.

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.

1.12.3.3. Display information

Properly organized and displayed information helps users better understand and find content, making your application easy to use. The mPaaS Mini Program provides different information display components, and you can select the appropriate components to display different types of information according to the page requirements.

List

List is a commonly used form of information organization. It divides the content into rows, and each row can jump to the detail page to display more information.

Lists can display information that exceeds the length of the screen by sliding, and can also group related content.

Visual style

The list consists of a title, a subtitle, and an icon. Icon is optional based on your needs.

Example

A list is a simple function collection page that displays many function collections on one page in the form of a list. Users can find different function items by navigating through the list.

Notice bar

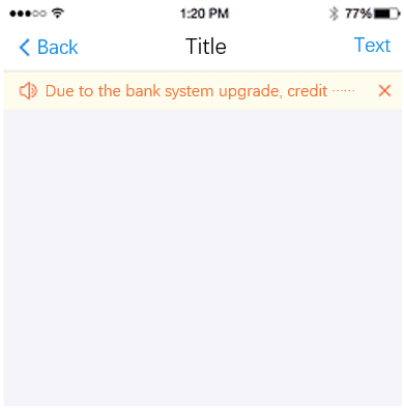
The top notice bar is inserted between the navigation bar and the page content to inform the user of important information and announcements.

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.

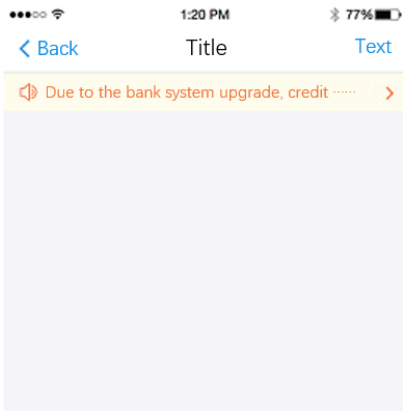
If there is more detailed content, users can click on the announcement to enter the details page. After returning, the announcement disappears.



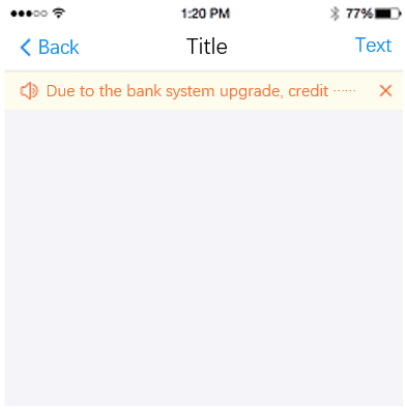
Visual style

The notice board can be equipped with an arrow to view details or a button to close it.

After the user clicks the arrow to view details and enters the details page, the announcement disappears.



After the user clicks close button, the announcement disappears.



Step bar

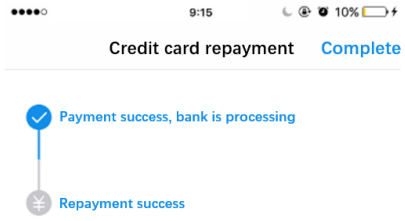
The step bar shows the number of steps and the current step.

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

Take credit card repayment as examples. After the user completes the operation, it needs to wait for a while before the task is truly completed. Therefore, it is necessary to inform the user that they need to wait through the task step bar on the result page.



1.12.3.4. Feedback

A very important point in the human-computer interaction process is the feedback of the operation. The application should give timely feedback to the user's operation. Instant response will increase the user's sense of trust.

mPaaS provides a series of feedback components, and you need to choose the correct feedback form in different scenarios.

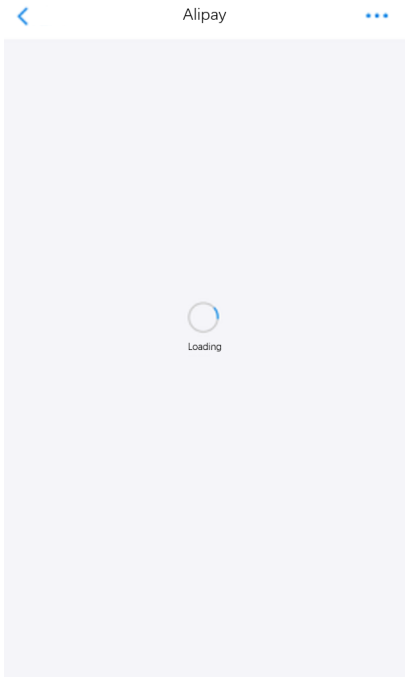
Feedback principle:

- Provide necessary, positive and immediate feedback for users at all stages of operation;
- 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.

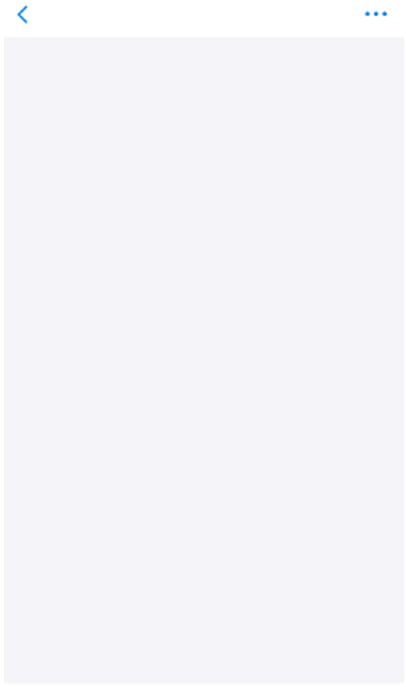
Necessary feedback

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.



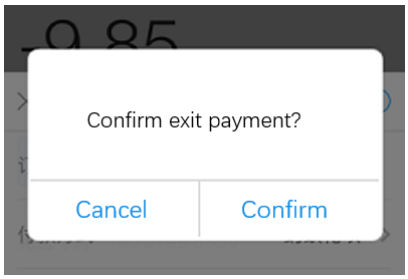
- Incorrect example: A blank page opens without any waiting prompt.



Avoid excessive feedback

Excessive feedback can cause unnecessary interruptions to the user and should be avoided.

- Incorrect 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.



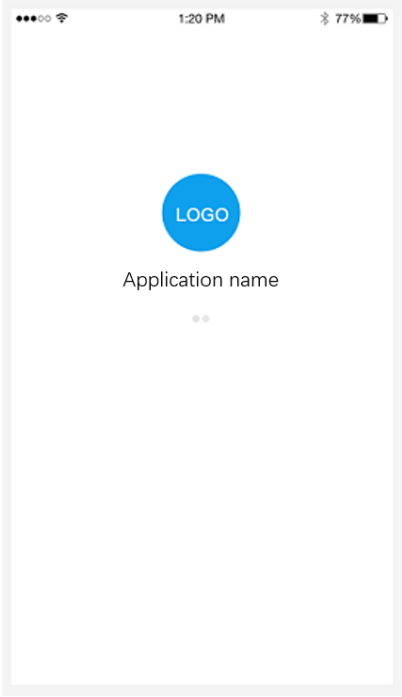
- Incorrect example 2: In the service window, when a service window is deleted, a deleted successfully toast is displayed. At this point the page state has changed significantly, this toast is completely unnecessary.



Start page loading

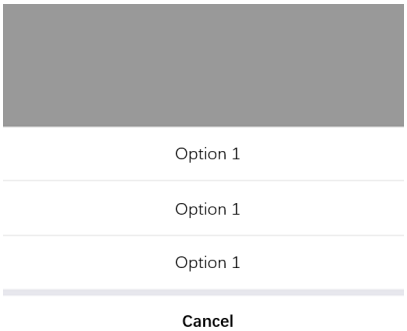
In an application, the startup page is one of the pages that shows the brand characteristics of the mini program to a certain extent.

This page will highlight the brand features and loading status of the Mini Program. In addition to the brand logo, all other elements on the startup page, such as the loading progress indicator, are uniformly provided by mPaaS and cannot be changed, so there is no need for 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 semi-transparent mask.

ActivityIndicator

Activity indicator is a loading component, which visualizes the loading process of the page content, reducing the anxiety of the user during the waiting period.

Definition and principle

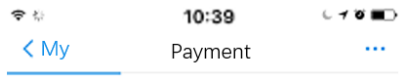
When a user enters a new page or submits an operation, and the page needs to be loaded and the user needs to wait, a progress bar should be used to inform the user of the loading progress. Otherwise, the user's wait will become confused and long, and the user will leave the page angrily.

We provide two styles, i.e. progress bar and progress ring, and define them separately.

Example

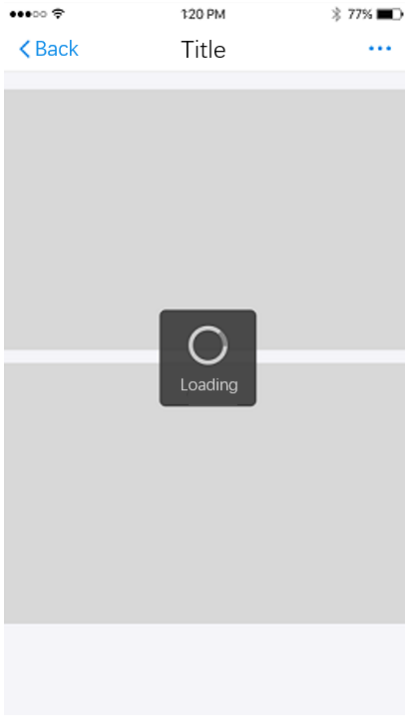
- Page startup loading progress bar:

  When a user uses the mPaaS framework to load a new online page, a linear page loading progress bar will appear below the navigation bar to show the loading progress of the current page content. The loading progress bar is provided uniformly by mPaaS and cannot be changed, so no development is required.



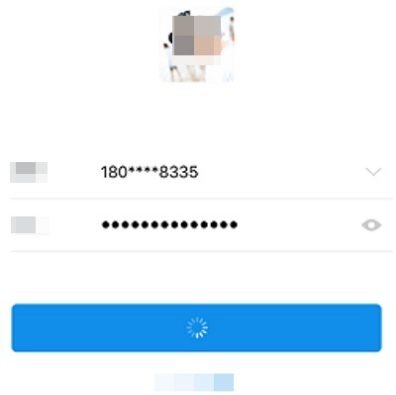
- Modal loading:

  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:

- Confirm and cancel important operations (such as whether to delete content);
- 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, main instruction, and action options. Special dialog boxes can include pictures and illustrations to better convey information to users.

Example

Warning and error

- When the user's operation may cause irreparable damage, a double confirmation dialog box should be appeared to alert users.
- 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.

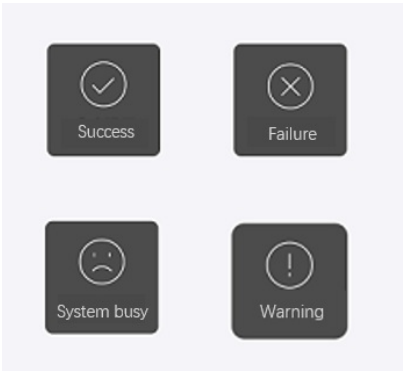
Special dialog with illustration

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 users can see status changes in a timely manner through the page, toast prompts should no longer be used to avoid excessive feedback.
- Toast loading is a blocking loading method and should be used less frequently. Try to use other methods instead.
- Toasts are displayed for a short time, and the text can only be displayed for up to 15 characters.

1.12.3.5. Gesture

Gestures can improve user efficiency as hidden shortcuts. Gestures can greatly improve the user experience. iOS and Android systems have already popularized different gestures to users. You can improve the user experience by following and supporting the existing operations of the system.

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 Program framework provides standard pull-to-refresh loading capabilities and styles. You can customize the pull-to-refresh interactions. For such interactions, mPaaS Mini Program will provide standard capabilities and styles.

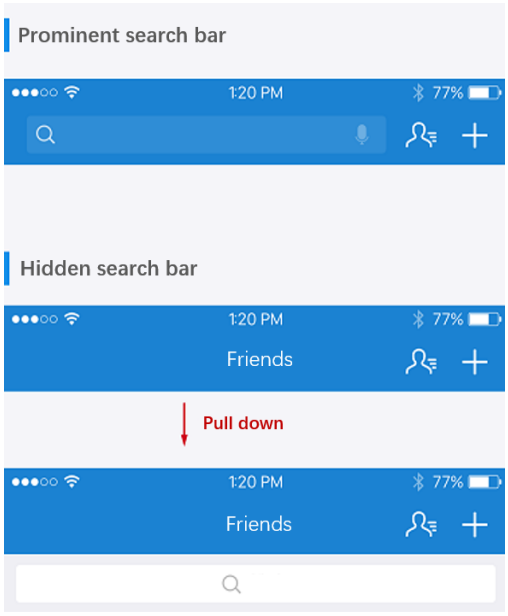
1.12.3.6. Differentiated design

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.

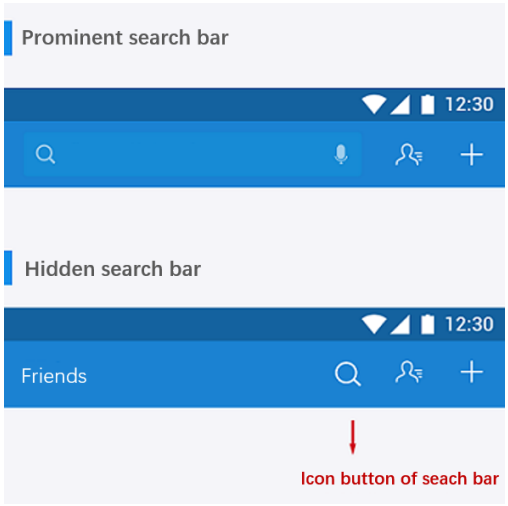
Search

There are two types of search bar, **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;

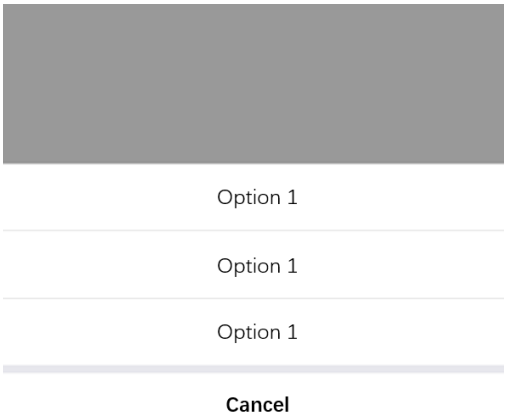


- The hidden search bar is used on pages where search needs are not particularly strong, so the space saved can be used to display more content.

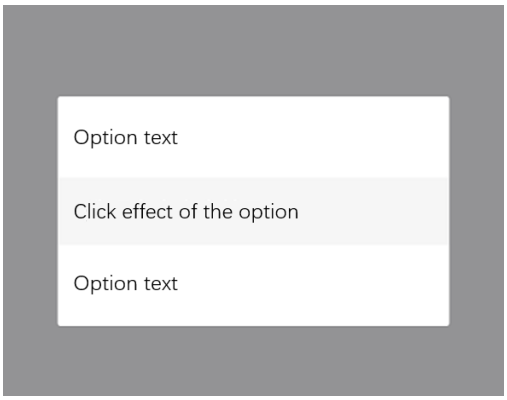


Operation list

- 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. Click the blank area outside the list or click 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:

- Ask whether to perform the current operation in the title;
- Alert users of risky consequences that may be caused by current operation in the main instruction if necessary;
- The button to confirm operation should be presented with the operation again.

Prohibition:

- Don't use ambiguous descriptions like: "Are you sure?";
- Do not explain or interpret the consequences of the operation;
- 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.

1.12.3.7. Combine components

Through the combination of different components, you can build several typical pages.

Result page

Definition and principle

After completing a task, the user needs to be informed of the current status of the task. Therefore, the page can display the successful status of the operation, or the current status of the task and how long it will take to complete.

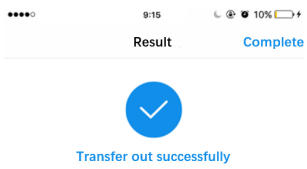
Follow-up operations

In addition to showing the final status of the task, the result page can also guide the user to perform other related operations.

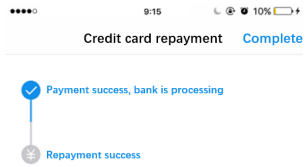
Example

There are two styles of result pages:

- Task success.



- A certain stage of the task is successful, and the next stage needs to wait.



1.12.3.8. aria-component

Aria attribute are technical specifications provided by the WAI-ARIA standard for accessible dynamic and interactive Web content. Some basic components of the Mini Program architecture support aria attribute, which can meet the accessibility needs of visually impaired users.

Limits

Mini Program components that support aria attribute include the following: view, text, icon, button, label, checkbox, switch, and image.

Usage

Common ARIA attribute

Role

The core of aria attribute is the role attribute, which represents the semantic role of a component. For example, when the role attribute is set to **img**, screen reader software will read out **image** when the component is focused. When set to **button**, screen reader software will read out **button** when focused.

```
<view class="button" onTap="defaultTap" role="button" aria-label="Confirm button"
  >Confirm button</view
>
```

In this example, the button is implemented with a view component. The role attribute sets this view component to express the semantics of a button. When a screen reader software focuses on this view component, it will identify that this view component is a button and read out **Button Confirm button**. Mini Programs use AXML syntax, which lacks many semantic tags available in HTML, such as H1, article, header, etc. These tags work with screen reader software to provide more semantic information to users. In Mini Programs, developers can use the role attribute to enhance the semantics of view components to achieve the same effect as developing with HTML tags. For example, HTML sample code for an article:

```
<article>
  <h1>Article Title</h1>
  <h2>Chapter One</h2>
  <p>First paragraph...</p>
</article>
```

For Mini Programs, although there are no components corresponding to article, H1, H2, and other tags, it can achieve the same effect through view components with role attribute. For example:

```
<view role="article">
  <view role="heading" aria-level="1">Article Title</view>
  <view role="heading" aria-level="2">Chapter One</view>
  <view>First paragraph...</view>
</view>
```

When a screen reader software focuses on the view component of **Article Title**, it will read out **Article Title Heading 1**. When focusing on the view component of **Chapter One**, it will read out **Chapter One Heading 2**. **Heading** in **Heading 2** is the semantic information provided by the role, and **2** is set by the aria-level attribute. The aria-level attribute belongs to aria attribute, providing additional information. In this example, aria-level expresses the heading level. Some role attribute values need to be used with aria attribute to take effect. For example, when using `role="img"`, you must configure the aria-label attribute. Otherwise, screen reader software will ignore the role attribute.

Aria-label

aria-label can replace the text content within a component, for example:

```
<view aria-label="aria-label content">Component text content</view>
```

When focused here, screen reader software will read out **aria-label content**, while the original text content in the component will be ignored. aria-label can also be used with components without text content to represent the text information attached to the component, which the system will automatically read out when focused. For example, for the image component, you can use aria-label to add descriptive information to the image.

```
<image src="{{src}}" role="img" aria-label="Landscape painting" />
```

`aria-label` is also used for view components with role attribute, for example:

```
<view
  style="background-image:url(./bg.jpg);"
  role="img"
  aria-label="Landscape painting"
></view>
```

For components with strong visual effects (such as image), simply let screen reader software read out the semantics is not very meaningful for visually impaired users. By providing text descriptions of the content through the aria-label attribute, the read-out content becomes more complete and effective.

Aria-labelledby

Some components are associated with other components and need to be read together to express a complete meaning. For example, the checkbox component often requires a text or view component to carry the text content selected by the checkbox. In this case, you can use the aria-labelledby attribute to associate the two components. After reading the content of the current component, screen reader software will continue to read the content in the component pointed to by aria-labelledby.

```
<label>
  <checkbox aria-labelledby="content" />
  <text id="content">Checkbox content</text>
</label>
```

After the checkbox component in the example gains focus, screen reader software will read out **Not selected Checkbox content Checkbox**. The value of aria-labelledby is the id attribute of another component. The component id attribute must be unique. Do not set duplicate id for components when rendering lists.

Aria-checked

aria-checked indicates whether components such as checkbox and switch are selected. After focusing on these components, screen reader software will read out the selected state of the component **Not selected** or **Selected**, telling users the current selected state of the component. Mini Program native components have this attribute pre-built in, for example:

```
<label>
  <checkbox aria-labelledby="content" value="{{checked}}" />
  <text id="content">Checkbox content</text>
</label>
```

If you are using custom components to develop checkbox, switch, and other components, you need to use the aria-checked attribute to allow screen reader software to obtain the component's selected state. For example, AXML sample code for a custom checkbox component:

```
<view
  class="my-checkbox"
  role="checkbox"
  onTap="handleTap"
  aria-checked="{{checked}}"
  aria-labelledby="myCheckboxText"
>
  <icon type="{{checked ? 'success' : 'clear'}}" size="12" />
  <text id="myCheckboxText">
    <slot> </slot>
  </text>
</view>
```

In the example, the value of the my-checkbox component's role attribute is set to checkbox, and the component's selected state checked is assigned to the aria-checked attribute. When a screen reader software focuses on this custom component, it can read out the selected information of the component. This attribute also applies to components such as switch and radio.

Aria-expanded

aria-expanded represents the expansion information of collapsible components. It is suitable for components with collapsible functionality, such as collapsible menus, dropdown menus with collapsible features, etc. Using the aria-expanded attribute allows screen reader software to read out the expanded state of the component.

Other aria attribute

To learn more about aria attribute, see WAI-ARIA.

Related information

The term accessibility comes from the English word "Accessibility," which means that anyone (whether able-bodied, disabled, elderly, or children) can use or access at any time and in any scenario. Standards organizations (such as W3C, WAI) have established standards for accessibility in the Web domain. Alipay mini programs provide accessibility features based on existing standards.

Accessibility standards

- Web Content Accessibility Guidelines (WCAG) are a set of guiding principles and best practices developed by accessibility experts, aimed at systematically explaining the meaning of accessibility and best practices.
- Web Accessibility Initiative - Accessible Rich Internet Applications (WAI-ARIA) is a technical specification that provides accessible dynamic and interactive Web content for people with disabilities. It defines methods for browser, media player, assistive technology developers, and Web content developers to achieve broader cross-platform accessibility.
- The accessibility features of Mini Programs support some attribute of WAI-ARIA. Developers can use the best practices summarized in WCAG to develop Mini Programs that are more accessible to people with accessibility needs.

Enable visual accessibility features

iOS and Android systems provide visual accessibility features, offering screen reading functionality for blind or visually impaired users, which can read out the content that users touch, select, or activate.

- iOS: Go to **Settings > General > Accessibility > VoiceOver** to enable the VoiceOver feature.
- Android: The screen reading feature in Android systems is called **TalkBack**. Simply enable it to use. (The location to enable TalkBack varies across different Android phone brands. Please refer to the help manual corresponding to your phone to find how to enable the TalkBack feature.)

On phones with voice screen reading enabled, when the screen reading function focuses on a Mini Program component, it will read out the content of the component.

1.12.4. SSO specifications

1.12.4.1. SSO user authorization guide

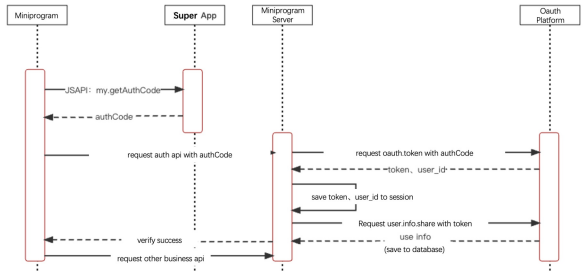
All reading and writing of user information on the Mini Program Open Platform requires the user's permission before it can be used by developers. User authorization is based on the international standard OAuth2.0 authorization mechanism. Based on this mechanism, developers can obtain SuperApp user information, etc.

Terminology

- **scope**: Authorization scope. A scope indicates the scope of permissions that the developer needs to request the user to authorize. A scope contains at least one OpenAPI interface or JSAPI interface. Multiple scopes can be combined for one authorization.
- **auth_code**: Authorization code, which is the same as the authCode obtained by JSAPI, a temporary user authorization credential.
- **access_token**: Access token, also known as auth_token (authorization token).

Integration process

Taking obtaining user information as an example, the overall access process is as follows (if you need to authorize other information, just use a different scope in the scopes parameter when calling my.getAuthCode).



scopes list

scopes	Description	Included OpenAPI interfaces
auth_base	Silent authorization, no authorization popup window will appear; Obtain auth_code on the client side, pass it to the server, and directly call oauth.token (exchange for an authorized access token) on the server side to obtain the unique user identifier user_id (open_id). **Important** In silent authorization scenarios, directly call oauth.token (exchange for an authorized access token).	oauth.token: Exchange for an authorized access token
auth_user	Active authorization requires the user to manually click to agree. After the user agrees, it can be used for operations such as logging into the mini program and obtaining user information.	• oauth.token: Exchange for an authorized access token • user.info.share: Member authorization information query interface

Step 1: Obtain authcode on the client side

Get user authorization by calling `my.getAuthCode` and get the authCode in the success callback.

Rules for using authcode

- auth_code is valid for one-time use only and cannot be reused.
- The validity period ranges from 3 minutes to 24 hours (the Open Platform rules will dynamically adjust the validity period of auth_code according to specific business scenarios, but it will not be less than 3 minutes or exceed 24 hours). An auth_code that exceeds its validity period will automatically expire even if unused.
- Each user authorization action generates a new auth_code.
- For security considerations, developers should call oauth.token (exchange for an authorized access token) to exchange for access_token as soon as possible after obtaining auth_code (user authorization code).

JavaScript code example

```
my.getAuthCode({
  scopes: ['auth_user'],
 // Active authorization: auth_user, silent authorization: auth_base or other scope. To obtain multiple user authorizations simultaneously, yo
u can pass multiple scope values in scopes.
  success: (res) => {
    if (res.authCode) {
      // Authentication successful
      // Call your own server interface to let the server perform backend authorization authentication, and use session, need to solve cross-d
omain issues
      my.request({
        url: 'https://xxx/auth', // This url is the developer's service address, which implements the function of the server taking the
authcode to the open platform for token verification
        data: {
          authcode: res.authCode,
        },
        success: () => {
          // Authorization successful and server-side login successful
        },
        fail: () => {
          // Deal with the errors according to your business scenario
        },
      });
    }
  },
});
```

Step 2: Obtain access_token and user_id (open_id) on the server side

The server calls oauth.token to exchange for the authorization access token. Developers can exchange the obtained auth_code for access_token (authorization token) and user_id (open_id).

auth_code serves as a ticket to exchange for access_token. Each time the user completes authorization, the auth_code in the callback address will be different. auth_code can only be used once and automatically expires if not used within one day.

Important request parameter description

Terminology

refresh_token: refresh token, used to refresh and obtain a new authorization token after the authorization token expires. The refresh token also has a validity period.

- grantType: Authorization type, supports the following types.
  - authorization_code: User authorization. With this type, pass the user authorization code auth_code in the code parameter to exchange for the user authorization token access_token. No need to pass refresh_token.
  - refresh_token: Refresh token. With this type, pass a valid refresh_token to refresh the authorization token. No need to pass code.
- code: User authorization code, the auth_code parameter obtained through the Mini Program interface.

Response example

```
{
    "oauth_token_response": {
        "user_id": "xxx",
        "open_id": "xxxx",
        "access_token": "xxxxxx",
        "expires_in": "3600",
        "refresh_token": "xxxxxxxx",
        "re_expires_in": "3600",
        "auth_start": "2010-11-11 11:11:11"
    },
    "sign":"xxxx"
}
```

Important response parameter description

- user_id (open_id): The unique identifier of SuperApp users.
- access_token: Access token, used to call interfaces that require authorization.
- expires_in: The validity period of the access token, in seconds.
- refresh_token: Refresh token, used to refresh the access_token.
- re_expires_in: The validity period of the refresh token, in seconds.
- auth_start: Authorization start time, used as the starting point for calculating the validity period.

Call server-side business interfaces

After obtaining the access_token, you can continue to use this token to call OpenAPI's authorized interfaces. Please note the permission scope and validity of the token.

> **Note**
> access_token is the auth_token in the **common request parameters** of the server-side interface.

Appendix: In-depth authorization mechanism

Authorization code auth_code

Description

auth_code is valid for one-time use. The validity period of auth_code ranges from 3 minutes to 24 hours (the Open Platform rules will dynamically adjust the validity period of auth_code according to specific business scenarios, but it will not be less than 3 minutes or exceed 24 hours). An auth_code that exceeds its validity period will be unusable even if it has not been used. Each user authorization action generates a new auth_code.

Recommendation

For security considerations, developers should call oauth.token (exchange for an authorized access token) to exchange for access_token (access token) as soon as possible after obtaining auth_code (user authorization code).

Authorization scope

Description

Scope is a public resource. Its use does not require signing a contract, but whether the interfaces it contains require signing a contract depends on the specific feature package or interface definition. Developers can include one or multiple user authorization scopes in an authorization request. Each authorization scope is called a scope. A scope contains several Open Platform interfaces. Multiple requested scopes are separated by commas. The authorization process is universal. In different business scenarios, the scope of authorization required is different. You need to replace the scope parameter in this document with the scope specified in the specific product integration documentation.

Scope validity period

The scope validity period here and the auth_code validity period mentioned earlier are two different concepts. The validity period of the scope will affect the validity period of the access_token and refresh_token that developers ultimately obtain. For the validity period of different scopes, please refer to the specific product documentation.

Recommendation

For product experience considerations, please request only the scopes you need. Too many authorization scopes can easily cause users to abandon authorization. It is recommended to use auth_base or auth_user for user acquisition in the product's login scenario, and then guide users to request specific scope user authorization according to specific business needs.

access_token

Validity period

- The access_token depends on the validity period of the scope specified during authorization. If multiple scopes are specified during authorization, the final validity period of the access_token depends on the scope with the shortest validity period.
- access_token expiration time = (authorization time point) + (expires_in returned by calling oauth.token (exchange for an authorized access token) after authorization).

> **Important**
> The user can cancel the authorization. After the user cancels the authorization, the access_token becomes invalid and cannot be used.

Token storage requirements

- Ensure the secure storage of access_token.
- Save access_token using appId + uid + single scope as an index. Otherwise, interface calls will report errors due to AppID token mixing and token overwriting between different scopes. If the authorization scope is the same in multiple authorizations, only save the access_token with the longest authorization expiration time. Authorization expiration time = authorization time point + expires_in returned by oauth.token (exchange for an authorized access token).

refresh_token

Description

When using auth_code to call oauth.token (exchange for an authorized access token) to obtain access_token, a refresh_token (refresh token) will be returned. Within the validity period of refresh_token, you can call oauth.token with refresh_token to refresh a new access_token.

Validity period

- The refresh_token depends on the validity period of the scope specified during authorization. If multiple scopes are specified during authorization, the final validity period of the refresh_token depends on the scope with the shortest validity period.
- refresh_token expiration time = (the time when the refresh_token was first obtained + re_expires_in returned by oauth.token (exchange for an authorized access token)).

Usage

- Use refresh_token to call oauth.token (exchange for an authorized access token).
- After refreshing, you can obtain a new access_token. The access_token expiration time is recalculated (correspondingly, you can see that expires_in will not change), and the original access_token will immediately become invalid. At the same time, you will receive a new refresh_token, and the original refresh_token will become invalid. The expiration time of the new refresh_token will not be recalculated (correspondingly, you can see that re_expires_in will decrease).

> **Important**
> Users can cancel authorization. After the user cancels the authorization, refresh_token cannot be used even if it is within its validity period.

FAQ

Q: How does a Mini Program implement user authorization?

A: Mini Programs **do not support** authorization by concatenating authorization links. It is recommended to use my.getAuthCode to implement user authorization, user login, etc.

Q: Can my.getAuthCode be used when the Mini Program onload?

A: Yes, but it must be silent authorization. Mini Program review prohibits forcing an authorization box to pop up immediately upon entry.

1.12.4.2. SSO interface documentation

oauth.token (Exchange for authorization access token)

Common request parameters

None.

Business request parameters

Parameter	Required	Field length	Description	Example value
grant_type	Yes	string[1,20]	The authorization type. • Use authorization code to exchange for an access token: authorization_code • Use refresh token to exchange for a new access token: refresh_token	authorization_code
code	Optional	string[0,40]	Authorization code, which can be obtained after the user authorizes the application. **Note** This parameter is required when grant_type is authorization_code. Do not fill in when grant_type is refresh_token.	xxx
refresh_token	Optional	string[0,40]	Refresh token, obtained when exchanging for access token last time. **Note** This parameter should not be filled when grant_type is authorization_code. It is required when grant_type is refresh_token, and its value comes from the return value app_refresh_token of this interface (which means you need to call this interface at least once with grant_type=authorization_code to obtain it).	xxx

Common request example

```
curl 'https://xxxx?charset=UTF-8&method=oauth.token&format=json&sign=${sign}&app_id=${appid}&version=1.0&sign_type=RSA2&timestamp=${now}' \
 -F 'refresh_token=xxx' \
 -F 'code=xxxxx' \
 -F 'grant_type=authorization_code'
```

Note
This example is for reference only.

Common response parameters

None.

Business response parameters

Parameter	Required	Field length	Description	Example value
access_token	Yes	string(40)	Access token, used to call interfaces that require authorization.	xxxxx
expires_in	Yes	string(16)	The validity period of the access token, in seconds.	3600
refresh_token	Yes	string(40)	Refresh token, used to refresh the access_token.	xxxx
re_expires_in	Yes	string(16)	The validity period of the refresh token, in seconds.	3600
user_id	Choose one between user_id and open_id	string(16)	The userId of the user.	xxxxx
open_id	Choose one between user_id and open_id	string(128)	The openId of the user.	xxxx

auth_start	Optional	date(32)	The start time of the authorization token, used as the starting point for calculating the validity period.	2010-11-11 11:11:11

Response example

```
{
    "oauth_token_response": {
        "user_id": "xxx",
        "open_id": "xxxx",
        "access_token": "xxxxxx",
        "expires_in": "3600",
        "refresh_token": "xxxxxxx",
        "re_expires_in": "3600",
        "auth_start": "2010-11-11 11:11:11"
    },
    "sign": "xxxxx"
}
```

> **Note**
> This example is for reference only.

user.info.share (Member authorization information query interface)

Used with the member authorization interface to query authorization information.

Common request parameters

None.

Business request parameters

None.

Common request example

```
curl 'https://xx?charset=UTF-8&method=user.info.share&format=json&sign=${sign}&app_id=${appid}&version=1.0&sign_type=RSA2&timestamp=${now}' \
 -F 'app_auth_token=${app_auth_token}' \
 -F 'auth_token=${auth_token}'
```

> **Note**
> This example is for reference only.

Common response parameters

None.

Business response parameters

Parameter	Required	Field length	Description	Example value
avatar	Yes	string(400)	The URL of the profile picture of the user. **Important** If there is no data (the user has not set it), this information will not be returned. Please handle this case properly.	-
city	Yes	string(20)	The city.	Shanghai
nick_name	Yes	string(50)	User nickname. **Important** If there is no data (the user has not set it), this information will not be returned. Please handle this case properly.	Xiao Er
province	Yes	string(20)	The province.	Anhui
user_id	Choose one between user_id and open_id	string(16)	The userId of the user.	2088102104794936
open_id	Choose one between user_id and open_id	string(128)	The openId of the user.	xxxxxx

gender	Optional	string(10)	The gender. Enumeration values are as follows: • F: Female • M: Male	F

Response example

```
{
    "user_info_share_response": {
        "code": "10000",
        "msg": "Success",
        "user_id": "xxxxxx",
        "avatar": "",
        "city": "Anqing",
        "nick_name": "Xiao Er",
        "province": "Anhui",
        "gender": "F"
    },
    "sign": "xxxxxxx"
}
```

Note

This example is for reference only.

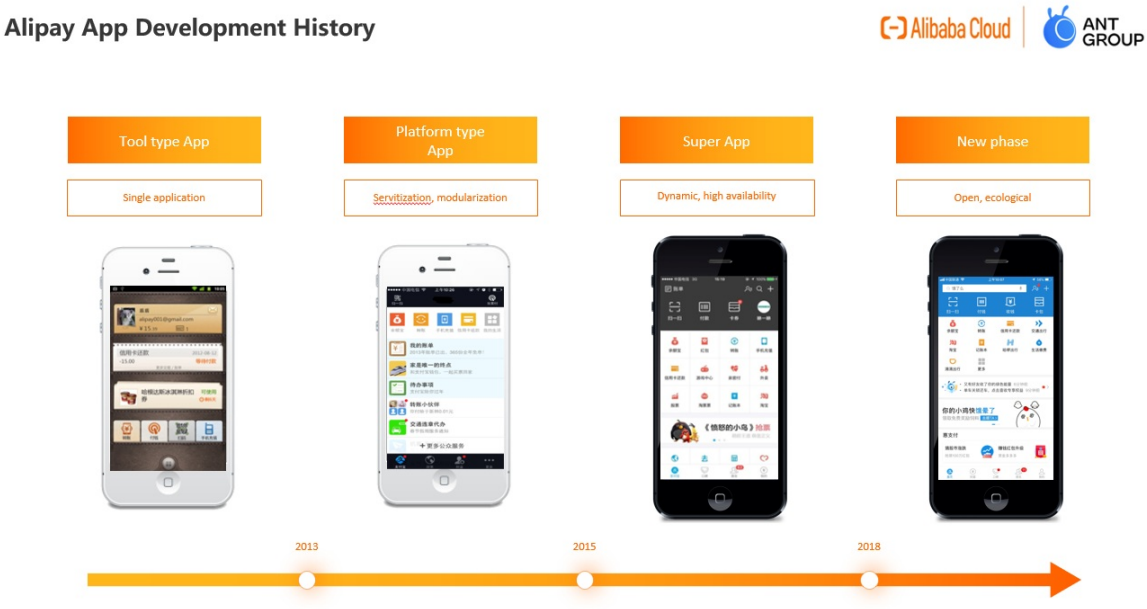
1.13. Deep Analysis of the Technical Architecture of mPaaS Mini Programs

As Mini Program technology becomes more mature, the advantages and typical scenarios of different platforms have their own emphasis. At the same time, more and more developers can combine their own business characteristics and use Mini Programs as business carriers to form a single platform or multi-platform collaborative relationship. Today, with the opening of Mini Program technology, mPaaS Mini Program framework, as a general framework for App, helps developers to implement Mini Program delivery for their own App. Not only that, the Mini Program code only needs to be written once, and can be multi-terminal delivered to its own App, Alipay, DingTalk and even other Mini Program open platforms.

This article focuses on the evolution of Alipay's mobile architecture and shares our thoughts and practices on App dynamics and improving R&D efficiency. At the same time, in view of the opening of mPaaS Mini Program capability, we will also introduce how to realize "write Mini Program code only once, multi-terminal delivery", which will bring developers a completely different development experience.

Alipay App Development History

First of all, let's review the specific development process of Alipay App in recent years.



At first, Alipay was only a tool App for single application, allowing users to complete Alipay-related business inquiries and operations on their mobile phones. After 2013, Alipay has gradually transformed into a platform-based App, which has the characteristics of "service, modularization and tool componentization". At this time, Alipay's business is not only to pay, but also to provide customers with a lot of life-related services, such as Yu'E Bao, pay electricity bills and so on. After 2015, Alipay has grown into a super App. At this time, Alipay needs to support a large number of complex businesses. In 2018, with the launch of Mini Programs, Alipay began to open up its business capabilities and use its own traffic to help partners. Therefore, the entire App faces the challenges of openness, dynamics, and high availability. Facing these challenges, we summarize it into the following three aspects:

Dynamics and Experience

- In the face of diverse requirements, how to ensure the rapid iteration of the business?
- On the premise of ensuring the dynamic update of App, how to ensure the user experience?

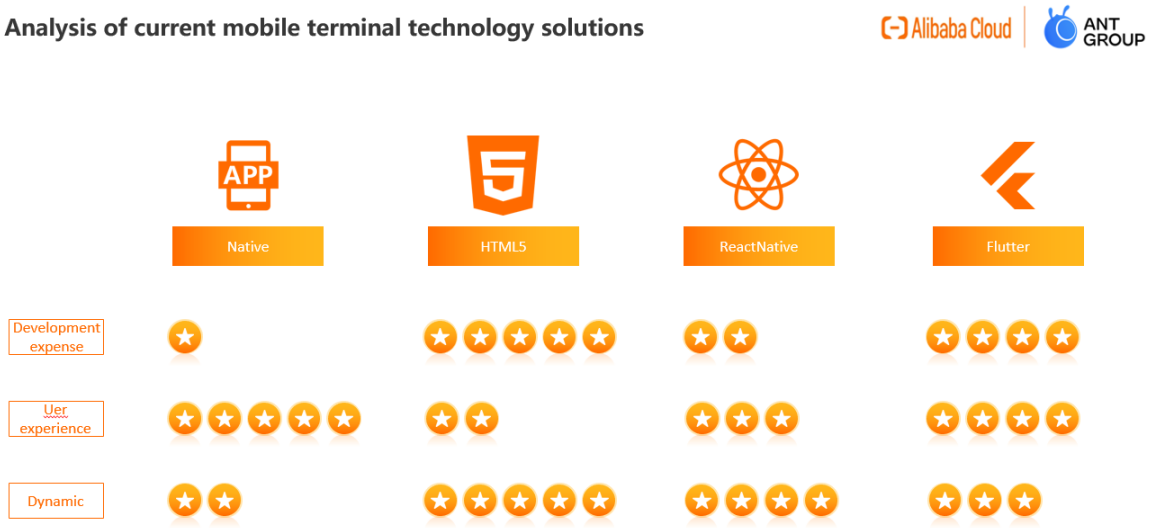
R&D efficiency

- How to write code once and reuse it in multiple terminals?
- Without client development experience, how can I improve development efficiency?

Open ecology

- How can we open up our capabilities to more developers?
- How to connect more ecological platforms and enrich your App scenarios?

If there are problems, we will solve these problems through technical means. We will proceed from the above three directions to select technology.



First of all, from the perspective of business development costs:

- As the most basic development mode, native development needs to be carried out at both ends, which is undoubtedly the highest cost.
- The second is ReactNative/Weex. Even if it is developed once and runs on both ends at the same time, there are still some differences in actual operation because JS is converted into Native component rendering. As a result, when developers write business interfaces, some differences need to be resolved through Native side custom development. Overall, ReactNative/Weex has helped businesses significantly reduce development costs, but there is still a lot of end-to-end adaptation work.
- Next is Flutter. From the perspective of business development, Flutter has really made great efforts for double-end alignment. In most scenarios, Android can run seamlessly on iOS after development, which is of course related to its self-developed engine. However, Flutter needs to be developed based on the Dart language, so for developers, the workload of porting some old businesses must be considered;
- Finally, HTML5, with mature language, mature development mode, double-end almost the same performance and other characteristics show that HTML5 is still the lowest development cost scheme that we can provide at present.

Let's talk about the user experience:

- First of all, the native experience is undoubtedly the best;
- The second is Flutter with its own rendering engine, which can be said to be no less than the native experience in terms of performance and display of controls.
- The next step is the ReactNative/Weex solution, which renders the front-end code into the local Natvie control. In the earlier version, App stuck due to the inadequate optimization of some controls, so the performance of user experience was insufficient;
- Finally, HTML5 is completely rendered through the browser kernel. With technologies such as preset resources and kernel optimization, HTML5 can achieve a close-to-native experience, but the overall performance is still different.

Next is dynamic support:

- First of all, the best dynamic is the HTML5 solution: you can access the online page, the server takes effect immediately, or you can dynamically update it by sending resources.
- The second is the ReactNative/Weex solution. Through certain customization, developers can hot deploy and hot update the front-end package. However, compared with the dynamic nature of "online + offline" in HTML5, there is still a certain gap in the scheme.
- Next is Flutter. Although there is a powerful hot reload mechanism, due to Google's limitations, the official version of iOS cannot be hot updated. Currently, iOS can be hot updated by modifying the engine, JIT and AOT, or by adopting run time parsing rendering to achieve dynamic. However, compared with the above two schemes, Flutter is slightly less dynamic.
- Finally, the native Android/iOS both ends can be dynamically updated through some technology means. However, due to the prohibition of iOS policy, the native scheme is not recommend for the time being in terms of dynamics.

After analyzing the different directions of the four schemes, the answer brought by mPaaS is: "A Hybrid architecture solution that takes into account dynamics, experience, development efficiency, and openness, that is, the mPaaS Mini Program."

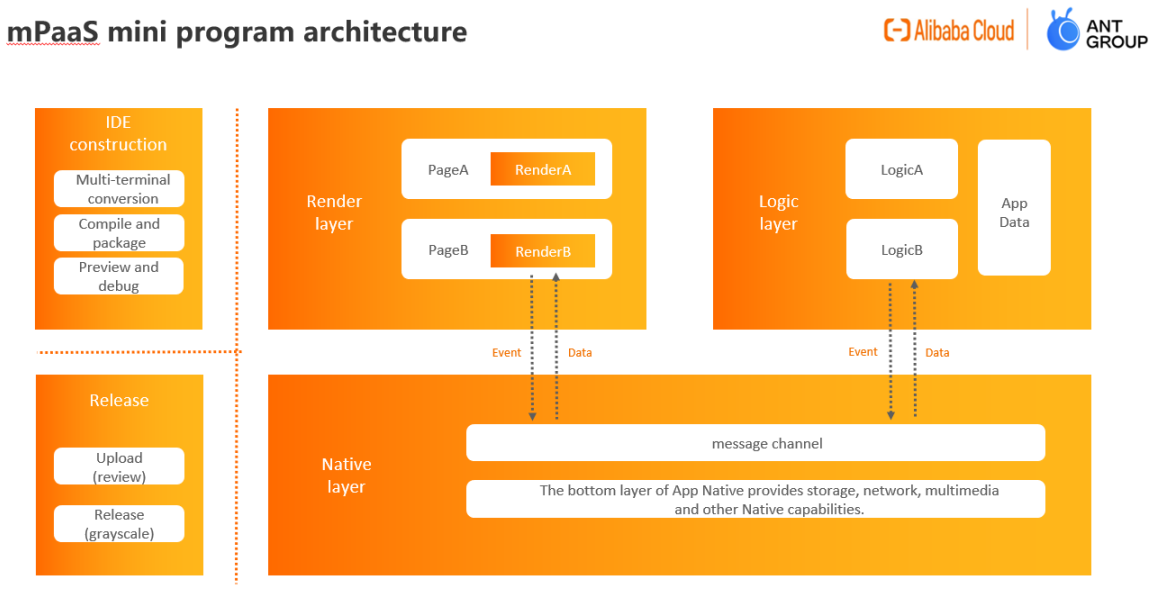
Technical analysis of mPaaS Mini Programs

What is a Mini Program?

According to the definition of Mini Programs in the w3c Mini Program white paper, Mini Program is a new mobile application program format that relies on Web technology and integrates native capabilities. It has the four characteristics of "convenient access, stable connection, safe and reliable, and excellent performance."

mPaaS Mini Program, based on Web technology, low learning cost. A set of Mini Program code, which supports both iOS and Android, close to the native experience. It also provides a wide range of components and APIs, such as obtaining user information, local storage, and payment functions.

Next, let's disassemble the complete technical architecture of the Mini Program and try to expand the overall architecture of the Mini Program through the "operation phase, development phase, and release phase".



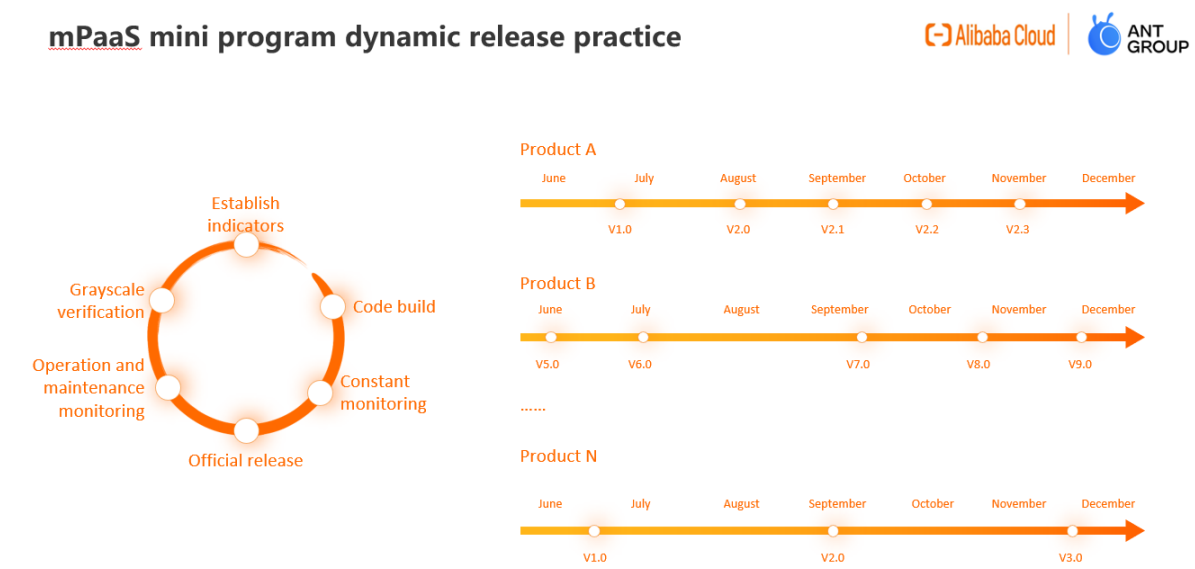
Operation phase The Mini Program uses a dual-thread mode to place page rendering and business logic in two separate threads. renderer runs in WebView and is responsible for rendering the interface. The Mini Program business logic runs on a separate worker thread and is responsible for event processing, API calling and lifecycle management. The two threads exchange data through postMessage and onMessage. Data can be passed from the worker thread to the render re-rendering interface, and the renderer can also pass the event to the corresponding worker for processing. A worker can correspond to multiple renderers to facilitate data sharing and interaction between pages.

For scenarios that require high rendering speed and interactive response, such as maps, Mini Programs embed native map components into WebView. Compared with rendering maps on Canvas, Mini Programs draw more quickly and efficiently.

In terms of resource loading, the Mini Program is loaded offline, that is, when the Mini Program is opened, the offline package of the Mini Program must be downloaded locally. Since each version is downloaded only once, on the one hand, the resource overhead of each request is saved, and on the other hand, the startup speed is greatly improved. When a new version is available, the release platform automatically compares the locally installed version with the latest version to generate and distribute a difference package. The client can update the Mini Program to the latest version without downloading the entire package.

Development and release phase Application development must be without the support of a complete tool chain. The Mini Program IDE integrates coding, debugging, preview, and release capabilities. After simple adaptation, the client can preview and debug Mini Programs in real-time in real-time applications.

After we have a preliminary understanding of the Mini Program architecture, let's take a look at how mPaaS Mini Programs will be dynamically released. This is precisely the highlight of the core of the mPaaS Mini Program. With the ability of "package release and management", the R&D and iteration efficiency of App can be deeply optimized.



As shown in the preceding figure, we have redefined the R&D model and release process. Each Mini Program can be used as an independent product and has its own release process without waiting for other teams. Each business team is completely split, and each business evolves and releases independently.

During the publishing process, follow the following process:

1. Metric linearity, defining business and performance metrics for each release;
2. Intelligent grayscale, internal grayscale, external grayscale, and specified grayscale;
3. Real-time monitoring, repair cycle;
4. Online operation and maintenance repair means technical bottom.

Then we will talk about the safety of Mini Programs.

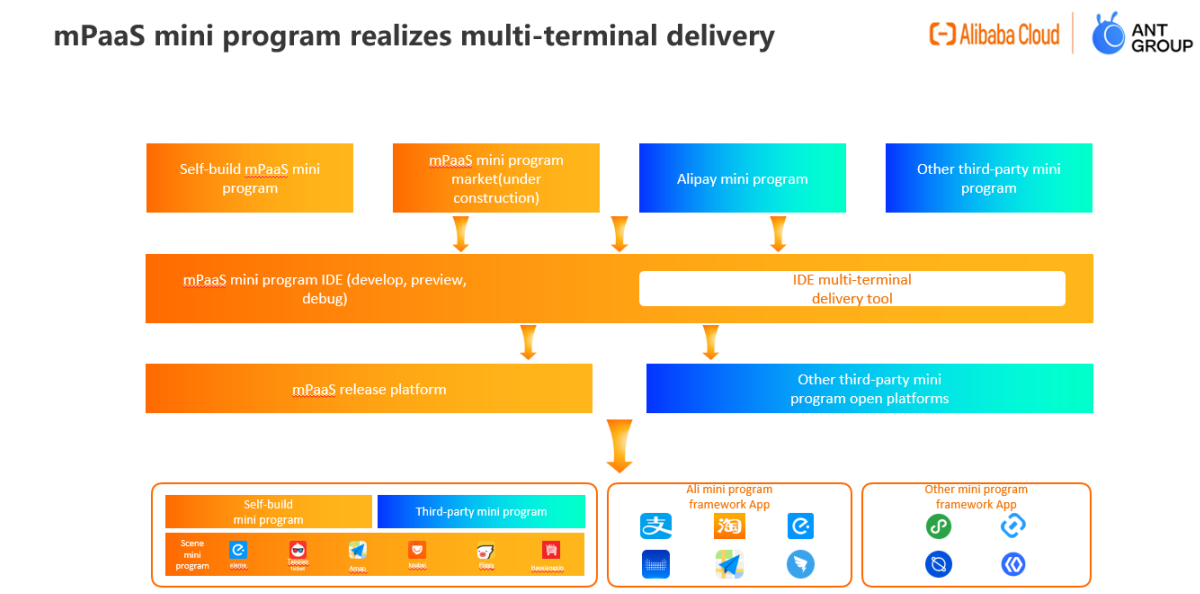
- **Connection security** Based on the capabilities of Alibaba's wireless bodyguard, the security of Mini Program requests is ensured. The tampered requests cannot pass verification.
- **Security of package body** The package body is encrypted and signed to ensure the security of the download process. Tampered Mini Program packages cannot be used.
- **Permission security** Complete permission management system, open different permissions for different Mini Programs, to ensure the privacy and security of users.

Then we turn to the Mini Program framework capability extension system.

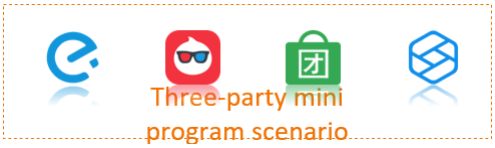
The mPaaS Mini Program itself has integrated nearly hundreds of commonly used APIs, including network, media, storage, positioning, code scanning, Bluetooth, etc. These APIs can also run perfectly in Alipay. Moreover, application developers can reveal their own functional mPaaS Mini Program extension capabilities to Mini Program developers. This expansion mainly includes three aspects:

- Capability extension: provides custom event capabilities and supports "Mini Program> native" and "native> Mini Program".
- Style extension: provides a variety of native style customizations, including native styles such as navigation bar, loading animation, and startup animation.
- Component extension: provides the ability to customize components and extend the Mini Program label.

Finally, let's talk about the multi-terminal launch and ecology of Mini Programs.



Based on the mPaaS Mini Program system, we can convert a lot of Mini Program standards into standard Mini Program products through tools, such as mPaaS Mini Programs written by developers themselves, or Mini Programs in the mPaaS Mini Program market, or Alipay or other three-party Mini Programs. After the conversion is completed through IDE, we can put it on different ends through two channels. Using the mPaaS publishing platform, you can put it into your own App, using other three-party open platforms, you can put it on the corresponding end, one-time development, only a small amount of adaptation, you can put it on multiple ends.

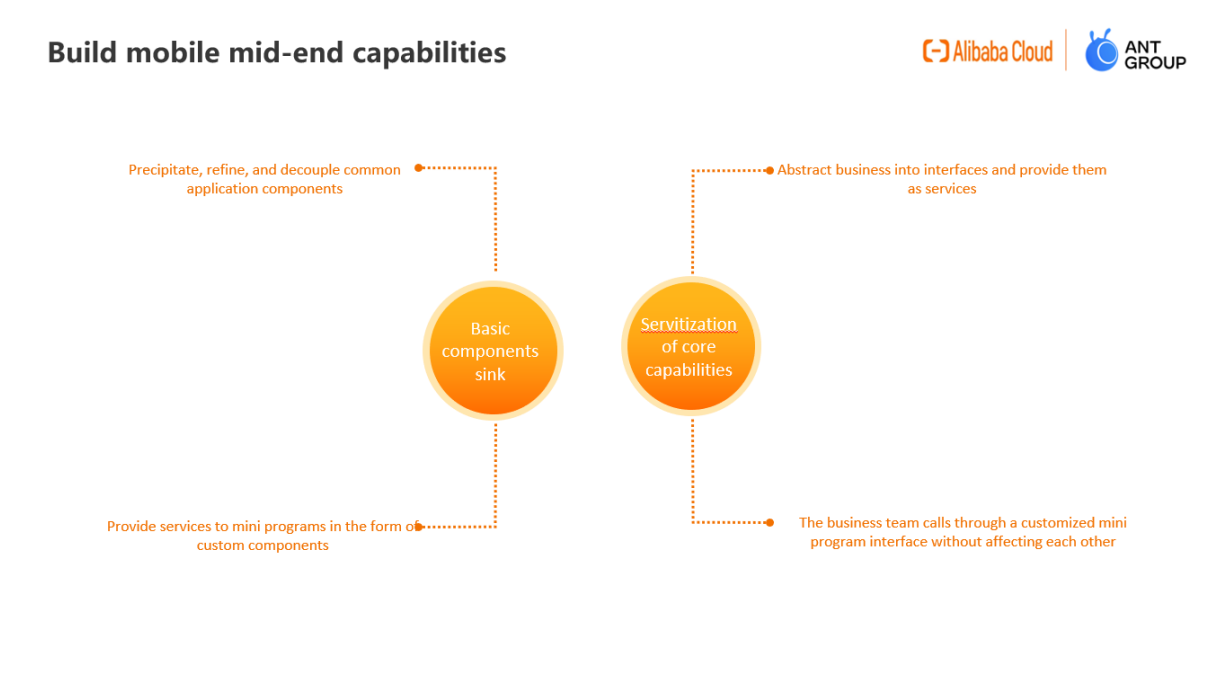


Of course, for the relatively lack of business scenarios in its own App, based on the unified Mini Program framework capability of mPaaS, the three-party business scenarios of Ali department can realize seamless delivery, thus meeting the needs of developers to enrich their own business scenarios.

Mobile capability construction based on mPaaS Mini Programs

After introducing the technical architecture and capabilities of mPaaS Mini Programs, let's talk about the specific research and development direction based on mPaaS Mini Programs.

Mobile mid-end capacity building



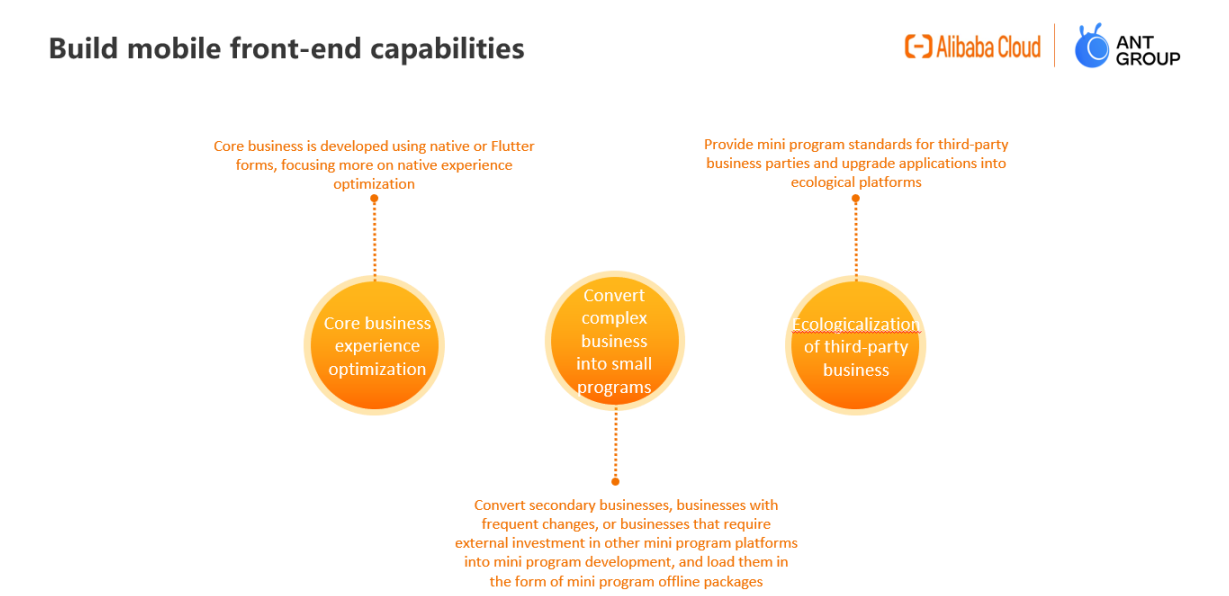
The so-called mobile mid-end capacity building, we hope to integrate the entire App architecture: at the basic level, further develop common components, avoid repeated work, and standardize service interfaces to provide high-quality, stable and standard services for more upper-level businesses. Then we need to deal with this matter from two aspects.

Basic components

We may have such a problem in the development process, that is, the two teams collaborate in development. Maybe everyone has some classic components. We can further develop these components. At the same time, we can also provide services to Mini Programs through the custom component capability of Mini Programs. **Core capability servitization**

After the components are precipitated, we need to service some core business capabilities and abstract the standard service interfaces or Mini Program APIs for other teams or third-party ecosystems to call. For example, Alipay's payment services, sesame credit services, etc., are based on the service, and ultimately good for other businesses to provide services.

Mobile front end construction



After we complete the mobile mid-end capacity building, the overall capability is already available, and the rest is to combine the Mini Program framework to build our mobile front end capability.

- Core business experience optimization

  For some very core business logic, such as Alipay payment, as well as some performance-demanding business, such as home page, or some special interactive pages. usually, we want to implement pages by using native pages or native technologies such as Flutter. Because these pages usually do not have major changes, the requirements for dynamic capabilities are not very strict, and the native can meet the needs of a variety of user experiences on these pages.

- Transform complex business into Mini Programs

  For some complex secondary businesses, the business itself may be iterated frequently, so the native development experience will be disastrous. At this time, we need to separate this part of the business, transform the business into a Mini Program through the front-end technology, and then publish the offline package to the application through the release service. In this way, it meets the complex and changeable scenarios of our business.

- Third-party ecologicalization

  We not only provide a variety of services ourselves, but also need to introduce third-party services to serve more people. The traditional H5 page is not as good as a standardized and simple Mini Program due to its overly broad front-end standards and certain technical threshold. At the same time, using the mobile mid-end construction we introduced above, we can provide a variety of self-mid-end capabilities for third-party Mini Programs to complete the diversification of scenarios.

Around how Mini Programs can help us transform our own business modules and gradually form dynamic updates, I believe everyone has a more comprehensive understanding.

> **Note**
> Currently, the mPaaS Mini Program is available for free trial. Welcome to the experience. In the integration test phase, if you have any Q&A requirements, you are also welcome to search the DingTalk search group 32843812 for communication.

1.14. FAQ

1.14.1. FAQ of using console

If the mini program package is not released to the application store, can it be started directly in the App after releasing the test package in the console?

After releasing the test package on the console, you can use the IDE to release the debugging or preview and obtain the mini program by scanning QR code with a mobile phone.

1.14.2. Mini Program FAQs

In some Windows environments, what can I do when the Mini Program integrated development environment (IDE) reports an error?

Close the Mini Program IDE completely, and then reopen the IDE using administrator privileges.

How can I reuse an earlier version of the Mini Program IDE?

- Download and install an earlier version of the Mini Program IDE in Download center.
- Downgrade the IDE version using the downgrade plan of the mobile PaaS (mPaaS) Mini Program IDE.
  i. Open the mPaaS Mini Program IDE, and select any mini program to enter the IDE.
  ii. In the top navigation bar, choose **Help** > **Local configuration**.
  iii. Enter the following code.

```
{
  "localAdaptorDebug": {
    "resourceCode": "mPaaS",
    "iterationId": 12000103,
    "productId": "tiny_app_ide_mac"
  }
}
```

  iv. Save and exit the mPaaS Mini Program IDE completely.
  v. Open the mPaaS Mini Program IDE again.

How can I obtain the ID of a running mini program from code?

Use the `update` method to obtain information about the running mini program.

1.14.3. FAQ about mini programs

How to solve the reported error "command not found: mpaas" when the mini program is integrated into iOS?

The `command not found: mpaas` error is reported because the command line tool is not installed. For more information, see the Prerequisites of accessing mPaaS based on existing projects. The mPaaS plugin has not been installed. Use the `sh <(curl -s http://mpaas-ios.oss-cn-hangzhou.aliyuncs.com/cocoapods/installmPaaSCocoaPodsPlugin.sh)` command to install the mPaaS plugin.

2.Mini Program Release

2.1. About Mini Program Release

The mPaaS Mini Program Release service is intended for Mini Program developers. The service supports **Mini Program encryption**, **switch configuration**, **whitelist**, and **release rule** management functions.

After you integrate the Mini Program function into your client, you can generate a package in mPaaS plugin, 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 grayscale release based on whitelist.

Features

- **Grayscale release**

  Before the official release, you can perform a small-scale release (such as internal employees) through a whitelist to verify whether the new package meets expectations. You can also do a time window grayscale release, which is released to a specified number of users within a specified time period. If it meets expectations, it can be delivered to the entire network.

- **Advanced filtering**

  When performing grayscale releases, you can also use advanced rules to define a more precise whitelist population. For example, you can send it only to users of Xiaomi phones. Multiple filtering rules can be superimposed, and the mini program package will only be delivered if all filtering rules are met.

- **Custom signature verification**

  To ensure security, a custom signature verification process is used to ensure the correctness of the script source. The mPaaS plugin provides resource packages and the function of signing the packages.

Benefits

- **Supports intelligent grayscale release and multiple upgrade policies**

  There are many rules to choose from, including whitelist grayscale, time window grayscale, population region, model, network, etc.

- **Provides incremental offline packages for updates**

  It reduces data redundancy and device bandwidth usage, thus showing advantages in scenarios where mobile network conditions are unstable.

- **Ensures high sensitivity and availability**

  The client RPC interface capability has been upgraded, with an availability rate of up to 99.999%, providing online minute-level reach capabilities.

- **Guarantees high performance**

  The reach rate reaches 99.999%, and the daily UV supports more than 200 million.

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

Log in to 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 **Configuration management**.
3. In the **Domain name management**, enter the virtual domain name, such as `example.com` .

   > **Note**
   > - Virtual domain names cannot be second-level or third-level domain names that begin with HTTP or HTTPS.
   > - You must use domain names that you registered yourself.

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**.

Next 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.

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**
> - The Mini Program ID is a 16-digit number, and it is recommended not to use numbers starting with 666 or 20000 for the Mini Program ID.
> - 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.

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 task** on the right.
3. On the **Create release task** page, complete the following configuration:
   - **Release type**: Choose **Grayscale 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 Grayscale 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 release.

Result

On the **Mini Program package list** page, you can see the status of the released Mini Program package is displayed in **Grayscale 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.

Next 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.

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 release 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** 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**.

> **Note**
> You can add up to 30 server domain names to the whitelist.

Edit/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**.

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**.

Edit/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 Analysis

3.1. About Mini Program Analysis

mPaaS Mini Program Analysis is a data analysis component for Mini Program developers and operators. This component provides data statistics capabilities for Alipay, WeChat, and mPaaS Mini Program platforms, supports comprehensive data analysis of Mini Programs on the three major platforms, and visualizes statistical analysis data, which is clear at a glance, assists in product operation decision-making, and drives product experience optimization.

Mini Program Analysis provides user analysis, page analysis, sharing analysis, event analysis, and funnel analysis functions, showing the mini program data performance such as the scale of new/active users, user sources, usage time, user retention, sharing and dissemination, helping you understand the overall operation of the mini program and providing data support for product optimization and operational decisions.

Features

Mini program management

The statistical analysis data of the Mini Program is displayed from two dimensions: App and single Mini Program. For a mini program, it provides a full tracing analysis path: "Customer acquisition > Activation > Sharing". You can easily add WeChat mini programs, Alipay mini programs, and mPaaS mini programs for data statistics and analysis. You can query, modify, and delete existing mini programs with a few clicks.

User analysis

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 analysis

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 analysis

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.

Event analysis

Mini Program event analysis tracks user behavior within the Mini Program, collects user behavior data, and conducts multi-dimensional and instant event analysis based on event attributes to meet the personalized analysis needs of the Mini Program. It also visualizes the analysis data through charts.

Funnel analysis

Funnel analysis is mainly used to analyze the conversion and loss of each step in a multi-step process, helping you locate the loss link of mini program users, discover the reasons for the loss, and then improve the user conversion rate through product optimization or operational activities.

Scenarios

This component is designed for developers who develop mini programs on multiple platforms. After integrating this component, the data of mini programs running on different platforms can be viewed in one console.

User activity analysis

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 and spreading analysis

By analyzing the sharing of Mini Program pages and the effect of their dissemination, we can evaluate the effect of valuable functions or content on the dissemination of Mini Programs. By analyzing indicators such as the number of shares, the number of sharers, the return flow of shares, and the number of new users brought by sharing, we can understand which users have shared the most times and which users have brought the most new users to the Mini Program, and then specify corresponding operation strategies for these users.

3.2. 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.
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.

Metric	Description
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.

3.3. 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.

3.4. 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 program greatly affects the user experience and in turn, also 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.

> **Important**
> As of August 26, 2024, the monitoring of new user numbers and cumulative user indicators of mini program will be offline.

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-application open exceptions, exceptions in JS API calls, exceptions in requests for application 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 mini program monitoring list

A user can monitor up to 100 mini programs. In other words, a user can have multiple applications and create multiple mini programs for each application, 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 application 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 data of today and yesterday on the number of active users, the number of startup times 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 application, 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-application open exceptions, the white screen rate, the exception rate, the number of exceptions in JS API calls, and the exception rate of requests for application 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).

Monitoring metrics

Metric	Description
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.
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.
PVs	The total number of visits to all pages of the mini program.
In-application open exceptions	The number of times that exceptions occurred when users open mini program pages in an application.
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‰
JSAPI exception error rate	JS exception rate = Number of JS exceptions/Number of PVs x 1000‰
JS exception count	The total number of JSAPI request exceptions for the mini-program.

Exception rate of requests for application packets	Exception rate of requests for application packets = Number of exceptions in requests for application data packets/Total number of requests for application data packets

5.Mini Program Open Platform (Public Preview)

5.1. Introduction to the Mini Program Open Platform

The mini program open platform is divided into developer platform and operation management platform. Most users of the developer platform are developers of third-party mini programs. After logging in quickly through the whitelist, the mini programs are synchronized to the operation management platform after creating mini programs, modifying mini programs, downloading configuration files, experiencing mini programs, trialling mini programs, gray scale, and official release. The operation management platform operator can launch the mini program after passing the audit. Functions such as the release of mini program packages and the testing of mini program rely on the mPaaS service of Ant Technology.

Functions and features

Manage mini program lifecycle

Provides complete lifecycle management of mini programs, from creation to development and debugging, from test preview to phased release and full shelf. Each step is very easy.

Unified management and review by multiple development groups

The management background supports unified review of mini programs from different sources and reviews based on the unified mini program quality and privacy policy to ensure that the host App provides a unified quality of user experience.

Stable and continuously optimized mini program container SDK

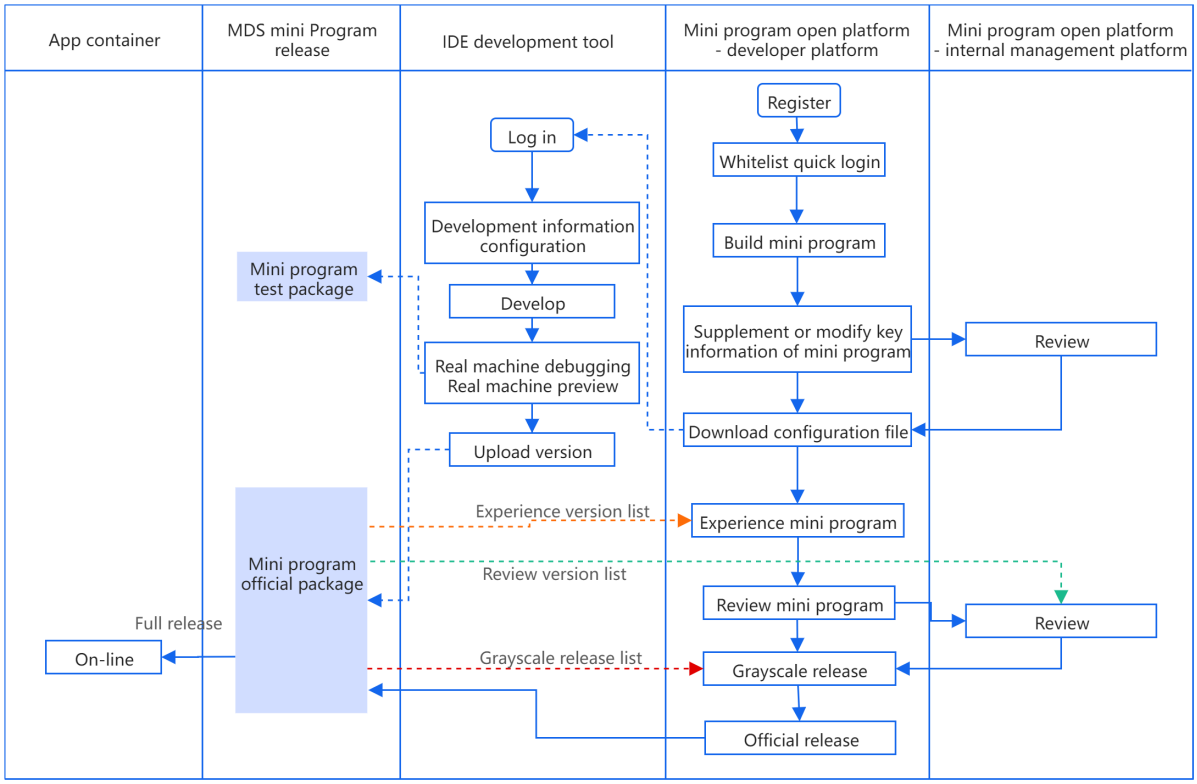
Provides a stable and efficient mini program container SDK to support the host App to run mini programs. You only need to integrate the container SDK into the App to obtain the ability to run mini programs, obtain a user experience comparable to native development, and have more system and device permission calling capabilities, which can greatly reduce development costs.

Unified management and review by multiple development groups

The management background supports unified review of mini programs from different sources and reviews based on the unified mini program quality and privacy policy to ensure that the host App provides a unified quality of user experience.

Procedure

The operation process of the mini program open platform is as follows.



5.2. Developer backend operation manual

5.2.1. Register for the developer console

Prerequisites

You must have an Alibaba Cloud account. If you don't have one yet, complete the account registration first.

Procedure

1. Log on to the mPaaS console and access the mPaaS application. In the navigation pane on the left, choose Mini Program > **Mini Program Open Platform** > **Developer Console**.
2. On the Mini Program Open Platform developer console page, click **Register Now**.

3. On the **Registration** page, enter your **registration mailbox** and obtain the **verification code**. Enter an **alias** and a **password**.

   The verification code will be sent to you as shown below:

4. Click **Register** to complete the registration.

5.2.2. Quick logon

Prerequisites

You must register for the Mini Program Open Platform developer console. If you have not registered yet, complete Register for the developer console first.

Procedure

1. Log on to the mPaaS console and go to your mPaaS application. In the navigation pane on the left, choose Mini Program > **Mini Program Open Platform** > **Developer Console**.
2. On the **Logon** interface, enter your **Account** and **Password**, and click **Log On** to log on to the Mini Program Open Platform developer console.
3. In the **Welcome To Mini Program Open Platform** dialog box, select a developer group.

> **Note**
>
> Developer groups are created in the **Operations Management Console** of the **Mini Program Open Platform**. After you select a developer group, you can publish and manage miniapps under the application. For more information about developer groups, see Developer group management.

4. Click **OK** to log on to the Mini Program Open Platform developer console. You can now publish and manage miniapps.

5.2.3. Homepage

After you log on to **Mini Program Open Platform** > **Developer Console**, you are redirected to the **Homepage**. On the **Homepage**, you can quickly create, configure, and publish miniapps.

Create and complete miniapp information

1. On the **Homepage**, click **+ Add Miniapp**.
2. In the **Add Miniapp** dialog box, enter the **Miniapp Name**, **Miniapp ID**, and **Miniapp Description**, and select the **Miniapp Purpose**.
3. Click **OK** to create the miniapp.
4. Click the newly created miniapp to add basic information about the miniapp. For more information, see View miniapp details.

Develop miniapps

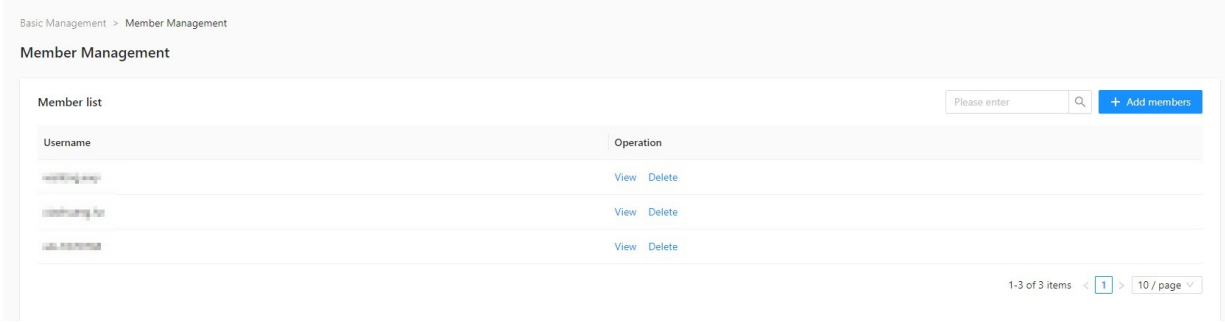
Add developer group members for the miniapp in Member Management, click **Development Configuration** on the **Homepage**, and configure the miniapp. Download IDE to develop, debug, and upload code.

Publish miniapps

1. On the **Homepage**, click **My Miniapps** to go to the **My Miniapps** page.
2. Click **View** in the Operation column of the **Miniapp List**.
3. On the **Intelligent Platform Miniapp Details** page, configure the trial version of the miniapp, submit the miniapp for review, implement a phased release of the miniapp, and complete the miniapp listing. For more information, see My miniapps.

5.2.4. Member Management

Members have the permission to open the development version and the experience version of the mini program. In the left-side navigation pane, choose **Basic Management** > **Member Management**. The **Member Management** page appears, then you can add and delete members on this page.

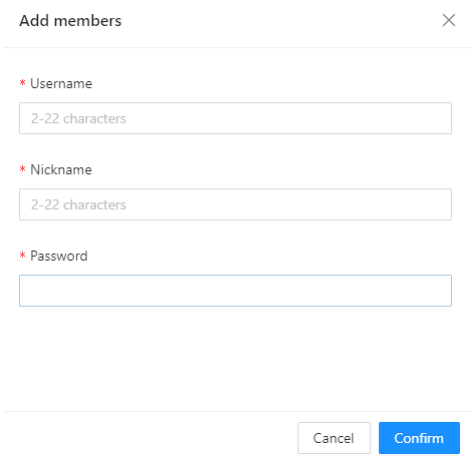


Add members

1. On the **Member Management** page, click **+ Add members**.



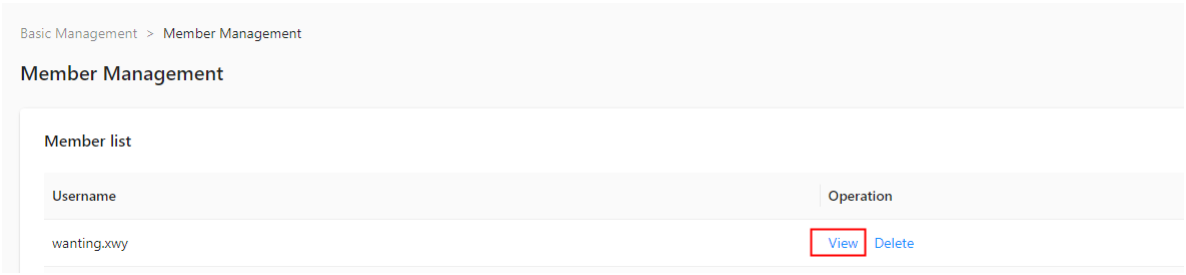
2. In the **Add Members** dialog box that appears, set **Username**, **Nickname**, and **Password**.

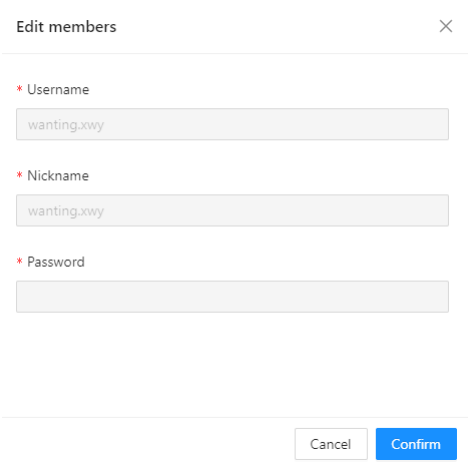


3. Click **Confirm**. The new member is added.

View members

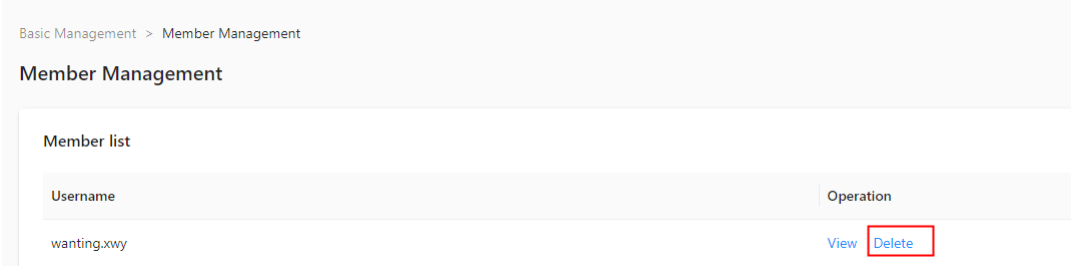
On the **Member Management** page, click **View** in the action column to view the username and nickname of a member.





Delete members

On the page of the **Member Management** page, click **Delete** in the action column. After the second confirmation, the member information will be deleted.



5.2.5. Mini Program management

5.2.5.1. My Mini Program

5.2.5.1.1. Overview

In the **developer backend** of the **Mini Program Open Platform**, click **Mini Program management** > **My mini programs** in the navigation pane on the left to access the **My mini programs** page.

On the **My mini programs** page, you can develop and manage mini programs, including creating mini programs, viewing mini program details, editing mini programs, enabling/disabling mini programs and phased release of mini programs.

On the **Intelligent Platform Mini Program Details** page, you can perform the following operations: mini program version management, mini program package management, mini program review history, mini program phased release, custom API management.

5.2.5.1.2. Create a mini program

The operation flow to create a mini program is as follows:

1. In the navigation pane on the left of the **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Programs** to access the **My Mini Programs** page.

2. On the **My Mini Programs** page, click **+ Add Mini Program** in the upper-right corner.

3. In the **Add Mini Program** dialog box, enter the **Mini Program Name**, **Mini Program ID**, **Mini Program Purpose**, and **Mini Program Description**.

4. Click **OK** to complete the mini program creation.

5.2.5.1.3. View mini program information details

1. In the navigation pane on the left of **Mini Program Open Platform** > **Developer Console**, click **Mini Program Management** > **My Mini Programs** to access the **My Mini Programs** page.
2. On the **My Mini Programs** page, click **View** in the operation column of the mini program list.

   On the **Smart Platform Mini Program Details** page, in the **Details** tab, you can view the cards for basic information, online version, review version, and trial version of the mini program.

Basic information

Basic information includes **Mini Program Name**, **Mini Program ID**, **Mini Program Purpose**, **Mini Program Introduction**, and **Detailed Description**.



Edit Mini Program basic information

The following procedure describes how to edit the basic information of a Mini Program.

1. Click **Basic Information** followed by **Edit**.

2. In the **Edit Mini Program Details** side panel, set the **Mini Program Introduction**, **Mini Program Detailed Description**, and **Mini Program Logo**.

3. Click **OK** to complete the Mini Program information editing.

> **Note**
> After updating the mini program information, you must submit the mini program for review for the information to be updated.

5.2.5.1.4. Mini program version management

1. In the navigation pane on the left of **Mini Program Open Platform** > **Developer Console**, click **Mini Program Management** > **My Mini Programs** to go to the **My Mini Programs** page.

2. On the **My Mini Programs** page, click **View** in the operation column of the mini program list to go to the **Intelligent Platform Mini Program Details** page.

   On the **Intelligent Platform Mini Program Details** page, the **Details** tab displays the basic information of the mini program and version cards for the production version, review version, and trial version.

In mini program version management, you can configure the production version, review version, and trial version, where:

- Production version: The mini program version that is currently in formal use in the production environment.
- Review version: The mini program version that is currently in the review process.
- Trial version: The version selected from the development version, which is a "test version" that can be opened only by designated personnel.

When configuring a mini program for the first time, you need to manually configure the mini program information. Upload the mini program code information through the IDE tool, set the uploaded mini program as a trial version or review version, and after the review is approved, you can publish the mini program as a production version.

Add a review version

You can add a mini program review version in the review version module on the **Intelligent Platform Mini Program Details** page. The procedure is as follows:

1. Click **Add Review** next to **Review Version**.

2. In the **Add Review - Select Version** dialog box, select an uploaded mini program version.

> **Important**
> - When the mini program resource status is Under Review or Review Approved, it cannot be set as a review version.
> - If you want the mini program to be automatically published after the review, select the **Automatically Publish After Review Approval** checkbox.

3. Click **OK**. The selected mini program is successfully set as a review version and submitted to the Mini Program Open Platform > Mini Program Operations Management Platform for review.

Try a mini program version

In the navigation pane on the left of **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Programs** to go to the **My Mini Programs** page. On the **My Mini Programs** page, click **View** in the operation column of the mini program list to go to the **Intelligent Platform Mini Program Details** page. On this page, you can configure or cancel a trial version in the trial version module.

Configure a trial version

On the **Intelligent Platform Mini Program Details** page, click **Configure Trial Version** next to **Trial Version**.

After configuration, click the QR code icon of the trial version and scan it with an App that has the mPaaS SDK installed to preview and test.

Cancel a trial version

On the **Intelligent Platform Mini Program Details** page, click **Cancel Trial Version** next to **Trial Version**, and confirm again to cancel the trial version.

> **Note**
> The trial version can be successfully canceled only when the mini program resource plan is synchronized to MDS.

Phased release of a mini program

In the navigation pane on the left of **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Programs** to go to the **My Mini Programs** page. On the **My Mini Programs** page, click **View** in the operation column of the mini program list to go to the **Intelligent Platform Mini Program Details** page. On this page, you can perform a phased release of the mini program version in the review version module. The procedure is as follows:

1. On the **Intelligent Platform Mini Program Details** page, click **Create Release** next to **Review Version**.
2. In the **Create Release** dialog box, select the release type, add a release description, and add advanced rules.
3. Click **OK** to perform a phased release of the mini program.

> **Note**
> You can create a phased release only when the current resource release status is Not Published.

Configure a production version

Publish a mini program version

A mini program version that has passed review can be published as a production version.

1. In the navigation pane on the left of **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Programs** to go to the My Mini Programs page.
2. On the **My Mini Programs Page**, click **View** in the operation column of the mini program list to go to the **Intelligent Platform Mini Program Details** page.
3. On the **Intelligent Platform Mini Program Details** page, click **Publish Version** next to **Review Version**, and confirm again to publish the mini program.

> **Important**
> The version number of the newly published mini program must be greater than that of the already published mini program.

Unpublish a production version

If a mini program has been published as a production version, you can perform an offline operation on the **Intelligent Platform Mini Program Details** page. After a mini program is unpublished, users will not be able to open the mini program through associated applications or other entry points. The procedure to unpublish a mini program is as follows:

1. In the navigation pane on the left of **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Programs** to go to the **My Mini Programs** page.
2. On the **My Mini Programs Page**, click **View** in the operation column of the mini program list.
3. On the **Intelligent Platform Mini Program Details** page, click **Unpublish Production Version** next to **Production Version**, and confirm again to successfully unpublish the version.

5.2.5.1.5. Mini Program code package management

The Mini Program Open Platform lets you upload Mini Program code packages for cloud compilation. It also supports uploading compiled code packages after local compilation through the developer tool. The flow for downloading code packages is as follows:

1. In the navigation pane on the left of **Mini Program Open Platform** > **Developer Console**, click **Mini Program Management** > **My Mini Programs** to access the **My Mini Programs** page.
2. On the **My Mini Programs** page, click **View** in the operation column of the Mini Program list.

On the **Smart Platform Mini Program Details** page, the **Code Package Management** tab displays the list of code package versions that have been uploaded for the current Mini Program. You can download specific versions of code packages.

3. In the Mini Program code package management tab, click **Download Code Package** in the operation column of the code package version list to download the code package.

5.2.5.1.6. Mini Program Audit History

In the mini program **Audit History** tab, you can view the audit information of historical versions of the current mini program. You can click **Details** in the operation column of the mini program audit version to view the detailed information of the corresponding version. You can also perform quick publish and listing operations for the mini program.

> **Note**
> Once the mini program passes the review, you can carry out the **On-shelf Review** operation. Then the mini program can be associated with the relevant App.

5.2.5.1.7. Mini program phased release

The phased release feature lets you set different mini program versions according to specific release plans. This enables different users to access different versions of mini programs.

You need to configure advanced rules for mini program phased release in **Developer Configuration** > **Publish Rule Management**. You can add corresponding rule information in **Publish Rule Management**. In **Publish Rule Management**, rules are provided for cities, device models, mobile system versions, network types, and platforms (Apple, Android). For more information, see Publish Rule Management.

The phased release tab displays the phased release list.



Scheduled phased release task list

The scheduled phased release task list displays the **Version Number**, **Version Guide**, **Scheduled Task Setting Time**, **Publish End Time**, **Scheduled Task Creation Time**, and **Scheduled Task Status** of the publish task. You can also click **Publish Details** in the operation column to view the publish task details.



Phased release list

The phased release list displays the **Version Number**, **Version Guide**, **Publish Start Time**, **Publish End Time**, and **Grayscale Status** of the phased release version. You can also click **Publish Details** in the operation column to view the publish task details. For release versions with a grayscale status of **In Progress**, you can quickly perform **Pause** or **End** publish operations.



5.2.5.1.8. Custom API management

The custom API management page displays **APIs that can be used directly**, **APIs that can be used after approval**, and the **API review list**. On this page, you can view API details and submit APIs for review.

APIs that can be used directly



APIs that can be used after approval



API review list



View API details

In the **APIs that can be used directly**/**APIs that can be used after approval**/**API review list**, click **Details** in the operation column of the API list.



Submit APIs for review

1. In the **APIs that can be used after approval** tab, select the APIs that you want to submit for review.
2. Click **Batch review**. After confirmation, the API review request is submitted successfully.



> **Note**
> The submitted APIs need to be reviewed in Custom API request review in **Mini Program development platform** > **Operations management console**.

5.2.5.1.9. Edit a mini program

1. In the navigation pane on the left of **Mini Program Open Platform** > **Developer Console**, click **Mini Program Management** > **My Mini Programs** to access the **My Mini Programs** page.
2. On the **My Mini Programs** page, click **Edit** in the Operation column of the mini program list.



3. In the **Edit Mini Program Details** side panel, configure the **Mini Program Introduction**, **Mini Program Detailed Description**, and **Mini Program Logo**.



4. Click **OK** to complete the mini program information editing.

> **Note**
> After updating the mini program information, you need to submit the mini program for review to update the information.

5.2.5.1.10. Enable/disable mini programs

In the navigation pane on the left of **Mini Program Open Platform** > **Developer Console**, click **Mini Program Management** > **My Mini Programs** to access the **My Mini Programs** page. If a mini program is no longer in use, click **Disable** in the operation column of the mini program list on the **My Mini Programs** page. After confirming, the mini program is successfully disabled. After the mini program is disabled, you can enable it again to resume using it.

5.2.5.2. H5 application

5.2.5.2.1. Overview

You can develop and manage H5 applications in the **Developer Console** of the **Mini Program Open Platform**.

This includes creating H5 applications, viewing H5 application details, editing H5 applications, enabling/disabling H5 applications, and phased release of H5 applications.

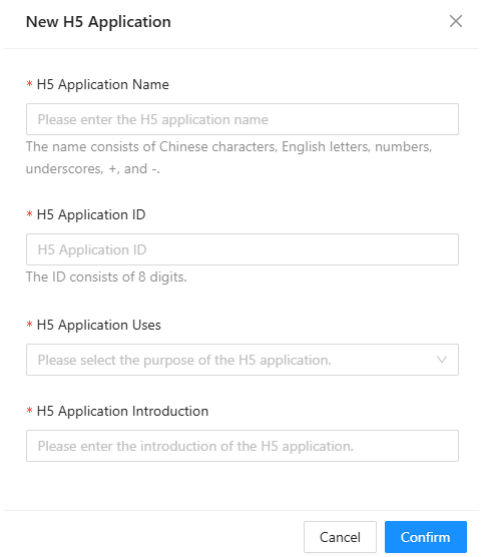
On the **Intelligent Platform H5 Details** page, you can perform the following operations: H5 application version management, H5 application package management, H5 application review history, and H5 application phased release.

Before using these features, you need to create an H5 application in the **Developer Console** of the **Mini Program Open Platform**.

5.2.5.2.2. Create an H5 application

Follow these steps to create an H5 application:

1. In the navigation pane on the left of the **Mini Program Open Platform Developer Console**, click **Mini Program Management** > **H5 Application** to access the **My H5 Application** page.
2. On the **My H5 Application** page, click **+ New H5 Application** in the upper right corner.
3. In the **New H5 Application** popup, enter the **H5 Application Name**, **H5 Application ID**, **H5 Application Uses**, and **H5 Application Introduction**.



4. Click **Confirm** to complete the H5 application creation.

5.2.5.2.3. View H5 Application Details

1. In the navigation pane on the left of the **Mini Program Open Platform Developer Console**, click **Mini Program Management** > **H5 Application** to access the **My H5 Application** page.
2. On the **My H5 Application** page, click **View** in the operation column of the H5 application list.
3. On the **Smart Platform H5 Application Details** page **Details** tab, you can view the basic information, online version, audit version, and experience version cards of the H5 application.

Basic information

The basic information includes **H5 Application Name**, **H5 Application ID**, **H5 Application Purpose**, **H5 Application Introduction**, and **Detailed Description**.

Edit H5 application basic information

To edit the basic information of an H5 application, follow these steps:

1. Click **Edit** next to **Basic Information**.

2. In the **Edit H5 Application Details** side panel, set the **H5 Application Introduction**, **H5 Application Detailed Description**, and **H5 Application Logo**.

3. Click **OK** to complete the editing of the H5 application information.

> **Note**
> After updating the H5 application's information, you need to submit it for review to complete the information update.

5.2.5.2.4. Manage H5 application version

1. In the **Mini Program Open Platform Developer Backend** left-side navigation pane, click **Mini Program Management** > **H5 Application** to enter the **My H5 Application** page.

2. On the **My H5 Application** page, click **View** in the operation column of the H5 application list to enter the **Smart Platform H5 Application Details** page.

On the **Smart Platform H5 Application Details** page, the **Details** tab displays cards for the basic information, online version, audit version, and experience version of the H5 application.

In H5 application version management, you can configure the online version, audit version, and experience version.

- Online version: The H5 application version currently used in the online environment.
- Audit version: The H5 application version currently in the review process.
- Experience version: A version selected from the development version, which is a "test version" that only specified personnel can access.

When configuring the H5 application for the first time, you need to manually configure the information of the H5 application. Upload the H5 application code information and set the uploaded H5 application as the experience version or audit version. After passing the audit, the H5 application can be put on the shelves as an online version.

> **Note**
> For more information about uploading H5 application code, see Upload code package.

Add new audit version

In the **Smart Platform H5 Application Details** page audit version module, you can add a new H5 application version for review. The procedure is as follows:

1. Click **Audit Version** followed by **Add A Review**.



2. In the **Add A Review - Select Version** pop-up window, select the uploaded H5 application version.



> **Important**
> - When the H5 application resource status is under review or approved, it cannot be set as the audit version.
> - If you need the H5 application to be automatically put on shelves after the review, select the **Automatically Put On Shelves After Approval** checkbox.

3. Click **Confirm**. The currently selected H5 application has been successfully set as the audit version and is submitted to the **Mini Program Development Platform** > **Operation Management Backend** for review. For detailed operations, see Mini Program On-shelf Review.

Experience H5 application version

In the **Smart Platform H5 Application Details** page **Experience Version** module, you can configure the experience version and cancel the experience version.

Configure experience version

1. On the **Smart Platform H5 Application Details** page, click **Experience Version** followed by **Configure Experience Version**.



2. Select a historical version and click **Confirm**.



After configuration, click the QR code icon of the experience version and use an app with the mPaaS SDK installed to scan the code for preview and testing.

Cancel experience version

On the **Smart Platform H5 Application Details** page, click **Experience Version** followed by **Cancel Experience Version**. After a second confirmation, the experience version is canceled.

> **Note**
> The experience version can only be successfully canceled when the H5 application resource package is synchronized to MDS.

Phased release of H5 application

On the **Smart Platform H5 Application Details** page, for **Approved** H5 applications, you can perform a phased release of the H5 application version in the audit version module on this page. The procedure is as follows:

1. On the **Smart Platform H5 Application Details** page, click **Audit Version** followed by **Create A New Release**.

2. In the **Create A New Release** pop-up window, select the release type, add the release description and advanced rules.

3. Click **Confirm** to perform a phased release of the H5 application.

> **Note**
> A phased release can only be created when the current resource's release status is not on shelf.

Configure online version

Put H5 application version on shelf

The H5 application version that has passed the review can be put on the shelves as an online version.

On the **Smart Platform H5 Application Details** page, click **Audit Version** followed by **Version On Shelves**. After a second confirmation, the H5 application is put on shelf.



> **Important**
> The H5 application version to be put on shelves must be greater than the H5 application version that has already been on shelves.

Remove the online version

If an H5 application has already been put on shelves as an online version, you can perform the remove operation on the **Smart Platform H5 Application Details** page. After the H5 application is removed, users will not be able to access the H5 application through the associated application or other entry points.

On the **Smart Platform H5 Application Details** page, click **Online Version** followed by **Online Version Off-shelf**. After a second confirmation, the version is successfully removed.

5.2.5.2.5. H5 Application Code Package Management

The **Developer Console** of the **Mini Program Open Platform** supports uploading H5 application code packages, enabling users to perform operations such as audit, release, and version listing. You need to upload the code package on the code management tab before initiating a version audit.

1. In the **Mini Program Open Platform**'s **Developer Console** left-side navigation pane, click **Mini Program Management** > **H5 Application** to access the **My H5 Application** page.
2. On the **My H5 Application** page, click **View** in the operation column of the H5 application list.

On the **Smart Platform H5 Application Details** page's **Code Package Management** tab, you can view a list of uploaded code package versions for the current H5 application, upload and download operations for specific versions.

Upload code package

1. On the Mini Program **Code Package Management** tab, click **+ Upload code package**.
2. In the **Upload code package** side panel, set the basic and configuration information for the H5 application code package.

Basic Information

Category	Description	Required
Version number	Enter the version number of the code package. The format refers to `1.0.0.1`.	Yes
File	Upload the code package in zip compressed file format.	Yes
Client effective range	◦ Enter the version effective range for iOS, Android, and Harmony. ◦ The H5 offline package will be effective within the client range limited by the entered minimum and maximum versions. ◦ If the maximum version is not filled in, it means supporting all subsequent new versions. It is recommended not to fill it in to avoid the offline package becoming invalid when the client is upgraded to a version higher than the filled maximum version.	The minimum version number is required

Configuration Information

Category	Description	Required
Main entry URL	The home page of the offline package.	No
Extended information	Page loading parameters in KV format, separated by commas for multiple KVs.	No
Download timing	Optional WIFI download or all network download.	Yes
Installation timing	Optional preload or not preload.	Yes

3. Check the box for **Confirmed the above information is accurate** and click **Confirm**.

Download code package

On the Mini Program **Code Package Management** tab, click **Download Code Package** in the operation column of the code package version list to download the package.

5.2.5.2.6. H5 Application Audit History

In the **Audit History** tab of the H5 application, you can view the review information for historical versions. By clicking **Details** in the operation column, you can view information for specific versions. Additionally, you can quickly publish and list the H5 application.

> **Note**
> Once the H5 application has passed the review, you can carry out the **Version on shelves** operation. The application must be put on shelves before it is linked to the relevant App.

5.2.5.2.7. Grayscale Release of H5 Application

In the **Grayscale release** tab, you can set different grayscale release versions of H5 applications according to different grayscale release plans. Different users can open different H5 applications according to different versions.

The advanced rules for grayscale release of H5 applications need to be configured in **Development Configuration** > **Release Rule Management**. You can add rule information in **Release Rule Management**. The **Release Rule Management** provides rules such as city, model, mobile phone system version, network type and platform (iOS, Android). For more details on release rules, please refer to Release Rule Management.

5.2.5.2.8. Edit H5 application

1. In the navigation pane on the left of the **Mini Program Open Platform Developer Console**, click **Mini Program Management** > **H5 Application** to access the **My H5 Application** page.
2. On the **My H5 Application** page, click **Edit** in the operation column of the mini program list.



3. In the **Edit H5 Application Details** side panel, set the **H5 Application Name**, **H5 Application ID**, **H5 Application Uses**, **H5 Application Introduction**, **Detailed Description Of H5 Application**, and **H5 Application Logo**.



4. Click **OK** to save the H5 application information.

> **Note**
> After updating the H5 application information, you need to submit the application for review to complete the information update process.

5.2.5.2.9. Enable/Deactivate H5 Application

In the **Mini Program Open Platform Developer Console**, click **Mini Program Management** > **H5 Applications** in the navigation pane on the left to access the **My H5 Application** page.

If the H5 application is no longer in use, click **Deactivate** in the operation column of the H5 application list on the **My H5 Application** page. After confirming the action, the H5 application will be successfully deactivated. A deactivated H5 application can only be used again after you enable it.



5.2.5.3. My mini game

5.2.5.3.1. Overview

In the Mini Program Open Platform developer platform, you can add game categories to provide process assistance and mechanism construction for host App traffic operation and mini game management.

In the navigation pane on the left, click **Mini Program Management** > **My Mini Games** to enter the **My Mini Games** interface.



On the **My Mini Games** page, you can add mini games, view intelligent platform mini game details, edit mini games, enable/disable mini games and implement phased release for mini games.

On the **Intelligent Platform Mini Game Details** page, you can perform the following operations: mini game version management, mini game package management, mini game review history, mini game qualification verification, mini game phased release, custom API management.

5.2.5.3.2. Add a mini game

1. On the **My Mini Games** page, click **+ Add Mini Game**.



2. In the **Add Mini Game** dialog box, enter the **Mini Game Name**, **Mini Game ID**, **Mini Game Description**, **Game Provider**, and **Customer Service Phone Number**, and select the **Mini Game Purpose**.



3. Click **OK** to create the mini game.

5.2.5.3.3. View details of mini games on the intelligent platform

1. On the **My Mini Games** page, click **View** in the Operation column of the mini game list.



2. The **Intelligent Platform Mini Game Details** page displays the mini game details, mini game version management, mini game package management, mini game review history, mini game qualification verification, mini game phased release, and custom API management.

On the **Intelligent Platform Mini Game Details** page, in the **Details** tab, you can view the basic information, qualification information, online version, review version, and trial version of the mini game.

Basic information

Basic information includes the mini game name, mini game ID, mini game purpose, mini game introduction, game publisher, customer service phone number, and detailed description.

Edit basic information of a mini game

Follow these steps to edit the basic information of a mini game:

1. Click **Edit** next to **Basic Information**.
2. In the **Edit Mini Game Details** side panel, set the **Mini Game Introduction**, **Mini Game Detailed Description**, **Game Publisher**, **Customer Service Phone Number**, and **Mini Game Logo**.
3. Click **OK** to save the mini game information.

> **Note**
> After updating the mini game information, you need to submit the mini game for review to apply the changes.

Qualification information

The Qualification Information tab displays the mini game type, game version number, business license, "Network Game Publication Number and Issuance" file, game software copyright registration certificate, and ICP filing.

Edit qualification information

Follow these steps to edit the qualification information of a mini game:

1. Click **Edit** next to **Qualification Information**.
2. In the **Edit Qualification Information** side panel, set the **Mini Game Type**, **Business License**, **Game Version Number**, **"Network Game Publication Number and Issuance" File**, **Game Software Copyright Registration Certificate**, **ICP Filing**, and **Access Token**.
3. Click **OK** to save the mini game qualification information.

> **Note**
> After updating the mini game information, submit the mini game for review to apply the changes.

Submit for review

Click **Submit for Review** below the qualification information, and confirm to complete the submission.

> **Note**
> The submitted mini game qualification verification needs to be reviewed in Mini Game Qualification Verification in the Operations Management Platform.

5.2.5.3.4. Mini game version management

1. In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to go to the **My Mini Games** page.
2. On the **My Mini Games** page, click **View** in the operation column of the mini game list to go to the **Intelligent Platform Mini Game Details** page.

   On the **Intelligent Platform Mini Game Details** page, the **Details** tab displays the basic information, qualification information, production version, review version, and trial version of the mini game.

In mini game version management, you can configure the production version, review version, and trial version, where:

- Production version: The mini game version that is currently in formal use in the production environment.
- Review version: The mini game version that is currently in the review process.
- Trial version: A version selected from the development version, which is a "test version" that can be opened only by designated personnel.

> **Note**
> When you add a mini game, you need to manually configure the mini game information. After you set an uploaded mini game as the trial version or review version and the review is approved, you can publish the mini game as the production version.

Add a review version

You can add a review version of the mini game in the review version module on the **Intelligent Platform Mini Game Details** page. The procedure is as follows:

1. Click **Add Review** next to **Review Version**.

2. In the **Add Review - Select Version** dialog box, select an uploaded mini game version.

> **Important**
> - When the mini game resource status is Under Review or Review Approved, it cannot be set as the review version.
> - If you want the mini game to be automatically published after the review, select the **Automatically Publish After Review Approval** check box.

3. Click **OK**. The selected mini game is successfully set as the review version and submitted to the Mini Program Development Platform > Mini Program Operation Management Platform for review.

Trial mini game version

In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to go to the **My Mini Games** page. On the **My Mini Games** page, click **View** in the operation column of the mini game list to go to the **Intelligent Platform Mini Game Details** page. On this page, you can configure or cancel the trial version in the trial version module.

Configure a trial version

1. On the **Intelligent Platform Mini Game Details** page, click **Configure Trial Version** next to **Trial Version**.

2. In the Configure Trial Version dialog box, select **Historical Version**.

3. Click **OK**.

After the configuration is complete, click the QR code icon of the trial version and scan the code with an app that has the mPaaS SDK installed to preview and test the mini game.

Cancel a trial version

On the **Intelligent Platform Mini Game Details** page, click **Cancel Trial Version** next to **Trial Version**, and confirm again to cancel the trial version.

> **Note**
> The trial version can be successfully canceled only when the mini game resource plan is synchronized to MDS.

Phased release of a mini game

In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to go to the **My Mini Games** page. On the **My Mini Games** page, click **View** in the operation column of the mini game list to go to the **Intelligent Platform Mini Game Details** page. On this page, you can perform a phased release of the mini game version in the review version module. The procedure is as follows:

1. On the **Intelligent Platform Mini Game Details** page, click **Create Release** next to **Review Version**.



2. In the **Create Release** dialog box, select the release type, set the scheduled release switch, add a release description, and add advanced rules.



3. Click **OK** to perform a phased release of the mini game.

> **Note**
> You can create a phased release only when the current resource release status is Not Published.

Configure the production version

Publish a mini game version

A mini game version that has passed the review can be published as the production version.

1. In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to go to the My Mini Games page.
2. On the **My Mini Games Page**, click **View** in the operation column of the mini game list to go to the **Intelligent Platform Mini Game Details** page.
3. On the **Intelligent Platform Mini Game Details** page, click **Publish Version** next to **Review Version**.



4. Set whether to schedule the release, and click **Confirm**.



After you set a scheduled task, the release operation will be executed at the release time.



The published mini program is displayed in the production version.



> **Important**
> The version of the newly published mini game must be greater than that of the mini game that has already been published.

One-click rollback

If a mini game has been published as the production version, you can perform a rollback operation on the **Intelligent Platform Mini Game Details** page. You can use the rollback feature to restore to the **No Production Version Published** state. The procedure is as follows:

1. In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to access the **My Mini Games** page.
2. On the **My Mini Games Page**, click **View** in the operation column of the mini game list.

3. On the **Intelligent Platform Mini Game Details** page, click **One-Click Rollback** next to **Production Version**, and confirm again to restore to the **No Production Version Published** state.

Unpublish the production version

If a mini game has been published as the production version, you can perform an offline operation on the **Intelligent Platform Mini Game Details** page. After a mini game is unpublished, users will not be able to open the mini game through associated applications or other entry points. The procedure to unpublish a mini game is as follows:

1. In the navigation pane on the left of **Open Platform**, click **Mini Program Management** > **My Mini Games** to go to the **My Mini Games** page.
2. On the **My Mini Games Page**, click **View** in the operation column of the mini game list.
3. On the **Intelligent Platform Mini Game Details** page, click **Unpublish Production Version** next to **Production Version**, and confirm again to successfully unpublish the version.

5.2.5.3.5. Mini game package management

The Mini Program Open Platform lets you upload mini game packages to the cloud. The operation flow for downloading packages is as follows:

1. In the **Mini Program Open Platform** left-side navigation pane, click **Mini Program Management** > **My Mini Games** to access the **My Mini Games** page.
2. On the **My Mini Games** page, click **View** in the operation column of the mini game list.

   On the **Intelligent Platform Mini Game Details** page, in the **Package Management** tab, you can view the version list of packages that have been uploaded for the current mini game. You can download packages of specific versions.
3. In the mini game package management tab, click **Download Package** in the operation column of the package version list to download the package.

5.2.5.3.6. Mini game review history

The mini game review history tab displays the historical version review information of the current mini game. You can click **Details** in the operation column of the mini game review version to view the detailed information of the corresponding version. You can also perform quick publish and listing operations for the mini game.

> **Note**
>
> After the mini game passes the review, you can execute the **Version Listing** operation. The mini game can be associated with relevant Apps only after it is listed.

5.2.5.3.7. Mini game qualification verification

The mini game qualification verification tab displays the current mini game's verification information. You can click **Details** in the operation column to view detailed qualification information of the mini game.

5.2.5.3.8. Mini game phased release

You can set different mini game phased release versions according to various phased release plans. Different users can access different mini games based on their assigned versions. Advanced rules for mini game phased release must be configured in **Developer Configuration** > **Publish Rule Management**. You can add corresponding rule information in **Publish Rule Management**. In **Publish Rule Management**, rules are provided for cities, device models, mobile system versions, network types, and platforms (Apple, Android). For more information, see Publish Rule Management.

The phased release feature tab displays the phased release list.

Scheduled phased release task list

The scheduled phased release task list displays the **version number**, **Version Guide**, **scheduled task setting time**, **publish end time**, **scheduled task creation time**, and **scheduled task status**. You can also click **Publish Details** in the operation column to view the publish task details.

Phased release list

The phased release list displays the **version number**, **Version Guide**, **publish start time**, **publish end time**, and **grayscale status** of the phased release version. You can also click **Publish Details** in the operation column to view the publish task details. For published versions with the grayscale status of **In Progress**, you can quickly perform **pause** or **end** publish operations.

5.2.5.3.9. Custom API management

The custom API management page displays **APIs that can be used directly**, **APIs that can be used after approval**, and the **API review list**. On this page, you can view API details and submit APIs for review.

APIs that can be used directly

APIs that can be used after approval

API review list

View API details

In the **APIs that can be used directly**/**APIs that can be used after approval**/**API review list**, click **Details** in the operation column of the API list.

Submit APIs for review

1. In the **APIs that can be used after approval** tab, select the APIs that you want to submit for review.
2. Click **Batch Review**. After confirmation, the API review request is submitted successfully.

> **Note**
> The submitted APIs need to be reviewed in **Mini Program Development Platform** > **Operations Management Console** > Custom API request review.

5.2.5.3.10. Edit mini game

1. In the navigation pane on the left of **Mini Program Open Platform**, click **Mini Program Management** > **My Mini Games** to access the **My Mini Games** page.
2. On the **My Mini Games** page, click **Edit** in the operation column of the mini game list.
3. In the **Edit Mini Game Details** slide-out panel, set the **Mini Game Introduction**, **Mini Game Detailed Description**, **Game Provider**, **Customer Service Phone Number**, and **Mini Game Logo**.
4. Click **OK** to save your changes to the mini game information.

> **Note**
> After updating the mini game information, you need to submit the mini game for review to apply the changes.

5.2.5.3.11. Enable/disable mini games

In the **Open Platform** left-side navigation pane, click **Mini Program Management** > **My Mini Games** to access the **My Mini Games** page.

Disable mini games

If a mini game is no longer in use, click **Disable** in the operation column of the mini game list on the **My Mini Games** page. After confirming twice, the mini game will be successfully disabled. Once disabled, the mini game can only be used again after you execute the enable operation.

Enable mini programs

For mini games that have been disabled, click **Enable** in the operation column of the mini game list on the **My Mini Games** page. After confirming twice, you can successfully enable the mini game.

5.2.6. Development configuration

5.2.6.1. Release rule management

Phased release supports rule configuration. You can define various configuration data required for phased release through release rules. You can add, view, and modify release rules for mini program development through development configuration.

Add a release rule

1. In the **Mini Program Open Platform Developer Console**, click **Development Configuration** > **Release Rule Management** > **+ Add Rule** in the left menu bar.
2. In the **Add Rule** dialog box, set the **Rule Name**, **Rule Type**, and **Platform**.
   - Rule Name: The name of different phased release rules.
   - Rule Type: To adapt to different customers' phased releases in their own environments, we provide four rule types: city, device model, mobile system version, and network type.
   - Platform: Supports four options: Apple, Android, All Platforms, and Platform Independent.
   - Rule Resource Value:
3. Click **OK**. The release rule is created successfully.

View release rule details

1. In the **Mini Program Open Platform Developer Console**, click **Development Configuration** > **Release Rule Management** in the left menu bar.
2. On the **Release Rule Management** page, click **Details** in the operation column of the release rule list. The **Rule Details** dialog box displays detailed information about the release rule.

Modify a release rule

1. In the **Mini Program Open Platform Developer Console**, click **Development Configuration** > **Release Rule Management** in the left menu bar.
2. On the **Release Rule Management** page, click **Modify** in the operation column of the release rule list.

3. In the **Modify Rule** dialog box, modify the **Rule Name** and **Platform**.
4. Click **OK**. The operation is successful.

5.2.6.2. Whitelist management

You can add and delete whitelists for development miniapps through development configuration.

Add a whitelist

1. In the **Mini Program Open Platform Developer Backend**, click **Development Configuration** > **Whitelist Management** > **+ Add WhiteList** in the left menu bar.
2. In the **Add Whitelist** dialog box, set the **UserId List** and **Whitelist File**.
3. Click **Confirm**. The whitelist is added successfully.

Delete a whitelist

In the **Mini Program Open Platform Developer Backend**, click **Development Configuration** > **Whitelist Management** in the left menu bar.

Delete a single whitelist

On the **Whitelist Management** page, click **Delete** in the operation column of the whitelist list. After confirmation, the whitelist is deleted successfully.

Delete whitelists in batch

1. On the **Whitelist Management** page, select the whitelists that you want to delete.
2. Click **Batch Delete**. The whitelists are deleted successfully.

5.2.6.3. Domain name management

To ensure the security of data accessed by the mini program, network connections requested in the mini program must be registered through ICP filing first. After ICP filing is completed, the mini program can normally access the corresponding server. When a mini program needs to request data from a server, you must configure the corresponding domain names through development configuration. Otherwise, the mini program will not be able to access the corresponding domain names.

Add a domain name

1. In the **Mini Program Open Platform Developer Backend**, click **Development Configuration** > **Domain Name Management** > **Add Domain Name** in the left menu bar.
2. Add a domain name in the **Add Domain Name** dialog box.

> **Important**
> When configuring a domain name, enter the domain name address with the protocol header. After the domain name takes effect, all directories and pages under this domain name will also take effect simultaneously.

3. Click **Confirm** to successfully add the domain name.

Modify a domain name

1. On the **Domain Name Management** page, click **Modify** in the operation column of the domain name.
2. Update the domain name information in the **Modify Domain Name** dialog box.
3. Click **Confirm** to successfully modify the domain name.

Delete a domain name

On the **Domain Name Management** page, click **Delete** in the operation column of the domain name. After secondary confirmation, the domain name is successfully deleted.

5.2.6.4. IDE Configuration Management

In the IDE Configuration Management section of development configuration, you can configure the IDE and download application configuration files.

Configure IDE

1. In the **Mini Program Open Platform Developer Backend**, click **Development Configuration** > **IDE Configuration Management** in the left menu bar.
2. On the **IDE Configuration Management** page, click **Modify** in the operation column of the configuration item.
3. Modify the configuration content.
4. Click **Save** in the operation column. The configuration is modified successfully.

Configuration items are as follows:

Configuration key name	Configuration key description
login_url	Logon address
uuid_url	Generate UUID address
debug_url	Connect WebSocket address
upload_url	Upload mini program resource plan address
applist_url	Obtain mini program list address

Download configuration file

1. On the **IDE Configuration Management** page, click **Download Configuration File**.

2. In the **Download Configuration File** dialog box, select the App for which you want to download the configuration file.

3. Click **Download**. The configuration file is downloaded successfully.

5.2.7. Operation logs

Operation logs allow you to view important or sensitive operations performed by platform users after logon. The operation flow is as follows:

1. In the **Mini Program Open Platform Developer Console**, click **Operation Logs** in the navigation pane on the left to access the **Operation Logs** page.
2. The Operation logs page displays the operation log list. This list contains operation date, operator IP, execution status, operation API, and operation content.

   - Operation date: The date when the behavior occurred.
   - Operator IP: The IP address of the user who performed the behavior.
   - Execution status: Whether the behavior was executed successfully or failed.
   - Operation API: The API that was executed for the behavior.
   - Operation content: The operation information of the sensitive behavior.

On the **Operation Logs** page, you can filter by operator and operation date, then click **Query** to retrieve the operation log information that meets your criteria.

5.2.8. Data Statistics

5.2.8.1. Mini Program management

The Mini Program management page displays the overall data analytics of all mini programs created through the mini program analysis module under the current development group. This includes summary data of four metrics: new users, active users, launch count, and sharing users, along with data overview of individual mini programs.



You can create mini programs for statistics and analysis, and manage the created mini programs.

The core metrics are described as follows.

Metric	Description
New users	The cumulative number of new users across all mini programs under the current app, with duplicates removed between different mini programs. For a single mini program, this refers to the number of users who access the mini program page for the first time. The same user accessing multiple times is counted only once.
Active users	The cumulative number of active users across all mini programs under the current app, with duplicates removed between different mini programs. For a single mini program, this refers to the total number of users who access the mini program. The same user accessing multiple times is counted only once.
Launch count	The cumulative number of launches across all mini programs under the current app, without removing duplicates between different mini programs. For a single mini program, this refers to the total number of times the mini program is opened. One launch is counted from when a user opens the mini program until it exits to the background. When the interval between two consecutive opens is less than 30 seconds, it is counted as one open.
Sharing users	The total number of users who share the mini program page. The same user sharing multiple times is counted only once.

View mini program data details

The mini program list displays all mini programs created through the mini program analysis module and their core metric data. You can sort by different metrics to view the data overview of individual mini programs.

1. In the **Mini Program Open Platform Developer Backend**, go to **Data Statistics** > **Mini Program Management** from the navigation pane on the left.
2. In the mini program list, select the target mini program, and click the mini program name or **View** in the operation column.

You will be directed to the Data Overview page to learn about the detailed statistics of the mini program.

Mini program code integration

Complete the mini program code integration to track the mini program metrics.

1. In the **Mini Program Open Platform Developer Backend**, go to **Data Statistics** > **Mini Program Management** from the navigation pane on the left.
2. In the mini program list, select the target mini program, and click **Code Integration** in the operation column.
3. In the **Code Integration** panel that appears, copy the integration code to the mini program for integration.

5.2.8.2. Data overview

The data analytics feature supports statistical analysis of miniapp usage data for the current App. Through the data overview page, you can understand the historical statistical data (T+1) and real-time data overview of a single miniapp.

To view the data overview, follow these steps:

1. In the **Mini Program Open Platform Developer Console**, access **Data Statistics** > **Data Overview** page from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp. By default, the system displays the analytical data of the most recently created miniapp.
3. Click the **Overview Statistics** or **Real-time Dashboard** tab to view the historical or real-time statistical data of the miniapp. The analytical data appears in the form of metric cards and line trend charts.

Overview statistics

The overview statistics page displays the core data metrics and trend changes of "Customer Acquisition > Activation > Sharing" for the currently selected miniapp. You can view the data performance and value conversion of the current miniapp, and identify data issues and locate abnormal data. The page supports selecting metric data for any past 1 day, 7 days, or 30 days.

Metric overview

Displays the historical statistical data of core metrics for the current miniapp in the three stages of customer acquisition, activation, and sharing during the specified time period in card format. This includes new users, active users, number of opens, average stay duration per session, number of sharing users, number of new users from sharing, along with the growth rate compared to the previous day, same period last week, or same period last month.

Line trend chart

Displays the change trend of various metric data during the specified time period in the form of a line chart. When you click a metric card, the line chart will display the data trend of the corresponding metric on the card.

Real-time dashboard

The real-time dashboard displays the real-time statistical data of five core metrics for the currently selected miniapp, and the hourly change trend of each metric.

Metric overview

Displays the real-time data of various metrics for the current miniapp in card format, including active users, number of opens, page visit count, average stay duration per session, average stay duration per user, and the growth rate compared to the same period yesterday.

Line trend chart

Displays the change trend of various metric data for today, yesterday, and 7 days ago in the form of a line chart. When you click a metric card, the line chart will display the data change trend of the corresponding metric.

Metric description

The descriptions of various metrics in the data overview are as follows.

Metric	Description
New users	The number of users who visit the miniapp page for the first time. Multiple visits by the same user are not counted repeatedly.
Active users	The total number of users who visit the miniapp. Multiple visits by the same user are not counted repeatedly.
Number of opens	The total number of times the miniapp is opened. A user opening the miniapp until it exits to the background is counted as one time. When the interval between two consecutive opens of the miniapp is less than 30 seconds, it is counted as one open.
Average stay duration per session	During the selected time period, the total duration of staying on the miniapp page for each open on average, that is, average stay duration per session = total stay duration of all users/number of opens.
Average stay duration per user	The total duration of staying on the miniapp page for each user on average, that is, average stay duration per user = total stay duration/number of active users.
Number of sharing users	The total number of people who share the miniapp page. Multiple shares by the same user are not counted repeatedly.
Number of new users from sharing	The number of new users who enter the miniapp for the first time through a shared link.
Previous day (%)	The growth rate of the cumulative data of the selected day compared to the cumulative data of the previous day. Previous day comparison = (data of the specified date - data of the previous day)/data of the previous day. When the divisor is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

Same period yesterday (%)	The growth rate of the cumulative data from 0:00 to the current hour compared to the cumulative data of the same time yesterday.
Same period last week (%)	The growth rate of the cumulative data of the past 7 days compared to the cumulative data of the past 8-14 days. Same period last week = (data of the past 7 days - data of the same period last week)/data of the same period last week. When the divisor is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period last month (%)	The growth rate of the cumulative data of the past 30 days compared to the cumulative data of the past 31-60 days. Same period last month = (data of the past 30 days - data of the same period last month)/data of the same period last month. When the divisor is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

5.2.8.3. User Analysis

The User Analysis page displays user access data for the currently selected miniapp (with T+1 timeliness), including seven metrics such as new users, active users, number of opens, page visits, and more. This helps you analyze in detail how user data changes at various stages. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view user analysis data, follow these steps:

1. In the **Mini Program Open Platform Developer Console**, navigate to **Data Statistics** > **User Analysis** from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp.
3. Select a time range to view statistical data for the corresponding period.

User analysis overview

This section displays statistical data for seven metrics of the miniapp during the specified period, including new users, active users, number of opens, page visits, average duration per session, average duration per user, and average opens per user. It also shows the growth rate compared to the previous day, same period last week, or same period last month.

The metrics are explained as follows.

Metric	Description
New users	The number of users who access the miniapp pages for the first time. Multiple visits by the same user are not counted repeatedly.
Active users	The total number of users who access the miniapp. Multiple visits by the same user are not counted repeatedly.
Number of opens	The total number of times the miniapp is opened. A session from opening the miniapp to exiting it to the background is counted as one open. When the interval between two consecutive opens is less than 30 seconds, they are counted as one open.
Page visits	The total number of visits to all pages within the miniapp. Jumps between multiple pages and repeated visits to the same page are counted as multiple visits.
Average duration per session	During the selected period, the average total duration spent on miniapp pages per open, calculated as: average duration per session = total duration of all users / number of opens.
Average duration per user	The average total duration each user spends on miniapp pages, calculated as: average duration per user = total duration / number of active users.
Average opens per user	During the selected period, the average total number of times each user opens the miniapp, calculated as: average opens per user = total number of opens by all users / total number of active users.
Previous day (%)	The growth rate of the selected day's cumulative data compared to the previous day's cumulative data. Previous day comparison = (user access data on the specified date - user access data on the previous day) / user access data on the previous day. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period last week (%)	The growth rate of the past 7 days' cumulative data compared to the cumulative data from 8-14 days ago. Same period last week = (user access data for the past 7 days - user access data for the same period last week) / user access data for the same period last week. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period last month (%)	The growth rate of the past 30 days' cumulative data compared to the cumulative data from 31-60 days ago. Same period last month = (user access data for the past 30 days - user access data for the same period last month) / user access data for the same period last month. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

User access trends

This section displays the trend of changes in each metric's data during the specified period as a line chart. Click on a metric card to display the data trend for the corresponding metric in the line chart.

Detailed data

This section displays user analysis statistical data for each hour or day within the selected time period in a table format. You can sort the data by time for easier viewing.

The time granularity of the data display depends on the selected time range. When the query time range is 1 day, the data display granularity is hourly. When the time range is 7 or 30 days, the data display granularity is daily.

5.2.8.4. Page analysis

Page analysis displays page view data for the currently selected miniapp (with T+1 data latency), including page visit count, page visitor count, and average page visit duration. This helps you understand which pages users visit most frequently and how long they spend on each page. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view page analysis data:

1. In the **Mini Program Open Platform Developer Console**, go to **Data Statistics** > **Page Analysis** from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp.
3. Select a time range to view the statistical data for the corresponding period.

   - **Page visit count**: The total number of visits to all pages within the miniapp. Navigation between multiple pages and repeated visits to the same page are counted as multiple visits.
   - **Page visitor count**: The total number of users who visited pages within the miniapp. Multiple visits by the same user are not counted repeatedly.
   - **Average page visit duration**: The average time a user spends on a miniapp page per visit. This is calculated as: average page visit duration = total time spent on pages during the specified period / total page visit count.

5.2.8.5. Share analysis

The sharing feature is an important channel for miniapp user acquisition and customer acquisition, and a key indicator for measuring the health of a miniapp. The share analysis page displays the sharing data of the currently selected miniapp (with T+1 timeliness), including five metrics such as number of users sharing, number of shares, and share return traffic, helping you understand the viral effect of your miniapp. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view share analysis data, follow these steps:

1. In the **Mini Program Open Platform**'s **Developer Backend**, access **Data Statistics** > **Share Analysis** page from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp.
3. Select a time range to view the statistical data for the corresponding period.

Share analysis overview

Displays statistical data for five metrics of the miniapp during the specified period: number of users sharing, number of shares, share return traffic, share return ratio, and new shares, along with the growth rate compared to the previous day, same period last week, or same period last month.

The metrics are explained as follows.

Metric	Description
Number of users sharing	The total number of users who shared the miniapp pages. Multiple shares by the same user are not counted repeatedly.
Number of shares	The total number of times the miniapp pages were shared.
Share return traffic	The number of times users access the miniapp by clicking on shared links is counted as return traffic for this share.
Share return ratio	This metric reflects the effectiveness of sharing. Share return ratio = Share return traffic/Number of shares × 100%.
New shares	The number of new users who entered the miniapp for the first time through shared links.
Previous day (%)	The growth rate of the selected day's cumulative data compared to the previous day's cumulative data. Previous day comparison = (Share data of the specified date - Share data of the previous day)/Share data of the previous day. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period last week (%)	The growth rate of the past 7 days' cumulative data compared to the cumulative data from 8-14 days ago. Same period last week = (Share data of the past 7 days - Share data of the same period last week)/Share data of the same period last week. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period last month (%)	The growth rate of the past 30 days' cumulative data compared to the cumulative data from 31-60 days ago. Same period last month = (Share data of the past 30 days - Share data of the same period last month)/Share data of the same period last month. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

Share trends

Displays the trend of each metric's data during the specified period in the form of a line chart. Click on a metric card to display the data trend for the corresponding metric in the line chart.

Detailed data

Displays the hourly or daily share analysis statistics for the selected time period in a table format, supporting data viewing sorted by time.

The time granularity of the data display depends on the selected time range. When the query time range is 1 day, the data display granularity is hourly. When the time range is 7 or 30 days, the data display granularity is daily.

5.3. Operations management console user guide

5.3.1. Log on

Enter the user **account** and **password** in the **Account login** page of the mini program open platform operation management platform, and click **Login** to log in to the Operation management platform of Mini program open platform .

5.3.2. Basic management

5.3.2.1. Member Management

The operations management backend lets you perform management operations on member information added in the miniapp open platform developer platform. You can invite developers, edit member permissions, freeze member permissions, recover member permissions, and reset passwords.

Invite developers

1. In the miniapp open platform internal management platform, click **Basic Management** > **Member Management**.
2. On the **Member Management** page, click **+ Add Member**.

3. In the **Add Member** dialog box, select **Link Validity Period** and **Permission Settings** in the **Invite Developer** section.

> **Note**
> The permission setting options are added in Role Management.

4. Click **Confirm** to generate an invitation link.

5. Click **Copy Link** to copy the link.

Edit member permissions

1. In the miniapp open platform internal management platform, click **Basic Management** > **Member Management**.
2. On the **Member Management** page, click **Edit Permission** in the operation column of the member list.

3. In the **Edit Permission** dialog box, select **Permission Settings**.



4. Click **Confirm** to successfully edit the permissions.

Freeze members

1. In the miniapp open platform internal management platform, click **Basic Management** > **Member Management**.
2. On the **Member Management** page, click **Freeze** in the operation column of the member list. After secondary confirmation, the operation is complete.

Recover member permissions

1. In the miniapp open platform internal management platform, click **Basic Management** > **Member Management**.
2. On the **Member Management** page, click **Recover** in the operation column of the member list. After secondary confirmation, the operation is complete.

Reset member passwords

1. In the miniapp open platform internal management platform, click **Basic Management** > **Member Management**.
2. On the **Member Management** page, click **Reset Password** in the operation column of the member list. After secondary confirmation, the operation is complete.

> **Important**
> After the password is reset, it will change to the default password in Custom Settings > Security Settings. Please reset and log on to change the password as soon as possible.

5.3.2.2. Role Management

The Mini Program Open Platform operation management backend supports custom roles and lets you differentiate permissions based on roles, enabling effective management of member account permissions.

Add a role

1. On the **Basic Management** > **Role Management** page, click **+ Add Role**.



2. In the **Add Role** side panel, set the **Role Name**, **Role Code**, and configure the corresponding **Role Permissions**.



3. Click **OK** to create the role.

Edit a role

1. On the **Basic Management** > **Role Management** page, click **Edit** in the operation column of the role list.
2. In the **Edit Role** side panel, select the role permissions.



3. Click **OK** to modify the role.

Delete a role

On the **Basic Management** > **Role Management** page, click **Delete** in the operation column of the role list. After confirmation, the role will be deleted.

Manage role accounts

On the **Basic Management** > **Role Management** page, click **Manage Account** in the operation column of the role list to add or delete members of that role.

Add a role member

1. In the **Manage Account** dialog box, click **+ Add Member**.

2. In the **Add Account** dialog box, select the member username.

3. Click **OK** to add members to the role.

Delete a role member

1. On the **Manage Account** page, select the username of the member you want to delete.
2. Click **Batch Delete**. After confirmation, the members will be deleted.

5.3.2.3. Review configuration

5.3.2.3.1. On-shelf review

The audit configuration function on the operation side lets you set the Mini Program on-shelf review node as a jointly sign node. This function supports the configuration of multiple roles. After configuration, at least one member from each role must review the Mini Program before approval can be granted (role joint signature).

The operation flow for **adding a role** is as follows:

1. On the **Basic Management** > **Audit Configuration** page, click **Mini Program Review** to switch to the mini program on-shelf review management page.
2. Click **+ Add Role** under **Jointly sign role**.

3. Select the corresponding roles in the **Add A Jointly Sign Role** dialog box.

4. Click **OK** to complete the node configuration for the mini program on-shelf review.

5.3.3. Developer group management

On the **Developer Group Management** page, you can add, delete, and modify mini program developer groups, and add members to the created developer groups.



Create a mini program developer group

1. In the navigation pane on the left of the Mini Program Open Platform, click **Developer Group Management**.



2. On the **Developer Group Management** page, click **Create Mini Program Developer Group**.



3. In the **Create Mini Program Developer Group** side panel, enter the mini program developer group name, select the mini program developer administrator, and select the mini program application.



4. Click **OK**. The mini program developer group is created successfully.

After the mini program developer group is created successfully, you can add developer members to the mini program.

Edit a mini program developer group

1. On the **Developer Group Management** page, click the (···) icon in the lower right corner of the mini program developer group.



2. Click **Edit** in the dropdown list.
3. In the **Edit Mini Program Developer Group** side panel, modify, add, or delete applications.



4. Click **OK**. The mini program developer group is edited successfully.

Delete a mini program developer group

1. On the **Developer Group Management** page, click the (···) icon in the lower right corner of the mini program developer group.
2. Click **Delete** in the dropdown list. The developer group is deleted successfully.

> **Important**
>
> When the developer group contains members, you cannot delete the developer group. To delete the developer group, you must first delete the developer group members through Developer Group Member Management.

Developer group member management

1. On the **Developer Group Management** page, click **Developer Member Management** in the lower left corner of the mini program developer group.



2. On the **Member Management** page, add or delete members.

5.3.4. Mini Program management

5.3.4.1. Mini program listing approval

The **Mini Program Listing Approval** page displays all approval information across five tabs: **All**, **Under review**, **Approved**, **Rejected**, and **Revoked**.

In the approval list on the **Mini Program Listing Approval** page, you can click **Details** in the operation column to open the **Version Details** page of the **Mini program/H5 application/Mini game**. This lets you view publishing information, basic information, review results, approval progress, and approval preview information.

Review mini program/H5 application/Mini game

For mini programs/H5 applications/mini games with the review status of **Under Review**, you can approve or reject them using the following procedure:

1. On the **Mini Program Listing Approval** page, click **Details** in the operation column of the mini program/H5 application/mini game with the status **Under review**.
2. On the **Version Details** page, you can view the test QR code of the mini program. You can scan this code with the App to load the mini program for trial use and verify whether it complies with regulations.
3. Click **Reject** or **Approve** in the lower right corner, fill in the **Review Comments**, and click **Confirm**.

> **Note**
> - The review of the submitted mini program is completed after all countersigning roles in Approval configuration have completed their reviews.
> - Only members with review permissions can perform review operations.

5.3.4.2. Domain name management

In the Mini Program Open Platform operations management console, click **Mini Program Management** > **Domain Name Management** in the navigation pane on the left. On the **Domain Name Management** page, operations personnel can **view and set blacklists and whitelists** and **view ICP filing information**.

- After setting whitelist domain names, enterprise mini programs can access domain names on the whitelist without requiring ICP filing for the domain names.
- After setting blacklist domain names, all enterprise mini programs will be unable to access these domain names.
- Operations personnel can use the **ICP Filing Management** feature to view all endpoints registered by enterprises across all environments.

Blacklist and whitelist management

Whitelist management

Add a whitelist entry

1. On the **Domain Name Management** page, click **Whitelist** on the **Blacklists And Whitelists** tab, and then click **+ Add**.
2. Add a domain name in the **Add Whitelist Entry** dialog box.
3. Click **OK**. The domain name is added successfully.

Add to blacklist

On the **Blacklists And Whitelists** tab of the **Domain Management** page, click **Add To Blacklist** in the Actions column of the whitelist. After you confirm the action, the domain name is successfully added to the blacklist.

Delete a whitelist entry

On the **Domain Name Management** page, click **Delete** in the Operation column of the **Blacklists And Whitelists** tab. After confirmation, the whitelist entry is successfully deleted.

Blacklist management

Add a blacklist entry

1. On the **Domain Name Management** page, click **Blacklist** on the **Blacklists And Whitelists** tab, and then click **+ Add**.
2. Add a domain name in the **Add Blacklist Entry** dialog box.
3. Click **OK**. The domain name is added successfully.

Delete a blacklist entry

On the **Domain Name Management** page, click **Delete** in the Operation column of the blacklist on the **Blacklists And Whitelists** tab. After confirmation, the blacklist entry is successfully deleted.

ICP filing management

Operations personnel can view ICP filing list information. On the **Domain Name Management** page, click **ICP Filing Management**. The **ICP Filing Management** tab displays the ICP filing list information.

5.3.4.3. Custom API request review

The **Custom API Request Review** page displays all custom API review information, including 5 tabs: **All**, **Under review**, **Approved**, **Rejected**, and **Revoked**.

In the review list on the **Custom API Request Review** page, you can click **Details** in the operation column to view custom API details in the side panel. For API requests that have not been reviewed, you can execute **Approve** or **Reject** operations.

5.3.4.4. Mini game qualification verification

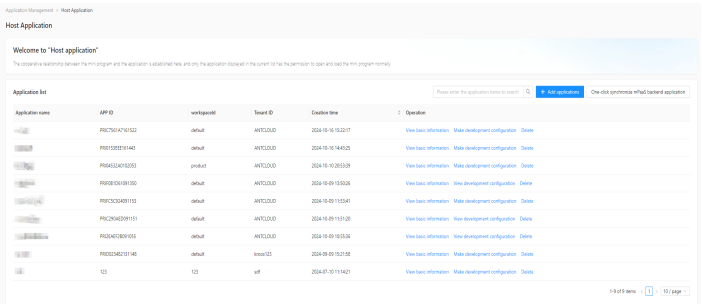
The **Mini Game Qualification Verification** page displays all mini program qualification verification information, including 5 tabs: **All**, **Under review**, **Approved**, **Rejected**, and **Revoked**.

In the verification list on the **Mini Game Qualification Verification** page, you can click **Details** in the operation column to view mini program qualification details in the side panel. For mini program qualifications that have not been reviewed, you can execute **Approve** or **Reject** operations.

5.3.5. Application Management

5.3.5.1. Host applications

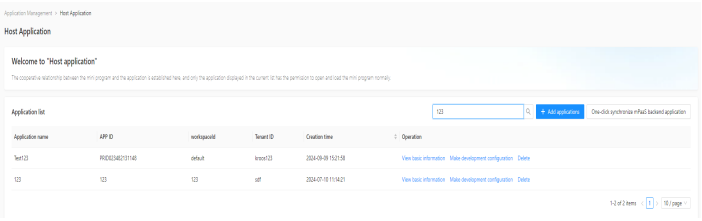
In the **Mini Program Open Platform Operations Management Console**, click **Application Management** > **Host Applications** to access the **Host Applications** page. Here you can establish cooperative relationships between mini programs and applications. Only applications displayed in the host application list have permission to normally open and load mini programs.



On the **Host Applications** page, you can query applications, add applications, synchronize mPaaS backend applications with one click, view application basic information, configure application development settings, view application development configuration details, and delete applications.

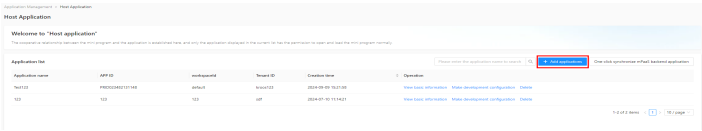
Query applications

On the **Host Applications** page, enter the application name in the input box, click ( ), and applications that meet the search criteria will be displayed in the application list.

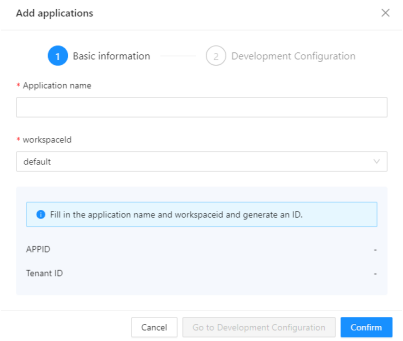


Add an application

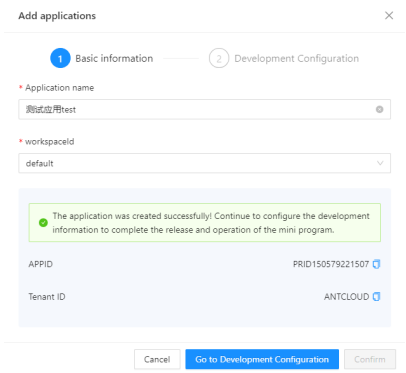
1. On the **Host Applications** page, click **+ Add Application**.



2. In the **Add Application** dialog box, set the basic information of the host application, including application name and workspaceId.

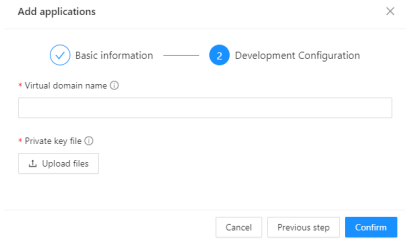


3. Click **OK**. The application is created successfully, and the APPID and tenant ID are automatically generated.



After the application is created successfully, you need to configure development settings for the application.

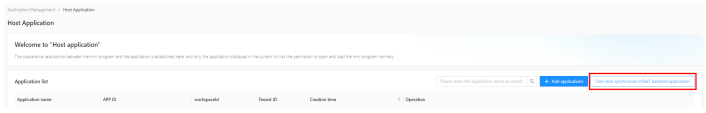
4. Click **Go to Development Configuration**.
5. In the **Development Configuration**, enter the **Virtual Domain Name** and upload the **Private Key File**.



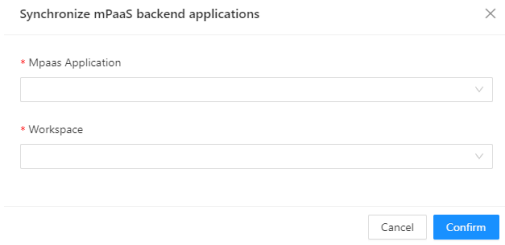
6. Click **OK**. The successfully created application will be displayed in the application list.

Synchronize mPaaS backend applications with one click

1. On the **Host Applications** page, click **Sync mPaaS Backend Applications with One Click**.



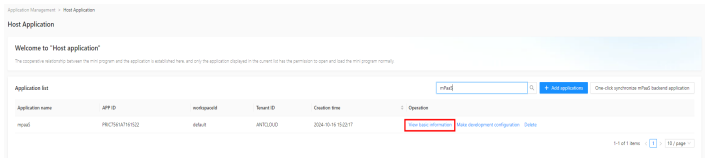
2. In the **Sync mPaaS Backend Applications** dialog box, select the **mPaaS Application** and **Workspace**.



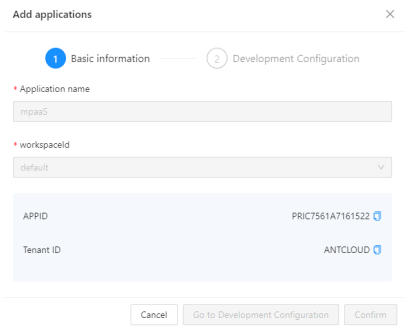
3. Click **OK** to successfully create the host application.

View application basic information

1. On the **Host Applications** page, click **View Basic Information** in the **Operation** column of the application.

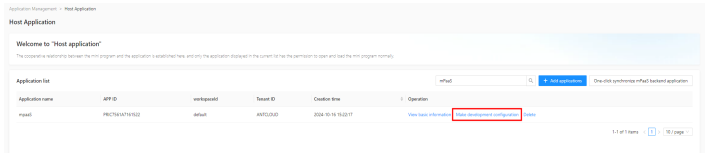


2. In the **Basic Information** section of the **Add Application** dialog box, you can view the **Application Name**, **workspaceId**, and copy the **APPID** and **Tenant ID**.

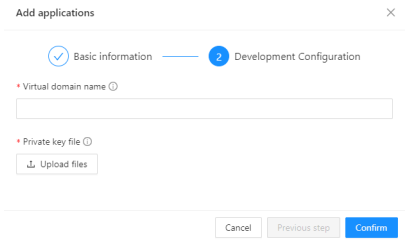


Configure application development settings

1. On the **Host Applications** page, click **Configure Development Settings** in the **Operation** column of the application.



2. On the **Development Configuration** page of **Add Application**, enter the **Virtual Domain Name** and upload the **Private Key File**.



3. Click **OK** to complete the development configuration for the application.

View application development configuration details

1. On the **Host Applications** page, click **View Development Configuration** in the **Operation** column of the application.



2. On the **Development Configuration** page of **Add Application**, view the virtual domain name and private key file.

Delete an application

On the **Host Applications** page, click **Delete** in the **Operation** column of the application. After secondary confirmation, the application is successfully deleted.



5.3.5.2. Custom API

By configuring custom APIs in the host application, you can inject capabilities into mini programs, enabling them to call relevant host application capabilities.

Click **Application Management** > **Custom API** in the navigation pane on the left to access the **Custom API** page under **Development Configuration**. On this page, you can add, edit, disable APIs, and view API details.



Add a custom API

1. On the **Custom API** page, click **Add Custom API**.



2. In the **Add Custom API** dialog box, enter the **API name**, **API description**, and select the **Visibility scope** and **Host App**.



3. Click **OK** to add the custom API. The successfully added API appears in the API list.



Edit a custom API

1. On the **Custom API** page, click **Edit** in the custom API list.
2. Set the API information in the **Edit Custom API** dialog box.

3. Click **OK** to save the changes.

Disable a custom API

On the **Custom API** page, click **Disable** in the custom API list. After confirmation, the API is disabled.

View custom API details

On the **Custom API** page, click **Details** in the custom API list. The custom API details dialog box displays detailed information about the custom API.

5.3.6. Information publishing center

The information publishing center is primarily used to publish operational announcements to the platform, display operational announcements, and system notifications. In the **Mini Program Open Platform Operations Management Console**, click **Information Publishing Center** in the left navigation pane to access the **Information Publishing Center** page. This page displays the information list, including title, category, publisher, and publication time.

> **Note**
> System notifications are configured in the miniapp code.

Publish operational announcements

You can publish operational announcements to miniapps in the information publishing center. Follow these steps:

1. On the **Information Publishing Center** page, click **Publish**.
2. In the **Publish New Announcement** dialog box, enter the **Announcement Title** and **Content**.
3. Click **Confirm** to publish the announcement.

View information details

On the **Information Publishing Center** page, click **Details** in the operation column of the information list.

The **Details** dialog box displays the detailed content of the published information.

Delete information

On the **Information Publishing Center** page, click **Delete** in the operation column of the information list. After confirming the action, the published information will be deleted.

5.3.7. Data statistics

5.3.7.1. Mini Program management

The Mini Program management page displays comprehensive data analytics for all mini programs created through the mini program analysis module under the current app. It includes summary data of four key metrics: new users, active users, launch count, and sharing users, along with a data overview of individual mini programs.

You can create mini programs for statistics and analysis, and manage the created mini programs.

The core metrics are described as follows.

Metric	Description
New users	The cumulative number of new users across all mini programs under the current app, with duplicates removed between different mini programs. For an individual mini program, it refers to the number of users who access the mini program page for the first time. Multiple visits by the same user are not counted repeatedly.
Active users	The cumulative number of active users across all mini programs under the current app, with duplicates removed between different mini programs. For an individual mini program, it refers to the total number of users who access the mini program. Multiple visits by the same user are not counted repeatedly.
Launch count	The cumulative number of launches across all mini programs under the current app, without removing duplicates between different mini programs. For an individual mini program, it refers to the total number of times the mini program is opened. A launch is counted from when a user opens the mini program until it exits to the background. When the interval between two consecutive openings of the mini program is less than 30 seconds, it is counted as one opening.
Sharing users	The total number of users who share the mini program page. Multiple shares by the same user are not counted repeatedly.

View mini program data details

The mini program list displays all mini programs created through the mini program analysis module and their core metric data. You can sort by different metrics to view the data overview of individual mini programs.

1. In the Mini Program Open Platform, from the navigation pane on the left, go to **Data Statistics** > **Mini Program Management** page.
2. In the mini program list, select the target mini program, and click the mini program name or **View** in the operation column.

   Enter the Data Overview page to understand the detailed statistics of the mini program.

Mini program code integration

Complete the mini program code integration to track the mini program metrics.

1. In the Mini Program Open Platform, from the navigation pane on the left, go to **Data Statistics** > **Mini Program Management** page.
2. In the mini program list, select the target mini program, and click **Code Integration** in the operation column.
3. In the **Code Integration** panel that appears, copy the integration code to the mini program for integration.

5.3.7.2. Data overview

The data analytics feature supports statistical analysis of usage data for your current App's miniapp, H5 application, or game. Through the data overview page, you can understand the historical statistical data (T+1) and real-time data overview of a single miniapp, H5 application, or game.

To view the data overview, follow these steps:

1. In the **Mini Program Open Platform Operations Management Console**, access **Data Statistics** > **Data Overview** page from the navigation pane on the left.
2. From the selection dropdown box in the upper left corner of the page, select the target miniapp, H5 application, or game. By default, the analytical data of the most recently created miniapp, H5 application, or game is displayed.
3. Click the **Overview Statistics** or **Real-time Dashboard** tab to view the historical or real-time statistical data of the miniapp, H5 application, or game. The analytical data is presented in the form of metric cards and line trend charts.

Overview statistics

The overview statistics page displays the core data metrics and trend changes of "Customer acquisition > Activation > Sharing" for the currently selected miniapp, H5 application, or game. You can view the data performance and value conversion of the current miniapp, H5 application, or game, identify data issues, and locate abnormal data. The page supports selecting metric data for any past 1 day, 7 days, or 30 days.

Metric overview

Displays the historical statistical data of core metrics for the current miniapp, H5 application, or game in the customer acquisition, activation, and sharing stages during the specified time period in card form. This includes new users, active users, open count, average stay duration per session, number of sharing users, new users from sharing, along with the growth rate compared to the previous day, same period last week, or same period last month.

Line trend chart

Displays the change trend of various metric data during the specified time period in the form of a line chart. When you click a metric card, the line chart will display the data trend corresponding to the metrics on that card.

Real-time dashboard

The real-time dashboard displays the real-time statistical data of five core metrics for the currently selected miniapp, H5 application, or game, and the hourly change trend of each metric.

Metric overview

Displays the real-time data of various metrics for the current miniapp, H5 application, or game in card form, including active users, open count, page visit count, average stay duration per session, average stay duration per user, and the growth rate compared to the same period yesterday.

Line trend chart

Displays the change trend of various metric data for today, yesterday, and 7 days ago in the form of a line chart. When you click a metric card, the line chart will display the data change trend of the corresponding metric.

Metric description

The descriptions of the various metrics in the data overview are as follows.

Metric	Description
New users	The number of users who visit the miniapp/H5 application/game page for the first time. Multiple visits by the same user are not counted repeatedly.
Active users	The total number of users who visit the miniapp/H5 application/game. Multiple visits by the same user are not counted repeatedly.
Open count	The total number of times the miniapp/H5 application/game is opened. From when a user opens the miniapp/H5 application/game until it exits to the background is counted as one time. When the interval between two consecutive opens of the miniapp/H5 application/game is less than 30 seconds, it is calculated as one open.
Average stay duration per session	During the selected time period, the average total duration that users stay on the miniapp/H5 application/game page per open, i.e., average stay duration per session = total stay duration of all users / open count.
Average stay duration per user	The average total duration that each user stays on the miniapp/H5 application/game page, i.e., average stay duration per user = total stay duration / number of active users.
Number of sharing users	The total number of users who share the miniapp/H5 application/game page. Multiple shares by the same user are not counted repeatedly.
New users from sharing	The number of new users who enter the miniapp/H5 application/game for the first time through a shared link.
Previous day (%)	The growth rate of the cumulative data for the selected day compared to the cumulative data for the previous day. Previous day comparison = (data for the specified date - data for the previous day) / data for the previous day. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.
Same period yesterday (%)	The growth rate of the cumulative data from 0:00 to the current hour compared to the cumulative data for the same time period yesterday.
Same period last week (%)	The growth rate of the cumulative data for the past 7 days compared to the cumulative data for days 8-14. Same period last week = (data for the past 7 days - data for the same period last week) / data for the same period last week. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

Same period last month (%)	The growth rate of the cumulative data for the past 30 days compared to the cumulative data for days 31-60. Same period last month = (data for the past 30 days - data for the same period last month) / data for the same period last month. When the denominator is 0, it is calculated as 1. Integer digits are retained before the percentage sign.

5.3.7.3. User Analysis

The User Analysis page displays user access data (with T+1 timeliness) for the currently selected miniapp/H5 application/game. It includes seven metrics such as new users, active users, number of opens, and page visits to help you analyze detailed data changes of users at various stages. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view user analysis data, follow these steps:

1. In the **Mini Program Open Platform Operations Management Console**, go to **Data Statistics** > **User Analysis** page from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp/H5 application/game.
3. Select a time range to view the statistical data for the corresponding period.

User analysis overview

This section displays statistical data for seven metrics of the miniapp/H5 application/game during the specified period. These include new users, active users, number of opens, page visits, average duration per session, average duration per user, and average opens per user, along with the growth rate compared to the previous day, same period last week, or same period last month.

The metrics are explained as follows.

Metric	Description
New users	The number of users who access the miniapp/H5 application/game pages for the first time. Multiple visits by the same user are not counted repeatedly.
Active users	The total number of users who access the miniapp/H5 application/game. Multiple visits by the same user are not counted repeatedly.
Number of opens	The total number of times the miniapp/H5 application/game is opened. From when a user opens the miniapp/H5 application/game until it exits to the background is counted as one time. When the interval between two consecutive opens of the miniapp/H5 application/game is less than 30 seconds, it is counted as one open.
Page visits	The total number of visits to all pages within the miniapp/H5 application/game. Jumps between multiple pages and repeated visits to the same page are counted as multiple visits.
Average duration per session	During the selected period, the average total duration of each open of the miniapp/H5 application/game staying on the miniapp/H5 application/game pages, that is, average duration per session = total stay duration of all users / number of opens.
Average duration per user	The average total duration each user stays on the miniapp/H5 application/game pages, that is, average duration per user = total stay duration / number of active users.
Average opens per user	During the selected period, the average total number of times each user opens the miniapp/H5 application/game, that is, average opens per user = total number of opens by all users during the period / total number of active users.
Previous day (%)	The growth rate of the cumulative data for the selected day compared to the cumulative data for the previous day. Previous day comparison = (user access data for the specified date - user access data for the previous day) / user access data for the previous day. When the divisor is 0, it is calculated as 1. Integer places are retained before the percentage sign.
Same period last week (%)	The growth rate of the cumulative data for the past 7 days compared to the cumulative data for the past 8-14 days. Same period last week = (user access data for the past 7 days - user access data for the same period last week) / user access data for the same period last week. When the divisor is 0, it is calculated as 1. Integer places are retained before the percentage sign.
Same period last month (%)	The growth rate of the cumulative data for the past 30 days compared to the cumulative data for the past 31-60 days. Same period last month = (user access data for the past 30 days - user access data for the same period last month) / user access data for the same period last month. When the divisor is 0, it is calculated as 1. Integer places are retained before the percentage sign.

User access trends

This section displays the trend of changes in each metric's data during the specified period as a line chart. When you click on a metric card, the line chart will display the data trend for the corresponding metric.

Detailed data

This section displays user analysis statistical data for each hour or day within the selected time period in a table format, supporting data viewing by time sorting.

The time granularity of the data display depends on the selected time range. When the query time range is 1 day, the data display granularity is hourly. When the time range is 7 or 30 days, the data display granularity is daily.

5.3.7.4. Page analysis

Page analysis displays page view data for the currently selected miniapp/H5 application/game (with T+1 data latency), including page visit count, page visitor count, and average page visit duration. This helps you understand which pages users visit most frequently and how long they spend on each page. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view page analysis data, complete the following steps:

1. From the navigation pane on the left of the **Open Platform Operations Management Console**, go to **Data Statistics** > **Page Analysis**.

2. From the dropdown box in the upper left corner of the page, select the target miniapp/H5 application/game.
3. Select a time range to view the statistical data for the corresponding period.

- **Page visit count**: The total number of visits to all pages within the miniapp/H5 application/game. Jumps between multiple pages and repeated visits to the same page are counted as multiple visits.
- **Page visitor count**: The total number of users who visit each page in the miniapp/H5 application/game. Multiple visits by the same user are not counted repeatedly.
- **Average page visit duration**: The average duration of each visit when a user opens and stays on a miniapp/H5 application/game page. This is calculated as: average page visit duration = total time spent on pages during the specified period / total page visit count.

5.3.7.5. Share analysis

The share feature is an important channel for miniapps, H5 applications, and games to attract new users and acquire customers. It is also an important metric to evaluate the health of these platforms. The share analysis page displays sharing data (with T+1 effectiveness) for the currently selected miniapp, H5 application, or game, including five metrics: number of users sharing, number of shares, share return traffic, and more, helping you understand the viral sharing effect of your platform. You can view statistical data for any past 1 day, 7 days, or 30 days.

To view share analysis data, follow these steps:

1. From the **Mini Program Open Platform**'s **Operations Management Console**, access **Data Statistics** > **Share Analysis** page from the navigation pane on the left.
2. From the dropdown box in the upper left corner of the page, select the target miniapp, H5 application, or game.
3. Select a time range to view statistical data for the corresponding period.

Share analysis overview

Displays statistical data for five metrics of the miniapp, H5 application, or game during the specified period: number of users sharing, number of shares, share return traffic, share return ratio, and new shares, along with the growth rate compared to the previous day, same period last week, or same period last month.

The metrics are explained as follows.

Metric	Description
Number of users sharing	The total number of users who shared the miniapp/H5 application/game pages. Multiple shares by the same user are not counted repeatedly.
Number of shares	The total number of times the miniapp/H5 application/game pages were shared.
Share return traffic	The number of times users access the miniapp/H5 application/game by clicking on shared links is counted as the return traffic for this share.
Share return ratio	This metric reflects the effectiveness of share propagation. Share return ratio = Share return traffic/Number of shares × 100%.
Add Share	The number of new users who enter the miniapp/H5 application/game for the first time through shared links.
Previous day (%)	The growth rate of the selected day's cumulative data compared to the previous day's cumulative data. Previous day comparison = (Share data of the specified date - Share data of the previous day)/Share data of the previous day. When the denominator is 0, it is calculated as 1. Only the integer part before the percentage sign is retained.
Same period last week (%)	The growth rate of the past 7 days' cumulative data compared to the cumulative data from 8-14 days ago. Same period last week = (Share data of the past 7 days - Share data of the same period last week)/Share data of the same period last week. When the denominator is 0, it is calculated as 1. Only the integer part before the percentage sign is retained.
Same period last month (%)	The growth rate of the past 30 days' cumulative data compared to the cumulative data from 31-60 days ago. Same period last month = (Share data of the past 30 days - Share data of the same period last month)/Share data of the same period last month. When the denominator is 0, it is calculated as 1. Only the integer part before the percentage sign is retained.

Share trends

Displays the trend of each metric's data during the specified period in the form of a line chart. Click on a metric card, and the line chart will display the data trend for the corresponding metric.

Detailed data

Displays share analysis statistical data for each hour or each day within the selected time period in a table format, supporting time-based sorting for data viewing.

The time granularity of the data display depends on the selected time range. When the query time range is 1 day, the data display granularity is hourly. When the time range is 7 or 30 days, the data display granularity is daily.

5.3.8. Operation logs

The operation logs page displays log information of users in the Mini Program management console. It includes operation date, operation user, IP, execution status, operation API, and operation content. You can query log information that meets specific conditions by filtering operation users or operation dates.



5.3.9. Custom settings

The Mini Program Open Platform supports setting brand information, registration and logon page, Mini Program settings, and Security Settings. In the **Mini Program Open Platform Operations Management Console**, click **Custom settings** in the left menu bar to configure related settings on the **Brand and security custom settings** page.

- On this page, you can modify brand information, registration and logon page, Mini Program settings, and Security Settings. Click **Save settings** to modify the platform information.
- You can also click **Restore default settings** to restore the default settings for the platform information.

Brand information

Registration and logon page

Mini Program settings

Security Settings