Ant Technology

Ant Cube Card User Guide

Document Version: 20260303

蚂蚁集团 ANT GROUP

Legal disclaimer

Ant Group all rights reserved©2022.

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.About Ant Cube Card

This article introduces Ant Cube Card in detail from the definition and benefits.

Product Definition

Ant Cube Card is originally a native high-performance rendering engine developed by Alipay. It can provide a variety of functions in different product forms in terms of input products, JS dynamic capability support, rendering data structure, and style support, and supports the combination of enhanced functions for different scenarios. With the requirements of new business scenarios for rendering capabilities and peripheral capabilities, Ant Cube Card is gradually expanded into "an application development technology stack based on a high-performance rendering engine". And "taking into account the user experience and research and development efficiency, the pursuit of the ultimate performance", it has become the technical goal of Ant Cube Card.

Product Benefits

- **Provide dynamic content display to improve development & operation efficiency**
  - Embed in native pages as cards.
  - Android/iOS dual-end consistency, high development efficiency, visible immediately after releasing.
  - Small size, good performance, less memory.
- **Deeply polished by Alipay's wallet business**
  - Multiple front-end development languages (Lite Vue).
  - Comprehensive development and debugging tools (compile, preview, debug, and release).
  - Client SDK + Server Card Management System.

2.Integrate into Client

2.1. Integrate into Android

2.1.1. Quick start

This topic describes how to integrate and use Ant Cube Card.

Ant Cube Card is in public preview starting from the mPaaS 10.2.3 baseline version. Currently, it only supports the mPaaS native AAR connection type. For more information about connection types, see Introduction to connection types.

Prerequisites

- You have activated and integrated mPaaS.
- You have installed the Ant Cube Card AntCubeTool. For more information, see About AntCubeTool.

Procedure

1. Select a baseline.
   i. Click **mPaaS** > **Native AAR Connection**. In the panel that appears, click **Start Configuration** under Connect/Upgrade Baseline to add the 10.2.3 baseline.
   ii. Add the **Cube Card** and **Cube Card-Network Image Loading Library** components.

   > **Important**
   >
   > The **Cube Card-Network Image Loading Library** is a sample library that helps you quickly integrate and try out cube cards. Replace it with your own imageLoader before you publish your application. This change reduces the package size, unifies image management, and lets you monitor image exceptions. For more information about setting the network image download Handler, see Client sends notifications to the card.

2. **Initialize the card.**

i. **The initialization process differs depending on the baseline version. Follow the steps for your specific version.**

- **For baseline versions 10.2.3 and later, add `MP.init(this);` in your Application class to initialize mPaaS. To set initialization parameters, see DPI engine initialization parameters.**
- **For baseline versions earlier than 10.2.3, add the following code in your Application class to initialize mPaaS.**

```
public class MainApplication extends Application {

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

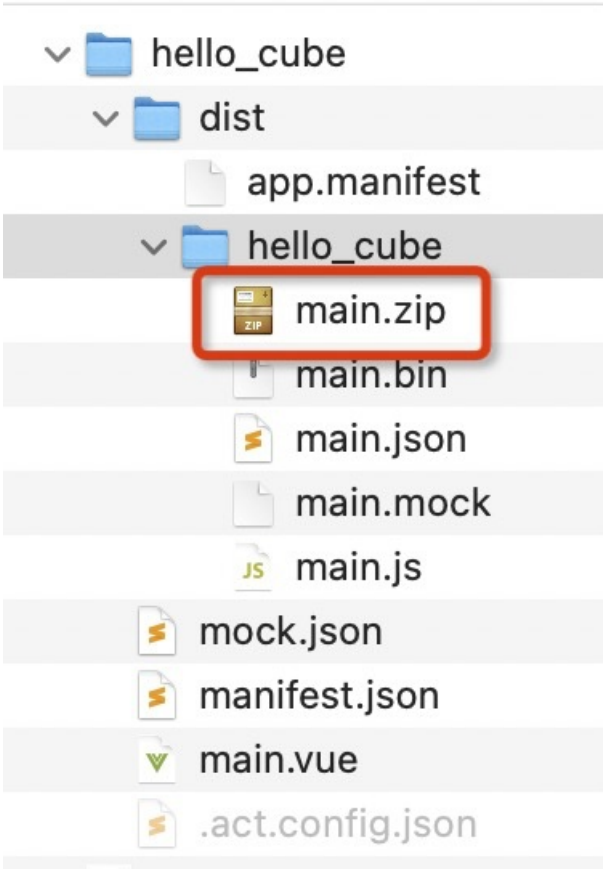
ii. **In the `AndroidManifest.xml` file in the App folder, add the following code to ensure that C layer crashes can be scraped.**

```
<!-- Required for Ant Cube Card to scrape C layer crashes -->
<receiver

android:name="com.alipay.mobile.common.logging.process.LogReceiverInToolsProcess"
    android:enabled="true"
    android:exported="false"
    android:process=":tools">
    <intent-filter>
        <action android:name="${applicationId}.monitor.command" />
    </intent-filter>
</receiver>
<receiver
    android:name="com.alipay.mobile.logmonitor.ClientMonitorWakeupReceiver"
    android:enabled="true"
    android:exported="false"
    android:process=":push">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="${applicationId}.push.action.CHECK" />
        <action android:name="${applicationId}.monitor.command" />
    </intent-filter>
</receiver>
```

3. **Build the card project.**

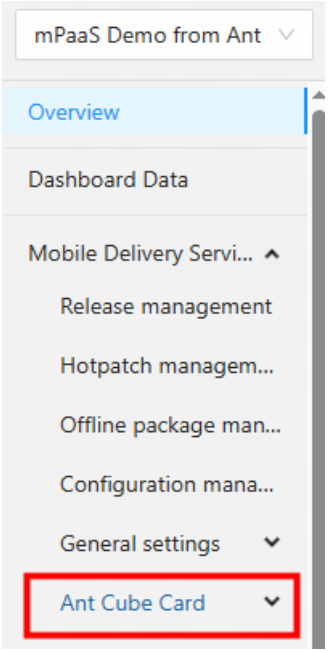
i. **Initialize the project. In the terminal, run the `act init` command.**
   - **Select Cube as the application type and choose the card template (VUE format).**
   - **Enter the application name. The name must be a combination of English letters, numbers, and underscores.**
   - **Select An extra folder for project source code is needed. This option creates an additional folder with the same name as your application. If you do not select this option, the project is initialized in the current folder.**

ii. **Build the project. Use the `cd` command to navigate to the card project directory that you just created. Then, run `act build` to build the project. The build output is located in the `/dist/` folder of your project.**



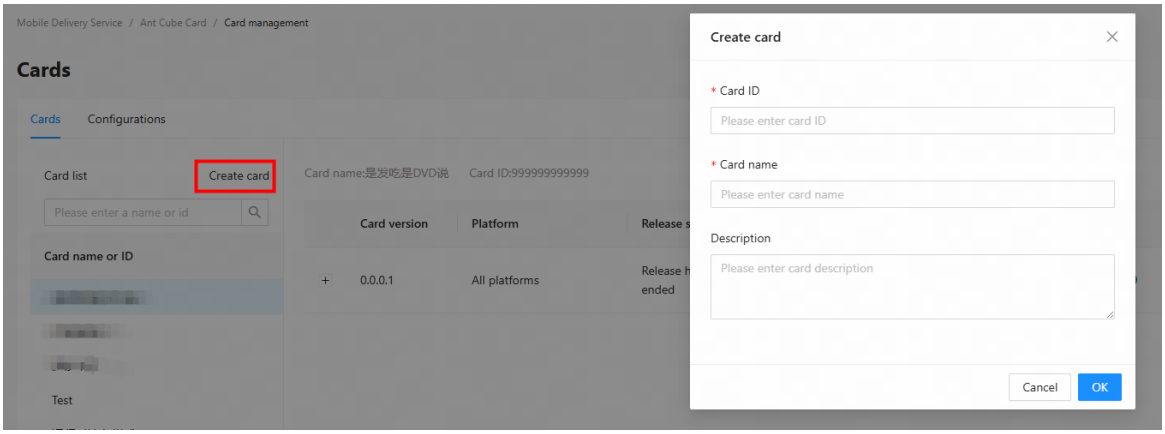
4. **Publish the card.**

i. **Go to the card console.**



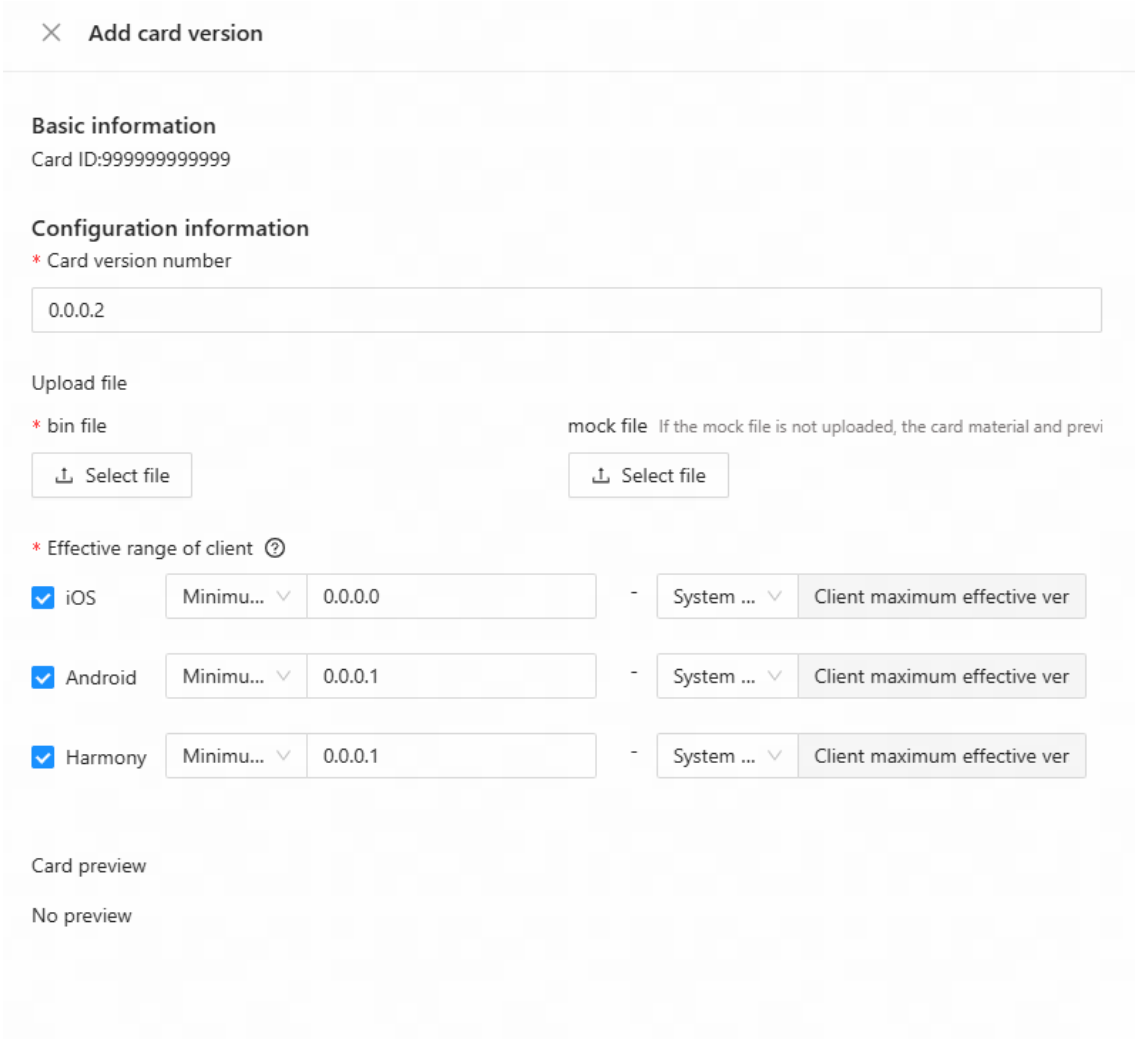
ii. **Click Create Card.**

Set the card ID and card name. The card ID must be at least 8 characters long and can only contain English letters, numbers, and underscores. The client uses this ID for rendering. The card name can be any string up to 20 characters long.



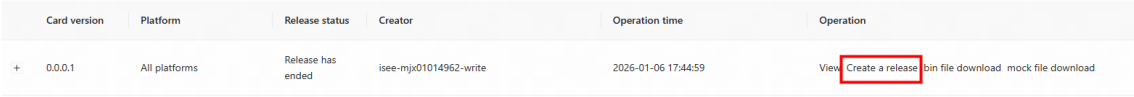
iii. **Add card resources.**

- **Use a 4-digit version number.**
- **Select the `main.zip` file that you just compiled.**
- **The Client Scope parameter specifies the range of client versions that can pull this card. To include all client versions, enter `0.0.0.0` as the minimum version.**

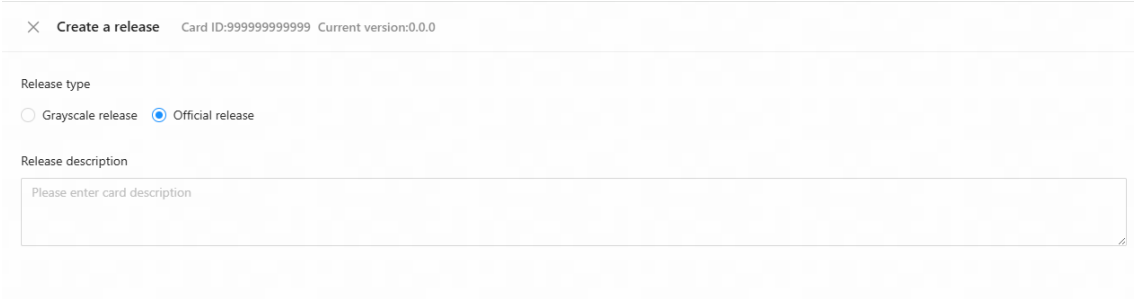


iv. **Publish the card.**

a. **Click Create a release.**



b. **Select Official release.**



After the card is published, the client can pull it.

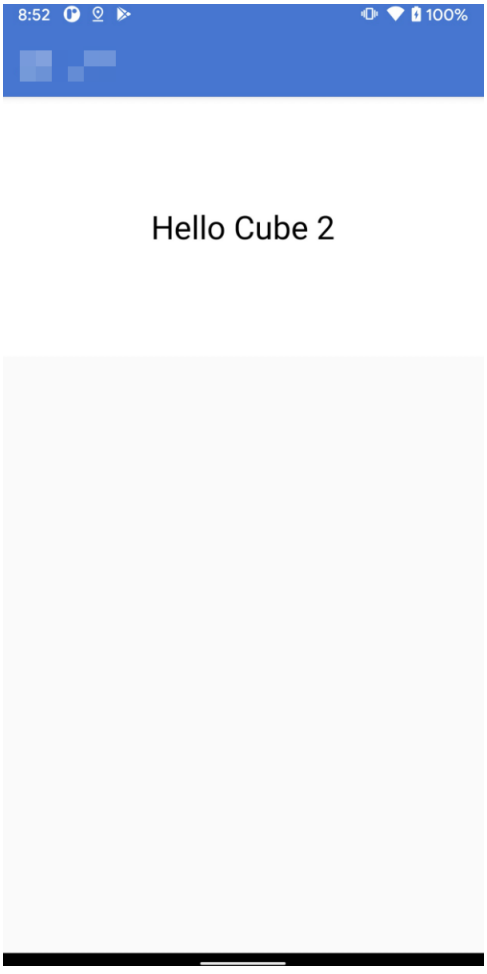
5. **Render the card.**

```
// Create card configuration
CubeCardConfig cardConfig = new CubeCardConfig();
// Card ID created in the console
cardConfig.setTemplateId("hello_cube");
// Card version
cardConfig.setVersion("1.0.0.0");
// Card width, set to screen width here
cardConfig.setWidth(MFSystemInfo.getPortraitScreenWidth());
// Card data (data for rendering the card, usually the content of mock.json)
JSONObject obj = new JSONObject("xxxxx");
cardConfig.setData(obj);
// Create card information
CubeService.instance().getEngine().createCard(cardConfig, new CCardCallback() {
    @Override
    public void onLoaded(final CubeCard crystalCard, CCardType cardType,
CubeCardConfig cubeCardConfig, CubeCardResultCode result
        if (resultCode == CubeCardResultCode.CubeCardResultSucc) {
            // Must run on the main thread
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    // Create the card View
                    CubeView view =
CubeService.instance().getEngine().createView(FastActivity.this);
                    // Add to the outer ViewGroup
                    mWrapperLl.addView(view);
                    // Render the card
                    crystalCard.renderView(view);
                }
            });
        } else {
            MPLogger.info("cube", "fail " + cubeCardConfig.getTemplateId() + " style
" + cardType + " error " + resultCode);
        }
    }
});
```

When the page is destroyed, you must manually revoke the card.

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if (mCard != null) {
        mCard.recycle();
    }
    int chidrenCount = mWrapperLl.getChildCount();
    for (int i = 0; i < chidrenCount; i++) {
        if (mWrapperLl.getChildAt(i) instanceof CubeView) {
            ((CubeView) mWrapperLl.getChildAt(i)).destroy();
        }
    }
    mWrapperLl.removeAllViews();
}
```

6. Preview.



2.1.2. Kernel version upgrade

Version list

Version Number	Minor Version	Corresponding Version Information	Changelog	Notes
10.2.90	Main baseline version	Main baseline version	The first-generation Cube kernel version with basic features.	No upgrade is required. This is the default version for the main baseline.
10.7.30 (beta)	20251105	Rendering engine: com.mpaas.cube.adapter:cubeadapter-build:1.0.0.250923200615 com.mpaas.cube.sdk:cubesdk-build:1.0.0.251104172537 Debug library: com.mpaas.cubedebug.sdk:cubedebug-build:1.0.0.251106163849	10.7.30 upgrade • Adds support for components. • Synchronized with Alipay upgrades and includes bug fixes.	

Kernel version upgrade

- Use the main baseline for the mPaaS baseline.
- The kernel version specifies the corresponding version information.

Set the baseline in the project's `build.gradle` file.

```
buildscript {
  ...
 ext.mpaas_artifact = "mpaas-baseline"
 ext.mpaas_baseline = "10.2.3-72" // Use the latest main baseline version.
  ...
}
```

Set the kernel version in the module's `build.gradle` file.

```
dependencies {
    ...
    implementation('com.mpaas.cube.adapter:cubeadapter-build:1.0.0.250923200615@aar') {
        force = true
    }
    implementation('com.mpaas.cube.sdk:cubesdk-build:1.0.0.251104172537@aar') {
        force = true
    }
   // Select the required kernel version number from the list above. The example uses v
ersion 10.7.30-20251105.
    ...
}
```

Kernel upgrade differences

10.2.90 to 10.7.30

minSdk

minSdkVersion ≥ 21

Card beforeCreate

When using `data` in the card's `beforeCreate` method, the original `this.xxx` should be changed to `this.data.xxx`.

Use the new ACT

Kernel versions 10.7.30 and later require ACT 4.0 or later for compilation.

New abstract method in CCardWidget

`CCardWidget` has a new `sizeOfWidgetView` abstract method. This method is not currently used, so you can simply return an instance.

```
@Override
public CSSize sizeOfWidgetView(Map<String, String> map, Map<String, Object> map1, Map<S
tring, Object> map2, int i, int i1) {
    return new CSSize();
}
```

Debug path

In the new kernel, the debug path is enabled and disabled by the upper layer, not by the kernel. When the debug path is enabled, you can quickly access locally developed cards through the innerStorage cache path. You must enable the debug path after Cube is successfully initialized.

To enable the debug path, call the following method:
`CrystalTemplateLoader.openDebugLoader();`

```
CubeInitParam cubeInitParam
= CubeInitParam.getDefault()
...
.setCubeInitCallback(new CubeInitParam.CubeInitCallback() {
 @Override
 public void onInit() {
  // Check if it is a debug package. Do not enable this for release packages.
  CrystalTemplateLoader.openDebugLoader();
 }

 @Override
 public void onError(Exception e) {

 }
});
```

Debug tool upgrade

Replace the old Debug library with the corresponding new version. You also need to import `cubedebug.aar` and `jsi-sdk.aar` (If there is Ariver Mini Program, this aar is not required).

> **Note**
>
> To import `cubedebug.aar` and `jsi-sdk.aar`, join the DingTalk group by searching for the ID 145930007362. Then, contact the support staff to request the files.

Add the following code to the module's `build.gradle` file.

```
configurations {
    all*.exclude group: 'com.alipay.android.phone.wallet', module: 'cubedebug-build'
}

dependencies {
 debugImplementation('com.mpaas.cubedebug.sdk:cubedebug-build:1.0.0.251106163849@aar')
{
     force = true
    }
}
```

> **Important**
>
> Please comment out any debug-related libraries before deployment and use the `debugImplementation` method to reference them.

2.1.3. APIs

2.1.3.1. CubeService

Introduction

This topic is about the card service class.

Methods

instance

```
/**
 * Gets the CubeCrystalService instance
 */
public static CubeCrystalService instance()
```

initEngine

```
/**
 * Global initialization
 * @param Initialization configuration
 * @param application The application
 */
public void initEngine(CubeEngineConfig config, Application application)
```

getEngine

```
/**
 * Gets the engine instance
 * @return The singleton instance of the engine
 */
public CubeEngine getEngine()
```

destroyEngine

```
/**
 * Globally destroy
 */
public void destroyEngine()
```

2.1.3.2. CubeEngine

Introduction

This is about the core class of the card engine.

Methods

createCard

```
/**
 * Creates a single card.
 * @param config The card configuration parameters.
 * @param callback The callback.
 */
public void createCard(final CubeCardConfig config, final CCardCallback callback)
```

createCards

```
/**
 * Creates cards in a batch.
 * @param configs The batch configuration.
 * @param callback The callback. The callback is invoked once for each card result.
 */
public void createCards(List<CubeCardConfig> configs, final CCardCallback callback)
```

createView

```
/**
 * Creates a rendering view.
 * @return The Cube rendering container view.
 */
public CubeView createView(Context context)
```

setCustomUnit

```
/**
 * Sets a custom unit.
 * @param unitName The name of the unit, such as sip.
 * @param unitRadio The unit ratio, such as 1.5.
 */
public void setCustomUnit(String unitName, float unitRadio)
```

registerModule

```
/**
 * Registers a custom module.
 * @param models The key is the module name, such as animation, and the value is the class name, such as CKAnitmationModule.
 * @param options
 */
public void registerModule(Collection<CubeModuleModel> models, Bundle options)
```

registerWidgets

```
/**
 * Registers a group of custom extension widgets.
 * @param widgets The information about the extension widgets.
 */
public void registerWidgets(Collection<CubeWidgetInfo> widgets)
```

sendEvent

```
/**
 * Sends a custom event from native to JavaScript.
 * @param componentData The component data. This is the `data` input parameter used when you create a component in `onCreateView` of `CCardWidget`. Pass through this parameter.
 * @param eventName The name of the custom event.
 * @param eventParams The parameters of the custom event.
 */
public void sendEvent(Map<String, Object> componentData, String eventName, @Nullable Map<String, Object> eventParams)
```

destroy

```
/**
 * Destroys the card instance.
 */
private void destroy()
```

2.1.3.3. CubeEngineConfig

Introduction

A configuration class for initializing the card engine.

Methods

getResourcePath

```
/**
 * Gets the path of the local resource plan.
 * @return
 */
public String getResourcePath()
```

setResourcePath

```
/**
 * Sets the path of the local resource plan.
 * @param resourcePath
 */
public void setResourcePath(String resourcePath)
```

getExceptionListener

```
/**
 * Gets the exception listener.
 * @return
 */
public CExceptionListener getExceptionListener()
```

setExceptionListener

```
/**
 * Sets the exception listener.
 * @param exceptionListener
 */
public void setExceptionListener(CExceptionListener exceptionListener)
```

getImageHandler

```
/**
 * Gets the image handler.
 * @return
 */
public ICKImageHandler getImageHandler()
```

setImageHandler

```
/**
 * Sets the image handler. If this parameter is not set, a default internal implementat
ion is used. Provide a custom implementation.
 * @param imageHandler
 */
public void setImageHandler(ICKImageHandler imageHandler)
```

setRpcHandler

```
/**
   * Sets a custom handler for RPC requests from the card.
   * @param handler
   * @returns
   */
public void setRpcHandler(ICRpcHandler rpcHandler)
```

2.1.3.4. CExceptionListener

Introduction

A listener for card exceptions.

Method

onException

```
/**
 * Called when an exception occurs.
 * @param info The exception information.
 */
public void onException(CExceptionInfo info);
```

2.1.3.5. CExceptionType

Introduction

The cards exception types.

Enumeration

```
/**
 * The exception types for cards.
 */
public enum CExceptionType {
    // JS exception
    JS_EXCEPTION,
    // Style exception
    STYLE_EXCEPTION
}
```

2.1.3.6. CExceptionInfo

Introduction

The card exceptions information class.

Methods

getCardUid

```
/**
 * Gets the card instance ID.
 * @return
 */
public String getCardUid()
```

getErrCode

```
/**
 * Gets the card exception type.
 * @return
 */
public CExceptionType getErrCode()
```

getTitle

```
/**
 * Gets the exception title.
 * @return
 */
public String getTitle()
```

getException

```
/**
 * Gets the exception information.
 * @return
 */
public String getException()
```

getExtParams

```
/**
 * Gets the extended exception information.
 * @return
 */
public Map<String, Object> getExtParams()
```

2.1.3.7. CubeCard

Introduction

The core class for cards.

Methods

renderView

```
/**
 * Renders a view. A CubeView object is required.
 * @param view The CubeView object generated by CubeEngine.
 */
public void renderView(CubeView view)
```

getSize

```
/**
 * Gets the width and height of the card.
 * @return Rect An object that contains the width and height of the card.
 */
public Rect getSize()
```

updateData

```
/**
 * Updates the rendering data.
 * @param jsonData The external data model required for cube rendering.
 */
public void updateData(JSONObject jsonData)
```

callJsFunction

```
/**
 * Calls a JS method.
 * @param methodName The method name.
 * @param params The call parameters.
 */
public void callJsFunction(final String methodName, final Object... params)
```

recycle

```
/**
 * Destroys the card and releases its resources.
 */
public void recycle()
```

getCardUid

```
/**
 * Gets the card instance ID.
 * @return The card instance ID.
 */
public String getCardUid()
```

notifyState

```
/**
 * Notifies the card of a state change when it appears on, disappears from, or moves between the foreground and background.
 * @param state The card state.
 */
public void notifyState(CCardState state)
```

getCubeCardConfig

```
/**
 * Gets the config parameter used to create the card.
 * @return The config object used to create the card.
 */
public CubeCardConfig getCubeCardConfig()
```

getBindView

```
/**
 * Gets the view temporarily attached to the card. The view can be reused by other cards.
 * @return The attached view.
 */
public CubeView getBindView()
```

2.1.3.8. CubeCardConfig

Introduction

This card configuration class.

Methods

getTemplateId

```
/**
 * Gets the unique ID of the template.
 * @return
 */
public String getTemplateId()
```

setTemplateId

```
/**
 * Sets the unique ID of the template.
 * @param templateId
 * @return
 */
public CubeCardConfig setTemplateId(String templateId)
```

getVersion

```
/**
 * Gets the version number of the template.
 * @return
 */
public String getVersion()
```

setVersion

```
/**
 * Sets the version number of the template.
 * @param version
 * @return
 */
public CubeCardConfig setVersion(String version)
```

getCardUid

```
/**
 * Gets the unique ID of the card.
 * @return
 */
public String getCardUid()
```

getData

```
/**
 * Gets the card data.
 * @return
 */
public JSONObject getData()
```

setData

```
/**
 * Sets the card data.
 * @param data
 * @return
 */
public CubeCardConfig setData(JSONObject data)
```

getWidth

```
/**
 * Gets the preset width of the card.
 * @return
 */
public int getWidth()
```

setWidth

```
/**
 * Sets the preset width of the card.
 * @param width
 * @return
 */
public CubeCardConfig setWidth(int width)
```

getHeight

```
/**
 * Gets the preset height of the card.
 * @return
 */
public int getHeight()
```

setHeight

```
/**
 * Sets the preset height of the card.
 * @param height
 * @return
 */
public CubeCardConfig setHeight(int height)
```

getExtOption

```
/**
 * Gets the extended parameters of the card.
 * @return
 */
public JSONObject getExtOption()
```

setExtOption

```
/**
 * Sets the extended parameters of the card.
 * @param extOption
 * @return
 */
public CubeCardConfig setExtOption(JSONObject extOption)
```

getEnvData

```
/**
 * Gets the environment variable data of the card.
 * @return
 */
public JSONObject getEnvData()
```

setEnvData

```
/**
 * Sets the environment variable data of the card.
 * @param envData
 * @return
 */
public CubeCardConfig setEnvData(JSONObject envData)
```

getLayoutChangeListener

```
/**
 * Gets the layout change listener.
 * @return
 */
public CCardLayoutChangeListener getLayoutChangeListener()
```

setLayoutChangeListener

```
/**
 * Sets the layout change listener.
 * @param layoutChangeListener
 */
public void setLayoutChangeListener(CCardLayoutChangeListener layoutChangeListener)
```

2.1.3.9. CCardType

Introduction

Specifies the source types for card templates.

Enumeration

```
/**
 * The source types for card templates.
 */
public enum CCardType {
    // Invalid card template.
    C_CARD_TYPE_NONE,
    // Template in memory.
    C_CARD_TYPE_CACHE,
    // Template in a local resource bundle.
    C_CARD_TYPE_BUNDLE,
    // Template on a local SD card.
    C_CARD_TYPE_FILE,
    // Cloud template.
    C_CARD_TYPE_CLOUD
}
```

2.1.3.10. CCardLayoutChangeListener

Introduction

Listener for layout change notifications.

Methods

onLayout

```
/**
 * Notifies of layout changes.
 * @param size The updated card size.
 */
void onLayout(Rect size);
```

2.1.3.11. CubeJSCallback

Introduction

This class sends a callback to a card after it calls a native method.

Method

```
/**
 * Sends callback data to the card.
 * @param data The callback data to send to the card.
 */
public void invoke(Object data)
```

2.1.3.12. CubeCardResultCode

Introduction

The card result code.

Enumeration

```
/**
 * Result codes for card requests.
 */
public enum CubeCardResultCode {
    // The card request is successful.
    CubeCardResultSucc,
    // A network error occurred.
    CubeCardResultNetworkError,
    // The request parameters are invalid.
    CubeCardResultParamError,
    // The card does not exist.
    CubeCardResultNotExist,
    // The card content is invalid.
    CubeCardResultContentInvalid
}
```

2.1.3.13. CubeModule

Introduction

A custom module must inherit from this class and implement the following methods.

```
// Annotation. uiThread specifies whether to invoke the callback on the main process.
    @JsMethod(uiThread = true)
    public void showToast(JSONObject object, final CubeJSCallback callback) {
        if (object != null && !object.isEmpty()){
            String msg = object.getString("message");
            Toast.makeText(ContextUtils.getInstance().getContext(), msg,
Toast.LENGTH_SHORT).show();
        }else {
            callback.invoke("message is null");
        }
    }
```

2.1.3.14. CubeModuleModel

Introduction

Module description information.

Methods

CubeModuleModel

```
/**
 * Constructor.
 * @param type The name of the module, such as animation.
 * @param fullClsName The full package path of the module class.
 * @param methods The JavaScript (JS) methods declared in the module.
 */
public CubeModuleModel(String type, String fullClsName, String[] methods)
```

2.1.3.15. CubeView

Introduction

A container class for card rendering.

Method

destroy

```
/**
 * Releases resources. Call this method explicitly.
 */
public void destroy()
```

2.1.3.16. CCardCallback

Introduction

Card creation callback class.

Method

onLoaded

```
/**
 * Callback notification for card creation.
 * @param card The card instance.
 * @param cardType The source type of the card template.
 * @param config The configuration parameters for the card.
 * @param errorCode The error code.
 */
public void onLoaded(final CubeCard card, CCardType cardType, CubeCardConfig config, Cu
beCardResultCode errorCode);
```

2.2. Integrate into iOS

2.2.1. Quick start

Ant Cube Card (hereinafter referred to as Cube Card) originated from the demand for dynamic native pages, and the product form is Cube Card. With the emergence of the Mini Program concept, Cube Card have been integrated into the Alipay Mini Program technology stack, and the product form is a lightweight Alipay Mini Program solution (as opposed to web Mini Program that are centered on browsing).

Ant Cube Card supports three integration methods: **integrate based on mPaaS framework**, **integrate based on existing projects and using mPaaS plugins**, and **integrate based on existing projects and using CocoaPods**. This topic describes how to integration Ant Cube Card and use Ant Cube Card.

Prerequisites

- You have connected your project to mPaaS. For more information, see the following topics:
  - Connect based on an existing project using CocoaPods
- You have installed the Cube Card AntCubeTool. For more information, see About AntCubeTool.

Connection steps

1. Select a baseline.
   i. Add the 10.2.3 main baseline.
   ii. Add the card component.
2. Initialize the card.
3. Build a card project.
   i. Initialize the project.
   ii. Build the project.
4. Publish a card.
   i. Go to the card backend.
   ii. Create a card.
   iii. Add card resources.
   iv. Publish the card.
5. Render the card.

1. Select baseline.

   Cube Card is now supported in the primary baseline 10.2.3. Please select the appropriate component addition method based on your integration method.

   - **Use mPaaS Xcode Extension**

     This method is applicable to integration methods: **Integrate based on mPaaS framework** or **Integrate by using mPaaS plugin based on existing projects**.

a. Click the Xcode menu item **Editor** > **mPaaS** > **Edit Project** > **Upgrade Baseline** to switch the project to the 10.2.3 baseline.

> **Note**
>
> If the "Upgrade Baseline" option is not clickable, please ensure that the project configuration has been imported. Refer to the Prerequisites section to open the project editing page.

b. After upgrading the baseline, click **Edit Module** > **Modify Module**, select the **Cube Card**, save, and then click **Start Editing** to complete the addition.

- **Use cocoapods-mPaaS Plugin**

  This method is suitable for integration method: **Integrate by using CocoaPods based on the existing project**.

  a. Perform the following operations in the Podfile.

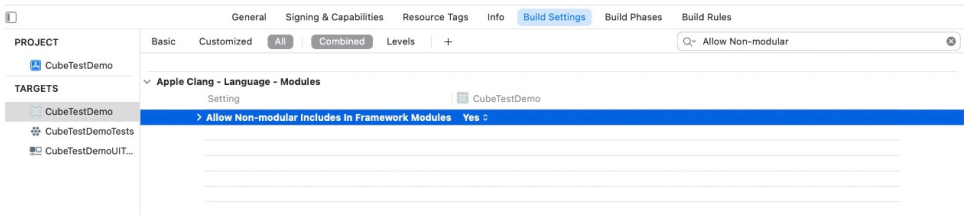
  a. Change **mPaaS_baseline** to `10.2.3`.

  b. Use **mPaaS_pod "mPaaS_Cube"** to add the **Cube Card** component dependency.

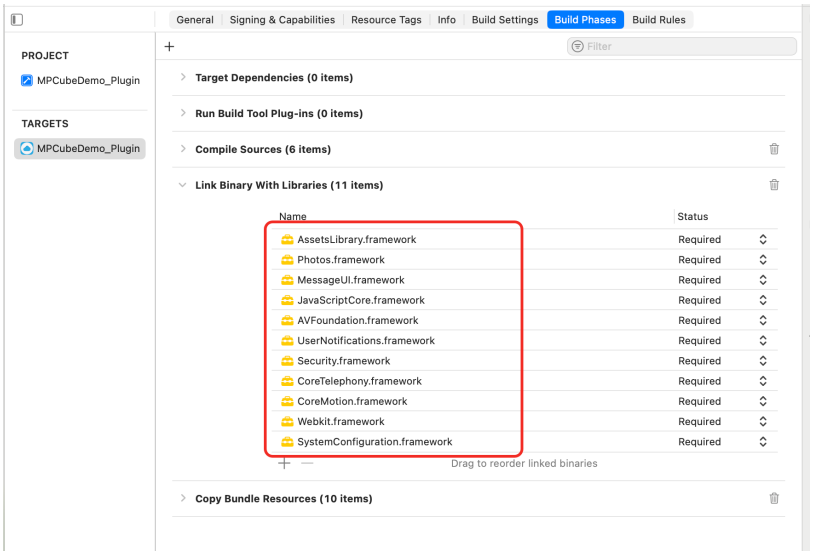
  b. Simply run `pod install` to complete the integration.

2. Initialize the card.

i. Compile the Xcode project. If the header file cannot be found, turn on the option to set the `Allow Non-modular Includes In Framework Modules` to `Yes`, as shown in the following figure.



ii. Add system dependencies.



iii. Import the libraries required for Cube Card.

```
#import <CubeCrystal/CubeEngine.h>
#import <AntCube/CubeService.h>
```

iv. Use a singleton to initialize the card engine.

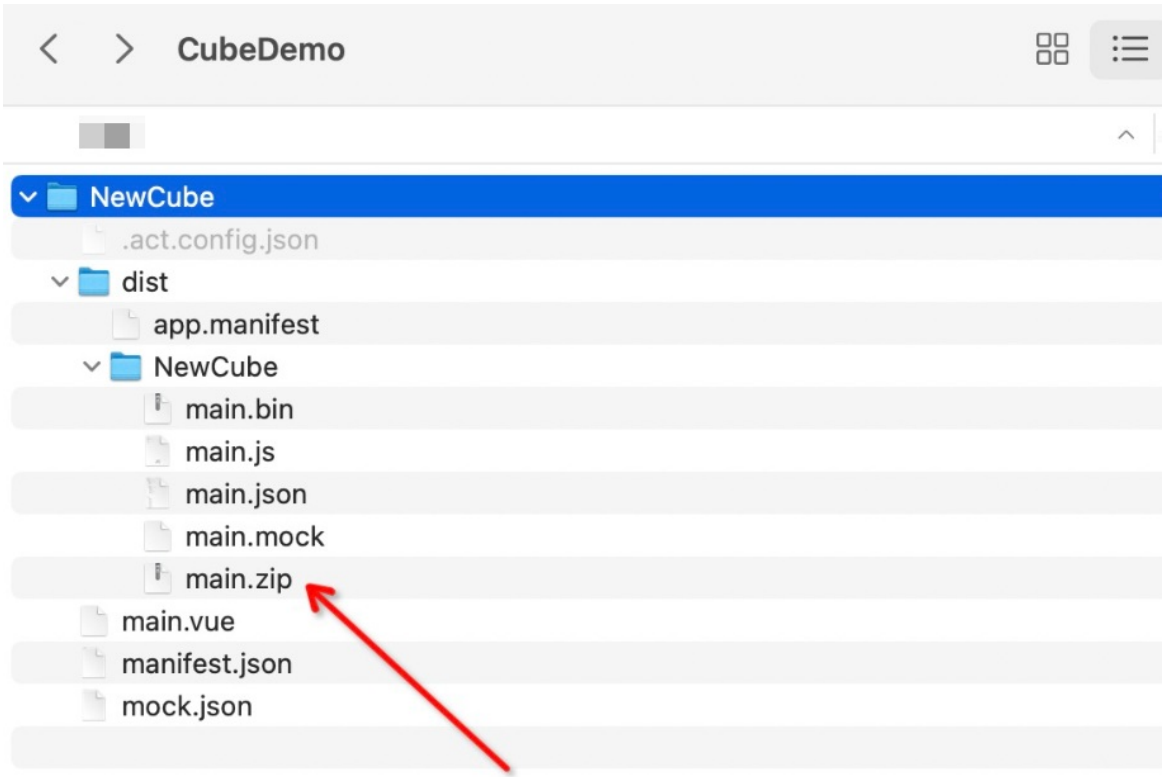
```
- (void)initEngie{
    static dispatch_once_t onceToken;
    NSString *mockBundlePath = [NSString stringWithFormat:@"%@/%@/crystal", [[NSBundle mainBundle] resourcePath], @"MPCubeDemo.bundle"];
    dispatch_once(&onceToken, ^{
        CubeEngineConfig* config = [[CubeEngineConfig alloc] init];
        [config setBundlePath:mockBundlePath];
        [[CubeService sharedInstance] initWithConfig:config];
    });
}
```

3. Build a card project.

i. Initialize a project. Execute `act init` command in the terminal.

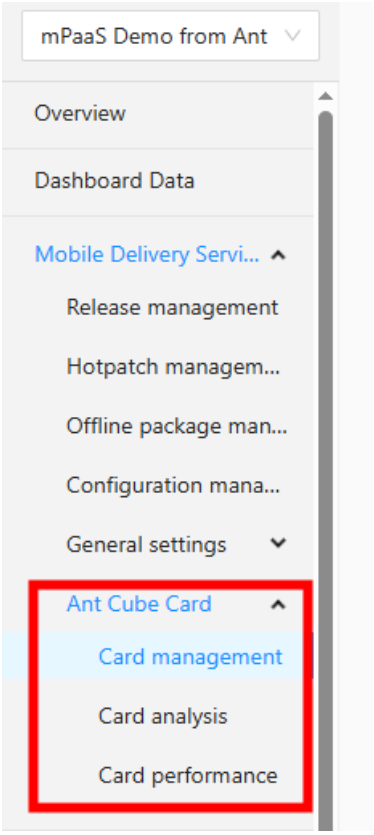
- Select Application Type as Cube and select Template Card (VUE format).
- Enter an application name. Enter a name for your project. We recommend that you use a combination of English, numbers, and underscores.
- Please select "Create an additional project source code folder". This will create an additional folder named after the application. If you select "Do not", the project will be initialized directly in the current directory.

ii. Construct the project. Run the `cd` command to open the created card project and run the `act build` to complete the construction. The constructed product will be in the `/dist/` folder of your project.

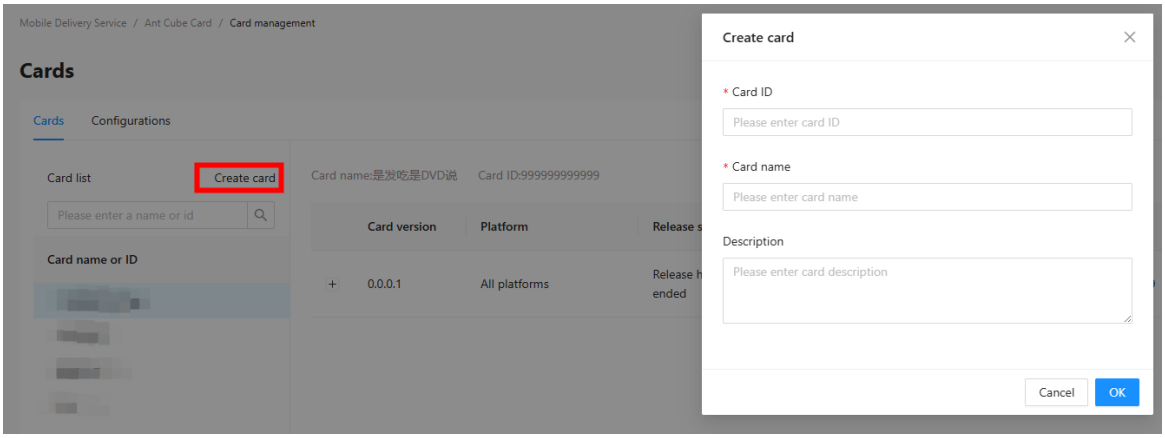


4. Release the card.

i. Go to the Ant Cube Card platform.



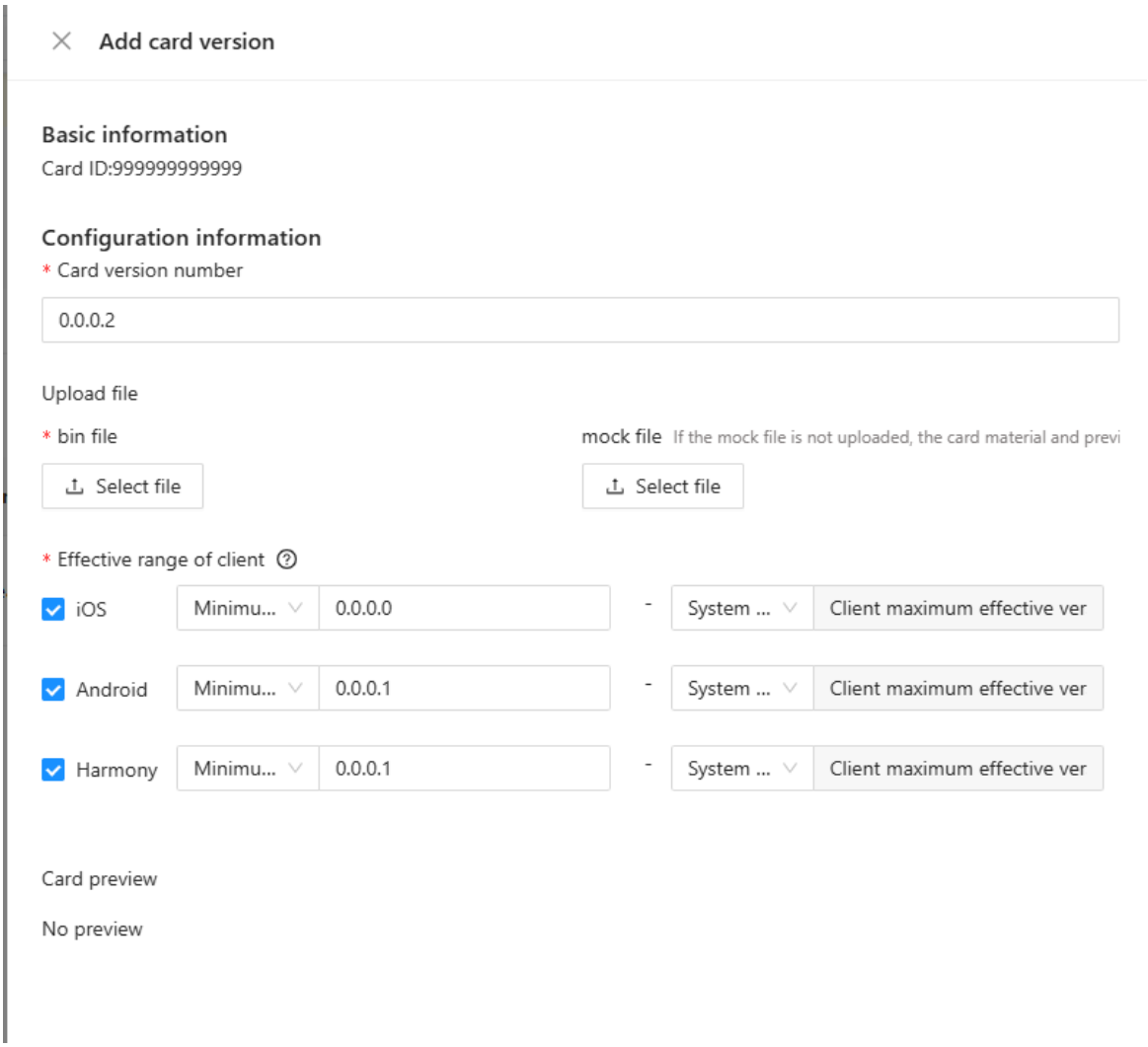
ii. Click **Create Card**.



We recommend that you use a combination of English, numbers, and underscores for the card ID. The client will rely on the card ID when rendering the card. The card name can take any value.

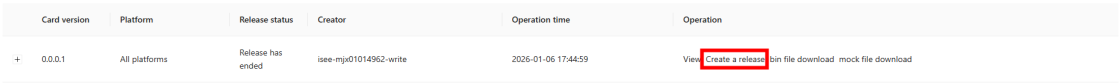
iii. Add card resources.

- We recommend that you use a 4-digit version number.
- Select the main.zip that you just compiled.
- The client range refers to the client version that can be pulled to the card. If you want to overwrite all client versions, enter `0.0.0.0` in the minimum version field.

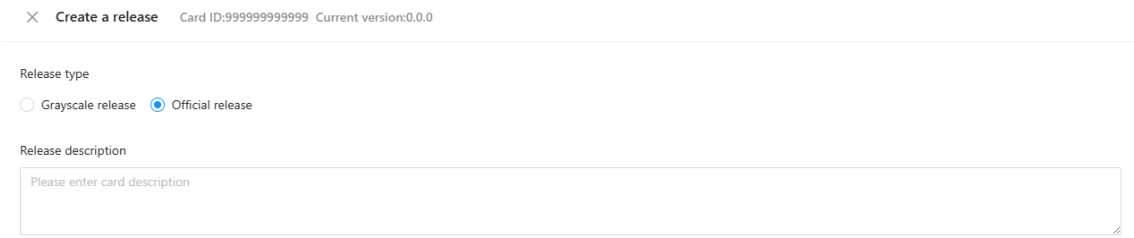


iv. Release the card.

a. Click **Create Release**.



b. Select **Official Release**.



After the card is released, the client can pull the card.

5. Render the card.

i. Copy the card ID and card version number from the webpage.

```
//Create card object.
CubeCardConfig *cardConfig = [[CubeCardConfig alloc] init];
//Configure the card version (required), copied from the console.
[cardConfig setVersion:@"1.0.0.0"];
//Configure card ID (required), copy from console.
[cardConfig setTemplteId:@"20211118"];
//Preset card width.
[cardConfig setWidth:[UIScreen mainScreen].bounds.size.width];
//Preset card height.
[cardConfig setHeight:350];
//Set the card data (required) parameter to business JSON data.
[cardConfig setData:@{}];
//Load the card.
[[[CubeService sharedInstance] getEngine] createCard:cardConfig callback:self];
```

ii. The card retrieves the key "iosImage", and will then obtain the value corresponding to the key "iosImage", as shown in the following code:

```
 [self.cardConfig
setData:@{@"iosImage":@"https://img.zcool.cn/community/013edb5c7b5b1ca801213f269fc887
pg@1280w_1l_2o_100sh.jpg"}];
```

iii. The delegate method after card creation needs to conform to the `CCardCallback` delegate protocol and implement the `onLoaded:cardType:config:erroCode` protocol method.

```
@interface ViewController () <CCardCallback>
@end
```

```
- (void)onLoaded:(CubeCard *)card cardType:(CCardType)cardType config:
(CubeCardConfig *)config erroCode:(CubeCardResultCode)erroCode {
    //Card creation failed
    if (!card) {
        NSLog(@"load card fail %@ style %d error %d", [config templteId], cardType,
erroCode);
        return;
    }

    //Creation successful
    NSLog(@"load card success %@ style %d error %d", [config templteId], cardType,
erroCode);

    dispatch_async(dispatch_get_main_queue(), ^{
        //Main thread UI operations: Rendering cards
  CubeView *view = [[[CubeService sharedInstance] getEngine] createView];
        CGSize size = [card getSize];
        [view setFrame:CGRectMake(0, 20, size.width, size.height) ];
        [self.view addSubview:view];

        [card renderView:view];
    });
}
```

iv. Build & Run.

- If compilation and execution are successful, the Xcode command line may report an RPC request exception. For details, please refer to RPC request exception.

6. Effect preview.

Demo Reference

If you need to refer to the demo of this integration method, please click: MPCubeDemo_Pod.zip.

2.2.2. Kernel version guide

For the iOS kernel, join the DingTalk group by searching for group ID 145930007362. Alternatively, submit a ticket in the ticket system to contact your business representative.

2.2.3. APIs

2.2.3.1. CubeService

This topic describes the core classes of Ant Cube Card Service.

Method

sharedInstance

```
/**
 Obtain a service instance

 @ param None
 @return CubeCrystalService
*/

+ (instancetype)sharedInstance;
```

initWithConfig

```
/**
 Initialize the engine

 @ param config Configuration parameters
 @ return None
*/

- (void)initWithConfig:(CubeEngineConfig *)config;
```

getEngine

```
/**
 Obtain engine instances

 @ param None
 @ return CubeEngine singleton engine instance
*/

- (CubeEngine *)getEngine;
```

destroyEngine

```
/**
 Release engine resources. You can call this operation to release resources when you en
sure that no more crystal-related interfaces are called.
*/

- (void)destroyEngine;
```

2.2.3.2. CubeEngine

This topic describes the methods of the core classes of the Ant Cube Card engine.

createCard

```
/**
 Create a single card

 @ param config Card configuration parameters
 @ param callback The callback for creating a card.
*/

- (void)createCard:(CubeCardConfig *)config callback:(id<CCardCallback>)callback;
```

createCards

```
/**
 Create a batch of cards. Each card result is called back once.

 @ param configs Configure parameters for cards in batches.
 @ param callback The callback for creating a card.
*/

- (void)createCards:(NSArray<CubeCardConfig *> *)configs callback:
(id<CCardCallback>)callback;
```

createView

```
/**
 Create a rendering view

 @return CubeView
*/

- (CubeView *)createView;
```

setCustomUnit

```
/**
 Set custom units

 @ param unitName The name of the company, such as sip
 @ param unitRadio Unit ratio, for example, 1.5
*/

- (void)setCustomUnit:(NSString *)unitName unitRadio:(float)unitRadio;
```

registerModules

```
/**
 Set up a custom extension module

 @ param modules Dictionary, key is the module name, such as animation, and value is th
e class name, such as CKAnitmationModule
*/

- (void)registerModules:(NSDictionary<NSString *, NSString *> *)modules;
```

registerWidgets

```
/**
 Register a component
 @ param widgets The configuration of the widget.
 */
- (void)registerWidgets:(NSArray<CubeWidgetInfo *> *)widgets;
```

sendEvent

```
/**
 Native sends custom event channels to the JS side
 @ param widgetData The component data, which is the input parameter data when the comp
onent is created by onCreateView in CCardWidget. You can pass it through here.
 @ param eventName The name of the custom event.
 @ param eventParams Custom event parameters
 */
- (void)sendEvent:(NSDictionary *)widgetData eventName:(NSString *)eventName eventParam
s:(NSDictionary*)eventParams;
```

2.2.3.3. CubeEngineConfig

This topic describes how to initialize the configuration class for Ant Cube Card Engine.

Properties

bundlePath

```
/// The path of the local resource package that stores the template.
@property (nonatomic, strong) NSString *bundlePath;
```

exceptionListener

```
/// Exception monitoring
@property (nonatomic, strong) id<CExceptionListener> exceptionListener;
```

imageHandler

```
/// Image handler. If this parameter is not specified, the internal default handler is
implemented. We recommend that you customize the handler.
@property (nonatomic, strong) id<CKImageHandler> imageHandler;
```

2.2.3.4. CExceptionListener

This article introduces the method of Ant Cube Card exception monitoring.

Method

onException

```
/**
 Exception monitoring

 @ param info The error message.
*/

- (void)onException:(CExceptionInfo *)info;
```

2.2.3.5. CExceptionType

This topic describes the exception types of Ant Cube Card.

Enumeration

```
/// Exception type
@property (nonatomic, assign) CExceptionType type;

typedef NS_ENUM(NSUInteger, CExceptionType) {
                // JS exception
    CExceptionTypeJavaScript = 0,
    // The card style is abnormal.
    CExceptionTypeStyle
};
```

2.2.3.6. CExceptionInfo

This topic describes the properties of the Ant Cube Card exception information class.

Internet Properties

exceptionMessage

```
/// The exception information.
@property (nonatomic, copy) NSString *message;
```

exceptionCardUid

```
/// The ID of the abnormal card.
@property (nonatomic, copy) NSString *cardUid;
```

exceptionExtraInfo

```
/// Extended parameters for exception information
@property (nonatomic, copy) NSDictionary *extraInfo;
```

2.2.3.7. CubeCard

This article describes the properties and methods of the Ant Cube Card core class.

Properties

cardUid

```
/// The unique ID of the card.
@property (nonatomic, readonly) NSString *cardUid;
```

Methods

renderView

```
/**
 Rendering view
 @param view
*/
- (void)renderView:(CubeView *)view;
```

getSize

```
/**
 Get card size
 @ return Card size
*/
- (CGSize)getSize;
```

updateData

```
/**
 Update data rendering
 @ param data The template data in the JSON format.
*/
- (void)updateData:(NSDictionary *)data;
```

callJsFunction

```
/**
 Call JS methods
 @ param function The name of the method.
 @ param arguments Call parameters
*/
- (void)callJsFunction:(NSString *)function arguments:(NSArray *)arguments;
```

notifyState

```
/**
 Notification card status
 @ param state The status of the card, which is notified when the card is displayed on the screen, displayed on the screen, and back and forth.
 */
-(void)notifyState:(CCardState)state;
```

getBindView

```
/**
 Obtain the view bound to the card (the view may be reused by other cards)
 @ return Bind view
 */
-(CubeView*)getBindView;
```

2.2.3.8. CubeCardConfig

This topic describes the properties of the Ant Cube Card configuration class.

Properties

templateId

```
/// The unique ID of the template. This parameter is required.
@property (nonatomic, strong) NSString *templateId;
```

version

```
/// Required. The version number of the template.
@property (nonatomic, strong) NSString *version;
```

data

```
/// Card data
@property (nonatomic, strong) NSDictionary *data;
```

width

```
/// The preset width of the card.
@property (nonatomic, assign) NSInteger width;
```

height

```
/// Card preset height
@property (nonatomic, assign) NSInteger height;
```

extOption

```
/// Card extension data
@property (nonatomic, strong) NSDictionary *extOption;
```

envData

```
/// Card environment variables
@property (nonatomic, strong) NSDictionary *envData;
```

layoutChangeListener

```
/// The card layout size change listener.
@property (nonatomic, weak) id<CCardLayoutChangeListener> listener;
```

2.2.3.9. CCardType

This topic describes the template source types of Ant Cube Card.

- **Statement**: `typedef enum{C_CARD_TYPE_NONE, C_CARD_TYPE_CACHE, C_CARD_TYPE_BUNDLE, C_CARD_TYPE_FILE, C_CARD_TYPE_CLOUD} CCardType;`
- **Description**: the result code of the card.
- **Enumeration Value:**

Enumeration Value	Description
C_CARD_TYPE_NONE	Invalid template
C_CARD_TYPE_CACHE	In-memory template
C_CARD_TYPE_BUNDLE	Template in local resource package
C_CARD_TYPE_FILE	Local physical storage template
C_CARD_TYPE_CLOUD	Cloud template

2.2.3.10. CCardLayoutChangeListener

This topic describes the method of the Ant Cube Card layout change notification listener class.

Method

onLayout

```
/**
 Card size change

 @ param size The new size of the card.
*/

- (void)onLayout:(CGSize)size;
```

2.2.3.11. CCardCallback

This article describes the method of the callback class created by Ant Cube Card.

Method

onLoaded

```
/**
 Card callback interface

 @ param card The instance of the card. The value is nil when the card fails to be crea
ted.
 @ param cardType The source of the card template.
 @ param config The configuration parameters for creating a card.
 @ param erroCode The error code.
*/

- (void)onLoaded:(CubeCard *)card cardType:(CCardType)cardType config:(CubeCardConfig *
)config erroCode:(CubeCardResultCode)erroCode;
```

2.2.3.12. CubeModuleProtocol

This article describes the properties of the Ant Cube Card implementation protocol `CubeModuleProtocol`.

Properties

cardUid

```
/// The unique ID of the card to be called.
@property (nonatomic, copy) NSString *cardUid;
```

CubeModuleMethodCallback

```
/// callback
typedef void (^CubeModuleMethodCallback) (id result);
```

2.2.3.13. CubeCardResultCode

This article describes the result code of the Ant Cube Card.

- **Statement**: `typedef enum{CubeCardResultSucc = 0, CubeCardResultNetworkError, CubeCardResultParamError, CubeCardResultNotExist, CubeCardResultContentInvalid}CubeCardResultCode;`
- **Description**: the result code of the card.
- **Enumeration Value:**

Enumeration Value	Description
CubeCardResultSucc = 0	The card request is successful
CubeCardResultNetworkError	Network errors
CubeCardResultParamError	Abnormal request parameters
CubeCardResultNotExist	The card does not exist
CubeCardResultContentInvalid	Card content is invalid

3.Client capabilities

3.1. Real Device Preview

3.1.1. Integrate into Android

This topic describes how to preview cards on a real device in an Android client.

Prerequisites

- You have enabled and integrated mPaaS.
- You have installed the AntCubeTool. For more information, see About AntCubeTool.
- You have completed the integration process as described in Getting Started.

Procedure

1. **Add the real-time preview dependency.**
   i. **Add the Cube Card-Developer Tool component.**
   ii. Add the third-party dependencies to the `build.gradle` file of the main project module. If a dependency conflict occurs, use your existing dependency versions.

```
dependencies {
    ......
    implementation "com.squareup.okhttp3:logging-interceptor:3.12.12"
    implementation 'org.jetbrains.kotlin:kotlin-stdlib-jdk7:1.3.72'
    implementation "com.squareup.okhttp3:okhttp:3.12.12"
    implementation 'com.squareup.picasso:picasso:2.5.2'
    implementation 'org.simple:androideventbus:1.0.5'
    .......
}
```

2. Run the following command in your project's directory to start the local debugging service:
   - macOS: `act prepare && act server`
   - Windows: `act prepare | act server`

   After you run the command, a QR code is generated in the terminal.
3. Start the client and scan the QR code to connect.

```
CubeCardDebug.openScanner(activity);
```

   Because an internal network connection is established, if the targetVersion is greater than 27, you need to downgrade it or add a configuration in the manifest.

   `android:usesCleartextTraffic="true"` `android:networkSecurityConfig="@xml/network_security_config"` and the corresponding `network_security_config.xml`.

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config cleartextTrafficPermitted="true">
        <domain includeSubdomains="true">${your_local_IP_address}</domain>
    </domain-config>
</network-security-config>
```

After you scan the QR code, a toast notification appears on the client: `Cube Socket: Connected`.

The terminal also shows that the device is connected.

4. **Preview**

   Modify the card's code and run `act build` to compile it. Then, run `act preview` to push the compiled content to the client.

3.1.2. Integrate real-device preview for iOS

This topic describes how to preview cards on a real device in an iOS client.

Prerequisites

- mPaaS has been activated and integrated.
- The AntCubeTool of Ant Cube Card has been installed. For more details, please refer to: About AntCubeTool.
- The integration process has been completed by referring to the Quick start.

Procedure

1. **Add the real-device preview dependency. The following example uses a pod.**

   i. Add the pod modules.

   ```
   mPaaS_pod "mPaaS_Cube"
   mPaaS_pod "mPaaS_Cube_Dev"
   ```

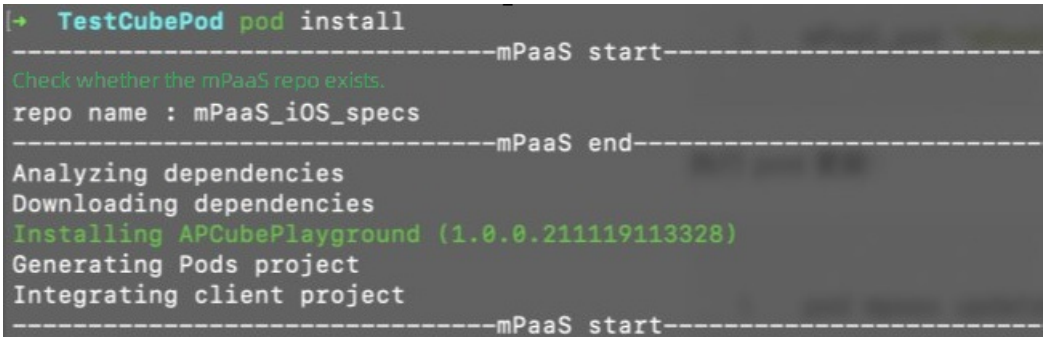
   > **Important**
   > - You must use mPaaS_Cube and mPaaS_Cube_Dev together.
   > - Before you publish the app, make sure to delete mPaaS_Cube_Dev.

   ii. Update the pod.

   ```
   pod mpaas update 10.2.3
   ```

iii. Run `pod install`.

```
pod install
```



iv. Add the dependency libraries required for the QR code scan preview.

```
#import <APCubePlayground/APCubePlayground.h>
#import <AudioToolbox/AudioToolbox.h>
#import <MPCubeAdapter/MPCubeAdapter.h>
```

v. Set the preview page.

> **Important**
> - Before you use the preview feature, ensure that the engine is initialized.
> - A custom extension module is invalid unless it is registered.

- If you use the default preview page, call the following code:

```
[[MPCubePreViewManager sharedInstance] setDefaultDebugPreview:true topOffset:0];
```

- If you define a custom preview page, register the `listener` and implement the `CubeCardPreviewListener Protocol`:

```
[APCubePlayground registPreViewListener:id<CubeCardPreviewListener>];

//  CubeCardPreviewListener
- (void)onPreViewFinish:(BOOL)rst cubeCard:(CubeCard*)cubeCard {

}
```

vi. Call the QR code scan preview method.

```
- (void)scan {
    [APCubePlayground launch];
    [APCubePlayground scan];
}
```

> **Important**
>
> The current app must have a `NavigationController`. Otherwise, the app will not respond after you call `scan`.

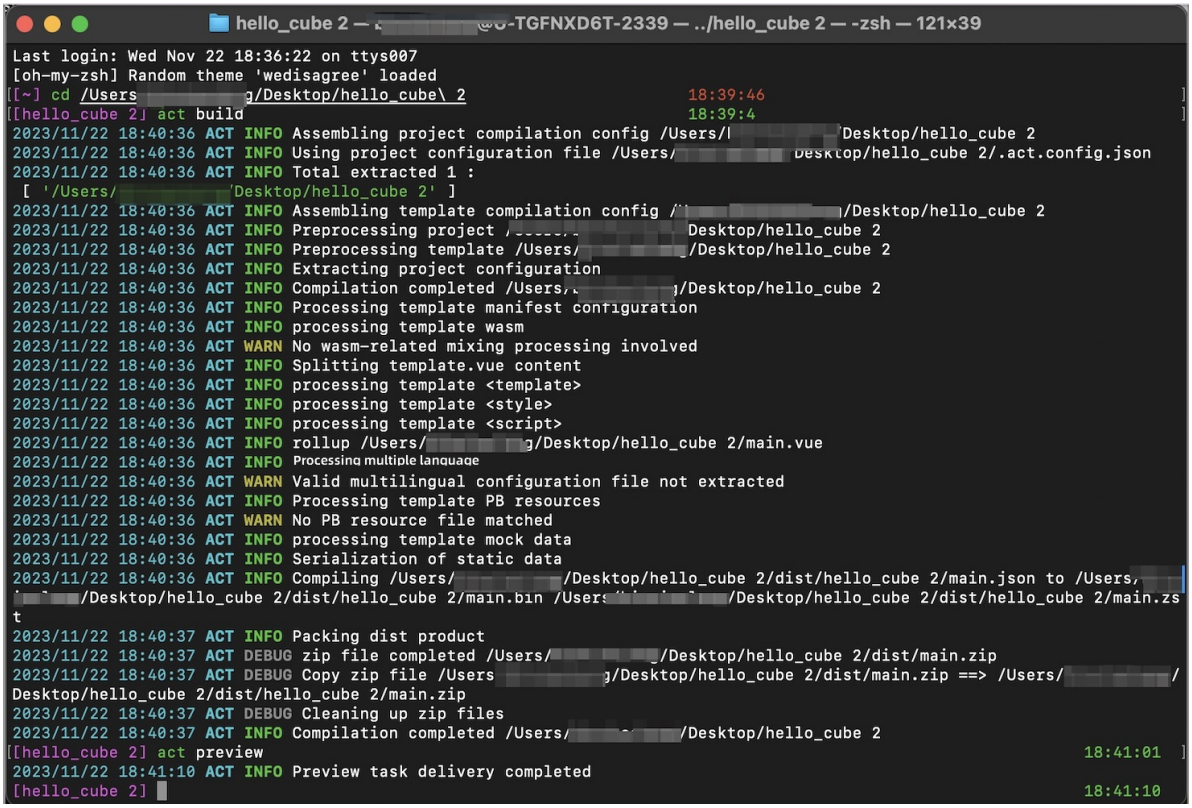
2. Use the command line to start the local debugging service. In the project path, run the command to start the service. The commands for macOS and Windows are as follows:
   - macOS: `act prepare && act server`
   - Windows: `act prepare | act server`

   After you run the command, a QR code is generated in the terminal.

3. Start the client and scan the QR code to establish a connection. The terminal prompts that the device is connected.

4. Preview the card.

   Open a new terminal window. Use the `cd` command to switch to the project folder and run `act build` to compile the project. Then, call `act preview` to push the compiled content to the client.



   - If you use the default preview page, the client automatically redirects to display the preview card.

- If you define a custom preview interface, the `CubeCardPreviewListener Protocol` callback is triggered. Your app can then retrieve the `CubeCard` instance and render it.

3.2. Engine Initialization Parameter

Initialize the engine

Android

```
CubeInitParam cubeInitParam = CubeInitParam.getDefault();
MP.init(this, MPInitParam.obtain().addComponentInitParam(cubeInitParam));
```

iOS

```
[[CubeService sharedInstance] initWithConfig:config];
```

Initialize engine parameters

Set the preset resource path

Android

```
CubeEngineConfig config = new CubeEngineConfig();
config.setResourcePath("cube");
cubeInitParam.setCubeEngineConfig(config)
```

iOS

```
 CubeEngineConfig* config = [[CubeEngineConfig alloc] init];
[config setBundlePath:mockBundlePath];// The path to local cards.
```

Set a custom JSAPI

Android

```
/**
 * Registers a Cube JSAPI.
 * @param cubeModuleModels
 */
public CubeInitParam setCubeModuleModels(Collection<CubeModuleModel> cubeModuleModels)
cubeInitParam.setCubeModuleModels(generateModuleModel())
```

iOS

```
NSDictionary *dic = @{@"MethodName": @"ImplementationClassName", @"abc":
@"MPDemoModule"};
[[[CubeService sharedInstance] getEngine] registerModules:dic];
```

Set custom tags

Android

```
/**
 * Registers a custom view (custom tag).
 * @param cubeWidgetInfos
 */
public CubeInitParam setCubeWidgetInfos(Collection<CubeWidgetInfo> cubeWidgetInfos)
cubeInitParam.setCubeWidgetInfos(generateWidget())
```

iOS

```
CubeWidgetInfo *widgetInfo = [CubeWidgetInfo new];
widgetInfo.tag = @"custom-widget";
widgetInfo.className = [MPDemoCustomCardView class];

NSMutableArray *widgetArray = [NSMutableArray array];
[widgetArray addObject:widgetInfo];
[[[CubeService sharedInstance] getEngine] registerWidgets:[NSArray
arrayWithArray:widgetArray]];
```

Set an image downloader

Android

```
cubeInitParam.setImageHandler(new ICKImageHandler() {
    @Override
    public String loadImage(String s, int i, int i1, Map<String, Object> map, LoadImage
Listener loadImageListener) {
        return null;
    }

    @Override
    public void cancel(String s) {

    }
```

iOS

You can use `CubeEngineConfig` to intercept image downloads and customize image parameters. The client must conform to the `CKImageHandler` proxy and implement the listener method.

```
// Properties of the CubeEngineConfig card engine class

/// The image handler. If this is empty, the default internal implementation is used. C
ustomize the handler.
@property (nonatomic, strong) id<CKImageHandler> imageHandler;
```

```
//  CKImageHandler.h
//  CubePlatform
//  Created by Joe on 2018/8/15.
//  Copyright © 2018 mQuick. All rights reserved.

#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

#ifndef CKImageHandler_h
#define CKImageHandler_h

extern NSString *const CKImageAppInstanceIDKey; // The AppInstanceID that corresponds t
o the triggered image loading.
extern NSString *const CKImagePageInstanceIDKey; // The PageInstanceID that corresponds
to the triggered image loading.
extern NSString *const CKImageInstanceOptionKey; // The instance option. Added for the
Falcon link.

typedef void(^CKImageHandlerCallback)(UIImage *image, NSError *error);

@protocol CKImageHandler <NSObject>

@required
/**
 @param url The image download URL.
 @param size The image size.
 @param option Extension parameters that depend on the image download library.
 @param callback The download callback. This callback is triggered after the download i
s complete.
 @return Returns the unique ID of the download task.
 */
- (NSString *)fetchImage:(NSString *)url size:(CGSize)size option:(NSDictionary *)optio
n callback:(CKImageHandlerCallback)callback;

@optional
/**
 @param fetchID The task ID, which is the return value of fetchImage.
 */
- (void)cancel:(NSString*)fetchID;

@end
```

```
@interface MPCubeManager () <CKImageHandler>

- (void)initEngine {
    CubeEngineConfig *engineConfig = [[CubeEngineConfig alloc] init];
    // Set the delegate.
    engineConfig.imageHandler = self;
    [[CubeService sharedInstance] initWithConfig:engineConfig];
}

- (NSString *)fetchImage:(NSString *)url size:(CGSize)size option:(NSDictionary *)optio
n callback:(CKImageHandlerCallback)callback {
    NSLog(@"Image URL: %@", url);
    NSLog(@"Image extension parameters: %@", option);

    return @"20211202";
}
```

Set an exception listener

Android

```
cubeInitParam.setExceptionListener(new CExceptionListener() {
    @Override
    public void onException(CExceptionInfo cExceptionInfo) {

    }
})
```

iOS

You can use `CubeEngineConfig` to listen for and catch frontend code exceptions. The client must conform to the `CExceptionListener` proxy and implement the listener method.

```
// Properties of the CubeEngineConfig card engine class

/// Exception listener
@property (nonatomic, strong) id<CExceptionListener> exceptionListener;
```

```
// CExceptionListener proxy class

//  CrystalExceptionProtocol.h
//  CubeCrystal
//  Created by hejin on 2021/9/10.

#import <Foundation/Foundation.h>
#import "CExceptionInfo.h"
#ifndef CExceptionListener_h
#define CExceptionListener_h

@protocol CExceptionListener <NSObject>

@required

/**
 Exception listener
 @param info The exception information.
 */
- (void)onException:(CExceptionInfo *)info;

@end
```

```
@interface MPCubeManager () <CExceptionListener>

- (void)initEngine {
    CubeEngineConfig *engineConfig = [[CubeEngineConfig alloc] init];
    // Set the delegate.
    engineConfig.exceptionListener = self;
    [[CubeService sharedInstance] initWithConfig:engineConfig];
}

// Implement the listener method to print listener information.
- (void)onException:(CExceptionInfo *)info {
    NSLog(@"Exception type: %lu", (unsigned long)info.type);
    NSLog(@"Exception message: %@", info.message);
    NSLog(@"Exception card ID: %@", info.cardUid);
    NSLog(@"Exception information extension parameters: %@", info.extraInfo);
}
```

Set a log listener

Android

```
cubeInitParam.setLogHandler(new ICKLogHandler() {
    @Override
    public void log(Context context, int i, String s, String s1, Throwable throwable) {
        // Android log
    }

    @Override
    public void jsLog(Context context, String s) {
        // JS log
    }
```

iOS

```
CubeEngineConfig* config = [[CubeEngineConfig alloc] init];
config.logHandler = self;
[[CubeService sharedInstance] initWithConfig:config];
#pragma mark - CKLogHandler
- (void)nativeLog:(NSString *)log type:(CKLogType)type {
}
- (void)jsLog:(NSString *)log {
}
```

Set an initialization callback

Android

```
cubeInitParam.setCubeInitCallback(new CubeInitParam.CubeInitCallback() {
    @Override
    public void onInit() {

    }

    @Override
    public void onError(Exception e) {

    }
})
```

iOS

Not supported.

Set the RpcHandler

Android

```
CubeInitParam cubeInitParam
= CubeInitParam.getDefault()
...
.setRpcHandler(new ICRpcHandler() {
    @Override
    public CKTemplateInfo.CKTemplateInfoResponse
fetchTemplateInfo(List<CKTemplateInfo.CKTemplateInfoRequestParam> list) {
        // CKRpcHandler() is the basic mPaaS implementation. Customers can extend it in
ICRpcHandler.
        return null;
    }
})
```

iOS

```
// 1. Set the rpcHandler.
CubeEngineConfig *engineConfig = [[CubeEngineConfig alloc] init];
engineConfig.rpcHandler = self;
[[CubeService sharedInstance] initWithConfig:engineConfig];

// 2. Implement CTemplateInfoProtocol. For the return value, see the log printed during
the default RPC request:
// rpc: templateInfo rpc result. You can also use the internal logic of the software de
velopment kit (SDK) when you implement your own RPC.
- (id)getTemplateInfoFromRpc:(NSArray *)params {

   // 1. Use the custom RPC logic.

    // 2. Use the default internal RPC logic of the SDK.
    Class clazz = NSClassFromString(@"CrystalTemplateInfoHandler");
    if (clazz) {
        id<CTemplateInfoProtocol> handler = [[clazz alloc] init];
        return [handler getTemplateInfoFromRpc:params];
    }

    return @{};
}
```

3.3. Render Cards

Assemble card information

Android

Create the configuration and set the parameters.

```
/**
 * Assemble card configuration.
 * @return
 */
private CubeCardConfig assembleCubeCardConfig(){
    // Create card configuration.
    CubeCardConfig cardConfig = new CubeCardConfig();
    // Card ID created on the backend.
    cardConfig.setTemplateId("hello_cube");
    // Card version.
    cardConfig.setVersion("1.0.0.0");
    // Card width. Set to the screen width.
    cardConfig.setWidth(MFSystemInfo.getPortraitScreenWidth());
    return cardConfig;
}
```

iOS

Create the configuration and set the parameters.

```
- (void)createCubeConfig {
                // Set card parameters.
    CubeCardConfig *cardConfig = [[CubeCardConfig alloc] init];
    // Card version.
    [cardConfig setVersion:@"1.0.0.0"];
    // Card ID created on the backend.
    [cardConfig setTemplteId:@"987654321"];
    // Preset width.
    [cardConfig setWidth:MP_Screen_Width];
    // Preset height.
    [cardConfig setHeight:CGFLOAT_MAX];
}
```

Create a card

Android

Request the card based on its configuration. The card engine retrieves the card template from the server. Use the `createCard` method to request a single card and the `createCards` method to request multiple cards.

```
/**
 * Request card information.
 * @param cardConfig
 */
private void requestCubeCard(final CubeCardConfig cardConfig){
    // Request the card.
    CubeService.instance().getEngine().createCard(cardConfig, new CCardCallback() {
        @Override
        public void onLoaded(final CubeCard cubeCard, CCardType cardType,
CubeCardConfig cubeCardConfig, CubeCardResultCode resultCode) {
            renderCubeCard(cubeCard,cardType,cubeCardConfig,resultCode);
        }
    });
}
```

iOS

Request the card based on its configuration. The card engine retrieves the card template from the server. Use the `createCard` method to request a single card and the `createCards` method to request multiple cards.

```
[[[CubeService sharedInstance] getEngine] createCard:cardConfig callback:self];
```

Render the card

Android

After obtaining the card information, a card View is generated and rendered on the main thread. This step also requires exception handling to prevent situations where card information is not successfully obtained.

```
/**
 * Render the card.
 * @param cubeCard
 * @param cardType
 * @param cubeCardConfig
 * @param resultCode
 */
private void renderCubeCard(final CubeCard cubeCard, CCardType cardType, CubeCardConfig
cubeCardConfig, CubeCardResultCode resultCode) {
    if (resultCode == CubeCardResultCode.CubeCardResultSucc) {
        // Must run on the main thread.
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                mCard = cubeCard;
                // Create the card View.
                CubeView view =
CubeService.instance().getEngine().createView(FastActivity.this);
                // Add to the outer ViewGroup.
                mWrapperLl.addView(view);
                // Render the card.
                cubeCard.renderView(view);
            }
        });
        MPLogger.info(TAG, "succ " + cubeCardConfig.getTemplateId() + " style " +
cardType);
    } else {
        MPLogger.info(TAG, "fail " + cubeCardConfig.getTemplateId() + " style " +
cardType + " error " + resultCode);
    }
}
```

iOS

After obtaining the card information, a card View is generated and then rendered on the main thread. Exception handling is necessary at this stage to prevent situations where card information is not successfully obtained.

```
#pragma mark - CrystalCardCallback
- (void)onLoaded:(CubeCard *)card cardType:(CCardType)cardType config:(CubeCardConfig *
)config erroCode:(CubeCardResultCode)erroCode {
    if (!card) {
        NSString *errMsg = [NSString stringWithFormat:@"Creation failed:
templteId=%@,style=%d, error=%d", [config templteId], cardType, erroCode];
        NSLog(@"Error message:%@", errMsg);
        return;
    }

    NSLog(@"Creation successful succ %@ style %d error %d", [config templteId],
cardType, erroCode);

    dispatch_async(dispatch_get_main_queue(), ^{
        //Refresh the UI on the main thread.
        CubeView *view = [[[CubeService sharedInstance] getEngine] createView];
        CGSize size = [card getSize];

        if (![view isEqual:[NSNull null]]) {
            [view setFrame:CGRectMake(0, 0, MP_Screen_Width - 40, size.height)];
            [card renderView:view];
        }

        [self.view addSubView:view];
    });
}
```

Notification Card Lifecycle

Notification didAppear

Android

```
mCard.notifyState(CCardState.CCardStateAppear);
```

iOS

```
[card notifyState:CCardStateAppear];
```

Notification disApear

Android

```
mCard.notifyState(CCardState.CCardStateDisappear);
```

iOS

```
[card notifyState:CCardStateDisappear];
```

Notify frontend

Android

```
mCard.notifyState(CCardState.CCardStateForeGround);
```

iOS

```
[card notifyState:CCardStateForeGround];
```

Notify backend

Android

```
mCard.notifyState(CCardState.CCardStateBackGround);
```

iOS

```
[card notifyState:CCardStateBackGround];
```

Release the card

Android

After you finish using the card, release its memory resources. Typically, call this in the page's `onDestroy` lifecycle method.

```
/**
 * Release card resources.
 */
private void releaseCubeCard(){
    if (mCard != null) {
        mCard.recycle();
    }
    int chidrenCount = mWrapperLl.getChildCount();
    for (int i = 0; i < chidrenCount; i++) {
        if (mWrapperLl.getChildAt(i) instanceof CubeView) {
            ((CubeView) mWrapperLl.getChildAt(i)).destroy();
        }
    }
    mWrapperLl.removeAllViews();
}
```

iOS

After you finish using the card, release its memory resources. Typically, call this in the page's `dealloc` lifecycle method or destroy it manually.

```
- (void)destoryCubeService {
    //Destroy the card engine.
    if (![[[CubeService sharedInstance] getEngine] isEqual:[NSNull null]]) {
        [[CubeService sharedInstance] destroyEngine];
    }

    //Delete the card view.
    [self.view removeAllSubviews];
}
```

3.4. Preset Cards

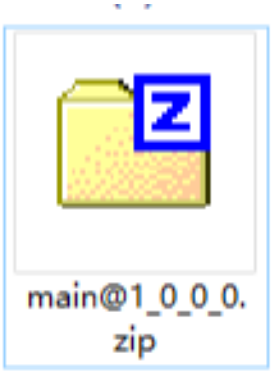
Download card resources

Method 1: Download the card file from the console. Click **Download bin file**. The downloaded file is a .zip file.

Method 2: Package the `dist/main.zip` file from your local card project.

Name the resource

1. Copy the downloaded file to resource directory.
2. Rename the downloaded file to `Card ID@Card Version`. For example, `main@1_0_0_0.zip`.



> **Note**
>
> In the future, the console will directly provide fully named zip packages, eliminating the need for users to manually modify the names; users can simply add them to assets.

Preset resources

Android

1. Set the properties of the `CubeEngineConfig` card engine class.

```
/**
* The path to the local resource package that stores the templates.
*
* @param resourcePath - Sets the resource file path.
*/
public void setResourcePath(String resourcePath) {
                this.resourcePath = resourcePath;
}
```

2. Set the path to the local Ant Cube Card assets.

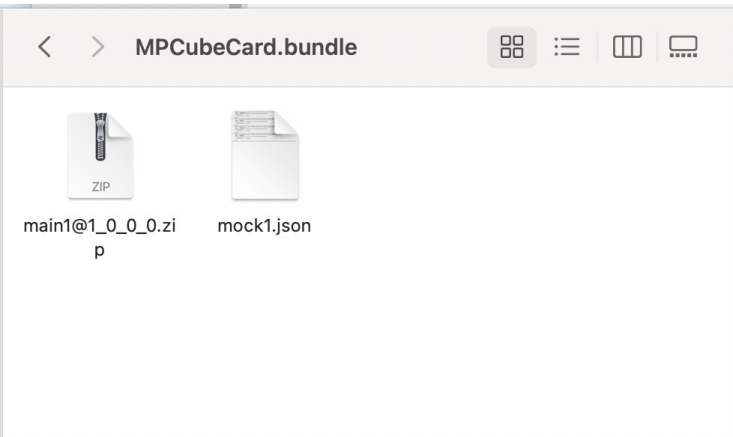
```
// Purge card data.
CubeService.instance().destroyEngine();
CubeEngineConfig config = new CubeEngineConfig();
// Set the path.
config.setResourcePath("The path where your resources are located");
CubeService.instance().initEngine(config, MainApplication.getApplication());
```

3. Place the .zip package in the specified path within assets directory.

iOS

1. Add the downloaded file to the Bundle.

   Create a new Bundle file. Compress the downloaded .bin file into a .zip package and name it `Card ID@Card Version`. For example, `main1@1_0_0_0.zip`. The card requires data from a JSON file. You can either use a built-in file or dynamically request the data by calling an API. Choose the data source method based on your needs.



2. Import the Bundle into the project and pass the Bundle path when initialize the engine.

```
- (void)initEngine {
    // The Bundle path for the local resource template.
    NSString *bundlePath = [NSString stringWithFormat:@"%@/%@", [[NSBundle mainBundle
] resourcePath], @"MPCubeBundle.bundle"];
    CubeEngineConfig *config = [[CubeEngineConfig alloc] init];
    [config setBundlePath:bundlePath];
    [[CubeService sharedInstance] initWithConfig:config];
}
```

3.5. Call client methods from a card

Custom JSAPI

Android

```
public class CustomCubeModule extends CubeModule {
    private static final String TAG = CustomCubeModule.class.getSimpleName();

    // Annotation. uiThread indicates whether the callback is in the main process.
    @JsMethod(uiThread = true)
    public void cubeToClient(final CubeJSCallback callback) {
        // Send a callback to the card.
        if (callback != null) {
            callback.invoke("cubeToClient callback data: " +
System.currentTimeMillis());
        }
    }
}
```

iOS

1. In the project, create an `NSObject` object and import the `<CubeModuleProtocol>` protocol.

   ```
   #import <Foundation/Foundation.h>
   #import <CubeCrystal/CubeModuleProtocol.h>

   NS_ASSUME_NONNULL_BEGIN

   @interface MPCubeModule : NSObject <CubeModuleProtocol>

   @end

   NS_ASSUME_NONNULL_END
   ```

2. Implement the method based on the template convention.

```
#import "MPCubeModule.h"
#import <CubeBridge/CKJSDefine.h>

@implementation MPCubeModule

// cardUid is a property of the module. When the module API is called, cardUid identi
fies which card initiated the call. This cardUid is the same as the cardUid in CubeCa
rd.
@synthesize cardUid;

// Configure the conventional method.
CK_EXPORT_METHOD(@selector(cubeToClient:))

- (void)cubeToClient:(CubeModuleMethodCallback)callback {
    // Execute the cubeToClient operation here.
    if (callback) {
        // The parameter to be passed back to the template.
        callback(@"cubeToClient");
    }
}

@end
```

Register the JSAPI

Android

The first parameter specifies the type, the second specifies the full path of the custom module, and the third specifies the name of the method to call. Do not obfuscate the full path or the method name.

```
Collection<CubeModuleModel> cubeModuleModels = new LinkedList<>();
cubeModuleModels.add(new CubeModuleModel("custom", CustomCubeModel.class.getName(), new
String[]{"cubeToClient"}));
CubeService.instance().getEngine().registerModule(cubeModuleModels, null);
```

iOS

```
- (void)registerModules {
    [MPCubeManager shareManager];

    // Register the custom module. The key is the module name (by convention with the s
erver-side), and the value is the client class name.
    NSDictionary *dic = @{@"custom": @"MPCubeModule"};
    [[[CubeService sharedInstance] getEngine] registerModules:dic];
}
```

Call the JSAPI

```
<script>
    // Register the module.
    const navigator = requireModule("custom");
    export default {
        data: {
            message: 'Hello Cube 1'
        },
        beforeCreate() {
            this.message = 'Hello Cube 2'
        },
        didAppear() {

        },
        methods: {
            onClick() {
                // Call the client method.
                navigator.cubeToClient(event=>{
                    this.message = event;
                })
            }
        }
    }
</script>
```

3.6. Client Calls Card Methods

JS implementation method

```
<template>
    <div class="root">
        <text class="message" :value="message" @click="onClick()"></text>
        <custom-widget ref="widget" class="custom" :value="message2" @on-
WidgetToCube="clientToCube(param)">
        </custom-widget>
    </div>
</template>

<script>

    export default {
        data: {
            message : 'This is the card text. Click me to call the custom widget.',
            message2 : 'This is the value passed to the custom widget.'
        },
        beforeCreate() {
            this.message = 'This is the card text. Click me to call the custom widget.'
            this.message2 = 'This is the value passed to the custom widget.'
        },
        didAppear() {

        },
        methods: {
```

```
            onClick() {
                console.info('invoke on-click event');
                this.$refs.widget.cubeToWidget({
                    'auto':"This is the parameter sent to the custom widget method."
                },ret=>{
                    this.message = "Received callback from the custom widget method. Par
ameter: "+JSON.stringify(ret);
                });
            },
            // Native method to call a custom tag (call JS)
            clientToCube(param){
                console.info(param.p1)
                this.message2 = "Received event callback from the custom widget. Paramet
er: "+JSON.stringify(param)
            }
        }
    }
</script>

<style>
    .root {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: white;
        width: 100%;
        height: 800rpx;
    }
    .custom {
        color: black;
        font-size: 20rpx;
        width: 100%;
        height: 600rpx;
    }
    .message {
        color: black;
        font-size: 20rpx;
        width: 100%;
        height: 200rpx;
    }
</style>
```

Call JS methods from a client

Android

Create a `CubeCard` instance `mCard` and call the following method.

```
mCard.callJsFunction(String methodName, final CJSFunctionCallback callback, Object... p
arams)
```

iOS

In the card loading delegate method, obtain the card instance, call the corresponding JS method, and send the data.

```
#pragma mark---CrystalCardCallback
- (void)onLoaded:(CubeCard *)card cardType:(CCardType)cardType config:(CubeCardConfig *
)config erroCode:(CubeCardResultCode)erroCode {
    if (!card) {
        NSString *errMsg = [NSString stringWithFormat:@"Creation failed:
templteId=%@,style=%d, error=%d", [config templteId], cardType, erroCode];
        NSLog(@"Error message:%@", errMsg);
        return;
    }

    NSLog(@"Creation successful succ %@ style %d error %d", [config templteId],
cardType, erroCode);

    dispatch_async(dispatch_get_main_queue(), ^{
        NSString *text = @"Client calls card JS method to pass parameters: Hello
World";
        if (![text isEqualToString:@""]) {
            NSArray *valueArray = @[text];
            // Call the card's JS method and pass the parameters.
            [card callJsFunction:@"clientToCube" arguments:valueArray];
        }
    });
}
```

3.7. Custom Native Tags

Implement custom tags on the client

Android

Inherit from `CCardWidget` and implement its abstract methods, the main one being `onCreateView`, where the corresponding custom View is returned.

```
public class CustomCubeWidget extends CCardWidget {
    private static final String TAG = "CustomCubeWidget";
    private Map<String, Object> createMap;
    private TextView root;

    public CustomCubeWidget(Context context) {
        super(context);
    }

    @Override
    public View onCreateView(Map<String, Object> params, int width, int height) {
        MPLogger.debug(TAG, "onCreateView:" + width + "," + height);
        createMap = params;
        for (String key : params.keySet()) {
            MPLogger.debug(TAG, key + ":" + params.get(key));
        }
        if (root == null) {
```

```
            root = new TextView(getContext());
            setText(root, params.get("value") + "");
            root.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Map<String, Object> eventParams = new HashMap<>();
                    eventParams.put("p1", "widget:" + System.currentTimeMillis());
                    CubeService.instance().getEngine().sendEvent(createMap, "on-WidgetTo
Cube", eventParams);
                }
            });
        }
        return root;
    }

    @Override
    public void onReuse(Map<String, Object> params, int width, int height) {
        MPLogger.debug(TAG, "onReuse");
    }

    @Override
    public void onUpdateData(Map<String, Object> params) {
        MPLogger.debug(TAG, "onUpdateData");
    }

    @Override
    public void onRecycleAndCached() {
        MPLogger.debug(TAG, "onRecycleAndCached");
    }

    @Override
    public boolean canReuse() {
        MPLogger.debug(TAG, "canReuse");
        return false;
    }

    @Override
    public void onDestroy() {
        MPLogger.debug(TAG, "onDestroy");
    }

    @JsMethod
    public void cubeToWidget(JSONObject param, CubeJSCallback callback) {
        if (root != null) {
            setText(root, "Received a custom method call from the card. Parameter: " +
param.toJSONString());
        }
        callback.invoke("cubeToWidget callback data: " + System.currentTimeMillis());
    }

    private void setText(TextView tv, String msg) {
        tv.setText("I am a native component. Click me to notify the card of the event c
allback, " + msg);
    }
}
```

```
}
```

iOS

Inherit `CCardWidget` and implement its protocol methods. The `onCreateView` and `updateData` methods must be implemented. Other methods are optional.

```
#import "CustomCardView.h"

@interface CustomCardView ()

@property (nonatomic, strong) UILabel *titleLabel;

@end

@implementation CustomCardView

// Required
/**
 * Creates the UI. This represents the view UI of the component to be displayed.
 *
 * @param data  A map containing data declared by the component on the card. This includes styles, events, and properties. The corresponding keys are DATA_KEY_STYLES, DATA_KEY_EVENTS, and DATA_KEY_ATTRS.
 * @param size  The size (width and height) of the view.
 * @return An active UIView.
 */
- (UIView *)onCreateView:(NSDictionary *)data size:(CGSize)size {
    NSLog(@"Print data:%@", data);

    UIView *customView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, size.width, size.height)];
    customView.backgroundColor = [UIColor redColor];

    self.titleLabel = [[UILabel alloc] initWithFrame:CGRectMake(0, 0, size.width, 30)];
    // The data source is the provided data.
    self.titleLabel.text = @"A";
    [customView addSubview:self.titleLabel];

    return customView;
}

// Required
/**
 * Updates the component data.
 * @param data
 */
- (void)onUpdateData:(NSDictionary *)data {
    NSLog(@"Print data:%@", data);
}

/**
 *  Prepares data for an existing control retrieved from the reuse pool before it is used.
 *  @param data Initialization data
```

```
    @param data Initialization data.
 *  @param size The size of the component when it is reused.
 */
- (void)onReuse:(NSDictionary *)data size:(CGSize)size {
    NSLog(@"Print data:%@", data);
    NSLog(@"Print size:%@", size);

    // The data source is the provided data.
    self.titleLabel.text = @"B";

}

/**
 *  Specifies whether the component can be reused.
 *  To improve efficiency, extension components can support reuse. For example, if a cu
stom tag component is removed from the current view due to a data update, the component
is placed in a reuse pool if it supports reuse. The next time the component is displaye
d, it is retrieved directly from the reuse pool.
 * @return YES: Reuse. NO: Do not reuse.
 */
- (BOOL)canReuse {
    return YES;
}

/**
 *  Cleans up a reusable component. If the component supports reuse (canReuse returns t
rue), the onRecycleAndCached method is called after the component enters the reuse pool
. This indicates that the component is off-screen and in the cache, and its resources n
eed to be cleared.
 */
- (void)onRecycleAndCache {
    // Clear the SubView content on the custom view.
    self.titleLabel.text = @"";
}
```

Register custom tags on the client

Android

The first parameter specifies the custom tag, which you can define. On the card, this tag is enclosed in angle brackets (<>). We recommend adding a prefix to prevent conflicts with the built-in tags of the dynamic card. The second parameter is the full path of the custom tag implementation class. Ensure that this class is not obfuscated.

```
Collection<CubeWidgetInfo> widgetInfos = new LinkedList<>();
widgetInfos.add(new CubeWidgetInfo("custom-widget", CustomCubeWidget.class.getName()));
CubeService.instance().getEngine().registerWidgets(widgetInfos);
```

iOS

The first parameter specifies the custom tag, which must be consistent with the tag used in the card. On the card, this tag is enclosed in angle brackets (<>). We recommend adding a prefix to prevent conflicts with the built-in tags of the dynamic card. The second parameter is the class name of the custom tag implementation class.

```
CubeWidgetInfo *widgetInfo = [CubeWidgetInfo new];
widgetInfo.tag = @"custom-widget";
widgetInfo.className = [CustomCardView class];

NSMutableArray *widgetArray = [NSMutableArray array];
[widgetArray addObject:widgetInfo];
[[[CubeService sharedInstance] getEngine] registerWidgets:[NSArray
arrayWithArray:widgetArray]];
```

Call custom tags from the card

To call a custom tag, specify the tag name that you defined. In this example, the tag is `custom-widget`.

```
<template>
    <div class="root">
        <text class="message" :value="message" @click="onClick()"></text>
        <custom-widget ref="widget" class="custom" :value="message2" @on-
WidgetToCube="clientToCube(param)">
        </custom-widget>
    </div>
</template>

<script>

    export default {
        data: {
            message : 'This is the text of the card. Click me to call the custom
component.',
            message2 : 'This is the value passed to the custom component.'
        },
        beforeCreate() {
            this.message = 'This is the text of the card. Click me to call the custom
component.'
            this.message2 = 'This is the value passed to the custom component.'
        },
        didAppear() {

        },
        methods: {
            onClick() {
                console.info('invoke on-click event');
                this.$refs.widget.cubeToWidget({
                    'auto':"This is the parameter sent to the custom component's method.
"
                },ret=>{
                    this.message = "Received the callback from the custom component's me
thod. Parameters: "+JSON.stringify(ret);
                });
            },
            // Handles the event callback from the custom component.
            clientToCube(param){
                console.info(param.p1)
                this.message2 = "Received the event callback from the custom component.
```

```
Parameters: "+JSON.stringify(param)
            }
        }
    }
</script>

<style>
    .root {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: white;
        width: 100%;
        height: 800rpx;
    }
    .custom {
        color: black;
        font-size: 20rpx;
        width: 100%;
        height: 600rpx;
    }
    .message {
        color: black;
        font-size: 20rpx;
        width: 100%;
        height: 200rpx;
    }
</style>
```

Methods for calling custom tags from the card

Card call

```
this.$refs.widget.cubeToWidget({
  'auto':"I am the parameter sent to the custom component method"
},ret=>{
  this.message = "Received the callback from the custom component method. Parameter: "+
JSON.stringify(ret);
});
```

Client reception

Android

```
@JsMethod
public void cubeToWidget(JSONObject param, CubeJSCallback callback) {}
```

iOS

```
// Configure the method based on convention.
CK_EXPORT_METHOD(@selector(cubeToWidget:callBack:))

// Custom tag calls a native method.
- (void)cubeToWidget:(NSDictionary *)params callBack:(CubeModuleMethodCallback)callback
{

}
```

Client-side callbacks for notification cards

Client notification

Android

```
CubeService.instance().getEngine().sendEvent(Map<String, Object> componentData, String
eventName, @Nullable Map<String, Object> eventParams)
```

iOS

```
/**
 * self.data: The data from the onCreateView method.
 * eventName: The method name on the card, for example, @"on-WidgetToCube".
 * eventParams: The parameters passed to the card, for example, @{@"p1":@"Custom tag ca
lls a cube method"}.
 */
[[[CubeService sharedInstance] getEngine]
     sendEvent:self.data
     eventName:eventName
     eventParams:eventParams];
```

Card receives notification

The card can use `@eventName` to accept notifications from the client. For more information, see the following code example.

```
<custom-widget ref="widget" class="custom" :value="message2" @on-
WidgetToCube="clientToCube(param)">
```

3.8. Custom fonts for cards

Set fonts

Android

Set the font after the card is initialized.

```
HashMap<String, String> fonts = new HashMap<>();
fonts.put("AlipayNumber", "local://antui_res/AlipayNumber.ttf");
CubeService.instance().getEngine().loadCustomFonts(fonts);
```

- The key in the map corresponds to the value of the card's `font-family`.
- The value in the map is the path of the font resource in your assets. The path must start with `local://`. For example, `local://antui_res/AlipayNumber.ttf` indicates that the resource path is `assets/antui_res/AlipayNumber.ttf`.

iOS

Set the font after the card is initialized.

```
    NSMutableDictionary *fontMap = [[NSMutableDictionary alloc] init];
    NSString *alipayNumberFont = [[NSBundle mainBundle]
pathForResource:@"AlipayNumber.ttf" ofType:nil];
    NSString *dncFont = [[NSBundle mainBundle] pathForResource:@"DNC57.ttf"
ofType:nil];
    NSURL *aliPayFontUrl = [NSURL URLWithString:alipayNumberFont];
    NSURL *dncFontUrl = [NSURL URLWithString:dncFont];
    [fontMap setObject:aliPayFontUrl forKey:@"AlipayNumber"];
    [fontMap setObject:dncFontUrl forKey:@"DNC57"];
    [[[CubeService sharedInstance] getEngine] loadFonts:fontMap];
```

Use fonts in cards

```
<template>
    <div class="root">
        <text class="message" :value="message" @click="onClick()"></text>
        <text class="message2" :value="message" @click="onClick()"></text>
    </div>
</template>

<script>
export default {
    data: {
        message: 'Hello Cube 1'
    },
    beforeCreate() {
        this.message = '0123456789'
    },
    didAppear() {

    },
    methods: {
        onClick() {
            this.message = "aaa"
        }
    }
}
</script>

<style>
.root {
    display: flex;
    flex-direction: column;
    align-items: center;
    background-color: white;
    width: 100%;
    height: 400rpx;
}

.message {
    color: black;
    font-size: 80rpx;
}

.message2 {
    font-family: AlipayNumber;
    color: black;
    font-size: 80rpx;
}
</style>
```

To use a font, add the `font-family` parameter to the CSS style. The value of this parameter is the name of a font that is preset on the client.

3.9. Client Sends Notification to Card

Register a notification listener on the card side using JSAPI

Import dependencies

```
const mp = requireModule("mpaas_jsapi");
```

postNotification

Sends a notification.

```
mp.postNotification(
  {
    name:'TEST_EVENT',
    data: {
      hello: 'hello world'
    }
  },
  res=>{

  }
);
```

addNotifyListener

Registers a notification listener in the `didAppear` lifecycle.

```
   didAppear() {
        mp.addNotifyListener(
            {
                name: 'TEST_EVENT'
            },
            res => {
                this.clientToCube(res)
            }
        )
    },
```

removeNotifyListener

Removes a notification listener.

```
didDisappear() {
        mp.removeNotifyListener(
            {
                name: 'TEST_EVENT'
            },
            res => {
                console.info(JSON.stringify(res))
            }
        )
    },
```

> **Note**
>
> The client performs verification, and the notification listener will be automatically released after calling `MPCube.recycleCard` .

Client sends notification

Android

```
MPCube.postNotification(String notifyName,JSONObject notifyParams);
```

iOS

```
[[NSNotificationCenter defaultCenter] postNotificationName:notifyName object:nil userIn
fo:notifyParams];
```

3.10. Set Card to Gray

Introduction

You can set cards to gray, which switches their display from color to gray. This feature is controlled by the client and requires no special operations on the card.

Android

After you retrieve the card in the `onLoaded` method, you can perform operations on the `CubeCard` .

```
cubeCard.setGrayFilter(isGray);
```

iOS

After you retrieve the card in the `onLoaded` method, you can perform operations on the `CubeCard` .

```
[cubeCard setGrayFilter:true];
```

3.11. Query Local Card Information

Introduction

Call `findCachePath` to retrieve all cached card information for the current card ID. Each subfolder in this path is a local version of the card. For example, the subfolder `cachePath/1_0_0_0` corresponds to version `1.0.0.0`. Note that you need to replace (`_`) in the folder with (`.`).

APIs

Android

```
MPCube.findCachePath(cardId)
```

iOS

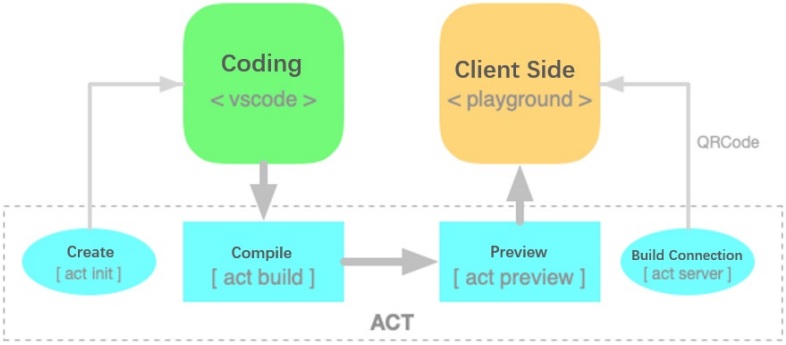
```
/**
 Gets the cache path for the template ID. Returns nil if no cache exists.
 @param templateId The template ID.
 */
- (nullable NSString *)findCachePath:(NSString *)templateId;

Example
NSString *cachePath = [cubeEngine findCachePath:@"cardId"];
```

4.Development Tools

4.1. ACT 2.0

4.1.1. About AntCubeTool

AntCubeTool (ACT) is a command line tool for developing, debugging, and previewing Ant Cube Card. This article provides a basic introduction to the tool.



ACT is available for macOS and Windows. The requirements for each platform are as follows:

- macOS
  - macOS 11.0 or later.
  - Command line: You can use the macOS Terminal or other command line tools.
- Windows
  - Must be at least Windows 10 and must be a 64-bit system.
  - Command line: Please use the Windows PowerShell command-line tool.

4.1.2. Install AntCubeTool

This topic describes how to install AntCubeTool.

Installation

To install ACT, run the `cnpm install xcube -g` or `cnpm install xcube@2.3.0-en.1 -g` command. After xcube is installed, you can run the `act` command on the command line.

```
cnpm install xcube -g
```

> **Important**
>
> Do not use sudo for the installation.

4.1.3. Use AntCubeTool

This topic describes the `act` commands for common scenarios in AntCubeTool (ACT).

Create a project

`act init` : Creates a project.

> **Note**
>
> The name fields in the configuration are used to create files or folders and can only contain letters, numbers, underscores (_), and hyphens (-).

```
➜  ~ act init -h
Usage: act init [options]

Initializes a Cube application in the current folder.

Options:
  -h, --help        display help for command
```

Start the service

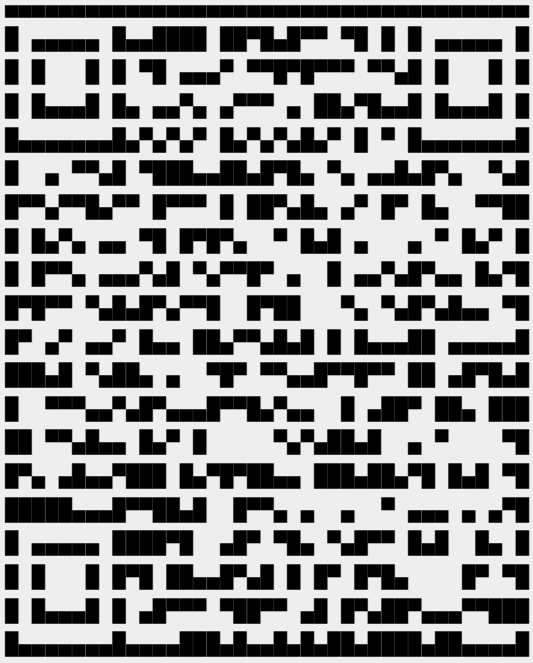
`act prepare` : Cleans up historical ports.

The `act server` Enables the connection listener.

```
➜  ~ act prepare --help
Usage: act prepare [options]

Prepares the environment.

Options:
  -h, --help  display help for command

➜  ~ act server --help
Usage: act server [options]

Starts the communication server.

Options:
  -h, --help  display help for command
```

On macqOS, run these two commands in the same terminal session. After running the commands, do not close the terminal window until the service has started. When the service starts, the following information is displayed. To run other `act` commands, you must open a new terminal window.

```
➜  ~ act prepare && act server
2021-1-7 16:38:25 [ACT] [INFO] cmd : kill -9 $(lsof -t -i:9001)
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec ... or kill -l [sig
spec]
2021-1-7 16:38:25 [ACT] [INFO] Port cleared.
2021-1-7 16:38:26 [ACT] [INFO] Starting service [30.30.XXX.XXX:9001]
2021-1-7 16:38:26 [ACT] [INFO] Communication channel is ready. Scan the QR code or ente
r the port number to connect.
2021-1-7 16:38:26 [ACT] [INFO]
```



```
2021-1-7 16:38:26 [ACT] [INFO] Routing module started.
2021-1-7 16:38:26 [ACT] [INFO] Service started. Do not close the terminal window.
```

Establish a connection

Establish a connection using the **Ant Dynamic Card-Developer Tool**. For more information, see:

- Android real device preview
- iOS real device preview

Compile the project

`act build` : Compile the project.

```
➜  ~ act build --help
Usage: act build [options] [path]

Compiles the card application in the specified path. If no path is specified, the curre
nt path is used.

Options:
  -p, --path [v]  The path to compile. Compatible with older tools. [Deprecated: Use bu
ild [path] directly.]
  -a, --all <v>   Batch compile. Compatible with older tools. [Deprecated: Use build [p
ath] --batch.]
  --batch         Specifies whether to use batch processing mode. (default: false)
  --watch         Specifies whether to enable real-time compilation. Not supported in b
atch processing mode. (default: false)
  -h, --help      display help for command
```

When the `--watch` parameter is configured, the real-time compilation is enabled, then the ACT will monitor file changes throughout the entire project directory and automatically trigger compilation when a file is saved.

> **Important**
>
> When real-time compilation is enabled, do not close the terminal window.

Preview the project

Preview the project using the **Ant Cube Card-Developer Tool**. For more information, see:

- Android real device preview
- iOS real device preview

Real-time preview

Perform a real-time preview using the **Ant Cube Card-Developer Tool**. For more information, see:

- Android real device preview
- iOS real device preview

View the current connection QR code

`act qrcode`: Displays the QR code for the current connection. Use this command when you need to reconnect a Playground device. If the `act server` command has been running for a long time, the log output may be too long to find the original QR code. In this scenario, you can use the `qrcode` command to display the QR code again.

```
➜  ~ act qrcode --help
Usage: act qrcode [options]

Displays the QR code for connection.

Options:
  -o, --onlyInfo  Outputs only the QR code content. This is disabled by default. When enabled, the emulated QR code image is not displayed. (default: false)
  -p, --pure      Outputs only the content without formatting information, such as time and prefixes. (default: false)
  -h, --help      display help for command
```

You can configure the parameters to output only the QR code content string. This allows other tools based on ACT to retrieve the QR code content.

```
act qrcode -o -p
```

View currently connected devices

`act device`: Queries the currently connected devices. You can use this command to check whether any Playground devices are connected to the server. Preview features, such as `preview` and `alive`, do not work if no devices are connected.

```
➜  ~ act device --help
Usage: act device [options]

Displays the list of currently connected devices.

Options:
  -h, --help  display help for command
```

View help

- You can view the supported instruction set.

  ```
  act -h
  ```

- View the parameters related to the command.

  ```
  act [command] -h
  ```

View information

```
act info
```

4.1.4. Update AntCubeTool

This topic describes the commands for updating AntCubeTool.

Run the `act update` command to update AntCubeTool.

```
➜  ~ act update --help
Usage: act update [options]

Check for and update to the latest version.

Options:
  -h, --help  display help for command
```

You can also run the `npm install xcube@en -g` command to reinstall the tool and obtain the latest version.

4.1.5. Uninstall AntCubeTool

Run the `npm uninstall xcube -g` command to uninstall AntCubeTool.

```
➜  ~ npm uninstall xcube -g
```

4.2. ACT 4.0 (Beta)

4.2.1. Use AntCubeTool

Create a new project

Use the `act init` command to create a new project.

Configuration fields, such as `name`, are used for file and folder names. These names can contain only letters, numbers, underscores, and hyphens.

```
➜  ~ act init -h
Usage: act init [options]

Initialize a Cube application in the current directory

Options:
  -h, --help        display help for command
```

Compile a project

Use the `act build` command to compile the Cube project in the specified directory.

The `path` parameter specifies the directory of the Cube project to compile. The `options` parameter specifies compilation configurations, such as enabling debug mode.

```
➜  ~ act build -h
Usage: @antdigital/cube build [options] [path]

Compile the Cube project in the specified path. The current path is used if a path is n
ot specified.

Options:
  --batch              Enable batch processing mode. Supported only for Card 1.0 project
s. (default: false)
  --watch              Enable real-time compilation. Not supported in batch processing m
ode. (default: false)
  --debug              Enable debug mode. (default: false)
  --bytecode           Enable JS bytecode. (default: false)
  --bizcode [bizcode]  The line-of-business to which the project belongs. (default: "")
  --page-id [id]       The ID of a single page.
  --page-sign          Securely sign the build artifact. (default: false)
  -h, --help           Display help information.
```

Compile dynamic assets

Use the `act build-asset` command to compile the Cube assets in the specified directory.

The `path` parameter specifies the project directory that contains the dynamic assets to compile.

```
➜  ~ act build-asset -h
Usage: @antdigital/cube build-asset [path]

Compile dynamic assets

Options:
  --only <step>  Build assets step by step. Steps: prepare | build | revert
  -h, --help     Display help information.
```

4.2.2. Update AntCubeTool

Method 1: Run the `act update` command to update ACT.

```
➜  ~ act update -h
Usage: @antdigital/cube update [options]

Update @antdigital/cube

Options:
  -h, --help  display help information
```

Method 2: Reinstall using the command `cnpm install @antdigital/cube -g` to obtain the latest version.

4.2.3. Uninstall AntCubeTool

You can run the `cnpm uninstall @antdigital/cube -g` command to uninstall ACT.

```
➜  ~ cnpm uninstall @antdigital/cube -g
```

5.Console Operations

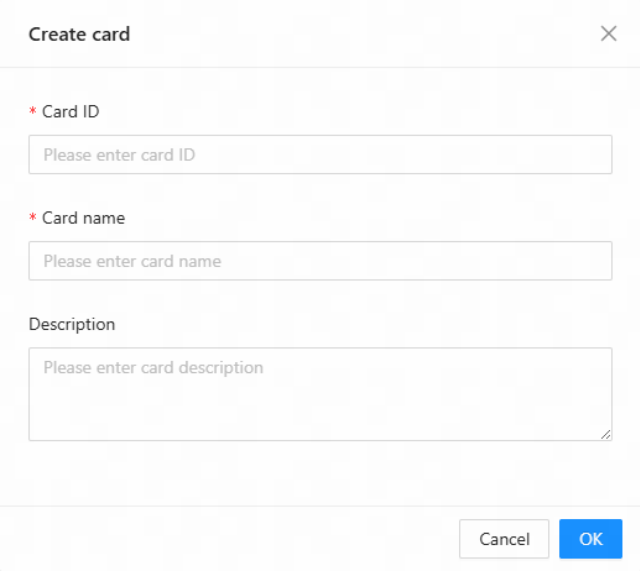
5.1. Card Management

5.1.1. Create a cube card

Learn how to create Ant Dynamic Cards in the mPaaS console.

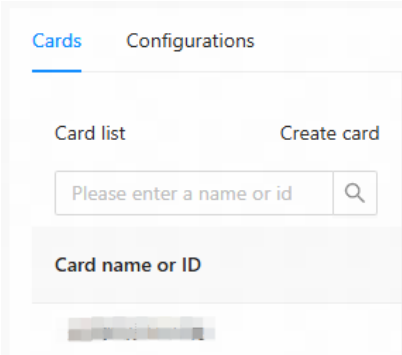
Procedure

1. Log on to the mPaaS console and select the target application from the application list.
2. In the left navigation pane, choose **Mobile Delivery Service** > **Ant Cube Card** > **Card Management**. The **Card Management** page appears.
3. On the **Card Management** page, click **Create Card**. In the **Create Card** form, enter the following information:
   - **Card ID:** Required. The ID must be at least 8 characters long and can contain only letters, digits, and underscores.
   - **Card name**: Required. Enter a name for the card.
   - **Description**: Optional.



4. After you enter the information, click **OK**.

The card is created and appears in the card list.



What to do next

After you create a card, you must upload its resources before you can publish it.

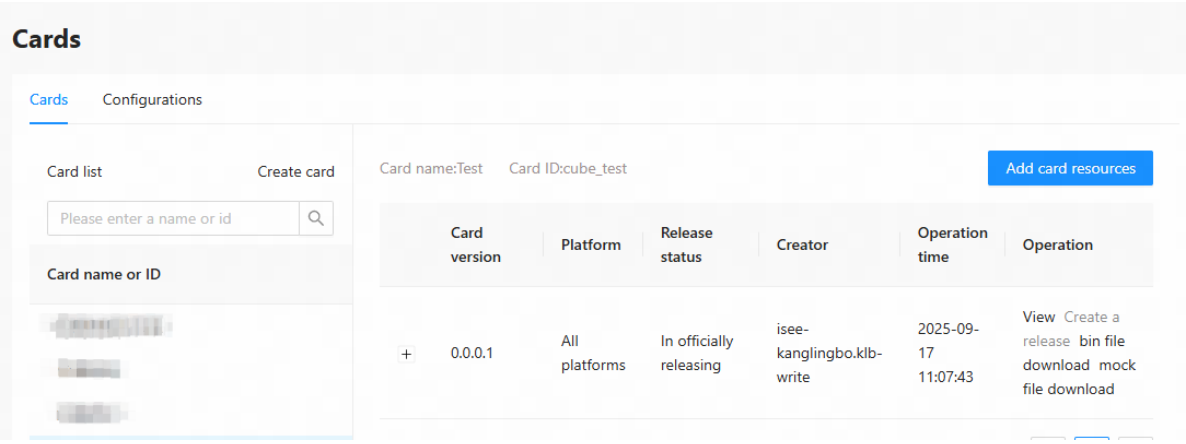
For more information, see Add cube card resources.

5.1.2. Add Cube card resources

This topic describes how to add card resource to a created card in the mPaaS console.

Procedure

1. Log on to the mPaaS console. In the left-side navigation pane, select target App.
2. In the left-side navigation pane, choose **Mobile Delivery Service** > **Ant Cube Card** > **Card Management**. The **Card Management** page appears.
3. In the **Card List** on the **Card Management** page, click the card to which you want to add resources.
4. Click **Add Card Resource** to go to the **Add Card Version** page.

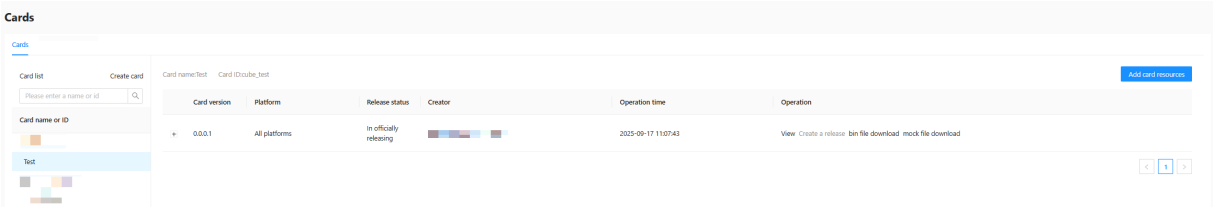


5. On the **Add Card Version** page, set the following parameters and upload a resource file: Click **Submit**.
6. - **Card Version Number**: Required. The version number must be a four-part number, such as `0.0.0.1`.
   - The card template ZIP file must be smaller than 10 MB. To develop the template, see the QuickStart documents for Android and iOS.
   - **Client Version Range**: The range of client versions that can pull the card, including the minimum and maximum version numbers. You must specify the client version scope for each platform.

- **Minimum Version**: The card is effective on the specified minimum client version and all later versions. To cover all client versions, enter `0.0.0.0`.
- **Maximum Version**: The card is effective on the specified maximum client version and all earlier versions. If you leave this field blank, the card is effective on all versions later than the minimum version.

> **Note**
>
> If you specify a maximum version, the offline package may not work if the client is upgraded to a later version.

After the card resource is added, the card information is displayed as shown in the following figure.



5.1.3. Delete a card

This topic describes how to delete an Ant Cube Card in the mPaaS console.

Procedure

1. Log on to the mPaaS console and select target App.
2. In the left navigation pane, choose **Mobile Delivery Service** > **Ant Cube Card** > **Card Management**.
3. On the **Card Management** page, in the **Card List**, hover over the target card and click the icon.
4. In the dialog box that appears, click **Confirm Deletion**.

Then the card is deleted successfully.

5.1.4. Create a release task

After you create anAnt Cube Card and add card resources, you must release the card so that clients can pull it. This topic describes how to create a publishing task for anAnt Cube Card in the mPaaS console.

Prerequisites

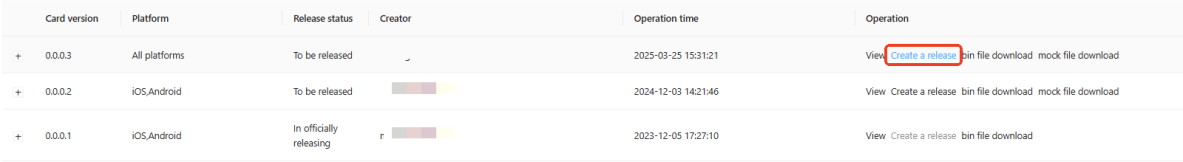
If you want to use the whitelist release model, you must create a whitelist before you release the card.

For more information about how to create a whitelist, see Whitelist management.

Procedure

1. Log on to the mPaaS console and select the target application from the application list.
2. In the navigation pane on the left, choose **Mobile Delivery Service** > **Ant Cube Card** > **Card Management** to open the **Card Management** page.
3. On the **Card Management** page, click the card that you want to release in the **Card List**.

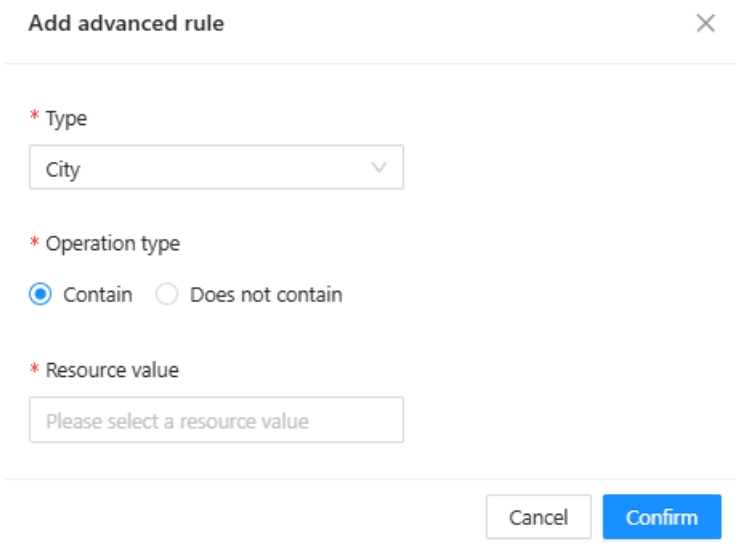
4. Click the **Create a Release** button to open the **Create a Release** page.



5. On the **New Release** page, configure the following parameters and click **Submit**.
   - **Release Type**: required. Grayscale release and official release. If you select grayscale release, you can configure the release model, whitelist, and advanced rules. If you select official release, you do not need to configure these parameters.
   - **Release Model**: required. You can select the release model of the whitelist or time window.
     - **Whitelist**: Releases the card to users on a whitelist. Only users on the whitelist can pull the card.
       - Whitelist Configuration: If you select **Whitelist** as the release model, select a pre-created whitelist.

       > **Note**
       > If a selected whitelist contains more than 100,000 users, only the first 100,000 users are included.

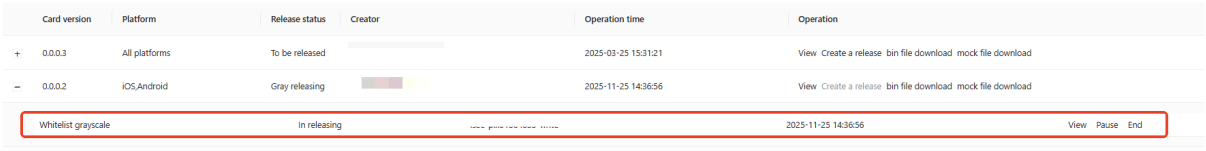
     - **Time Window**: Performs a canary release to a limited number of users within a specified time frame.
       - If you select the Time Window model, specify the window closing time and the number of users for the canary release.
   - **Release Description**: Optional. Add a description for the current release task.
   - **Advanced Rules**: Optional. You can release the card based on specific rules. You can configure Platform, Type, Operation Type, and Resource Value, as shown in the following figure.



     - **Type**: required. City, model, network, and device system version.
     - **Operation Type**: required. Contain or not contain. Under the equipment system version type, the operation type has additional in-range and out-of-range options.

- **Resource Value**: required. You can enter the resource value directly. You can also configure the resource mapping relationship in the release rule in advance and call this operation here.
- For more information about how to add resource values for advanced rules, see Release rule management.

You have now completed releasing the card. After releasing, the task status will be "In releasing" as shown in the image below. For tasks that are "In releasing" you can view, pause, or end the task. When a version of a card is in a "In releasing" task, you cannot create another release task. You must wait for the current release task to finish, or end the current "In releasing" task before creating a new task.



5.2. Card Analysis

This topic describes how to view analysis data for Ant Cube Card in the mPaaS console.

View card analysis

1. Log on to the mPaaS console and select the target App from the App list.
2. In the navigation pane on the left, click **Mobile Delivery Service** > **Ant Cube Card** > **Card analysis** to open the **Card Analysis** page.

   This page displays data for Ant Cube Card, including the number of new users, active users, card visits, and card clicks, along with their trend graphs. You can query card analysis data by card name, card version, platform, App version, and date range.

   - New users: The number of new users (device IDs) who accessed the card.
   - Active users: The total number of users (device IDs) who accessed the card. Each user is counted only once, regardless of the number of visits.
   - Card visits: The total number of times the card was accessed.
   - Card clicks: The total number of times the card was clicked.

Query card analysis data

1. On the **Card analysis** page, you can set the card name, card version, platform, and App version, and select a date range or time period.
2. Click **Query** to view the details of the card analysis.

5.3. Card Performance

This topic describes how to view Ant Cube Card performance in the mPaaS console.

View card performance

1. Log on to the mPaaS console. In the application list, select the target App.
2. In the navigation pane on the left, choose **Mobile Delivery Service** > **Ant Cube Card** > **Card performance**.

   This page displays business data, package pull exceptions, and card errors for Ant Cube Card. You can query card performance by card name, card version, platform, application version, and date range.

- Business data
  - Active users: The total number of users (device IDs) accessing the card; multiple accesses by the same user are not counted repeatedly.
  - Visits: The total number of times the card was accessed.
- Package pull exceptions
  - Number of package pull exceptions: The number of exceptions that occurred during card package pull requests.
  - Package pull exception rate: Number of card package pull exceptions / Total number of card package pull requests
- Card errors: This section displays the number of card errors, the number of affected users, an error trend curve by type, an error distribution by type, and a list of detailed error messages.
  - Number of errors: The total number of card errors, which is the sum of JavaScript (JS) exceptions and style exceptions.
  - Number of affected users: The number of unique users, identified by device ID, affected by card errors.

Query card performance

1. On the **Card performance** page, set the card name, card version, platform, application version, and date range.
2. Click **Query** to view the card performance details.

6.OpenAPI

6.1. Overview

Common requests and return values

Request instructions

Common requests must include the `tenantId`, `workspaceId`, and `appId` parameters.

> **Note**
>
> - For POST requests, include these three parameters in the query string. Do not add them to the request body.
> - If the method for passing parameters is not specified for a POST request, the default `Content-Type` is `multipart/form-data`.

Name	Type	Description
tenantId	String	The tenant ID.
workspaceId	String	workspaceId
appId	String	The application ID.

Return value description

The return value is in JSON format.

```
{
    "data":{
        "data":"data",
        "errorCode":null,
        "requestId":"",
        "resultMsg":"",
        "success":true
    },
    "requestId":""
}
```

The `success` parameter indicates whether the request was successful.

- If the value is `true`, the request was successful. Retrieve the returned data from the `data` parameter.
- If the value is `false`, the request failed. Retrieve the error message from the `resultMsg` parameter.

> **Note**
>
> In subsequent sections, only the content of the `data` parameter is described. The complete return value structure is omitted for brevity.

API request signing

All OpenAPI request URLs must include four URL parameters.

- appId
- workspaceId
- timestamp: The UNIX timestamp, in seconds, that indicates when the signature was generated.
- sign: The signature generated using an RSA 2048 private key.

  The `sign` value is generated by signing the following string. The signature is then converted to a hexadecimal (hex) string.

  ```
  appId=<your APPID>&workspaceId=<your workspaceId>&timestamp=<the request timestamp>&url=<the request URL without parameters>
  ```

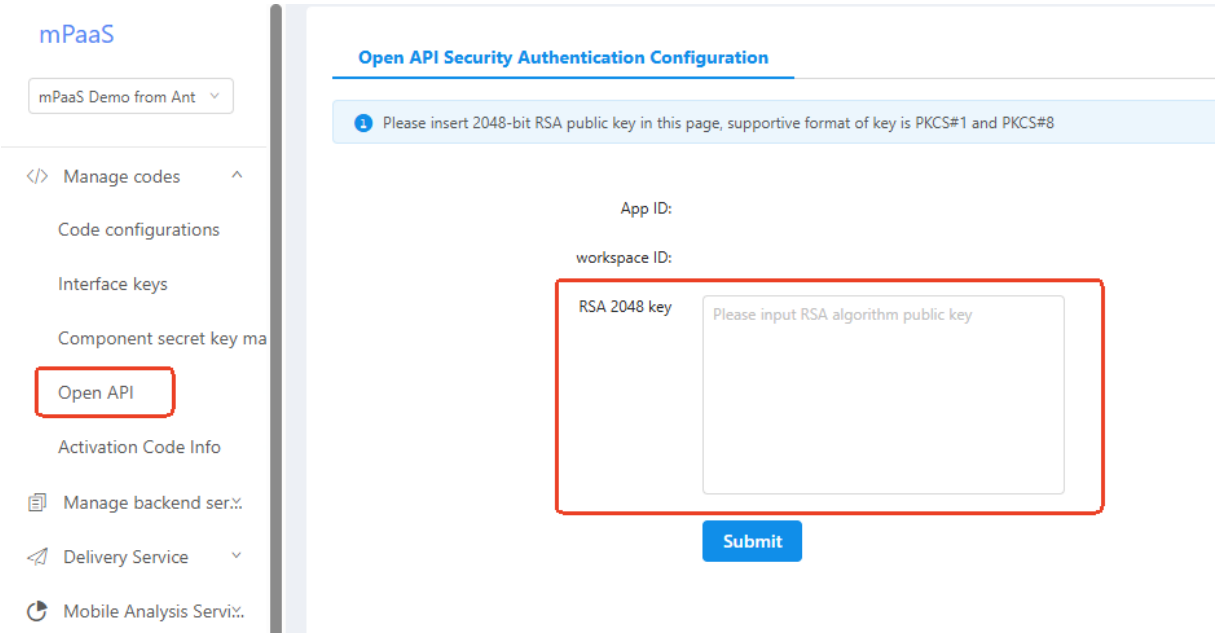
Complete URL example

```
http://11.167.24.43:8281/openapi/nebula/getNebulaResourceList?
appId=4B7AAC1231527&sign=53df44dc861099d276710c1233261ea42d5553b5f38714c8688134c72f8d0b34
e3de7ef4b13cc5ed1e241bb9416934737faf0ae915301660a968f476722eaa945437178dd8abc39912e6d6573
09be389d212350b6b18b87277ae05abe5e41aa270cf2966c86048025c317070449afb022aeb963d5209d8cbbd
f0b3522d509a28651e1b16490eb87b565e6abb974535c0d6f894e372283b910b1e372b458e82f9b53ce5443a9
eba6830cf271904ecc5b44cff8f8cad2d3a0e869019dc668b70baebc6d7bda8f1f739b8dc9aacf9bb84caf0f1
a5c5adacfa35814872582c681e5f9b9e0445cd82860f3489d1474130c6882662ea295b1196c228ab48&timest
p=1558937883&workspaceId=sit
```

RSA key pair generation

```
To generate a key pair:

1. Generate a 2048-bit RSA private key.
openssl genrsa -out private_key.pem 2048

2. Convert the private key to PKCS#8 format.
openssl pkcs8 -topk8 -inform PEM -outform DER -in private_key.pem -out private_key.der
-nocrypt

3. Convert the public key to DER format.
openssl rsa -in private_key.pem -pubout -outform DER -out public_key.der

4. Convert the private and public keys to Base64 format.
openssl base64 -in private_key.der -out private_key_base64.der
openssl base64 -in public_key.der -out public_key_base64.der

5. View the public key content.
cat public_key_base64.der

6. View the private key content.
cat private_key_base64.der
```

After you generate the key pair, upload the public key to mappcenter for signature verification.



6.2. Create, Query and Delete Card

Create a card

- Request path: /openapi/cubecard/createTemplate
- Request method: POST
- Request parameters

Name	Type	Required	Description
templateId	String	Yes	The card ID.
templateName	String	Yes	The card name.
templateDesc	String	No	The description of the card.

- Return value

```
success
```

Query card list

- Request path: /openapi/cubecard/listTemplates
- Request method: GET
- Request parameters

Name	Type	Required	Description
pageNum	Integer	Yes	Current page, starting from 1
pageSize	Integer	No	Number of items per page, default is 20

- Return value

```
{
  "firstPage":true,
  "lastPage":true,
  "nextPage":1,
  "pageNo":1,
  "pageSize":20,
  "prePage":1,
  "totalCount":7,
  "totalPage":1,
  "list":[
    {
      "id":123, // The primary key
      "templateName":"Card name",
      "templateId":"Card ID",
      "templateDesc":"Card description"
    }
  ]
}
```

Return value description

Parameter	Type	Description
firstPage	boolean	Whether the current page is the first page ◦ Yes: true ◦ No: false
lastPage	boolean	Whether the current page is the last page ◦ Yes: true ◦ No: false
nextPage	Integer	Page number of the next page
pageNo	Integer	Current page number
pageSize	Integer	Number of items returned per page
prePage	Integer	Page number of the previous page
totalCount	Integer	The total number of cards
totalPage	Integer	The total number of pages
list	List	Specific content returned, the following are fields in the content
id	Long	The primary key of the card
templateName	String	The name of the card
templateId	String	The card ID
templateDesc	String	The description of the card

Delete a card

- Request path: /openapi/cubecard/deleteTemplate
- Request method: POST
- Request parameters

Name	Type	Required	Description
templateId	String	Yes	The card ID.

- Return value

```
Deleted successfully
```

6.3. Upload and publish card resources

6.3.1. Upload card resources

- Request path: /openapi/cubecard/createResource
- Request method: POST
- Request parameters

Name	Type	Required	Description
templateId	String	Yes	Card ID
file	MultipartFile	Yes	ZIP file generated by the card
templateResourceVersion	String	Yes	Current resource version number
templateResourceDesc	String	No	Current resource description
platform	String	Yes	The card supports Android and iOS platforms, separated by `,`. For example: `Android,iOS`, in the following order: Android, iOS.
iosMaxVersion	String	No	Maximum iOS version supported, referring to the application version, not the system version

iosMinVersion	String	Required when platform includes iOS or all	Minimum iOS version supported, referring to the application version, not the system version
androidMaxVersion	String	No	Maximum Android version supported, referring to the application version, not the system version
androidMinVersion	String	Required when platform includes Android or All	Minimum Android version supported, referring to the application version, not the system version
harmonyMaxVersion	String	No	Maximum Harmony version supported, referring to the application version, not the system version
harmonyMinVersion	String	Required when platform includes Harmony	Minimum Harmony version supported, referring to the application version, not the system version
extendInfo	String	No	JSON format string, extension parameters

- Return value

```
152
```

The returned content is the primary key of the uploaded resource.

6.3.2. Query the card resources list

- Request path: /openapi/cubecard/listResources
- Request method: GET
- Request parameters

Name	Type	Required	Description
templateId	String	Yes	Card ID
pageNum	Integer	Yes	Current page, starting from 1
pageSize	Integer	No	Number of items per page, default is 20

- Return value

```
{
  "firstPage":true,
  "lastPage":true,
  "nextPage":1,
  "pageNo":1,
  "pageSize":20,
  "prePage":1,
  "totalCount":7,
  "totalPage":1,
  "list":[
    {
      "id":123, //primary key
      "templateName":"Test Card",
      "templateId":"test_template",
      "templateDesc":"Test",
      "templateResourceVersion":"1.1",//resource version
      "binFileUrl":"http://oss.down.net/xxxxxxxxx",
      "zipFileUrl":"http://oss.down.net/xxxxxxxxx",
      "templateResourceDesc":"Test",
      "operator":"default",
      "gmtCreate": "2022-05-17 12:34:09",
      "platform":"iOS,Android",
      "iosMaxVersion":"10.1",
      "iosMinVersion":"1.1",
      "androidMaxVersion":"9.9",
      "androidMinVersion":"1.1",
      "extendInfo":"{\"key\":\"value\"}",
      "binFileMd5":"md5 value of bin file",
      "minCubeSdkVersion":"1.0",
      "resourceStatus": 1,
      "previewPictureUrl":"https://xxxx",
      "mockDataDownloadUrl": "https://xxxxxx"
    }
  ]
}
```

Return value description

Parameter	Type	Description
firstPage	boolean	Whether the current page is the first page. ◦ Yes: true ◦ No: false
lastPage	boolean	Whether the current page is the last page. ◦ Yes: true ◦ No: false

nextPage	Integer	The page number of the next page
pageNo	Integer	The current page number
pageSize	Integer	The number of items returned per page
prePage	Integer	The page number of the previous page
totalCount	Integer	The total number of cards to query
totalPage	Integer	The total number of pages to query
list	List	The specific content returned, the following are fields in the content
id	Long	The primary key of the card resource
templateName	String	The card name
templateId	String	The card ID
templateDesc	String	The card version
templateResourceVersion	String	The version of the card resource
binFileUrl	String	The download URL of the bin file in the card resource
zipFileUrl	String	The download URL of the ZIP file in the card resource
templateResourceDesc	String	Card resource description
operator	String	The person who uploaded. The value is "default" when uploading by OpenAPI
gmtCreate	String	Upload date
platform	String	Platform type: Android, iOS, ALL (supports both platforms)

iosMaxVersion	String	Maximum iOS version supported, referring to the application version, not the system version
iosMinVersion	String	Minimum iOS version supported, referring to the application version, not the system version The minimum version must exist, the maximum version can be empty
androidMaxVersion	String	Maximum Android version supported, referring to the application version, not the system version
androidMinVersion	String	Minimum Android version supported, referring to the application version, not the system version The minimum version must exist, the maximum version can be empty
harmonyMaxVersion	String	Maximum Harmony version supported, referring to the application version, not the system version
harmonyMinVersion	String	Minimum Harmony version supported, referring to the application version, not the system version The minimum version must exist, the maximum version can be empty
extendInfo	String	Extension parameters
binFileMd5	String	The md5 value of the bin file
minCubeSdkVersion	String	The minimum cubeSdk version supported by the current bin file. If empty, there is no restriction
resourceStatus	Integer	Current resource status ◦ 0: Initial state ◦ 1: In grayscale release ◦ 3: In official release ◦ 5: Release ended ◦ 6: Release paused
previewPictureUrl	String	The URL of the generated preview image
mockDataDownloadUrl	String	The download URL of the mockData file

6.3.3. Query the content by resource ID

- Request path: /openapi/cubecard/getResourceInfoById
- Request method: GET
- Request parameters

Name	Type	Required	Description
templateId	String	Yes	Card ID
resourceId	Long	Yes	Primary key of the resource to query

- Return value

```
{
  "id":123, //Primary key
  "templateName":"Test Card",
  "templateId":"test_template",
  "templateDesc":"Test",
  "templateResourceVersion":"1.1",//Resource version
  "binFileUrl":"http://oss.down.net/xxxxxxxxx",
  "zipFileUrl":"http://oss.down.net/xxxxxxxxx",
  "templateResourceDesc":"Test",
  "operator":"default",
  "gmtCreate": "2022-05-17 12:34:09",
  "platform":"ALL",
  "iosMaxVersion":"10.1",
  "iosMinVersion":"1.1",
  "androidMaxVersion":"9.9",
  "androidMinVersion":"1.1",
  "extendInfo":"{\"key\":\"value\"}",
  "binFileMd5":"MD5 value of the bin file",
  "minCubeSdkVersion":"1.0",
  "resourceStatus": 1,
  "previewPictureUrl":"https://xxxx",
  "mockDataDownloadUrl": "https://xxxxxx"
}
```

Response parameters

Parameter	Type	Description
id	Long	Primary key of the card resource
templateName	String	The name of the card
templateId	String	The ID of the card

templateDesc	String	The card description
templateResourceVersion	String	Version of the card resource
binFileUrl	String	Download URL of the bin file in the card resource
zipFileUrl	String	Download URL of the ZIP file in the card resource
templateResourceDesc	String	Description of the card resource
operator	String	The person who uploaded. The value is "default" when uploading by OpenAPI
gmtCreate	String	Upload date
platform	String	Platform type: Android, iOS, ALL (supports both platforms)
iosMaxVersion	String	Maximum iOS version. This refers to the application version, not the system version
iosMinVersion	String	Minimum iOS version. This refers to the application version, not the system version. The minimum version must exist, while the maximum version can be empty
androidMaxVersion	String	Maximum Android version. This refers to the application version, not the system version
androidMinVersion	String	Minimum Android version. This refers to the application version, not the system version. The minimum version must exist, while the maximum version can be empty
harmonyMaxVersion	String	Maximum Harmony version. This refers to the application version, not the system version
harmonyMinVersion	String	Minimum Harmony version. This refers to the application version, not the system version. The minimum version must exist, while the maximum version can be empty

extendInfo	String	Extension parameters
binFileMd5	String	MD5 value of the bin file
minCubeSdkVersion	String	Minimum cubeSdk version supported by the current bin file. If empty, there is no restriction
resourceStatus	Integer	Current resource status ◦ 0: Initial state ◦ 1: In grayscale release ◦ 3: In official release ◦ 5: Release ended ◦ 6: Release paused
previewPictureUrl	String	URL of the generated preview image
mockDataDownloadUrl	String	Download URL of the mockData file

6.3.4. Create a release task

- Request URI: /openapi/cubecard/createTask
- Request method: POST
- Request parameters:

Name	Type	Required	Description
publishType	Integer	Yes	The release type. ◦ 2: Canary release ◦ 3: Full release
publishMode	Integer	Yes	The canary release mode. ◦ 0: Full release ◦ 1: Whitelist ◦ 2: Time window
taskDesc	String	No	The release description.

greyEndtimeData	String	Required when `publishMode` is 2.	The end time for a time-window canary release. The format is "YYYY-MM-dd HH:mm:ss". The time must be later than the current time and within 7 days of the current time.
greyNum	Integer	Required when `publishMode` is 2.	The number of users for the time-window canary release.
whitelistIds	String	Required when `publishMode` is 1.	The primary key IDs of the whitelists. Separate multiple IDs with a comma (`,`).
templateResourceId	Long	Yes	The primary key ID of the resource plan to be released.
greyConfigInfo	String	No	The advanced rule conditions for the release. This is a JSON string. For details, see the following table: [{"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"}]

Advanced rule description

Name	Type	Description
ruleElement	String	The rule type. ◦ city: City ◦ mobileModel: Device model ◦ netType: Network ◦ osVersion: Device OS version
value	String	The rule value. Separate multiple values with a comma (,). When `operation` is 3 or 4, the `value` is in the `aa-bb` format, where `aa` is the lower value and `bb` is the higher value.

operation	Integer	Related operations ◦ 1: Includes ◦ 2: Excludes ◦ 3: Within range ◦ 4: Outside range If `ruleElement` is `city`, `mobileModel`, or `netType`, `operation` can only be 1 (Includes) or 2 (Excludes). If `ruleElement` is `osVersion`, `operation` can be any of the four values.

- Return value

```
456
```

The return value is the primary key of the created release task.

6.3.5. Query the release task list

- Request path: /openapi/cubecard/listTasks
- Request method: GET
- Request parameters

Name	Type	Required	Description
templateResourceId	Long	Yes	The ID of the card resource.

- Return value

```
[
  {
    "gmtCreate":"2019-04-24 17:43:54",
    "gmtModified":"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\":fal
}],\"defaultResult\":false}",
    "greyEndtimeData":"",
    "greyNum":0,
    "id":212,
    "taskDesc":"Card release",
    "templateResourceId":572,
    "publishMode":1,
    "publishType":2,
    "taskStatus":1,
    "whitelistIds":"931,932",
    "operator":"Operator"
  }
]
```


Return value description

Return value	Type	Description
gmtCreate	String	The creation time
gmtModified	String	The update time
greyConfigInfo	String	Advanced grayscale rule information. For more information, see greyConfigInfo details
greyEndtimeData	String	The end time of the time window grayscale release
greyNum	String	The number of users at the end of the time window grayscale release
id	Long	The primary key of the release task
taskDesc	String	The description of the release task
templateResourceId	Long	The primary key of the card resource package corresponding to the task

publishMode	Integer	The grayscale release mode ◦ 1: Whitelist ◦ 2: Time window
publishType	Integer	The release type ◦ 2: Grayscale release ◦ 3: Official release
taskStatus	Integer	The current task status ◦ 1: Releasing ◦ 2: Ended ◦ 3: Paused
whitelistIds	String	The whitelist IDs for whitelist release, separated by commas for multiple IDs
operator	String	The operator. All tasks created by OpenAPI are set to default

greyConfigInfo details

Name	Type	Description
operator	String	The rule relationship. "and" indicates the AND operation on the results in subRules
defaultResult	boolean	The default result to return
subRules	List	The rules collection
operator	String	The rule name ◦ contains: Contains ◦ excludes: Does not contain ◦ vLimitIn: Within the range ◦ vLimitOut: Outside the range

left	◦ When operator is contains or excludes, it is a List character set where each element represents a rule value. ◦ When operator is vLimitIn or vLimitOut, it is an object where lower represents the lower value and upper represents the higher value	See the description in the type column
right	String	The rule type name
defaultResult	Boolean	The default result

6.3.6. Query the task details by task ID

- Request path: /openapi/cubecard/getTaskDetail
- Request method: GET
- Request parameters

Name	Type	Required	Description
taskId	Long	Yes	Primary key of the task ID to be queried

- Return value

  The parameter description are consistent with those in Query the release task list.

```
{
  "gmtCreate":"2019-04-24 17:43:54",
  "gmtModified":"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\":fal
}],\"defaultResult\":false}",
  "greyEndtimeData":"",
  "greyNum":0,
  "id":212,
  "taskDesc":"Card release",
  "temlateResourceId":572,
  "publishMode":1,
  "publishType":2,
  "taskStatus":1,
  "whitelistIds":"931,932",
  "operator":"Operator"
}
```


6.3.7. Change the status of a publishing task

- URI of the request: /openapi/cubecard/changeTaskStatus
- Request method: POST
- Request parameters

Name	Type	Required	Description
templateResourceId	Long	Yes	The ID of the card resource.
templateTaskId	Long	Yes	The ID of the card task.
taskStatus	Integer	Yes	The target status. ◦ 1: Publishing ◦ 2: Finished ◦ 3: Paused

- Return value

```
success
```

7.Card Syntax

7.1. Card Basics

7.1.1. Project management

This topic describes the project configurations of Ant Cube Card.

Project directory structure

A valid card project consists of a configuration file `.act.config.json` located under the project root directory and a set of card description file `.vue`, `.css`, `.json`, etc. The directory structure is as follows:

```
.
├── dist // The compilation result folder (automatically generated when you perform the compilation operation)
| ├── app.manifest // The application configuration information (the naming format is fixed)
| ├── test_cube
| | ├── main.bin // The binary file of the compiled product.
| | ├── main.json // The JSON file that is compiled.
| | ├── main.mock // The compilation product of mock.json
| | ├── main.js // The JS logic segment of the compiled product, which facilitates run time troubleshooting of JS segment exceptions.
| | └── main.zip // The overall package file for all products of the card.
├── test_cube
| ├── main.vue // [Required] card source code file, file name cannot be changed
| ├── mock.json // [Optional] Card mock data
| ├── manifest.json // [Required] The card compilation configuration file. The file name cannot be changed.
| └── main.css // [Optional] card style file
└── .act.config.json // [Required] project configuration file, file name cannot be changed
```

.act.config.json

The `.act.config.json` file is the configuration file for the card project. It is currently generated by the Ant Cube Card command-line tool and does not need to be modified or paid attention to.

> **Note**
>
> The `.act.config.json` must be under the root directory of the Works.

```
//.act.config.json

{
  "type": "templates", // Required. The project type. Valid values: templates.
}
```

manifest.json

The `manifest.json` is the compiled configuration file of the corresponding card. It is currently generated by the Ant Cube Card command-line tool and does not need to be modified or concerned.

> **Note**
>
> The `manifest.json` must be under the same path as the corresponding card `main.vue`.

```
//        manifest.json
{
  "name": "my-card", // Optional. The card name. After the card is published, the card ID prevails.
  "version": "x.x.x", // Optional. The card version. After the card is published, the background version of the card prevails.
  "compilerType": 1, // Optional. The card compilation mode. 0 (static card) | 1 (dynamic card. JS is supported. Recommended). Default value: 0.
  "jsformat": 1, // Optional. The card JS compilation format. 0 (expression export) | 1 (IIFE export. JS import is supported. Recommended). Default value: 0.
}
```

Project example

test_cube.zip

7.1.2. Template writing

The writing syntax of a template is borrowed from VUE. After simplification, the template contains only three fields: <template>, <style>, and <script>.

<template> Field

The relevant logic for the template mode page layout should be placed within the <template></template> segment.

The elements supported within the template pattern <template> segment are determined by the Card component.

< style > Field

For more information about style support, see Style syntax.

< script > Field

The JS-related logic is placed in the script. For more information about JS-related capabilities, see JS capabilities.

7.1.3. Unit

This paper introduces the definition of Ant Cube Card in terms of value, time, length and color.

Numeric Unit

Some attributes in the card need to be implemented in pure numerical units, such as flex and lines. In this case, px and vw are not added after the value.

For more information about the value ranges and value types of pure numeric values, see the property definitions of each component in . If a impure value is encountered, it is processed as 0 by default.

Example:

```
lines:5;
flex:1;
```

Time Unit

Some attributes of cards need to be described by time, for example, animation-duration. Cards support time units in seconds and milliseconds.

Example:

```
animation-duration:2s;
```

Length Unit

The card has several different units to indicate the length. The length consists of a number and a unit. No space can appear between the number and the unit. The length units are divided into built-in units and custom units.

Built-in Unit

Relative length

Relative length units specify the properties of one length relative to another. For different equipment relative length is more suitable

- vw:viewpoint width, window width, 1vw=1% of the window width.
- %: Percentage of window height /width.

Example:

```
font-size: 20vw;
background-size:50%;
```

Absolute length

An absolute length unit is a fixed value that reflects a real physical dimension. The absolute length unit depends on the output medium and does not depend on the environment (monitor, resolution, operating system, etc.)

- px: pixel (1px = 1/96th of 1in)
- rpx: calculated based on 750, 2rpx = 1px

Example:

```
font-size: 17px;
line-height:40rpx;
```

Click here detailLength.zip for the complete sample code.

Color Unit

Colors in cards can be specified in the following ways:

- Hexadecimal
- RGB Color
- Color Name

Hex Color

There are several ways to specify the components of a hexadecimal color:

- # RRGGBB, where RR (red), GG (green) and BB (blue). All values must be between 0 and FF.

For example, the #0000FF value is rendered blue because the blue component is set to the highest value FF and the others are set to 0.

- # RGB, where R (red),G (green) and B (blue). All values must be between 0 and F.

For example, the #00F value appears blue because the blue component is set to the highest value F and the others are set to 0.

- 0xrrggbb, where rr (red),gg (green) and bb (blue). All values must be between 0 and ff.

For example, the 0x0000ff value appears blue because the blue component is set to the highest value ff and the others are set to 0.

Example:

```
background-color:#ff0000;
border-top-color:0x0000ff;
border-bottom-color:#00f;
```

RGB Color

There are several ways to specify the color of an RGB component:

- RGB (red, green, blue). Each parameter (red, green and blue) defines the brightness of the color, as an integer between 0 and 255. For example, an RGB (0,0,255) value is rendered as blue because the parameter for blue is set to the highest value of 255 and the others are set to 0.
- RGBA (red, green, blue, alpha). The alpha parameter is between 0.0 (fully transparent) and 1.0 (fully opaque). For example, an RGB (0,0,255,0.5) value appears as translucent blue.

Example:

```
background-color:rgb(0,0,255);
border-color:rgba(255,0,0,0.5);
```

Color Name

The card supports 147 color name definitions, including 17 standard colors and 130 other colors. The following table lists the names and corresponding hexadecimal values for all colors.

Standard Color

Parameter	Hexadecimal
red	0xffff0000

blue	0xff0000ff
black	0xff000000
...	Etc.

Others

Parameter	Hexadecimal
red	0xffff0000
blue	0xff0000ff
black	0xff000000
...	Etc.

Example

```
background-color:red;
```

> **Note**
>
> - Only the color formats listed above are supported. None of the other color formats are supported.
> - If an unsupported color format is encountered, the card will be treated as the default transparent color (except for some styles that have separate default color rules).

Click here detailColor.zip for the complete sample code.

Appendix-Color Value Definition

Parameter	Color value
red	0xffff0000
darkred	0xff8b0000
tan	0xffd2b48c

linen	0xfffaf0e6
sienna	0xffa0522d
indianred	0xffcd5c5c
teal	0xff008080
grey	0xff808080
green	0xff008000
gray	0xff808080
darkgrey	0xffa9a9a9
darkgreen	0xff006400
beige	0xfff5f5dc
orange	0xffffa500
darkgray	0xffa9a9a9
orangered	0xffff4500
khaki	0xfff0e68c
seagreen	0xff2e8b57
gold	0xffffd700
darkorange	0xffff8c00
darkkhaki	0xffbdb76b
indigo	0xff4b0082

goldenrod	0xffdaa520
maroon	0xff800000
gainsboro	0xffdcdcdc
lime	0xff00ff00
greenyellow	0xffadff2f
darkgoldenrod	0xffb8860b
slategrey	0xff708090
slategray	0xff708090
salmon	0xfffa8072
darkseagreen	0xff8fbc8f
seashell	0xfffff5ee
darksalmon	0xffe9967a
tomato	0xffff6347
thistle	0xffd8bfd8
darkslategrey	0xff2f4f4f
cyan	0xff00ffff
forestgreen	0xff228b22
dimgrey	0xff696969
darkslategray	0xff2f4f4f

mistyrose	0xffffe4e1
dimgray	0xff696969
darkcyan	0xff008b8b
black	0xff000000
magenta	0xffff00ff
limegreen	0xff32cd32
coral	0xffff7f50
darkmagenta	0xff8b008b
azure	0xfff0ffff
blue	0xff0000ff
oldlace	0xfffdf5e6
cornsilk	0xfffff8dc
darkblue	0xff00008b
skyblue	0xff87ceeb
firebrick	0xffb22222
orchid	0xffda70d6
lightgrey	0xffd3d3d3
lightgreen	0xff90ee90
lightyellow	0xfffffffe0

lightgray	0xffd3d3d3
darkorchid	0xff9932cc
royalblue	0xff4169e1
aqua	0xff00ffff
steelblue	0xff4682b4
bisque	0xffffe4c4
crimson	0xffdc143c
slateblue	0xff6a5acd
dodgerblue	0xff1e90ff
blanchedalmond	0xffffebcd
lightseagreen	0xff20b2aa
lightslategrey	0xff778899
lightslategray	0xff778899
brown	0xffa52a2a
lightsalmon	0xffffa07a
snow	0xfffffafa
lightcyan	0xffe0ffff
rosybrown	0xffbc8f8f
sandybrown	0xfff4a460

darkslateblue	0xff483d8b
yellow	0xffffff00
lightcoral	0xfff08080
mintcream	0xfff5fffa
aquamarine	0xff7fffd4
saddlebrown	0xff8b4513
honeydew	0xfff0fff0
pink	0xffffc0cb
lightblue	0xffadd8e6
cadetblue	0xff5f9ea0
wheat	0xfff5deb3
lawngreen	0xff7cfc00
white	0xffffffff
aliceblue	0xfff0f8ff
chocolate	0xffd2691e
yellowgreen	0xff9acd32
moccasin	0xffffe4b5
navy	0xff000080
chartreuse	0xff7fff00

ivory	0xfffffff0
palegreen	0xff98fb98
lavender	0xffe6e6fa
hotpink	0xffff69b4
olive	0xff808000
fuchsia	0xffff00ff
mediumseagreen	0xff3cb371
silver	0xffc0c0c0
olivedrab	0xff6b8e23
darkturquoise	0xff00ced1
turquoise	0xff40e0d0
violet	0xffee82ee
violetred	0xffd02090
darkviolet	0xff9400d3
palegoldenrod	0xffeee8aa
whitesmoke	0xfff5f5f5
springgreen	0xff00ff7f
burlywood	0xffdeb887
peru	0xffcd853f

floralwhite	0xfffffaf0
lightpink	0xffffb6c1
darkolivegreen	0xff556b2f
ghostwhite	0xfff8f8ff
mediumblue	0xff0000cd
mediumorchid	0xffba55d3
lightsteelblue	0xffb0c4de
lightslateblue	0xff8470ff
transparent	0x00000000
deepskyblue	0xff00bfff
lightskyblue	0xff87cefa
lightgoldenrodyellow	0xfffafad2
plum	0xffdda0dd
mediumaquamarine	0xff66cdaa
mediumslateblue	0xff7b68ee
blueviolet	0xff8a2be2
midnightblue	0xff191970
deeppink	0xffff1493
lemonchiffon	0xfffffacd

antiquewhite	0xfffaebd7
paleturquoise	0xffafeeee
powderblue	0xffb0e0e6
navajowhite	0xffffdead
mediumspringgreen	0xff00fa9a
cornflowerblue	0xff6495ed
palevioletred	0xffdb7093
mediumvioletred	0xffc71585
purple	0xff800080
rebeccapurple	0xff663399
lavenderblush	0xfffff0f5
mediumturquoise	0xff48d1cc
peachpuff	0xffffdab9
mediumpurple	0xff9370db
papayawhip	0xffffefd5

7.1.4. Data binding

This article describes the form of data binding in the template pattern.

Single data source binding

Following the Vue format, template mode supports both interpolation and directive for single data binding.

- Interpolation formats can be used only separately, and cannot be mixed, such as <text>ab{{var1}}cd</text>.

- Directive support abbreviated formats.

When used, the data field that can be bound (that is, the name in the following table) is determined by the current component, and the bound data variable (that is, variable in the following table) can be defined by using expressions.

Type	Data structure	Short form	Cascade	Examples
Interpolation	{{variable}}	None	. or []	<text>{{var1.var2}}</text>
Directive	v-bind:name="variable"	:name="variable"	. or []	<text :value="var1[num1]"></text>

The supported formats for text-based components (such as `text`) when populating content include:

- [1] <text:value="var"></text> (where var="hello_1")
- [2] <text>{{var}}</text> (where var="hello_2")
- [3] <text>hello_3</text>

The resolution priority is [1] < [2] < [3] , which means that the `hello_3` will eventually be displayed.

Double data source binding

The template supports submitting two sets of data as data sources to be bound simultaneously. This mainly solves the problem of injecting additional control data into the template from the native side during actual business development. When such a requirement is encountered, this function is enabled.

The data injected from the template will be merged with the mock data sent by the server. The injected data has higher priority, and if there are identical fields, it will overwrite the corresponding fields in the mock data. The usage is the same as that of the fields in the mock data.

```
// The injected data.
{
title: "title"
}

// The data extraction method.
<text :value=title></text>
```

Example code

Click here detailBindData.zip for the complete sample code.

7.1.5. Event binding

This topic describes the types of events that can be bound to Ant Cube Card and the binding methods.

Common events

In a card, there are two types of events that can be bound to the template:

- Click
- Longpress

Template usage

In template mode, event binding only supports the command format. The command form supports the shorthand format. The name of the event that can be bound to the card. The name of the event that can be bound to the card (@ click in the following sample code) is determined by the card-registered component. For more information about the card-registered components, see div.

- The bound handler parameter (param in the following sample code) supports expressions.
- You can define the corresponding business JS method in the <script> section, with no restrictions on the parameters of the method.

```
<template>
...
        <text @click="click(param)"></text>
...
</template>

<script>
    export default {
        data: {
        },
        beforeCreate() {
        },
        methods: {
            click(p) {
                console.info("onclick");
            }
        }
    }
</script>
```

Example code

Click here detailBindEvent.zip for the complete sample code.

7.1.6. Logical rendering

This topic describes the types of logical rendering and their corresponding directives.

Conditional rendering

v-show

The `v-show` directive conditionally renders node content. The node is always rendered, regardless of whether the `v-show` expression evaluates to `true` or `false`.

- When the `v-show` expression evaluates to `true`, it is equivalent to adding `display: flex` to the node's CSS style.
- When the `v-show` expression evaluates to `false`, it is equivalent to adding `display: none` to the node's CSS style.

```
<div class="div" v-show="exist(exist4)"></div>
```

v-if

The `v-if` directive conditionally renders a block of content. The block is rendered only if the directive's expression evaluates to `true`.

- You can use v-else-if as an "else-if block" for `v-if`. You can also chain multiple `v-else-if` blocks.
- You can use `v-else` as an "else block" for `v-if`.

```
<div class="div" v-if="exist(a)">
  ...
</div>
<div class="div" v-else-if="exist(b)">
  ...
</div>
<div class="div" v-else>
  ...
</div>
```

> **Important**
>
> - A `v-else-if` block must immediately follow a `v-if` block. Otherwise, it will not be recognized.
> - A `v-else` block must immediately follow a `v-if` or `v-else-if` block. Otherwise, it will not be recognized.
> - Do not use `v-if` on the root node of a template.

List rendering

v-for

The `v-for` directive renders a block of content repeatedly based on an array, a `number`, or an object. The following six syntaxes are supported:

- v-for="index in 10"
- v-for="index in number"
- v-for="item in array"
- v-for="(item,index) in array"
- v-for="item in object"
- v-for="(item,key) in object"

```
<div class="vforStyle" v-for="(value,index) in obj">
    <text class="content">{{index + value}}</text>
</div>
```

Note

- The `v-for` directive does not support nesting. Do not use a `v-for` directive inside another `v-for` rendering block.
- When rendering data based on an object, the display order of the items is not guaranteed. To display items in a specific order, traverse an array or `number` instead.
- Do not use `v-for` on the root node of a template.

Sample code

Click detailLogicalRender.zip to download the complete sample code.

7.2. Card Tags

7.2.1. Basic tags

7.2.1.1. div

The tag defines a division or section in a document. This tag partitions the document into independent parts. It is used for organizational purposes and has no associated format.

Embedded component support

You can nest any other component within this component.

Styles

The component supports all common styles.

Properties

None.

Events

The component supports all common events.

Example

```
<div>
    <div class="box"></div>
</div>


.box {
    border-width: 2px;
    border-style: solid;
    border-color: #BBB;
    width: 250px;
    height: 250px;
    margin-top: 250px;
    margin-left: 250px;
    background-color: #EEE
  }
```

Sample code

Click detailDiv.zip to download the complete sample code.

7.2.1.2. text

The <text> component is a built-in component of the Ant Dynamic Card Engine that renders text with specified styles. This component can only contain text values. You can use `{{}}` to insert variable values as text content.

Support for embedded components

Other components cannot be nested in this component.

Styles

The <text> component supports all the styles described in General styles, in addition to the following special styles:

Font-related

Property	Description	Value type	Default value	Optional values	Syntax	Notes
font-size	Font size	Length unit	16px		font-size: 10px;	

font-weight	Font weight	string	normal	normal, bold, 100, 200, 300, 400, 500, 600, 700, 800, 900. `normal` is equivalent to `400`. iOS supports all nine values. On Android, a value of `400` is rendered as `normal`, `700` or higher is rendered as `bold`, and values between 400 and 700 are rendered with a FakeBold effect.	font-weight: bold; font-weight: 700;	
font-style	Font style	string	normal	normal, italic	font-style: normal;	
font-family	Sets the font	string	Platform default font		font-family: PingFangSC-Regular;	Consistency of this setting across different platforms and devices is not guaranteed. If the specified font is not available on the platform, it falls back to the platform's default font.

Layout-related

Property	Description	Value type	Default value	Optional values	Syntax	Notes
lines	Number of text lines	int	0, which means no line limit		lines: 10;	
text-align	Text alignment	string	left	left, center, right	text-align: center;	
text-overflow	Sets the ellipsis style for overflowing content	string	clip	clip, ellipsis	text-overflow: clip;	`ellipsis` is currently supported only for single-line text. **Note** On some Android phones, the ellipsis may appear in the middle of the text. This is determined by the operating system.

line-height	Sets the line height of the text	Length unit + number	0		line-height: 12px;	• If a number is specified, the value is `fontSize` × `value`. • If a length unit with a `px` suffix is specified, the value is the `value` itself. • If a percentage is specified, the value is `fontSize` × `value`.
				• normal: Whitespace is collapsed and text wraps automatically. Existing line breaks and paragraph breaks are ignored.		

white-space	Controls the line break and whitespace policy in the text	string	pre-wrap	• nowrap: Collapses whitespace characters. The text does not wrap and is displayed on a single line, which can extend beyond the background box. • pre: Whitespace characters do not collapse, line breaks occur between text segments, and text within each segment does not auto wrap (each segment is displayed on a single line and can extend beyond the background).	white-space: nowrap;	-

				• pre-wrap: Whitespace characters are preserved, line breaks are preserved, and text wraps automatically. pre-line: Whitespace characters are collapsed, line breaks are preserved, and text wraps automatically.		

word-wrap	Controls how lines are broken, such as whether to break words	string	break-word	• normal: Breaks lines at the end of words. Can extend beyond the backgro und box. • break-word: Breaks lines at the end of words. If the line is still too long, it can break within the word. • anywher e: Can break lines at any point.	word-wrap: break-word:	-
word-break	Controls how words are broken when a line breaks	string	None	• normal: Words are not broken at line breaks. Can extend beyond the backgro und box. • break-all: Words can be broken at any point to wrap. • keep-all: Same as `normal` .	word-break;	Does not differentiat e between Chinese, English, and mixed Chinese-English text.

letter-spacing	Controls the spacing between characters. Can be positive or negative.	string	0	normal: No extra space. Single value length unit: Can be positive or negative.	letter-spacing:5px;	-
text-indent	Indentation of the first line of text. Can be positive, negative, or a percentage of the parent element's width.	string	0	Single value length unit + percentage: Can be positive or negative.	text-indent:30%;	-
				baseline|sub|super|length|percentage|top|bottom|middle|top|bottom **Important** The baseline is related to the font. Use this property with caution.		

vertical-align	Vertical alignment style of the text	string	baseline	• `middle` aligns the middle of the element with the baseline of the parent plus half the x-height of the parent. • `baseline` aligns the baseline of the element with the baseline of its parent. The HTML specification does not detail the baseline of some replaced elements, such as `<textarea>`. This means the behavior of these elements with this value may vary across browsers.	vertical-align:middle	-

				• `sub` aligns the baseline of the element with the subscript-baseline of its parent. • `super` aligns the baseline of the element with the superscript-baseline of its parent. • `top` aligns the top of the element and its descendants with the top of the entire line. • `bottom` aligns the bottom of the element and its descendants with the bottom of the entire line. For elements that do not have a baseline, the bottom margin edge is used instead.		

Effects-related

Property	Description	Value type	Default value	Optional values	Syntax	Notes

color	Font color	Color unit	0x000000		color:red;	
					color:#333;	
					color:rgb(255,0,255);	
text-decoration	Text decoration	string	none	underline, none, line-through, overline	text-decoration: underline;	-
text-shadow	Text shadow	Length unit & Color unit		Supports the format `${x} ${y} ${size} ${color}`. `x`, `y`, and `size` are length units, and `color` is a color unit.	text-shadow: 2px 2px 3px gray;	The color defaults to the font color. `x` and `y` are required parameters. Others are optional.
text-shadow-color	Text shadow color	Color unit	Same as `color`		text-shadow-color: blue;	Optional parameter
text-shadow-offset	Text shadow offset	Length unit		-	text-shadow-offset: 2px 2px;	Required parameter
text-shadow-radius	Text shadow radius	Length unit	0		text-shadow-radius: 3px;	Optional parameter

Properties

Property	Description	Value type	Default value	Syntax	Notes
value	The text content of the component	string		<text value="Text content string"> </text>	

line-space	Line spacing, such as 4px	Length unit		<text line-space="4px" ></text>	

Events

The <text> component supports all the events listed in Common Events.

Example

```
<text class="text">
                The Cube engine is a simple, easy-to-use, cross-platform development
solution. It lets you build high-performance, scalable native applications with a web d
evelopment experience. Vue is a lightweight and powerful progressive frontend framework
.
</text>


.text {
    lines: 3;
    color: #666666;
    font-size: 32px
  }
```

Differences from web

The main differences between Ant Dynamic Cards and the web are described below.

- **Ant Dynamic Cards does not differentiate between Chinese and English text when handling `word-break`.**
- **Ant Dynamic Cards does not differentiate between mixed Chinese and English text when a long word overflows the background box.**

Web	Ant Dynamic Card
If `word-wrap` is not specified, the behavior is the same as `word-wrap: normal`.	If `word-wrap` is not specified, the behavior is the same as `word-wrap: break-word`.
`word-wrap: normal`: Words do not break and overflow the background box. A word is defined as a sequence of English characters.	`word-wrap: normal`: Words do not break and overflow the background box. A word is defined as a sequence of English characters and can include Chinese characters.
`word-break: normal`: Chinese text wraps to the next line. English words do not break and overflow the background box.	`word-break: normal`: Chinese text wraps to the next line. English words do not break and overflow the background box.

`word-break: keep-all`: Neither Chinese text nor English words wrap. They overflow the background box.	`word-break: keep-all`: The behavior is the same as `normal`.
-	`letter-spacing` is supported on Android 5.0 and later.

Sample code

- Click detailFontSize.zip to download the sample code for font size.
- Click detailFontWeight.zip to download the sample code for font weight.
- Click detailTextAlign.zip to download the sample code for text alignment.

7.2.1.3. image

The <image> component is a built-in component of the Ant Dynamic Card Engine that renders a single image with specified styles.

Embedding support

This component does not support nesting other components.

Styles

The <image> component supports all styles in Common styles.

Properties

Property	Description	Value type	Default value	Valid values	Syntax
src	The online or local address of the component image, or Base64-encoded data.	string		• URL("https:/xxx"): A CDN address. • URL("./xxx"): A relative address in an offline package. • URL("data:"): Base64-encoded data.	src="https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png"

resize	The position of the image.	string	stretch	stretch|cover|contain|top|bottom|center|left|right|top left|top right|bottom left|bottom right. If a value is outside this range, the default value cover is used.	resize: contain
placeholder	The online address or Base64-encoded data of the component's placeholder image.	string		• URL("https://xxx"): A CDN address. • URL("data:"): Base64-encoded data.	placeholder="https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png"

Events

The <image> component supports all events in Common events.

Example

```
<image class="image" resize="contain"
src="https://gw.alicdn.com/tfs/TB1dZ4WowoQMeJjy0FnXXb8gFXa-950-1267.jpg">
</image>

.image {
  width: 600px;
  height: 400px
}
```

> **Note**
> - The <image> component does not support SVG images.
> - The original image is downloaded in the following cases:
>   - The value of resize is not cover, contain, or stretch.
>   - The width or height of the node is not specified.

Download the complete sample code: detailImage.zip.

7.2.1.4. richtext

<external-richtext> is a built-in component of the Ant Cube Card Engine that renders rich text.

Embedded component support

Nesting other components is not supported.

Styles

The <external-richtext> component supports all common styles and some special styles.

Format

This component supports the HTML tag format, including the following font-related tags.

Tag	Description	Syntax	Notes
br	Line break	 To break lines in a paragraph, use the br tag. 	-
span	Groups inline elements in a document.	`some text.` some other text.	-
div	Divides the document into separate and distinct sections.	`<div style="color:#00FF00">` This is a header This is a paragraph. `</div>`	-
b	Bold text	This is normal text - this is bold text.	-
del	Deleted text	a dozen is `<del>20</del>` 12 pieces	-
h1	Heading 1	This is Heading 1	-
h2	Heading 2	This is Heading 2	-
h3	Heading 3	This is Heading 3	-
h4	Heading 4	This is Heading 4	-

h5	Heading 5	This is Heading 5	-
h6	Heading 6	This is Heading 6	-
i	Italic text	Italicized mailbox cn42ducn4***@163.com mt3@42du.online	-
p	Defines a paragraph	This is some text in a very short paragraph	-
img	Defines an image	src: The download link. width/height: The rendering width and height of the image. 	The default width and height are the same as the font height. If the width and height are set to be greater than the line height, the rendering does not expand the line height.
a	Defines a link	href: The link URL. <a href='https://www.alipay.com'>	-

Font-related styles

Property	Description	Value type	Syntax	Notes
font-size	Font size	Length unit	<span style=\"font-size:30px;\">Inline 30px	-

Effects

Property	Description	Value type	Syntax	Notes

color	Font color	Color unit	<span style=\"color:#F00\">Inline #F00	-
font-weight	Font weight	string	<span style=\"font-weight:100\">Inline #F00	The values are the same as the styles of the text tag.
font-family	Font	string	<span style=\"font-family:thinFont\">Thin font	The values are the same as the styles of the text tag.

Properties

Property	Description	Value type	Default value	Syntax	Notes
text	The value of the component, which is the text content.	string		<external-richtext text="richtextContent"></external-richtext>	-
line-space	Line spacing, such as 4px.	Length unit		<external-richtext line-space="4px"></external-richtext>	-
detectEmotionEmoji	Specifies whether to detect custom emoji.	1/0	0	<external-richtext text="richtextContent" detectEmotionEmoji="1" ></external-richtext>	-
linkColor	Sets the font color for links (a tags).	Color unit	0xff108ee9	<external-richtext text="richtextContent" linkColor="#FF0000"></external-richtext>	-

highlightedColor	Sets the highlight color for clicked links.	Color unit	0xffa9a9a9	<external-richtext text="richtextContent" highlightedColor="#0000FF" > </external-richtext>	-

Note

- If the height that you set for an img tag is greater than the line height, the line height is not expanded during rendering. By default, the image height is the same as the font height.
- If you set the width and height for an img tag, the image is scaled to fill the specified dimensions.
- The font color set for a link within a span tag has a higher priority than the linkColor property.
- The highlightedColor property controls the highlight color, including for links in a tags. If this property is not set, the default color is gray.
- The size of an emoji is the same as the font height.

Events

The <richtext> component supports all common events.

Example

```
<external-richtext
:text="richTextContent"
:line-space="4px"
></external-richtext>

data: {
    richTextContent:
      'Inline 30pxInline 30rpxInline #F00
Global 30px,color:#F00
Global 30rpx,color:#0F0
Inline 30px,color#0F0Global 20px,color#0F0
      <div style=\"color:#F00\"><div><div>Outermost #F00</div></div></div>
      <div><div><div style=\"color:#F00\">Innermost #F00</div></div></div>
      b<del>del</del><div>div</div><h1>h1</h1><h2>h2</h2><h3>h3</h3><h4>h4</h4>
<h5>h5</h5><h6>h6</h6>i<p>p</p>span'
  }
```

Sample code

Click detailRichtext.zip to download the complete sample code.

7.2.1.5. slider

The <slider> component displays multiple images sequentially in a single view.

Embedded component support

The <slider> component supports advanced features but can only nest <cell> child components. The <cell> component defines a child item in the slider. For improved performance, the Ant Dynamic Card engine efficiently revokes the memory of <cell> components.

Styles

Some features in common styles are not supported. These include the padding property in the box model, flex containers and flex members in layouts, background-image properties, hover effects, or animations.

Properties

Property	Description	Value type	Default value	Syntax	Notes
auto-play	Specifies whether to automatically play the image carousel.	boolean	true	auto-play="true"	None
interval	Specifies the time interval for the automatic carousel in milliseconds. This property takes effect only when auto-play is set to true.	number	500 ms	interval="500"	If you set a value less than 500 ms, the interval defaults to 500 ms.
infinite	Specifies whether the carousel loops.	boolean	true	infinite="true"	None
show-indicators	Specifies whether to display indicators.	boolean	false	show-indicators="true"	None

scrollable	Specifies whether you can switch between pages using swipe gestures.	boolean	true	scrollable="true"	None
index	Specifies which page of the slider to display.	number	0	index="2"	None
previous-margin	Specifies the margin to expose the previous page.	Length unit	0	previous-margin="200px"	Cannot be used with `infinite="true"`.
next-margin	Specifies the margin to expose the next page.	Length unit	0	next-margin="200px"	Cannot be used with `infinite="true"`.

Events

Common events are not supported. The following specific events are supported:

Name	Description	Parameters
on-change	This event is triggered when the carousel index changes.	index: The index of the displayed image. The value is a number.

Example

```
<template>
    <div class="root">
        <slider class="testSlider" :index="index" show-indicators="false"
scrollable="true" duration="1000" auto-play=true  @on-change="onChange(index)" >
            <cell class="cell" v-for="(item, i) in imageList">
                <image class="image" resize="contain" :src="item.src"></image>
            </cell>
        </slider>
    </div>
</template>
<script>
export default {
    data: {
    },
}
</script>>
<style>
    .root {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: white;
        width: 100%;
    }

    .image {
        width: 100%;
        height: 100%;
    }

    .cell {
        width: 100%;
        height: 100%;
        background-color: blue;
        flex-direction: column;
        align-items: center;
    }

    .testSlider {
        width: 100%;
        background-color: red;
        margin-top: 100px;
        height: 400px;
    }
</style>
```

7.2.1.6. scroller

The <scroller> component is a container that enables horizontal or vertical scrolling for its child components. It provides smooth scrolling and efficient memory management, making it ideal for displaying long lists.

Embedded component support

You can embed any component.

Styles

The component does not support some common styles. These include padding, flex container and member layout properties, the background-image property, hover effects, and animations.

Properties

Property	Description	Value type	Default value	Optional values	Syntax	Notes
show-scrollbar	Sets whether the scrollbar is displayed.	Boolean	false	-	show-scrollbar="true"	-
scroll-direction	Sets the scroll direction of the component.	String	vertical	vertical horizontal	scroll-direction="vertical"	-
upper-threshhold	Sets the distance from the top or left edge to trigger an event.	String	50 px	-	upper-threshhold="50px"	-
lower-threshhold	Sets the distance from the bottom or right edge to trigger an event.	string	50 px	-	lower-threshhold="50px"	-
offset-accuracy	Specifies the behavior during component scrolling. callback frequency	String	10 px	-	offset-accuracy="10px"	-

allow-bounce	Enables or disables the bounce effect.	Boolean	false	-	allow-bounce="true"	10.2.28 Support
always-bounce	Specifies whether to allow scrolling when the content does not fill the container. Note: This property takes effect only when `allow-bounce` is set to true.	Boolean	false	-	always-bounce="true"	10.2.28 Support

Events

The component does not support common events. The following table lists the supported events.

Name	Description	Parameters
on-scroll	Triggered during scrolling.	contentSize: The width and height of the content area. contentOffset: The offset of the display area.
on-scrollstart	Triggered when scrolling starts.	contentSize: The width and height of the content area. contentOffset: The offset of the display area.

on-scrollend	Triggered when scrolling ends.	contentSize: The width and height of the content area. contentOffset: The offset of the display area.
on-scrolltoupper	Triggered when the scroll position is less than the threshold from the top.	-
on-scrolltolower	Triggered when the scroll position is less than the threshold from the bottom.	-

Example

```
<template>
    <scroller class="root" ref="scroller" show-scrollbar="true" scroll-
direction="horizontal" @on-scroll="onScroll()" @on-scrollstart="onScrollStart()" @on-sc
rolltoupper="onScrollToUpper()" @on-scrollend="onScrollEnd()" @on-
scrolltolower="onScrollToLower()" offset-accuracy="40px">
        <text class="message" :value="message" @click="onClick()"></text>
        <image class="image"
src="https://gimg2.baidu.com/image_search/src=http%3A%2F%2F1812.img.pp.sohu.com.cn%2Fimag
%2Fblog%2F2009%2F11%2F18%2F18%2F8%2F125b6560a6ag214.jpg&refer=http%3A%2F%2F1812.img.pp.so
.com.cn&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=jpeg?
sec=1623901796&t=5e352084411390819560 42a69907f7f5"></image>
        <text class="message" :value="message" @click="onClick()"></text>
        <text class="message" :value="message" @click="onClick()"></text>
        <text class="message" :value="message" @click="onClick()"></text>
    </scroller>
</template>

<script>
    export default {
        data: {
            message: 'Hello Cube 1'
        },
        beforeCreate() {
            this.data.message = 'Hello Cube 2'
        },
        didAppear() {

        },
        methods: {
            onClick() {
                // NOTE: Use act debug to view console logs.
                console.log('invoke on-click event');
            },

            onScroll(data) {
                console.log("onScroll---" + JSON.stringify(data));
            },
```

```
            onScrollStart(data) {
                console.log("onScrollStart---" + JSON.stringify(data));
            },

            onScrollEnd(data) {
                console.log("onScrollEnd---" + JSON.stringify(data));
            },

            onScrollToUpper() {
                console.log("onScrollToUpper---");
            },

            onScrollToLower() {
                console.log("onScrollToLower---");
            },
        }
    }
</script>>

<style>
    .root {
        display: flex;
        flex-direction: row;
        align-items: center;
        background-color: white;
        width: 100%;
        height: 400px;
    }
    .message {
        color: black;
        font-size: 50rpx;
    }
    .image {
        width: 200px;
        height: 400px;
    }
</style>
```

> **Important**
>
> Nesting scrollers with the same scroll direction is not supported.

7.2.2. Component tags (Beta)

7.2.2.1. Development and configuration

> **Note**
>
> - Use ACT 4.0 or later.
> - ACT requires `npm 8.0` or a later version to compile component cards. If your npm version is earlier than 8.0, compilation errors may occur. You must upgrade your npm tool.
>   - Check your current npm version: `npm --version`
>   - Upgrade npm to the latest version: `npm install npm -g`

Overview

A card component is a structured way to organize cards. The component feature of Cube cards lets you break down a large card into multiple atomic components for easier management and maintenance. These components can be referenced by different cards, which improves the reusability of common card effects.

Structure

A component is like a mini-card. It mainly consists of two files: the **main component file (main.vue)** and the **component configuration file (manifest.json)**.

A component has the same capabilities as a card. Basic features such as layout, styling, tags, and logic processing, along with advanced features such as CSS and JavaScript imports, also apply to components.

> **Note**
>
> A component cannot exist on its own. It is only detected and parsed during compilation when it is applied to a card. As a result, the usage of certain file types differs between components and cards.

File	Purpose	Component scenario	Card scenario
.act.config.json	ACT compilation configuration file	Not required	Required
mock.json	Mock data for template previews	Not required. Set it directly in the card's mock.json file.	Required

Usage

Components can be divided into two types based on how they are managed and maintained: **local components** and **cloud components**.

Local components

Developers can maintain the source code for a card's components within the card's source project. This type of component is called a **local component**.

File structure

A local component mainly consists of two files: `main.vue` and `manifest.json`.

```
.
├── main.vue
└── manifest.json
```

Create a component

You can run the act command in the folder where you want to store the component.

```
➜  ~ act init
? Select an application type
  Cube Template Card (VUE format)
❯ Cube Card Component (VUE format)


➜  ~ act init
? Select an application type Cube Card Component (VUE format)
? Enter the component name local-component
? Select a component management method (Use arrow keys)
❯ Built into the card project
  Share to the NPM cloud
```

Associate a component with a card

To use a component in a card, declare the association in the `useComponents` field of the card's `manifest.json` file.

```
# Card's manifest.json file

{
    ...
    "useComponents": {
        "local-component": "./components/LocalComponent",
        "local-component-2": {
            "path": "./components/LocalComponent2",
            "type": "local"
        }
    },
    ...
}
```

```
# Card's main.vue file

<template>
  <div class="root">
   <local-component class="comp"></local-component>
   <local-component-2 class="comp2"></local-component-2>
  </div>
</template>
```

The value of `useComponents` is a key-value pair. The `Key` is the **tag name of the component** used in the card, and the `Value` is the **source information of the associated component**.

The Value supports both **String** and **Object** formats.

- **String**: The path to the component's root directory. This is the directory that contains the component's `manifest.json` file.
  - Relative path: The path to the component's root directory relative to the card's `manifest.json` file.
  - Absolute path: The absolute path to the component's root directory.
- **Object**: The source configuration of the associated component.
  - path: The path to the component's root directory. This is the directory that contains the component's `manifest.json` file. Both relative and absolute paths are supported.
  - type: local. This field is optional. The default value is `local` when you configure a path.

Associate a component with another component

You can associate a component with another component, just as you can with a card. For clarity, the component that uses another component is called the host component, and the component being associated is called the associated component.

To use an associated component in a host component, declare the association in the `useComponents` field of the host component's `manifest.json` file.

```
# Host component's manifest.json file

{
  ...
  "isComponent": true,
  "useComponents": {
    "other-component": "../LocalComponent2"
  },
  ...
}
```

```
# Host component's main.vue file

<template>
  <div class="root">
    <other-component class="comp"></other-component>
  </div>
</template>
```

> **Important**
>
> In the configuration, the relative path of the associated component must be the path to the **root directory of the associated component** relative to the **host component's** `manifest.json` file.

Cloud components

Developers can publish and maintain component source code on an npm package management platform. This type of component is called a **cloud component**.

File structure

A cloud component mainly consists of three files: the **main component file (main.vue)**, the **component configuration file (manifest.json)**, and the **npm configuration file (package.json)**.

```
.
├── main.vue
├── manifest.json
└── package.json
```

> **Note**
>
> For more information about how to use the npm configuration file, see the npm Docs.

Create a component

You can run the act command in the folder where you want to store the component.

```
➜  ~ act init
? Select an application type
  Cube Template Card (VUE format)
❯ Cube Card Component (VUE format)


➜  ~ act init
? Select an application type Cube Card Component (VUE format)
? Enter the component name cloud-component
? Select a component management method
  Built into the card project
```

Associate a component with a card

To use a component in a card, declare the association in the `useComponents` field of the card's `manifest.json` file.

```
# Card's manifest.json file

{
  ...
  "useComponents": {
    "cloud-component": "@ali/card-component-cloud-component",
    "cloud-component-2": {
      "path": "@ali/card-component-cloud-component-2",
      "type": "cloud"
    }
  },
  ...
}
```

```
# Card's main.vue file

<template>
  <div class="root">
   <cloud-component class="comp"></cloud-component>
   <cloud-component-2 class="comp2"></cloud-component-2>
  </div>
</template>
```

The value of `useComponents` is a key-value pair. The `Key` is the **tag name of the component** used in the card, and the `Value` is the **source information of the associated component**.

The Value supports both **String** and **Object** formats.

- **String**: The npm package name of the associated component.
- **Object**: The source configuration of the associated component.
  - **path**: The npm package name of the associated component.
  - **type**: cloud. This field is optional. The default value is `cloud` when you use an npm package name.

Associate a component with another component

You can associate a component with another component, just as you can with a card. For clarity, the component that uses another component is called the **host component**, and the component being associated is called the **associated component**.

To use an associated component in a host component, declare the association in the `useComponents` field of the **host component's** `manifest.json` file.

```
# Host component's manifest.json file

{
  ...
  "isComponent": true,
  "useComponents": {
    "cloud-component": "@ali/card-component-cloud-component"
  },
  ...
}
```

```
# Host component's main.vue file

<template>
  <div class="root">
    <cloud-component class="comp"></cloud-component>
  </div>
</template>
```

Install a cloud component

Unlike local components, cloud components are maintained on the npm platform. Before you compile a card, you must download its cloud component dependencies to your local card project. In npm, this procedure is called installing dependencies.

Before you install component dependencies in a card project, create a `package.json` file in the root directory of the card project. The root directory is the directory that contains the card's `.act.config.json` file.

You can create a `package.json` file using the `npm init` command, or you can manually create a `package.json` file as long as it is in the correct JSON format.

```
➜  ~ npm init -h

npm init [--force|-f|--yes|-y|--scope]
npm init <@scope> (same as `npx <@scope>/create`)
npm init [<@scope>/]<name> (same as `npx [<@scope>/]create-<name>`)

aliases: create, innit
```

When you **first install** an npm package for a cloud component, run the `npm install [<@scope>/]<pkg>@<tag> --save` command in the root directory of the card project. This command automatically places the cloud component package in the `node_modules` folder and adds its dependency information to the `dependencies` field of the project's `package.json` file.

> **Note**
>
> - The `node_modules` folder is a temporary folder generated by the npm package management tool when you run the install command. You do not need to maintain it in the Git repository. Add it to `.gitignore`.
> - To ensure that the component behaves as expected when used by a card, set the cloud component's npm package dependency version to `latest`.

Code example

> **Important**
>
> The package may not be available for download from npm.

```
npm install @ali/card-component-my-cloud-component@latest --save
```

```
# package.json file in the card project's root directory

{
  ...
  "dependencies": {
    "@ali/card-component-my-cloud-component": "latest"
  },
  ...
}
```

To **update** or **reinstall** a cloud component package that is already configured in the `package.json` file, run `npm install` in the root directory of the card project.

> **Note**
>
> To ensure that all configured components are updated when you reinstall, delete the `node_modules` folder before you run `npm install`, or run one of the following commands:
>
> - macOS: `rm -rf ./node_modules && npm install`
> - Windows: `rm ./node_modules -r -fo; npm install`

Publish a cloud component

Publishing a cloud component is the same procedure as publishing a regular npm package. For more information, see the official npm documentation on publishing packages.

> **Note**
>
> When you publish a cloud component, ensure that the required `main.vue` and `manifest.json` files are included in the published package.

7.2.2.2. Component syntax

Component development

The basic syntax is the same as the card development syntax.

If you use `requireModule('animation')` in a component's JavaScript (JS) code, the animation is overwritten if the component's caller also uses this built-in JS module.

> **Note**
>
> This issue will be resolved in a future version of the DPI engine.

Property declaration (props)

A component can use `props` to declare properties that can be modified externally. Properties that are not declared in props cannot be modified externally.

```
props: {
  author: {
    default: function () {
      return { name: 'name' }  // The default value is a JSON object. The key is name a
nd the value is 'name'.
    }
},

props: {
  author: {
     default: "name" // The default value is the string 'name'.
  }
},

props: {
 author: {}  // The default value is undefined.
}

props: [ 'name', 'title' ]; // Both name and title are undefined.
```

> **Important**
>
> - Only the four syntax formats for props are supported.
> - If the props syntax is incorrect, the card cannot transfer data to the component. Properties on the component node in the card will not take effect.
> - If a property on a component node is not declared in the component's props, the property is invalid.
> - A component's data comes from three sources: external input (properties on the component node within the card), default values defined in props, and data in the data segment. The priority is **data** > **external input** > **props default value**. These three data sources are merged. Fields with the same name are overwritten.

Lifecycle

A component lifecycle consists of four stages:

- beforeCreate: This method is executed before the component is created.
- onCreated: This method is executed after the component is created. At this stage, only the JavaScript part is created, and the component has not yet been rendered to the screen.
- onUpdated: This method is called when props are updated. The parameters are the changed field and its new value.
- onDestroyed: This method is called before the component is destroyed. If a component has resources to release, you can implement this method to do so.

> **Important**
>
> - onCreated: This method is similar to the onCreated lifecycle method of a card. The call order is: **onCreated of the child component** > **onCreated of the parent component** > **onCreated of the card** (if components are nested).

- onDestroyed: The onDestroyed method is called directly when a component node is destroyed by vif or vfor logic. If a component is destroyed because a card is released, the call order is: **onDestroyed of the child component** > **onDestroyed of the parent component** > **onDestroyed of the card** (if components are nested).

Using components

Syntax

The following example uses the Button component.

```
<Button :title="title" @buttonClicked="onButtonClicked"><Button>
```

> **Note**
> Only vif, vfor, props, and event can be attached to a component node.

Component notifications to a card (child to parent)

- A component uses the emit method to send an event to its parent card. For example, if a component uses a buttonClicked method (which you must define) to emit an event, you can listen for that event on the component as follows:

```
this.$emit("buttonClicked", {"title":"Test"});
```

- The callback method for an external listener is as follows:

```
<Button :title="title" @buttonClicked="onButtonClicked"><Button>
```

Data update (updating a component externally)

- The component must declare in its props the properties that can be updated externally.

```
props:{
  // Button text
  title:{
    default: "Default"
  }
}
```

- You can update component properties externally.

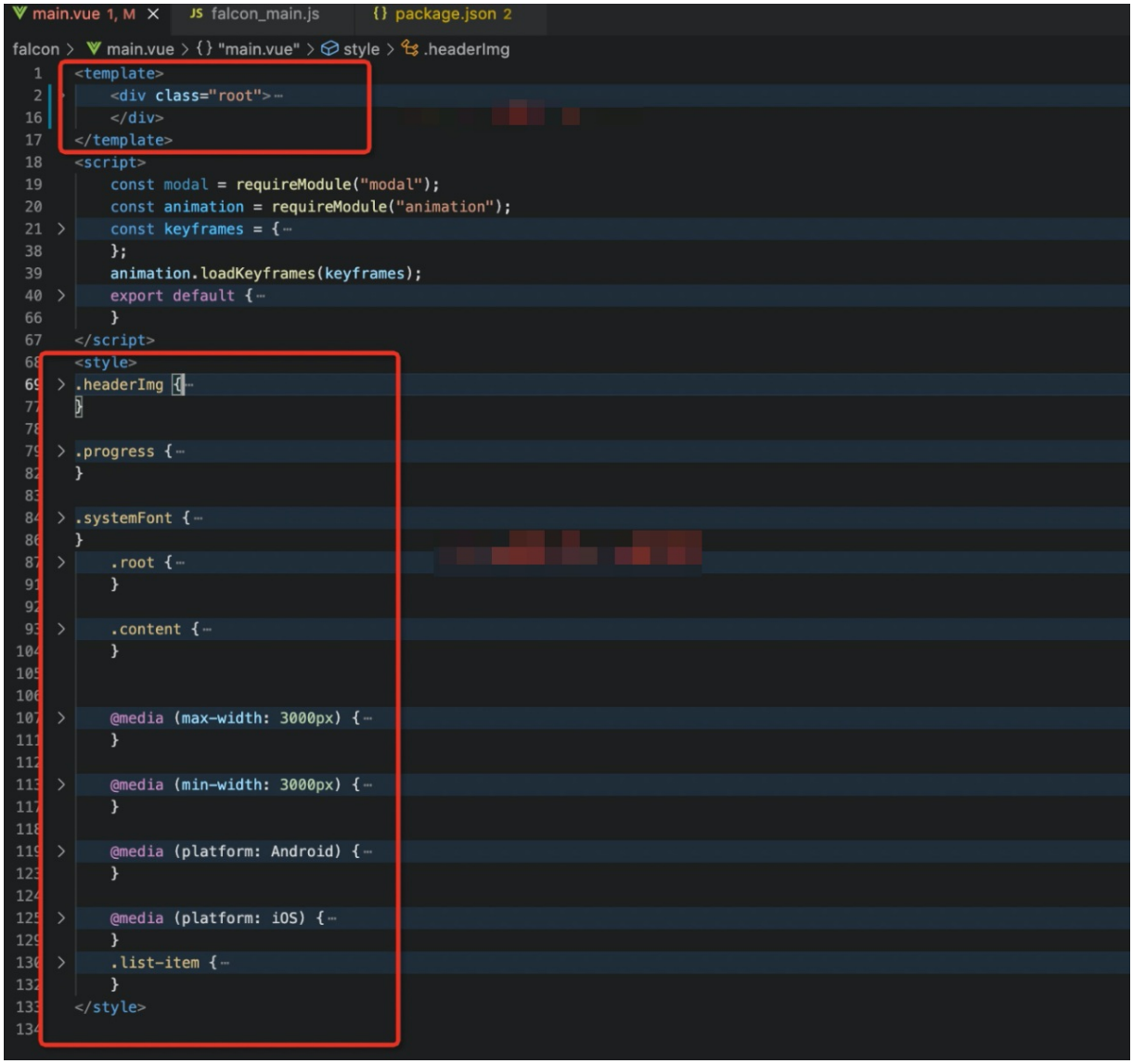
```
<Button v-if="showComponent" :title="buttonTitle" borderRadius="20px"
:buttonColor="buttonColor" @buttonClicked="componentButtonClick()"></Button>
```

7.3. Card Style

7.3.1. Style syntax

In template mode, you can define CSS for page styles in the <style></style> section. The supported styles are based on common styles.

In template mode, you can apply styles to components within the <template></template> section using classes. For example, `<text class="mytext">` .



CSS styles

The `class`, `id`, and `type` selectors are supported. More complex combinations, such as parent-child or state selectors, are not supported. The following code shows examples of the three supported selectors:

```
// class
.class {
}

// id
#id {
}

// type
div {
}
```

Inline styles

Template mode lets you inject styles at runtime using the built-in `style` field of a component.

The syntax for inline styles is the same as in popular frontend frameworks, such as VueJS and ReactJS. Both bound and non-bound formats are supported.

- Bound inline styles
  - You can group the style fields for binding into a single JSONObject.
  - The keys for the style fields must be in camelCase. For example, you must convert `background-color` to `backgroundColor`.
- Non-bound inline styles
  - You can group the style fields for binding into a single string that follows CSS syntax.
  - The keys for the style fields must use hyphen-separated words as specified in CSS, for example, `background-color`.

The order of style priority from highest to lowest is inline style, id, class, and type.

```
// main.vue
<template>
  <div class="root">
    <div class="div1" :style="style"></div>
    <div class="div2" style="width: 100px; background-color: blue"></div>
  </div>
</template>

// mock.json
{
  "style": {
    "height": "100px",
    "backgroundColor": "red"
  }
}
```

For inline styles, the template accepts a JavaScript (JS) object as the attribute value.

```
<div class="root" :style="{height: height}"> // Example from above
```

Dynamic class binding

You can dynamically bind different selectors to a component's CSS styles.

```
<div :class="mydiv">
```

Media queries (@media)

Template mode includes the media query feature from the CSS specification. This feature is mainly used for mobile UI adaptation.

The media query capabilities in template mode are more limited than those in the CSS specification and require specific usage patterns as described in the following sections. For more information about media queries, see Introduction to @media.

- Media types

You do not need to specify the media type. The default value is `all`.

Media type	Supported
all	Yes
screen	No
print	No
speech	No

- Media features

  In cards, media features are based on firmware attributes, which is different from how they work in frontend browsers.

Media feature	Value	Description
platform	ios | android	Adapts to a specific platform. The usage differs from the CSS specification. Set the platform value directly after `@media`. For example, `@media android`.
support	safearea	Adapts to the screen security zone on the iOS platform.

- Media operators

Operator	Support
and	Yes
not	No
only	No

The following example shows how to adapt styles by combining CSS attributes and `@media` queries.

```
<template>
    <div class="banner"></div>
</template>
<style>
    @media android {
        .banner {
                  width: 100px;
                        height: 100px;
            background-color: #00fff0;
        }
    }
    @media ios and (support: safearea) {
        .banner {
            width: 100px;
                        height: 100px;
            background-color: #00fafb;
        }
    }
    .banner {
              width: 100px;
        height: 100px;
        background-color: green;
    }
</style>
```

Importing styles

Before you import styles, you need to understand the following two types of styles:

- Imported styles: These are styles in a .css file that can be centrally managed and imported.
- Scoped styles: These are styles in the <style></style> section of a .vue file that apply only to that template.

Syntax format

The syntax for importing styles is as follows:

```
<style src="[relative path to .css file]" />
```

- File structure

```
.
└── template_name                                                  // Template folder (nam
ed after the template ID)
    ├── main.vue                                                     // Template layout
and style description file
    ├── manifest.json                                              // Template
configuration file
    └── mock.json                                                   // Test data for
template binding
    └── common.css                                                // Common template sty
le file
```

- Template code

```
<template>
        ... [Template layout description]
</template>

<style src="./common.css" />

<style>
        ... [Styles that apply only to this template]
</style>
```

Cascading rules

When compiling style resources in a .vue file, template mode cascades and merges only style fields that share the same selector. Different selectors remain unchanged.

The cascaded result is a combination of imported styles and scoped styles, as shown in the following table.

Imported styles	Scoped styles	Cascaded result
`.special { color: red; width: 100px; } .others { color: red; width: 100px; }`	`.special { background-color: red; width: 200px; } .scoped { background-color: red; width: 200px; }`	`.special { color: red; background-color: red; width: 200px; } .scoped { background-color: red; width: 200px; } .others { color: red; width: 100px; }`

Code example

You can download the FalconDemo code example.

7.3.2. Common style

7.3.2.1. Background

Ant Cube Card provides several properties to customize the background of an element.

Background styles

There are several ways to set the background style of an element:

- background-color defines the background color of an element.

Property	Value type	Default value	Syntax
background-color	Color unit	transparent	background-color: red;

- background-image defines the background image of an element.

Property	Value type	Default value	Accepted values	Syntax	Notes
			`url("https:/xxx")` is used for a CDN address, `url("./xxx")` is used for a relative address of an offline package, and `url("data:")` is used for Base64 encoding. `url("data:") linear-gradient(s1, s2, ..., slast)`	background-image: linear-gradient(45deg, red 0%, #333 50%, rgb(255,0, 255) 80%, green 100%); background-image: linear-gradient(to top, red, #333, rgb(255, 0, 255), green);	

background-image	string	none	The format is linear-gradient(s1, s2, ..., slast). The first parameter specifies the gradient angle, which can be an angle value ending in deg or a direction keyword such as top, to top, right, to right, bottom, to bottom, left, or to left. The second parameter specifies the starting color of the gradient. If a percentage is provided for the starting color, it must be 0%. The intermediate parameters specify the intermediate colors. You can provide percentage values for these colors in increasing order. If you do not provide percentages, the colors are evenly spaced. The final parameter specifies the ending color of the gradient. If a percentage is provided for the ending color, it must be 100%. none: Clears the background.	background-image: url("https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png");	None

background-size	string or length unit	auto	A single keyword: `cover`, `contain`, or `auto`. Two values, such as `${x}px ${y}px`. You can use length units or percentages. ${x}px A single length or percentage value.	background-size: contain; background-size: 100px 200px;	If you specify only one value, the other value defaults to `auto`.
background-position	string	0	A single keyword: `top`, `right`, `bottom`, `left`, or `center`. Two keywords, such as `bottom right`. ${x}px length value / ${y}px length value plus a descriptive value. ${x}px: a length value or a percentage. ${y}px: a length value or a percentage.	background-position: top; background-position: bottom right; background-position: 30px left; background-position: 100px; background-position: 50px 50px;	If you specify only one value, the other value defaults to `center`.
				One value: `background-repeat: repeat-x;`	None

background-repeat	string	repeat	repeat-x, repeat-y, no-repeat, repeat	Two values, such as `background-repeat: repeat no-repeat;`. The first value applies to the x-axis (`repeat` or `no-repeat`) and the second value applies to the y-axis (`repeat` or `no-repeat`).	None

- background shorthand:
  - You can combine `background-color` and `background-image` related properties. The order does not matter.
  - You can combine `background-image` related properties, such as `background-image` and `background-repeat`.
  - Use `none` to clear the background.

Examples

```
background: url('https://img.alicdn.com/tfs/TB1uCUdfND1gK0jSZFyXXciOVXa-151-164.png') r
epeat-x;
background: url('https://img.alicdn.com/tfs/TB1uCUdfND1gK0jSZFyXXciOVXa-151-164.png') #
f0f no-repeat;
background: url('https://img.alicdn.com/tfs/TB1uCUdfND1gK0jSZFyXXciOVXa-151-164.png') #
00f repeat-x bottom;
background: #ff0 url('https://img.alicdn.com/tfs/TB1uCUdfND1gK0jSZFyXXciOVXa-151-164.pn
g') repeat-y right;
```

Basic background usage example

```
div
{
 background-image: url('img_tree.png');
 background-repeat: repeat;
}
```

Important

- When you set both a background image and a gradient, the gradient takes precedence over the image.
- **[v-alipay-10.2.0]** Gradient backgrounds support multiple syntaxes, such as: `background: linear-gradient(#FF6010, 50%, #FFD2B3, #FFF2E9, #FFFFFF);` .

Shadows

You can set the shadow for an element.

Property	Value type	Default value	Valid Values	Syntax	Notes
box-shadow	Length units & color units	none	The format is `${x} ${y} ${size} ${color}` . The x, y, and size values are length units, and the color value is a color unit.	box-shadow: 10px 20px 10px red;	All four values are required.

Important

- The built-in components of Ant Cube Card support this style on both iOS and Android platforms.
- Each element supports only one shadow effect.

Basic shadow usage example

```
div
{
 width: 300px;
 height: 100px;
 background-color: yellow;
 box-shadow: 10px 10px 5px #888888;
}
```

Transparency

Property	Value type	Default value	Accepted values	Syntax
opacity	float	1	A floating-point number from 0 to 1.	opacity: 0.5;

7.3.2.2. filter

The CSS filter property applies graphical effects such as blurs or color shifts to an element. Filters are often used to adjust the rendering of images, backgrounds, and borders.

Supported functions

Function	Meaning	Default value	Value range
grayscale()	Image Grayscale A value of 100% makes the image completely grayscale.	0	0% to 100%
opacity()	The transparency level of the image.	1	0% to 100%
invert()	Inverts the image. A value of 100% completely inverts the image.	0	0% to 100%
sepia()	Converts the image to sepia. A value of 100% makes the image completely sepia.	0	0% to 100%
saturate()	The saturation of the image. A value of 0% makes the image completely unsaturated. A value of 100% leaves the image unchanged. Other values are linear multipliers for the effect. Values over 100% provide higher saturation.	1	0% to +∞
contrast()	The contrast of the image. A value of 0% renders the image completely black. A value of 100% leaves the image unchanged. Values over 100% lower the image contrast.	1	0% to +∞

brightness()	Applies a linear multiplier to the image, making it appear brighter or darker. A value of 0% creates a completely black image. A value of 100% leaves the input unchanged. Other values are linear multipliers for the effect. A value greater than 100% provides a brighter result.	1	0% to +∞

Unsupported functions

Compound functions and the blur, drop-shadow, hue-rotate, or url functions are not supported.

Example

```
<template>
  <div class="root">
    <text>normal</text>
    <image
      class="image-normal"
      src="https://pic49.photophoto.cn/20181202/0021033888940147_b.jpg"
    ></image>

    <text>grayscale:100%</text>
    <image
      class="image-gray"
      style="bg"
      src="https://pic49.photophoto.cn/20181202/0021033888940147_b.jpg"
    ></image>
  </div>
</template>
<script>
export default {
  data: {},
  methods: {},
};
</script>
<style>
.root {
  display: flex;
  align-items: center;
  justify-content: center;
}
.image-normal {
  flex-direction: column;
  flex-shrink: 0;
  align-content: auto;
  width: 350rpx;
  height: 350rpx;
}
.image-gray {
  flex-direction: column;
  flex-shrink: 0;
  align-content: auto;
  width: 350rpx;
  height: 350rpx;
  filter: grayscale(100%);
}
</style>
```

Click detailImageFilter.zip to download the complete sample code.

7.3.2.3. Box model

The box model in Ant Dynamic Cards is based on the CSS box model, which represents all elements as rectangular boxes. Other styles determine the size, position, and properties of these boxes, such as color, background, and border.

The box model describes the space that an element occupies. Each box has four edges: the margin edge, border edge, padding edge, and content edge.

Margin

The margin is the empty space between elements and is controlled by the `margin` property. Both shorthand and longhand formats are supported.

Property	Description	Value type	Default value	Optional values	Syntax	Notes
					margin: 10px 10px 10px 10px;	The four values correspond to the top, right, bottom, and left margins.

margin	Margin	Length unit	0	auto; A single length unit or a percentage	margin: 10px 10px 10px;	The three values correspond to the top, left and right, and bottom margins.
					margin: 10px 10px;	Specifies the distance for the top and bottom borders, and the left and right borders, respectively.
					margin: 10px;	The four margins are the same.
margin-left		Length unit	0	auto; A single length unit or a percentage	margin-left: 10px;	-
margin-right		Length unit	0	auto; A single length unit or a percentage	margin-right: 10px;	-
margin-top		Length unit	0	auto; A single length unit or a percentage	margin-top: 10px;	-
margin-bottom		Length unit	0	auto; A single length unit or a percentage	margin-bottom: 10px;	-

Padding

The padding is the distance between the content and the border and is controlled by the `padding` property. Both shorthand and longhand formats are supported.

Property	Description	Value type	Default value	Optional values	Syntax	Notes
padding	Padding	Length unit	0	auto; A single length unit or a percentage	padding: 10px 10px 10px 10px;	The four values correspond to the top, right, bottom, and left padding.
					padding: 10px 10px 10px;	The three values correspond to the top, left and right, and bottom padding.
					padding: 10px 10px;	The two values correspond to the top and bottom, and the left and right padding.
					padding: 10px;	The four sides have the same padding.
padding-left		Length unit	0	auto; A single length unit or a percentage	padding-left: 10px;	-
padding-right		Length unit	0	auto; A single length unit or a percentage	padding-right: 10px;	-
padding-top		Length unit	0	auto; A single length unit or a percentage	padding-top: 10px;	-

padding-bottom		Length unit	0	auto; A single length unit or a percentage	padding-bottom: 10px;	-

Content margin

The content area is the space that remains after subtracting the border and padding from the total width or height. The formula is: Content Area = Width/Height - Border - Padding.

Width and height

For cards, the `box-sizing` property supports only the `border-box` value. This means that the `width` and `height` properties set the width and height of the border area.

Property	Description	Value type	Default value	Optional values	Syntax
width	Element width	Length unit	0	auto; A single length unit or a percentage	width: 100px;
min-width	Minimum width limit	Length unit	-	-	min-width: 50px;
max-width	Maximum width limit	Length unit	-	-	max-width: 200px;
height	Element height	Length unit	0	auto; A single length unit or a percentage	height: 100px;
min-height	Minimum height limit	Length unit	-	-	min-height: 50px;
max-height	Maximum height limit	Length unit	-	-	max-height: 200px;

Border

You can specify the style of the border, including its width, color, style, and radius. Shorthand properties are supported, such as `border`, `border-left`, `border-top`, `border-bottom`, `border-right`, `border-style`, `border-width`, `border-color`, and `border-radius`.

Property	Description	Value type	Default value	Optional values	Syntax	Notes

border	Border	string	none	-	border: 1px solid #f32600;	The width, line style, and color can be in any order.
border-left	Left border	string	none	-	border-left: 1px solid #f32600;	
border-right	Right border	string	none	-	border-right: 1px solid #f32600;	
border-top	Top border	string	none	-	border-top: 1px solid #f32600;	
border-bottom	Bottom border	string	none	-	border-bottom: 1px solid #f32600;	
border-style	Border style	string	none	dotted solid dashed none	border-style: solid dotted dashed solid	The four values correspond to the style of the top, right, bottom, and left borders.
					border-style: solid dotted solid	The three values correspond to the style of the top, left and right, and bottom borders.
					border-style: solid dashed	The two values correspond to the style of the top and bottom, and the left and right borders.

					border-style: solid	The four borders have the same style.
border-left-style		string	none		border-left-style: solid	-
border-top-style		string	none		border-top-style: solid	-
border-right-style		string	none		border-right-style: solid	-
border-bottom-style		string	none		border-bottom-style: solid	-
border-width	Border width	Length unit	3px	-	border-width: 1px 1px 1px 1px	The four values correspond to the width of the top, right, bottom, and left borders.
					border-width: 1px 1px 1px	The three values correspond to the width of the top, left and right, and bottom borders.

					border-width: 1px 1px	The two values correspond to the width of the top and bottom, and the left and right borders.
					border-width: 1px	The width of the four borders.
border-left-width		Length unit	3px	-	border-left-width: 1px	-
border-right-width		Length unit	3px	-	border-right-width: 1px	-
border-top-width		Length unit	3px	-	border-top-width: 1px	-
border-bottom-width		Length unit	3px	-	border-bottom-width: 1px	-
border-color		Color unit	0x000000	-	border-color: red #333 rgb(255, 255, 0) green	The four values correspond to the color of the top, right, bottom, and left borders.
					border-color: red #333 rgb(255, 255, 0)	The three values correspond to the color of the top, left and right, and bottom borders.

	Border color				border-color: red #333	The two values correspond to the color of the top and bottom, and the left and right borders.
					border-color: red	The color of the four borders.
border-left-color		Color unit	0x000000	-	border-left-color: red;	-
border-right-color		Color unit	0x000000	-	border-right-color: red;	-
border-top-color		Color unit	0x000000	-	border-top-color: red;	-
border-bottom-color		Color unit	0x000000	-	border-bottom-color: red;	-
border-radius		Length unit	0	-	border-radius: 10px 10px 10px 10px;	See the description of `border-radius` values below this table.
					border-radius: 10px 10px 10px;	
					border-radius: 10px 10px;	
					border-radius: 10px;	

border-top-left-radius	Border radius	Length unit	0	-	border-top-left-radius: 10px;	-
border-top-right-radius		Length unit	0	-	border-top-right-radius: 10px;	-
border-bottom-left-radius		Length unit	0	-	border-bottom-left-radius: 10px;	-
border-bottom-right-radius		Length unit	0	-	border-bottom-right-radius: 10px;	-

The values for `border-radius` are described as follows:

- One value: Sets the radius for all four corners.
- Two values: The first value is for the top-left and bottom-right corners. The second value is for the top-right and bottom-left corners.
- Three values: The first value is for the top-left corner. The second value is for the top-right and bottom-left corners. The third value is for the bottom-right corner.
- Four values: The values are for the top-left, top-right, bottom-right, and bottom-left corners, in that order.

Basic usage of borders

The following example shows how to use borders.

```
div {
    border-style:solid;
    border-color:#ff0000;
    border-width:10px;
    border-radius:5px;
}
```

Note

- When you use a shorthand property, any styles that you do not specify are set to their default values. For example, `border: 5px red;` has no effect because the border-style property defaults to none, whereas `border: 5px solid;` displays a 5px black solid border because the border-color property defaults to black.
- When shorthand and longhand properties are used together, the property that is declared later takes precedence. For example:

```
// Displays a 5 px black solid border (black overwrites red)
border:5px red solid;
border-color:black;

// Displays no border (none overwrites dotted)
border-style:dotted;
border:5px red
```

Basic usage of the box model

The following example shows how to use the box model.

```
div {
    background-color: lightgrey;
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

Sample code

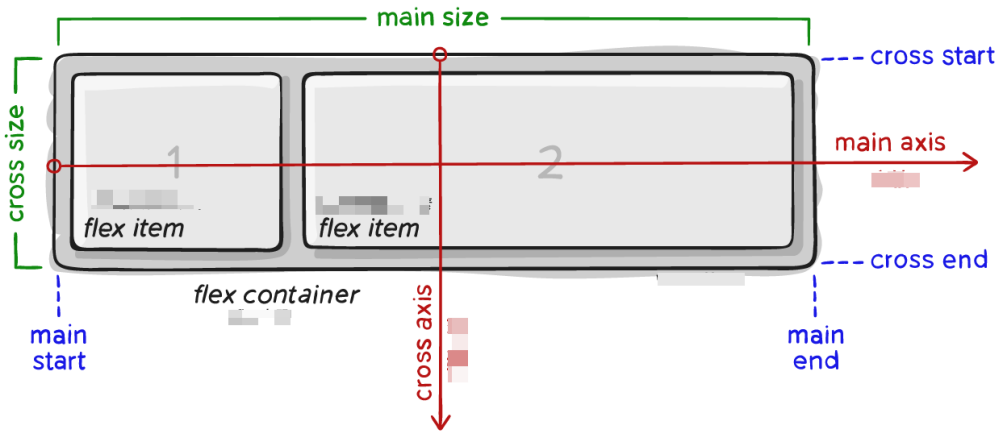
Click detailBoxModel.zip to download the complete sample code.

7.3.2.4. Layout

This topic describes the layout styles in Ant Dynamic Cards.

Flexbox

The card layout model is based on CSS Flexbox, a one-dimensional layout model. This model ensures that the layout of all page elements is consistent and predictable, and can adapt to different devices or screen sizes.



Flex containers

Flexbox is the only layout model used in cards. You must set display:flex. A card element becomes a Flex container if it can nest other elements.

Property	Description	Value type	Default value	Optional values	Syntax	Remarks
display	Flex layout	string	flex	flex	display:flex	Explicitly specify the node as a flex layout.

The main difference from web development is the calculation of text nodes. In cards, the size of a text node cannot exceed the size of its parent node.

Property	Description	Value type	Default value	Optional values	Syntax	Remarks

flex-wrap	Determines whether flex members are distributed on a single line or multiple lines.	string	nowrap	wrap, nowrap	flex-wrap:wrap;	
flex-direction	Defines the arrangement direction of flex members.	string	column	column, row, row-reverse, column-reverse	flex-direction:row;	
align-items	Defines how flex members are arranged on the cross axis to handle white space.	string	stretch	stretch, center, flex-start, flex-end, baseline	align-items:center;	
align-content	Defines how space is distributed between and around content items along the cross axis.	string	flex-start (web: stretch)	auto, stretch, center, flex-start, flex-end	align-content:center;	This property has no effect when flex-wrap is set to nowrap.
justify-content	Defines how flex members are arranged on the primary axis to handle white space.	string	flex-start	flex-start, flex-end, center, space-between, space-around	justify-content:center;	

Flex members

The differences from the web are as follows:

- Text node calculation: If the width and height properties are not set, the size of a text node in a card is calculated by Yoga. The result is then adjusted based on flex constraints. On the web, a text node's size is calculated based on its content and is not subject to flex constraints.
- flex-basis difference: In a card, if the width and height properties are not set, the flex-basis value is used as the initial value for flex style calculations, and the actual content size is ignored. On the web, the layout is based on the actual content size, and flex constraints are not considered.

Property	Description	Value type	Default value	Optional values	Syntax	Remarks
flex	Defines the size of a flex member relative to the remaining space in the container.	Length unit or percentage	0		flex:1; flex:1 1 30px;	Supports the shorthand for flex: <flex-grow> | <flex-shrink> | <'flex-basis>.
flex-grow	Defines the stretch ratio of a flex member when there is available space.	Length unit	0		flex-grow: 1;	
flex-shrink	Defines the ability of a flex member to shrink.	Length unit	0 (web: 1)		flex-shrink: 1;	
flex-basis	Defines the default size of a flex member before the remaining space is distributed.	Length unit or percentage	auto	auto, pixel value, percentage	flex-basis: auto; flex-basis: 50px; flex-basis: 30%;	

align-self	Lets an individual flex member overwrite the default alignment.	string	auto	auto, center, stretch, flex-start, flex-end	align-self:flex-start;	

Positioning

The `position` property sets the positioning type for an element. You can then use the `top`, `bottom`, `left`, and `right` properties to set the element's coordinates.

Property	Description	Value type	Default value	Optional values	Syntax
position	Positioning type	string	relative	relative, absolute, fixed	position: fixed;
top	Offset from the top	Length unit or percentage	0		top: 10px;
bottom	Offset from the bottom	Length unit or percentage	0		bottom: 10px;
left	Offset from the left	Length unit or percentage	0		left: 10px;
right	Offset from the right	Length unit or percentage	0		right: 10px;

Other

Property	Description	Value type	Default value	Optional values	Syntax
overflow	Controls whether content is clipped when it overflows the element's box.	string	visible	visible, hidden	overflow:hidden;
visibility	Specifies whether an element is visible.	string	visible	visible, hidden	visibility:hidden;

Basic usage of Flexbox

The following is a basic example of how to use Flexbox.

> **Important**
>
> - When you use the shorthand for a property, any styles not included in the shorthand are set to their default values.
> - If you define both shorthand and non-shorthand versions of a property, the one that appears later overwrites the earlier one.

```
.flex-container {
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
    flex-direction: row;
}

.flex-item {
    background-color: cornflowerblue;
    width: 100px;
    height: 100px;
    margin: 10px;
}


<div class="flex-container">
  <text class="flex-item">flex item 1</text>
  <text class="flex-item">flex item 2</text>
  <text class="flex-item">flex item 3</text>
</div>
```

Sample code

Click detailFlex.zip to download the complete sample code.

7.3.2.5. Hover

Ant Cube Card allows you to configure the hover property and use the hover property to configure the node style. When the gesture clicks on a node with the hover configuration, the node displays the style configured by the hover property. The original style is restored when the gesture leaves the node.

Styles

Currently, the following two hover styles are supported:

Property	Data type	Default value	Writing	Remarks
background-color	color unit	0	background-color:red;	The change in the background color when a hover occurs.
color	color unit	-	color:rgb(255,0,255);	The font color changes when a hover occurs.

Examples

```
<text
  class="normal-text"
  value="06.hover + touchcancel"
  :hover="hoverDic"
></text>

data: {
    hoverDic: {
      backgroundColor: "#7b8b6f",
      color: "white"
    }
 }
```

Precautions

When you use the hover property, note the following points:

- The does not support gesture movement to detect hover changes, that is, hover only supports click state changes.
- Upward pass-through is not supported. That is, leaf nodes respond to hover events first, and at most one node responds to hover events at the same time.

  For example, if both nodes A and B have the hover property and B is a child node of A, when B triggers a hover event, A no longer triggers a hover event.
- Follow the touch event blocking rules. That is, if a child node responds to a hover or touch event, the parent node no longer responds to the hover or touch event.

  For example, there are two nodes A and B, and B is a child node of A. If B responds to a hover event or a touch event, A no longer responds to a hover event or a touch event.
- Platform differences. Affected by platform differences in existing gesture capabilities, the response of a node with the hover property during animation (displacement animation) behaves differently on different platforms. The Android platform supports the hover response during the displacement animation track; the iOS platform does not support the hover response during the displacement animation track.

Example code

Click here detailHover.zip for the complete example code.

7.3.2.6. Animation

Ant Dynamic Card supports animation properties. These properties allow you to gradually change an element's size, rotation, translation, and color from one value to another.

Transition

Property value	Value type	Default value	Optional values	Syntax
transition-property	string	Empty	background-color, opacity, transform, all	transition-property: all; transition-property: background-color, opacity; transition-property: background-color, opacity, transform;
transition-duration	number	0		transition-duration: 200;
transition-delay	number	0		transition-delay: 200;
transition-timing-function	string	ease	ease, ease-in, ease-out, ease-in-out, linear, cubic-bezier(x1,y1,x2,y2)	transition-timing-function: ease-in; transition-timing-function: cubic-bezier(0.3, 0.3, 0.9, 0.9);

Transition example

```
.panel {
    margin: 10px;
    top:10px;
    align-items: center;
    justify-content: center;
    transition-property: background-color;
    transition-duration: 0.3s;
    transition-delay: 0s;
    transition-timing-function: cubic-bezier(0.25, 0.1, 0.25, 1.0);
  }
```

Transform

Property	Value type	Default value	Optional values	Notes

transform	string		translateX({<length/percentage>}) |translateY({<length/percentage>}) |translateZ({<length>}) |translate({<length/percentage>} {<length/percentage>}) |translate3D({<length>}, {<length>},{<length>}) {<length/percentage>}) |scaleX(<number>) |scaleY(<number>) |scale(<number>) |rotate(<angle/degree>) |rotateX(<angle/degree>) |rotateY(<angle/degree>) |rotateZ(<angle/degree>) |rotate3D(<angle/degree>, <number>, <number>,<number>) |transform-origin (center) |matrix(n,n,n,n,n,n)	translateX({<length/percentage>}): Translates the element along the X-axis. Supports length units or percentages. translateY({<length/percentage>}): Translates the element along the Y-axis. Supports length units or percentages. translate({<length/percentage>} {<length/percentage>}): Translates the element along the X-axis and Y-axis. This is a shorthand for translateX and translateY. scaleX(<number>): Scales the element along the X-axis. The value is a number that represents the scaling factor. Percentages are not supported. scaleY(<number>): Scales the element along the Y-axis. The value is a number that represents the scaling factor. Percentages are not supported. scale(<number>): Scales the element along the X-axis and Y-axis. This is a shorthand for scaleX and scaleY. rotate(<degree>): Rotates the element around a fixed point specified by the transform-origin property without deforming it. The specified angle defines the amount of rotation. A positive angle rotates the element clockwise. A negative angle rotates it counter-clockwise. transform-origin: Sets the origin for an element's transformations. Only `center` is supported. matrix: A 2D transformation matrix.

				translateZ and rotateZ take effect only when transform-style is set to preserve-3d. The translateZ property does not support percentage values.
transform-origin	string	center	left, right, top, bottom, center, or a numeric value. Both single-value and double-value syntax are supported.	
transform-style	string	flat	preserve-3d, flat	
perspective	length	none	none | <length>	The value must be positive. A negative value or 0 has the same effect as none.
perspective-origin	string	center	left, right, top, bottom, center, or a numeric value. Both single-value and double-value syntax are supported.	

Transform example

```
.transform {
    align-items: center;
    transform: translate(150px, 200px) rotate(20deg);
    transform-origin: 0 -250px;
    border-color:red;
    border-width:2px;
  }
```

3D animation example

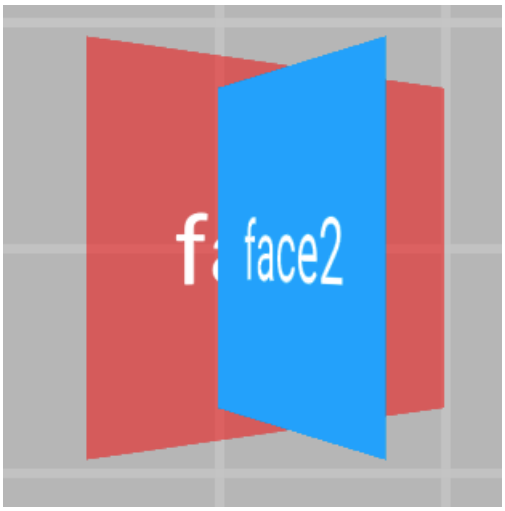
The following code provides a complete example.

```
.div {
    width: 300px;
    height: 300px;
    transform-style: preserve-3d;
    transform: rotateX(45deg) rotateZ(30deg) translateZ(-50px);
    perspective: 600px;
}
```

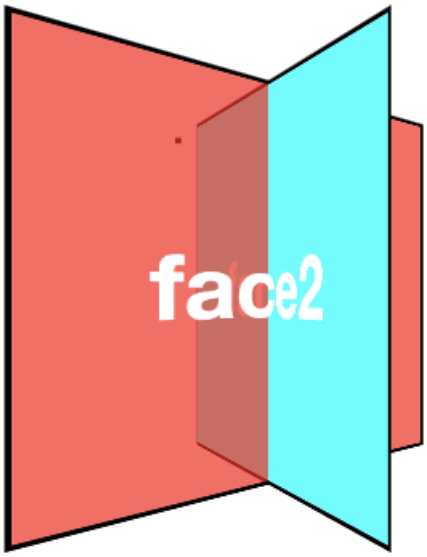
3D animation constraints

- Animation nesting is limited to two layers. This means that a parent node and a child node can have animations at the same time. If a parent node, a child node, and a grandchild node all have 3D animations, the animation of the grandchild node may not render as expected.

- Due to platform limitations, a View cannot be split on Android clients. This means a View can only be fully displayed or completely covered by another element. The following figure shows an example.



On Android, face2 completely covers face1.



The following figure shows the expected effect on CSS and iOS.

Animation

Property	Value type	Default value	Optional values	Syntax
animation-name	string			animation-name: demo;
animation-duration	number	0		animation-duration: 100;

animation-delay	number	0		animation-delay: 200;
animation-timing-function	string	ease	ease, ease-in, ease-out, ease-in-out, linear, cubic-bezier(x1,y1,x2,y2)	animation-timing-function: ease-in; animation-timing-function: cubic-bezier(0.3, 0.3, 0.9, 0.9);
animation-iteration-count	number		A number infinite (equivalent to 9999)	animation-iteration-count: infinite; animation-iteration-count: 10;
animation-direction	enum	normal	normal, alternate	animation-direction: alternate;
animation-fill-mode	enum	forwards	forwards, backwards, both, none	animation-fill-mode: backwards;

Animation example

```
.moving-node01 {
  width: 200rpx;
  height: 100rpx;
  background-color: red;
  margin-top: 50rpx;
  animation-name: moving-horizontal;
  animation-duration: 5000ms;
  animation-delay: 2000ms;
  animation-timing-function: ease;
  animation-iteration-count: infinite;
  animation-direction: normal;
  animation-fill-mode: forwards;
}
```

Keyframe animations

```
<template>
  <div class="root">
    <div class="line">
      <div class="subline"></div>
    </div>
  </div>
</template>

<script>
```

```
        const animation = requireModule("animation"); // Get the module.
    const keyframes = {
        'moving-horizontal': {
            "transform": [
                {
                    "p":0,
                    "v":"translateX(-200px)"
                },
                {
                    "p":0.5,
                    "v":"translateX(-100px)"
                },
                {
                    "p": 1.0,
                    "v": "translateX(0px)"
                }
            ]
        }
    };
    animation.loadKeyframes(keyframes); // Load the module.
</script>


<style>
    .root {
        display: flex;
        align-items: center;
        justify-content: center;
    }

    .line{
        width:200px;
        height:10px;
        overflow: hidden;
        background-color:gray;
    }
    .subline{
        transform:translate(-200px,0px);
        width:200px;
        height:10px;
        background-color:red;

        animation-name: moving-horizontal;
        animation-duration: 2000ms;
        animation-delay: 000ms;
        animation-timing-function: linear;
    }
</style>
```

Animation end callback

You can define the @on-animationEnd event for a node. When the animation ends, a callback is triggered and passes the parameter `{"status":"finish/interrupt"}` to the corresponding method. The value `finish` indicates that the animation completed normally. The value `interrupt` indicates that the animation was interrupted or canceled.

Example:

```
<template>
  <div class="root">
    <div class="anim_node" @on-animationEnd="onAnimationEnd()"></div>
  </div>
</template>

 <script>
   ...
  methods: {
    onAnimationEnd(param){
      if(param.status == "finish") {
         console.info("Animation finished.");
      } else if (param.status == "interrupt") {
         console.info("Animation interrupted or canceled.");
      }
    },
  },
};
</script>

<style>
  ...
</style>
```

Notes

Note the following when you use animation properties:

- The root node does not support animations.
- Entity widgets and external widgets, such as input and slider, do not support animations.
- The skew animation is not supported. You can use the matrix property to achieve a similar effect.
- Applying a CSS animation while a keyframe animation is running has no effect.
- On iOS, gestures are not supported during an animation. The system responds to gestures only after the animation is complete.

Sample code

Click detailTransitionAnimation.zip to download the complete sample code.

7.3.2.7. Accessibility

Ant Cube Card provides accessibility mode.

Accessibility mode supports the following properties:

Property	Description	Type	Valid value
role		string	input list slider switch header button img link search
aria-label	Used to describe the tags added to the current element	string	
aria-hidden	Used to hide an element from being recognized	string	
aria-disabled	Used to control whether the element is active	string	

Examples:

```
<div class="scroll">
    <text class="case_title"> This is an example </text>
    <image class="image" :src="src" aria-label="This is an image"></image>
    <div class="div" :aria-label="This is a div" ></div>
    <div class="row" v-for="row in rows">
      <text class="rowtext"  >{{row}}</text>
    </div>
    <text class="case_title" :role="role"> This is an example to test the role</text>
    <text class="case_title" :aria-disabled="true" :role="role"> This is an example. Te
st aria-disabled</text>
    <text class="div" :aria-hidden="true" value="test aria-hidden"></text>
</div>
```

Example code

Click here detailBarrierFree.zip for the complete example code.

7.4. JS Capabilities

7.4.1. JS capabilities

This topic describes how to use JS capabilities in Ant Cube Card.

If the card requires JS capabilities, you only need to write JS code in the <script> section of the template.

- Add the script to the vue file in the following format:

```
<template>
    <div class="root">
        <text class="message" :value="message" @click="onClick()"></text>
    </div>
</template>

<script>
    export default {
        data: {
            message: 'Hello Cube 1'
        },
        beforeCreate() {
            this.message = 'Hello Cube 2'
        },
        didAppear() {

        },
        methods: {
                        // methods is a custom JS method inside and a card lifecycle m
ethod outside.
            onClick() {
                console.info('invoke on-click event');
            }
        }
    }
</script>

<style>
    .root {
        display: flex;
        flex-direction: column;
        justify-content: center;
        align-items: center;
        background-color: white;
        width: 100%;
        height: 400rpx;
    }
    .message {
        color: black;
        font-size: 50rpx;
    }
</style>
```

- Print logs

  Currently, `console.info`, `console.warn`, and `console.error` are supported to print logs. For more information about the usage, see the preceding example.

Example code

Click here detailJSCapacity.zip for the complete example code.

7.4.2. Lifecycle settings

The card provides a variety of JS lifecycle functions. If you implement the following lifecycle function in the template, you can call the method at the corresponding time.

beforeCreate

beforeCreate is a lifecycle function provided during the initialization of the JS environment before the template node is created. Its main function is to preprocess the data sent by the server.

The beforeCreate lifecycle method has the following limits:

- JS APIs cannot be called.
- You cannot use asynchronous methods, such as `setTimeout setInterval`.
- `beforeCreate` method has no parameters and no return value.
- If the field sent by the server does not exist, you must `try catch` the field when you use it. Otherwise, the template rendering fails when an exception occurs.

onCreated

This method is an interface that is called after the template node is created. The writing method of JS is unlimited, and a template is called only once. Also, do not modify DOM nodes within this method, which may cause concurrency problems.

didAppear

This method is called when the template view enters the screen. If you enter the screen multiple times, it will be called multiple times.

didDisappear

This method is called when the template view leaves the screen. If you leave the screen multiple times it will be called multiple times.

onBackground

This method is called when the application enters the backend. If you enter the backend multiple times, it will be called multiple times.

onForeground

This method is called when the application enters the frontend. If you enter the frontend multiple times, it will be called multiple times.

onUpdated

This method is called when the card is updated. The field whose parameter is changed.

onDestroyed

This method is called when the card is destroyed.

The onDestroyed method does not have the ability to call JS APIs, component methods, and timers with asynchronous logic. If necessary, it can be used to print a log console.info or cancel business logic such as timers.

Precautions

The preceding lifecycle methods only support the following methods, but do not support arrow functions.

```
onCreated() {
        console.info();
}
```

Example code

Click here detailLifeCycle.zip for the complete example code.

7.4.3. Timer

The card has the ability to delay execution and timing execution. The methods of delayed execution and timed execution are described below, respectively.

setTimeout

This method allows a function to be called or a section of code to be executed after a period of time.

This use case indicates that the arrow function is executed after 1 second.

```
setTimeout(() => {
  console.info("setTimeout");
}, 1000);
```

setInterval

This method allows a function to be called or a code segment to be executed repeatedly at the same time intervals.

This use case indicates that the arrow function is executed every 1 second.

```
setInterval(() => {
  console.info("setTimeout");
}, 1000);
```

Clear timer

For setTimeout, if you need to cancel the timer before the function is triggered, you need to manually call clearTimeout to cancel the timer. If the trigger is triggered after the function is executed, you do not need to manually clear it.

For setInterval, you must call clearInterval to cancel the timer if canceled. Otherwise, a memory leak occurs.

Example:

```
// setTimeout
var timer1 = setTimeout(() => {
  console.info("setTimeout");
}, 1000);

clearTimeout(timer1);

// setInterval
var timer2 = setInterval(() => {
  console.info("setInterval");
}, 1000);

clearInterval(timer2);
```

Example code

Click here detailTimer.zip for the complete example code.

7.4.4. JS API

7.4.4.1. Dom

The dom module is used to perform some specific operations on the component nodes of the card. For example, you can use it to query specific ref node information.

selectorQuery(queries, callback)

This method can query related information of the node, including information such as the location area and the location of the page.

Parameter	Type	Description	Remarks
queries	array	The array of node information to query. • ref: the node that you want to query. • type: the type of information that you want to query. Valid values: rect, scroll, and viewport. Each element is a key-value pair.	Only scrollable components can be queried for scroll type.

callback	function(e)	The callback that is fired when the execution is complete. e.result_key is result, and value is an array of the queried data.	The value keys of different query types are different. The following signature algorithms require different message digest algorithms: • rect • left • top • bottom • right • width • height • scroll • scrollLeft • scrollTop • viewport • width • height

Example code

```
<template>
    <scroller class="root">
        <text class="message bgColor" ref="text" :value="data" @click="onClick()">
</text>
    </scroller>
</template>

<script>
          // Introduce a module.
    const dom = requireModule("dom");
    export default {
        data: {
            data : "Point I refresh"
        },
        onCreated() {
            dom.selectorQuery([{ref:"text", type: "rect"}, {ref:"text", type:
"viewport"}], (e)=>{
                var results = e.result;
                for (var i = 0; i < results.length; i++) {
                    if (i == 0) {
                        var r = results[0];
                        this.textRect = JSON.stringify(r);
                    }
                    if (i == 1) {
                        var r = results[1];
                        this.textViewPort = JSON.stringify(r);
                    }
                }
            });
        }
    }
</script>

<style>
    .root {
        display: flex;
        flex-direction: column;
        align-items: center;
        background-color: white;
        width: 100%;
        height: 500px;
    }

    .bgColor {
        background-color: gray;
    }

    .message {
        color: black;
        font-size: 50rpx;
    }
</style>
```

Click here detailDom.zip for the complete example code.

7.4.4.2. animation

The animation module provides an interface for running animations on child components. It supports simple transformations on a component, such as changes to its position, size, rotation angle, background color, and opacity.

You can call the loadKeyframes(keyframes) method to load the animation keyframes.

Parameter	Type	Description
keyframes	object	The keyframe parameters. name{string} is the animation name. values{list}_NativeStyleKeyFrameProperty is the data. It is a dictionary in the format {'p':number; 'v':string}, where p represents percent and v represents value.

The keyframes object defines multiple animation effects. Each outer key, such as fluctuate, is the name of an animation effect. The value of the key is a list of changing animation properties, such as transform. The inner 'p' and 'v' keys represent the progress and property value of the animation, respectively. For more information, see Animation.

7.4.4.3. Card communication JSAPI

Import dependency

```
const mp = requireModule("mpaas_jsapi");
```

postNotification

Send a notification.

```
mp.postNotification(
  {
    name:'TEST_EVENT',
    data: {
      hello: 'hello world'
    }
  },
  res=>{

  }
);
```

addNotifyListener

Registration Notifications.

```
    didAppear() {
        mp.addNotifyListener(
            {
                name: 'TEST_EVENT'
            },
            res => {
                this.clientToCube(res)
            }
        )
    },
```

removeNotifyListener

Remove notifications.

```
didDisappear() {
        mp.removeNotifyListener(
            {
                name: 'TEST_EVENT'
            },
            res => {
                console.info(JSON.stringify(res))
            }
        )
    },
```

> **Note**
>
> The client performs verification, and the notification listener will be automatically released after calling MPCube.recycleCard.

7.4.5. Keyframe animation

This topic describes how to implement a JavaScript (JS) keyframe animation, using a click animation on an online payment confirmation page as an example.

1. Import the Animation module outside of `export default` (line 19).
2. Define the animation effects for the keyframe (line 21).
3. Load the animation by calling the `loadkeyframe` method of the `animation` object to load the defined animation content (line 135).
4. Assign the loaded animation to a node (line 159). This example dynamically assigns the keyframe animation. Alternatively, you can write the animation properties directly in the node's class.
5. For more information about the animation properties supported in keyframe animations, see Animations.

The following code is an example:

```
<template>
    <div class="card_root">
        <div class="white_bg"  @click="clickAnimation()">
            <div class = "right_content_div">
```

```
                    <text value="Click the floating water drop to run the animation"></t
ext>
                    <div class="image_tip">
                        <image class="image_tip_icon" :style="imageTipeAnimationStyle" r
esize="cover"
src="
clNwYWNlRGlzcGxheVAzAAAokWNgYFJJLCjIYWFgYMjNKykKcndSiIiMUmB/yMAOhLwMYgwKicnFBY4BAT5AJQwwG
8u8bACKIv64LMOiU1tUm1XsDXYqbw1YuvRJsw1aMArpTU4mQg/QeIU5MLikoYGBhTgGzl8pICELsDyBYpAjoKyJ4D
dD2BtA7CQI+whYTUiQM5B9A8hWSM5IBJrB+API1klCEk9HYkPtBQFul8zigpzESoUAYwKuJQOUpFaUgGjn/ILKosz
BIFR2AopSp45iXr6SgYGRiaMzCAwhyi+nMgOCwZxc4gxJrvMzDY7v////9uhJjXfgaGjUCdXDsRYhoWDAyC3AwMJ3
JBYlgoWYgZgpLY2B4dNyBgbeSAYG4QtAPdHFacZGYHlGHicGBtZ7//9/VmNgYJ/MwPB3wv//vxf9//93MVDzHQaGA
AFSFl7jXH0fsAAAA4ZVhJZk1NACoAAAAIAAGHaQAEAAAAAQAAABoAAAAAAAKgAgAEAAAAAQAAAFCgAwAEAAAAAQAA
UAAAAA2sjhBQAAIoVJREFUeAHNfAt0XdV55r7ve/W4ekvWw5Yf2LzCw2A7hISAEwikmRmSZpXpyiSdaVnTZmaSrGR
m5IUkNoASeTaSkhpU2Tpl2zZhZda2baEgYCxEDAIWCeBmOwsSXL1sOSLOlKurpX93Hm+/6z/6N9ryRjwAa2fe7e+9
/+9/f+ffz3OOQuZ9Gmb33tNZGj+4yQsVi6G2Dc+kz/yjsfejqaH3m1HeGw81zBx98IZYLPsfE7WRFV7ZM/mZ4kB+P
j+Y7L/7L93Otm3k82v68AnHj05q2J2Oh/T3WmLjD1KWPCYR+rYsmYyVmTHcruni813ND08Tsef7+A+L4A8JVXvPiq
+6LZHO/3GsJ500sZgxIcc0D3B5+MnPm/n+iVw2E7ltbM1X7li/fn3+vQbSsfK9MWXkyTvX1Xlv3FXTFb/GNNXB62g
gWQWQGw7Mf0xuMzZubo3EO50rovtV15w+vvjeV+re8pgBOPbf/tVM3EXyR60z0mEfdBcy1SEGkrPVA9EeOimcuZuU
Q3PZ+q+2XHXHve8ViK6575oN3sBAarb/hzcnmgo3RLvTYROJoG6YUm1NhDRcBCwA0IJJWrFgCgMT3uxo6M5C7DPfa
69V2fYKpNPu0gTv/6h2dHzKEfpVbFLzfpmkqvcz0uGjal6ZwJIQ7XJI0pFH3b1AsJKEEso2tjgpnpn92VLXb9p46r
rptDfCqeBdBXDqse2fS6Snf5BYXd8RTBS0wLWCIEYjZuKVw2bwN/tlIu6+9GyTPrMbIGL88+xYqB4p3gna3LzJ9U2
Wbqbmy68s/+HkoJ9WkPrumnrbLxp+5PJ8u/+tN4c/FL6LKYKNBlFTjX6yJh4wGkw7/cY6aPTZl1V15gSqWyOfTIS6
ZYvpufxcY8BjigCMAyIh0u4tXbpoSkenzMwxc89E5PJta7Z+ZvK0NcoqPu0ATjz5wwuTsf4fJXsTl5g6dEUCJhcsc
LRU1ueMIceuhFk2ptML0fOcuECBZm5VKhbPoff9kUJmbM6k9caBLtjcbMFyrHRQWyBC/NzJmZvuyzuXz3F9s+8Y3d
PE0wigF848dsvvJ5tndsR665tNNGq9zgKo/ZbLFlzjL/aZwd0HTM8HN5gmdlcuVwgKA3nQrcdePWKGn91vujefYZr
20MvFMu7c7BuAi5fN7kDk1NzUzWbmu75ra/RuV021MeTguAA7seaG4yv9pR01G6PrQCa7uQ70m+x9kqGQEULzePLv
yWVyZs3HzjPxxlofPMEOPxZDwTsOL4UX9sFLa5tqTc/Hzjch0ARsAQ86XRBLRVMeypjMoPezudiWG7u2fu6U76dPO
Tj9+5JVZz9K9Sa5IbTSrhd1PWol1XfQBddu7ouOnH+Fa/stV0X3ImeFBIr9Kg4AUoooCgwzOP7NpnsoPjphfjZLKz
/S0o3B44LIWXqaXXp2TzbX/cWOq7ftUvWnIj6FAHrhyce/+4c1LbO3x1bVNcjajl1PxjnG1lzpsmEz9uwBc2zPYdP
bNMem2HnWG1SY7nOSRNqjdOHhg2QwCy44Je07xxrb+k4QRD4BVExrwpGDPRpWemMzXfbP/x7XebfwxhjHjnQZv1jj
7NzZGEs8/P3aTu96044uyEMAana9jnl4TzmbNwPwuuJ80fSiC0brucZz2sLGq1VsPIONFqVjETMP7+rHEJBIxqRLh
7yx0I0ZmpQ0HEuOqNTJupkfDPMu2f/lrv+ZdNiM538KOmvm0VIw/ffX59w8DfpNbEt5ga3Y5BbQV4yAO87OFRM/Do
ZhXadZsXmdXye9Q61wgXItErotVB6NuVuBgsGn95vpQ8Nm1dbzTApLHpPHwruiS6OeEoTYpWdyJtOX3Z0pbviDlVd
Y9b1VtNq+lvVU74x3+547O1reN3J1bXtlfOstYDycWlCMIoGnj89UHTfdk5pq63dcFLpBRmyLrXMUcBovtpOohtQv
w5ia6T9mBp941bSe3W1aLz7D77q8QfRAAROxgojJa7ZvemRqpvUPuz+5/Z/FjLfx41j8VqS9yORjt9xY0zF3a6ynN
FG7ot5NXrZNxDHl2M27GBR16Ubr3yig+YCL3UnSiCammKi4imLYPbnaVbKl0VgB9eXsQQcRheTsfsgTdGajCRzWOI
nwEEjWz7iAcbF/Oj+ZSd/Yec2tP7R3URWeVPyWAeTZXc/UzT9Idxf+i2nBoacsUVCXgsf+aNdt2b4Rc+TxvabP7B7
nHNghcEpoH3ZCxwwVOPrAZReWgH0Bt57qCZhMev/Og5pmZlG0BElyYPu7CAaNMYFwuDGTM1mvpu20/vuPmtTi48Bj
MPgvu2tavbt+kl4b+gPTiMFfZ1mNiQZvPfJjv3ndHHvhELZfHzCNZ3YtBk8At1WLxxJJewX5qnIWk4exZ2PLEkTAh
lup5Wk2yuNwO4gZyBa7oxLjKwPAhUFDKR2rhJhrOX/+d1Oxsylz/5yLP33QyUTy6cNIADuwZS6dQ//Cx9RvTfmjq7
PgBAwYg5OTcr5gBh543szP5k3vNRtNohkLae4qgsCG+4YveC0KLUli8kreIQZ5LXRilgWCFiEsZ+LplGlcu8KM7Ok
4dGTHpVmwlhrBQvpIgTQpi96yK5Sy4JPdTwzIZdD/c9enIgStWOniWTO3d60Qsj3/pJ4/rQF0yt63lgV/BgWH5k0h
+EXMsitM+8WcZdlNcAlgVI3q3Ma6tbtpslZ4CvMOQbov8iTJ5aRJcMtZN240Z+lZLNxXX3WBiTVgqcUuTdvkgsNxX
NLo/OmJHh5O1dn9pxE4x9U0/0p0gavGzwQhd43/6zxtXeFwwH5MB7IKDAALyZ/YOm74HnTMfm9aYde9VgrFHPIEAK
JpYbBMyhvYVU1A3iUxXX2CRSAtMF3YYzefu8oc+PluMzd03BhuAUXGtoO9CSuGcEutaWma3Tb485u+HFR9goRrwpJ
x7+zvVNK/M/jnbYBTIrkq5rK8YDoIkXD5nRPX1m1ZUXmmQbjqu0y4qBtgoXeLcmvQkurSJtvUtp4l3MqKf5Sd/rmL
ltN/6GHCiwigTcMLDz/+iln14bNN/WrsgPCgyuezXshlDsbM2YHM/Hi++7O9V3/7PkguG07ogYO//P7F6ZbZH0Tbe
M1orGAhh43vju/Wb05cNm7ac2m2RL/QJ4UqUFT2+1ZlnGNPVJrHknRtIPrhApVkYKHXmf2dcZKLVEYsiAxXU99s1r
6BOfzEXpN5Y8jxROqy+nByVNtZG6/zjvz1wM4fozstH5YFcN8T/1RfGxu6J7GyLu2DByXQH4CIu3n8+YNmbN9RgLf
GvRvdld1FjhtRUzvSgsSVzgWk4+EEMiSENsKX6XJo5I7yqaGszOa9Bb+ne9ikcBo7JelYoJoPaweMw0dyc7ozPP/d
/R4at3RYFsDW3K5t6d7Exf7pMRS73gfPm35t0AxjmbL2kxeZaEoXx4qeU5nKsTEaNK2x0peNq+onWotkFxEWtLFIi
FCSIeoa654nzTh4Pa3GjGB1H48MOYQOI0qa3V+3hy3ze+sqCsMrUkgIMP7Liotnn+qwbrI1FGhRqwVMmPTpkBdIE1
1oYnVYTNPzJNjKNW2pFZHocvlQqvo1VgG5H/zRSwtOEKt+YWFGL0eGIGK5VdfeYLq2nGneePgFU8IzFd12+l0ZcgA
lhjaqLHbxrc9ZdnORqCIA773Xi8Rig7cmu+tqRBHvhAZ4k4f1VT8ONDs3nWFSPFoPwAMTDWMIbFZZjf3iil8WWbEK
```

MidgJZ4akqpy5XrqqYIkEAiM1rOkwak0nfTjzMk1MkCFCGPYcXtofNHcmG0rFXbv0dYBPI2sQiAD9Ud9Mn0u3hayg
qhNuyH2tsO/3ienxtyeBbOtalW+IF+FDA3zW6ccleBVsftMFpHqMs3b4gWFSOmNrCAiI/U7RMrmCmblRetMDhPM2J
/qRCFvLqeIiunK4pfPr79bdcAWpFqADQA8LJxPQ34u21RM1n1ErRdXNYP00cHDErL4U3y8ElLKARywVutxhUx3K8L
LBJwfvSkaa5HqpAFL6Q3KVWCJmEBzmYNetBanREdf6jPzU7NVXRlyWB+mW1MRkzuybTs2Fa6mCgD7k1+/PN0SuUwm
JVGT2I098VOPmVEw5ZX5HHVVeVXqpMvKOqxVXZKi02uxzTUpVAZDl2VR6UI4ED3RR2J83ru82Rp/YBMPY+y0gMeOH
4Zkcev1he98WFUwrgAwGc/+Ubyt1qepAnLB+6YPDuMUuWSaN+BggAtlAqEXeU4UAmPBJHrx49Ioy3w1Teks0DKJNU
pQL0q/1vxqri6Mpd5/Wa6ePTZnZoXMY+AY56eMELG1pSkdL08BdhjGpfAPDV//nV1am60DU8wwsaqWwwYgRrvg4+S
4QKNYVuWhoFSGpYxfilYp5efId1K8yzAtQ15UldSDH/SoCEBqP6fXDD73ht9OYUZD1Qtx4JCI5H/r0M4/71U9gQfW
f/dbqjBvswi5qNiHwW258S1k4Nq3GuJotlVCgVq5olYpYHCEjGYbLCb6ZDJKplrRrxfkdlUBdpqrhKtiobSAs7fnJ
47HDdlM1sxhqSbLGgEPnMQDXbu5MZ4uTRz4NyorAHpwyUSqfK2s+1hCIa0MU/vY3sOmCcdCMjYoXTRUZBZkVLsby9
Epy0llGdXpoI8rYsMMyVd9JBUj2gysagvCpBG3DYGsHqo7G33YzuG5Chq6JRwCWOE/VIcfpaXdIIgC//5IaeRG1sc
lU90QKM/mTBbHVAJg4H2oTButvBKTjgQvbehSfBU8wlylT2mi1PlRuo1Zh5OUtObdMsecgN/RGiihrThIaIMXTg0e
NN7ZDBeIhdWDRc2nR70514fcJOIjW1k1vqm5LY8zqexxuI8W4G3TeKY6wYX0XjzBsYWGHB8nSyqUw1mAHd0aU8Wqb
ndjTQcVWGbq0DLGqlOr0bzLo2VwkhrMyDx4nT2Gd5Pk3UUtRIwema6Lps3Mwc2kigfiROrSSH3VfpnKwTyFJ131XX
T0C1YkpqWGSgtUroYLJZZV827+qplhFhEIWuhTZmpFdQCRMgWpYKckC0hS4P07zQ1voVTWZqAAcN3J3Y4LEATlVbE
efvpDJKPUC8XiZqPsPCjtAoXlSnZsytStAIA8J2PQuyux1ugX+eXk0byWI9YkE4EO5VsiVn65aVZGFbvyUm7lVYZZ
cq9dPVvEG56icb0lgXpgHgDCcSRwYbWZGIwDtDXvFCs90Lh5/422/UxRLhtbI8YZ0aAGQRD6A5+6Ya2H2doyrlESN
hUFeZvWvBjBjBIoY/NKro5FreW3kU9Cxs2rqgp+R45JvZRX5ZXuxjwXwS6L3biAZ8clHrhyaGPQGN06bIprj/3Ooz
lDfYEYmFsT4hEy5VDuY8tjURnPuFE3hdQnYeKJTKbEylkmfChqXyLNI7WQG0FLBw6Ut4WWaDKysi+NHioMzSmA+CT
qoAcMfkLKwITeFk/G5f3EAibRRc6FbhwOe23HjzzRFq6NRjqisQhcDMFVDqY81kMxPgdxxgG/JvsrdolVPsGVZ1rz
I0W4+WSxl5SXcvZmyQ13qredxyTZPHphkF9Wlay8hHmtIRMzi0ENocgePkZ+as51kZ8sG5ouFQTSg/2xotmVJLIoH
4U1SDpkChCOyecHKNDyilcwqNS6Nxno4tIApVEhBZEnSZL8YdrSAsWW7pf6vFpWUYSM5jUWfZZOmoKouqTc/gQywm
rKzLjzR7X3EO3/Ko7UHFOH+NmXA5m2mKxhLROm5hxCo2ipXbvl7EeVmIWzsxiJVQkwYaYPMEVcoWZJXLr9PhrSwIc
isNTPUoxBYzdNmtDxo2naKRfLLJ2FLl3yVlbkLF/AQ6JfHsZYV5KvA5x2kw8hAoxCpflUlDtAKRY6fxYaIbyiGGQr
EfN/ApLRBbkPXViGarEkxabG+QD4BotD9ixAIhyCKhaY2VIHn+4FJ7tIyabJEoDdLkZRl/bJCkzQtd0wvlys6YpaF
BT1SuU5vgkvh1ykBheglZd1Cn5FpPtimFCQDIOwCAgyBSj5/JK1QIIiQS0RclWZ8jBWPkm7BSyz5WqXxA4tKEdCWi
K2f5qIRl8kSCaWb9OITYwzKOGAQTqDD4smX/HZt8uJDPZnhM5RsrGsDhM8XwsGieYwDIQblfaGmWLkaQicE3QPitM
sXyS6hE/ztk7hrUq7fEFaeRx5AhDU5dDBWmkH8hX2WV6lBcaxDkzG8/Mmgtk42IFRn/B62PEhE09Ph0te3Wg+B6gr
Ahli1xfHJawDsu8nGLla0wSCsWxdRNORu7NJEFIQCJPM4l5fyxwS2TtCvrpEUJZRya8guZdNVp00uVC4+Wy34DTS6
s4I41yF0Nu0DiaBTaFQmg9FI+PRbCF2rDg/nwGxSTZ2CiSEEtj/FnN542ExzWndN4a1Sb/0wZL5x+aVbm1eMmK3Zx
wXZjJS8qt8xKVxuDPMopK/I2EdxI5Cv4bN7VIWl/lyE6sIQroM0evjmJ8zMzPQMQ1eADNhj1JkOtPWPhycI5Y/O54
4mVUkSuiBeGRJmTzeQ/bHO1ZOQxlXpy2NDPgv5fbGBbwgL5YlDYyBTgqTZvVVx2yMNIjlLp+mEVeAR7otYyKQdeRJ
jAi1VJbjqL6TWCtaDfhf2xkXwAEMCWyqHBqZovT4Z/6ytfyWO3sr/iIRGNRkX8BoMgznJPKJ5jDdHKLJ/fWDV0CR4
TDe8iASy119yitFvsGiP5AljWJa5qsI8uxuFWUOn3ZFFzCrSycPkQWA06OTsoXFRAt91LlQZxlLm1I5+tqmTabIDm
8+Xn+Xiv8u5Qwj+VyOB7DL93qjHVMRgrjHbz4KVHKAAak1/TGiuNeaFZPVounqW6GWsaCanfzWva8gX6kBc9VgZKK
6lAegTeOzs3o9RLH28/sU2oU5w5TD8ecBiicA5ufCv5mbRDcVBVTMSvCDuaWhpw0f703gI0C8T8fAsuBCQtKMcUkF
0awwLlteXayKAeKxPIunw2rbwaq24ULwaNMkvUuaSswxeU+x2tiEPV/NSMqetoEgwqu2/ZTE3nTTSZfhoa/PPAwdn
1OTuePiqlTGi5bwVKKtAXZ6ONbC+yM6AQRjFtksvxpBrdJIxErTvKjFj+StXAVNZW2ZyJOGi0HlXBny6FXV1USuwg
qqzQy1CpNKlAnsZlcJAaw/CVqMcrfXCioF7IeMWiyea9kbrVF+NVBgvgNV+7Z3hutvhs8IEKlYrbluVktqG7zYwfG
+xSqDpBoWdAfQKMdjf5YF5WqgQ0PS70ZWhnm5yFNdBprYs1wMAQHBymoa2YDu0tQu0Eiu6M4kxCNm/NCQaeLrwPgn
NGuy+P9wql6NM9H/y945D2PZB9OTsX+X/eDNZ8buNZAe5A21krzeThEVPmOMmgRmgcGAgByjDPAVvLNa7gIy/5nFj
XxKXzK29UGNBPK4l6tL6S4tSAMFTB55bBxmMWQ1r8bzbw5bQZ1wJrQpy3VxrO5+Ysb6ZAxkYjJXd//YCI5fqj0Ip7
HY1yrDV+cMh/4EyUtOKgAmu40qmeaTG6CkxtiCsbyFnd1WWL8vQ8W4fGyiO6HM9kObJij7VdPUt0oEiU4bnvsQNHT
eFE2ka2UrRznf+6ACgI5Pzk9H0qseFBH8BAA+fOzu/VOZwhMB6hUeZEznBevM0EsHsQaiJQhiFH447gSGkw6a5jUm
KFRqsoa3lJk8vqUp6ArryOHpAW63BsER2q18rb+hSQhRuAci6e0dvGXz9iVnxgrRzr++2wOtHuIk6o8+XYzrOe+1Y
pcQALh9e6g8k43+XX4cL9e4Xgh5gtq4ptNEcLQ1+hqel/KISw0MQGBFTgMUIGmoS4c+6lS5YEJiI1mGH+rRtOoJ6r
Kh/wUw5XcEOtLupxdCx4HnkXrhD2/UN7++Q75Lq2Ztl9+baAB/bwYGFiEn8EI9X4UwOsoFVCACBzh0Mr/mV4ePaAw
TNERBQfnKS/AG0/NYc/NlREpKI2xc0SBW6l7Ku2Cw31jyoMzllUZZGooXgHZkg7ogLKBRj5ZTxtEZ8HLSUD7y+LLc
Evg8zNzpnx1/D57cVnVox9OnmUsduYzJqX6y/9QtB9oWWhCzNz7fU7pjOzkbuLU1nfC/XDPIKI9+fSq9pNuqvF9P8
//gyVRFwwmEGGgbRZnAYDWcsZYjzfI3u1Sv6loEtqPb5VG91o4K8EQHqwZ4NBtjX99Te00b3s5KNWHZhpmWdhFcsR
wDS++SuH6+5aufJSLJgXQoUHkjweWvfTwaOzB/khnt9YEOXuQhkU937kPJMZGjPH38CyBmslAUAMssBQSdAQ0OROW
qG+/ml0lbve7NJK/WIUBZG8Ea3FSh+16nXrQw5nH9R9uwXcVpyxCGpVJ2znReuAF/PkVXIihnnRjOSvj7C2OZ8t62
7+H2yeGxYBuPX3t08en4nflh+HF/IlSo6HYjyU4U5EcT629oqN5tCTe/B3DjBeBmtD2yjllQaiKgVG8pZnKVC1XPh
8ZK11jBYx7/A/0QEVCF7ssvTBaW15aJNyLNp42ZiYwZeeF1vGB5gYlw3cf2osyz4HEHxrGvXNN+61J/em8RgEQ3Xn

```
3x8dzv2kH6G1QrqyvCW9CUNVn+K8I6157kE7gTHkUOGuBkqBIAaDTkqoIBqLJ6DQdjXxWACEUSt5BWSVTTjgnx5JM
/lkvs16HeGMZEmMOpysHHnjcrN59taloa8baBfx4gbRbHwcwLjzw+F3lw+LN3/iOxqQ5LAnjudefOZ0otN4wPzmRD
M+iNYgGoZJpGvTWaYee8XXHnwGYwMA5BV4FhtLfhjtAqlp20i/gTAJbAuTiaYRQ0UlCNRZRVOgrE4CpCBVgu93Z9p
OZh7/6Hn8GLRF2mdcMq4+E7Y9rqy9LuMl4umjdDY/mMqVv79a2hkD0MQP1OWBJAlm/8/F88M5yJ3lbAhBJC13Xvig
u7Vm60YTQ5fe9+DTPojYK3PglfGFd10aBWMCDyAAuBRIoVvayaRdsMhvxzF62yLgpJy6YY9cfjqMZxzzGHZe/8XTp
O0H3xOQAPqwrwy6Qhnoc0HGcaf4pl1jT8ydmf/u6yfxZgWQAJYih29Y6BocIv8Aq7bOl8EGEIAcD4GMLaaP01W0wU
nenz9limhIiG8zwQiCKI1yGy0eClmhAViN6akWDJ/mN3ahW2uesj4g5FsATW+S1W11CbBaP+rgmJcrl8xrDzxlGnB
vLS843HJRnaIXXRDgII8ArouiPT0f8z2nnPXcRiuYB+d+Kw+97bV3VF9+3sXJla6yXwfAB3z/+TTMCe3Rb7Rw93te
58zU0TGz4arNpqa+Rv6gjn+KDR7WwpOcIPLTPkF+l/4BHv46I0gIvsJMYBgYyaV5xASX5QIey9Ft8dFQZnLGHHrsO
XekwXPI9DkT884aaI5xE8nDYDvIGx4t5Q66aPrb3yqyOsZrngtGQ5FmNeuvfGD3Wlxu5vWVHT6PE9GYAmz0gIIIHB
SAPvTca2YQC+1VHzzHtK3FZhwHjzKmsBYLoADCH9I0uGniwLzFQ1kW8gTGUgUkpitpAXjoKSGuErDLGDkwYI69dMB
znbtKzvNR4PBWTYUPCgA55YRm8bOV48lk2tu3rDp7a/ENS/TMI1fRkWn7z/f3/t2rbU5P9qaE0lPXyIJyDSOP2bCX
hPEwc+SYeGOqBX9ADEDG8UzB01McBbEaQFaxlCUKFMuDtAOWAEdvIwOCAIoMgUEIA7gc1nADT++FV+XMqo9uxIeGm
FPH+Ika4rQxLGeUwaYzgsyIS7Prvh2h0PiZI3+VnK7GVF9v/TDf+uIzn5t/XN8UQZLxXK3WWX1k9h6ZHwxCIW4Ud+
dM9o9guXOGadvQgxdJ0D7+LRcGAhnUXJVeQMrn5a8CpAnJaze15QoeysKpmAEc5tj+w2Z8X79pPqMH37ecJe8CyM0
7/LALOCh2fA41PF2cly2+fP/Mx/+78LlZ84FTTjxGwLpYfuu/F3W2ITP65vitd63M5x0iCI2p2JDLs0tkfTR0fN0a
wQPqgunACUfzqhUYNjFTyyNDIIp0KADzRKagof5/a4jNBDOt31VDuHklTCDH+wbxGW4/nnvX4Js+/NUOeh0nC45zC
B42THyRA3/Ph0aWoy1Pl76//Vbf+80No3T53I6mWl+x78ziebzNDfpRsj7WV8nIwHzP6EAhADIAkMt3qgTfYPYbW/
QwvjSt6TJNqztNshZfeXLpgL8NKN6gtYlF+KFHUQdjDUwyb8tYF+vmk53cTNZMALjMwDETwyTWfv56fFyN1x65juX
tihIn9MJvACXsmEcXAyPm2O5OJdn1/5ie8+plWdbCzmniyzy9f30C0bG73BnzbUmws8Hm8JiGiJeCLazpjeyJhjJo
hsfRpfrw1v9xw9dG6jpbsRhvMUkcXkb49wUZ2Dh7cTIQA0WV1QVQATuGiSK2kvhzyCPjZnYYb9QDiFroa97Qa1LYV
2lSFDAefN4vJGPA91gJ/LsPFs5DeF1Or/0LX1m/jG660Hse+ti/kS+x//SVtT/oU/b0zkPhdJRU0Z3TmYdauBpDcR
wlHEzOjIyZ6YERMzc2KSe9UYAcw18EidfWCLhh5EUXqpIH2fAibqvm4WnzWbx6DADxXS4+uW0y6Z4OU9vejB0GbgK
4AnNwE40VsD4KznhbGFm8uVzGwp9TeRlo9+vXnTdXjw/fbCOwJQqsSL1qOXbbu+NjR1a6ou3O7RQzAGSuMJGjzPnz
3lGCNFz0WF5YvBYwQ/IxIq95vBFQRF7ey+PCFkL0ZoIVxWencTwpSzTUyZF7NIl1KZUSMHiTHACIx6Ea63mypKHXQ
I3TYEL5yZ845kTcMfd1z5PZyu+M82aNnbCdKktyNYLTPw1I71Tbmjt8TDud+N1UTRyQCkrBdRBRspF6RYI0FRBaQT
6DwC9pLauOCYr1IhnDmK4ADALMs5OzTMsxUYSBUz5bKuW95D+UEt1/0vyRbYer1b+dfNCOtyO8WMYLTT1x8yfj5fF
qPFD5sEdikEEgD5Sx1IBEDaqiXCj5u1igmFBDcRpFESgKdpdlcw8AJ4IXgeqyvlSzh5iezMeXV/2vqxHY/4Sk/Nrz
1ChTLbs9L3bmk9/67YQ3/aVwuPCRCL5VBpKYCP3uGIBI1GiBWOGY4iQpVxHEw0AhXdJI8L8FjTS++0lPLs4Vy8Vy6
CqPbOBy773n3XhU7NX6107XFNdemnJM0/GbU5+e0r4uXZf4/vKq6JJE2rHMDCO6T9dA8EWQtKIvhhYnGw4AlYLGUe
k2qWjiUIDuWsqXh0vl8P3zJvmz+p3ff8J9CLRY6TujnFYAXdOyL97RE50bvyrs5T+Fvwx9CTYv3aEkPJNBgGD3Q9q
tNRRAsJkkAuVOuwKLBl5Tl5hnEYvrer5IXvK8STj9Rt+d6wz316f981AN1mZF69qyU1c+Q8rF+24JOpizxTOhNe1A
GjGPxGUno2OlCmI8466hXPbyKMLbTqGjAPw1eOOzWEQ+HU00vRzauB1fB7674T0BsLqJO3fujF7R8IsmbEfbwuH5V
A3QSPS4e9coIbiHAkPFcsedORcHkiGomPThsz9sDBD05cd911WL+8t+H/A+Q3uJ3ZZ3xhAAAAAElFTkSuQmCC"/>

                        <text class="single_line award_number"
:style="imageTipeAnimationStyle">{{animationText}}</text>
                        <text class="plus_number" :style="plusNumberAnimationStyle">{{pl
usAward}}</text>
                    </div>
                </div>
        </div>
    </div>

</template>

<script>
    // Import the animation module
    const animation = requireModule("animation");
    // Define the keyframe animation
    const keyframes = {
        'fluctuate': {
            "transform": [
                {
                    "p":0,
```

```
                    "v":"translateY(0rpx)"
                },
                {
                    "p":0.5,
                    "v":"translateY(-10rpx)"
                },
                {
                    "p":1.0,
                    "v":"translateY(0rpx)"
                }
            ]
        },
        'integralClicked': {
            "transform": [
                {
                    "p":0,
                    "v":"scale(1.0)"
                },
                {
                    "p":0.5,
                    "v":"scale(1.2)"
                },
                {
                    "p":1.0,
                    "v":"scale(1.0)"
                }
            ],
            "opacity": [
                {
                    "p":0,
                    "v":"1.0"
                },
                {
                    "p":0.3,
                    "v":"1.0"
                },
                {
                    "p":0.6,
                    "v":"0.0"
                },
                {
                    "p":0.9,
                    "v":"0.2"
                },
                {
                    "p":1.0,
                    "v":"1.0"
                }
            ]
        },
        'plusUp': {
            "transform": [
                {
                    "p":0,
```

```
                    "v":"translateY(0rpx)"
                },
                {
                    "p":1.0,
                    "v":"translateY(-48rpx)"
                }
            ],
            "opacity": [
                {
                    "p":0,
                    "v":"0"
                },
                {
                    "p":0.0873,
                    "v":"1.0"
                },
                {
                    "p":0.7205,
                    "v":"1.0"
                },
                {
                    "p":1.0,
                    "v":"0"
                }
            ]
        },
        'rightContentShow': {
            "transform": [
                {
                    "p":0,
                    "v":"scale(0)"
                },
                {
                    "p":0.5704,
                    "v":"scale(1.1)"
                },
                {
                    "p":1.0,
                    "v":"scale(1.0)"
                }
            ],
            "opacity": [
                {
                    "p":0,
                    "v":"1"
                },
                {
                    "p":1.0,
                    "v":"1"
                }
            ]
        }
    };
```

```
    // Load the keyframe animation
    animation.loadKeyframes(keyframes);
    export default {
        data: {
            animationText:"123",
            plusAward: "123",
            shouldAnim: true
        },

        onCreated: function() {
            this.imageTipeAnimationStyle = {
                animationName: "fluctuate",
                animationDuration: "2000ms",
                animationDelay: "000ms",
                animationTimingFunction: "linear",
                animationIterationCount: "infinite",
            };
        },

        didAppear() {

        },

        methods: {
            clickAnimation: function() {
                this.imageTipeAnimationStyle = {
                    animationName: "integralClicked",
                    animationDuration: "750ms",
                    animationDelay: "0ms",
                    animationTimingFunction: "ease-in-out",
                    animationIterationCount: "1",
                    animationFillMode: "backwards"
                };
                this.plusNumberAnimationStyle = {
                    opacity: 1.0,
                    animationName: "plusUp",
                    animationDuration: "350ms",
                    animationDelay: "0ms",
                    animationTimingFunction: "ease-in-out",
                    animationIterationCount: "1",
                    animationFillMode: "backwards"
                };
            }
        },
    }
</script>

<style>
    .card_root {
        display: flex;
        width: auto;
        flex-direction: row;
        padding-right: 24rpx;
        padding-left: 24rpx;
```

```
    }
    .white_bg {
        display: flex;
        flex-direction: row;
        justify-content: center;
        width: 100%;
        height: 144rpx;
        padding-right: 16rpx;
        padding-left: 16rpx;
        border-width: 0;
        border-radius: 16rpx;
        padding-top: 20px;
    }
    .left_div {
        display: flex;
        flex-direction: column;
        justify-content: center;
        flex:1;
    }
    .left_content {
        font-weight:600;
        font-size: 28rpx;
        color : #333333;
        width: auto;
    }
    .tips {
        margin-top:6rpx;
        display:flex;
        font-size: 24rpx;
        color : #99999999;
        width: auto;
    }
     .right_award {
        font-size: 40rpx;
        color : #999999;
        display: flex;
        width: auto;
        flex-shrink:0;
    }
    .right_content_div{
        display:flex;
        flex-direction: row;
        flex-shrink: 1;
        justify-content: center;
    }
    .right_content_first{
        display:flex;
        flex-shrink: 0;
        flex-direction: row-reverse;
        position: absolute;
        right: 0rpx;
    }
    .right_div {
        padding-left:1rpx;
```

```
        display:flex;
        flex-direction: row;
        flex-shrink: 0;
        width: 272rpx;
        overflow: visible;
    }
    .image_tip{
        position: relative;
        justify-content:center;
        width:80rpx;
        height: 85rpx;
        margin-left: 20px;
    }
    .image_tip_wrapper {
        width:80rpx;
        height: 85rpx;
        position: absolute;
        top: 29rpx;
        right: 17rpx;
    }
    .image_tip_icon {
        width:80rpx;
        height: 85rpx;
    }
    .arrow{
        display:flex;
        flex-shrink: 0;
        align-self:center;
        color:#CCCCCC;
        font-size: 14pit;
        font-family: IconFont;
        margin-left: 6rpx;
        margin-right: -7rpx;
    }
    .single_line {
        overflow: hidden;
        flex-shrink: 1;
        text-overflow:ellipsis;
        lines: 1;
    }
    .content_icon{
        width: 36rpx;
        height: 38rpx;
        align-self:center;
    }
    .award{
        margin-left: 10rpx;
        font-size:30rpx;
        color:#999999;
    }
    .award_number{
        position: absolute;
        top: 0;
        right: 0;
        left: 0;
```

```
        left: 0;
        bottom: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        font-size:36rpx;
        color:#806A00;
        text-align: center;
        line-height: 85rpx;
    }
    .plus_number{
        position: absolute;
        top: 0;
        right: 0;
        left: 0;
        bottom: 0;
        display: flex;
        justify-content: center;
        align-items: center;
        font-size:36rpx;
        color:#806A00;
        text-align: center;
        line-height: 85rpx;
        opacity: 0.0;
    }
</style>
```

Click detailKeyFrame.zip to download the complete sample code.

8.FAQ

8.1. Resolve the "Permission Denied" error during AntCubeTool installation

If you receive a "Permission Denied" error during a global installation on macOS, you can grant administrator permissions by running the following commands.

```
sudo mkdir -p /usr/local/{share/main,bin,lib/node_modules,include/node}
sudo chown -R $USER /usr/local/{share/main,bin,lib/node_modules,include/node}
```

8.2. Resolve the "Operation not permitted" error when using AntCubeTool

On macOS, if you encounter the "**shell-init: error retrieving current directory: getcwd: cannot access parent directories: Operation not permitted**" error, please grant file access permissions to the terminal. Go to **System Preferences** > **Security & Privacy** > **Privacy** > **Full Disk Access** and select **Terminal**.