Ant Technology

Ant Cube Card User Guide

Document Version: 20250303

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.Introduction to Ant Cube Card

This article introduces Ant Cube Card in detail from the definition and product advantages, and also provides a series of videos to learn more about Ant Cube Card.

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.

Benefits

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

Introduction Video

Overview

You can quickly locate the specific content by referring to the timestamp provided in the following table.

Content	The timestamp of the coordinates. Unit: seconds.
Definition and characteristics of Ant Cube Card	00:00:53~00:01:23
Application framework	00:01:24~00:03:01
System architecture	00:03:02~00:05:23
Core modules	00:05:24~00:07:03

Thread model	00:07:03~00:09:03
Data models	00:09:04~00:11:26
High-performance concurrent rendering	00:11:27~00:13:28
Card production /Workflow	00:13:29~00:14:15
Card development and debugging	00:14:16~00:15:09

2.Integrate into client

2.1. Introduction to the integration process

This topic describes the overall process of integrating Ant Cube Card into a client.

When you use Ant Cube Card, follow the following procedure:

1. Select Custom Baseline.
   i. Add a card baseline.
   ii. Add a card widget.
2. Initialize the card.
3. Build a card project.
   i. Initialize a project.
   ii. Construction Engineering.
4. Publish the card.
   i. Go to card background.
   ii. Create a card.
   iii. Add card resources.
   iv. Publish the card.
5. Render the card.

Other references

- Integrate into Android
- Intrgrate into iOS

2.2. Integrate into Android

2.2.1. Quick Start Guide

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

Ant Cube Card has been public preview by using the mPaaS 10.2.3 baseline version. However, only the mPaaS native AAR is supported. For more information, see Overview.

> **Note**
>
> If there are more integration-related questions, please search group number 32843812 to join DingTalk group for consultation and exchange. The mPaaS public cloud Q&A assistant has been added to the DingTalk group to quickly answer common integration questions.

Prerequisites

- You have activated and integrated to mPaaS.
- Ant Cube Card AntCubeTool tool has been installed. For more information, see About AntCubeTool.

Procedure

1. Select a baseline.
   i. Choose **mPaaS** > **Native AAR integration**. In the integration panel, click **Start Configuration** under Integrate /Upgrade Baseline to add the 10.2.3 baseline.
   ii. Add the **Rubik's Cube Card** and **Rubik's Cube Card -Network Image Loading Library** widgets.
2. **Initialize the card.**
   i. **The initialization process varies based on the baseline version. Perform the corresponding operations based on your baseline version.**
      a. **In the baseline version 10.2.3 and later, you only need to add `MP.init(this);` to the application to initialize mPaaS. For more information about how to configure initialization parameters, see Extend initialization parameters.**
      b. **In baseline versions earlier than 10.2.3, you must add the following code to Application to initialize mPaaS.**

```
public class MainApplication extends Application {

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

ii. **Add the following code to the `AndroidManifest.xml` under the App directory to ensure that the flash back of the C layer can be caught.**

```
!" -- Ant Cube Card Capture Level C Flash Back Required -->
<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 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.**
      - **Select "Need to create additional project source code folder". An additional folder will be created with the application name. If you select Not Required, the system is initialized 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 build. The built product will be in the `/dist/` folder of your project.**

4. **Publish the card.**

   i. **Go to card background.**

ii. **Click Create Card.**

We recommend that you use a combination of English, numbers, and underscores for the card ID. The card ID must be no less than 8 characters in length. The client relies on the card ID when rendering the card. The card name can be any value and can be up to 20 characters in length.

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. **Publish the card.**

a. **Click Create Release.**

b. **Select Official Release.**

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

5. **Render the card.**

```
// Create a card configuration.
CubeCardConfig cardConfig = new CubeCardConfig();
// The ID of the card created in the background.
cardConfig.setTemplateId("hello_cube");
// The card version.
cardConfig.setVersion("1.0.0.0");
// The width of the card. Select the width of the screen.
cardConfig.setWidth(MFSystemInfo.getPortraitScreenWidth());
// Card data (the data used to render the card, usually the content in 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) {
            // need to run in the main thread
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    // Create a card view.
                    CubeView view =
CubeService.instance().getEngine().createView(FastActivity.this);
                    // Add it 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 destroying the page, you need to manually recycle 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. Effect preview.

2.2.2. Advanced Guide

2.2.2.1. Extend Initialization Parameters

In the `cp_change_23596.28` and later versions of the baseline, you do not need to manually initialize the card before using it. You only need to install the card component. This article describes the implementation of setting initialization parameters in this case.

If you need to set initialization parameters, refer to the following code:

```
// Set the initialization parameters.
CubeInitParam cubeInitParam
    = CubeInitParam.getDefault()
    // Engine initialization configuration
    .setCubeEngineConfig(generateCubeEngineConfig())
    // Register the card-to-client channel.
    .setCubeModuleModels(generateModuleModel())
    // Register a custom label.
    .setCubeWidgetInfos(generateWidget());
// Initialize mPaaS
MP.init(this, MPInitParam.obtain().addComponentInitParam(cubeInitParam));
```

setAutoInitCube

```
/**
 * Set whether the framework automatically initializes the cube. By default, the cube i
s automatically initialized.
 * @param autoInitCube
 * @return
 */
public CubeInitParam setAutoInitCube(boolean autoInitCube)
```

setCubeEngineConfig

```
/**
 * Set CubeEngineConfig, please refer to the CubeEngineConfig introduction
 * @param cubeEngineConfig
 * @return
 */
public CubeInitParam setCubeEngineConfig(CubeEngineConfig cubeEngineConfig)
```

setCubeModuleModels

```
/**
 * Register cube jsapi
 * @param cubeModuleModels
 */
public CubeInitParam setCubeModuleModels(Collection<CubeModuleModel> cubeModuleModels)
```

setCubeWidgetInfos

```
/**
 * Register a custom view (custom label)
 * @param cubeWidgetInfos
 */
public CubeInitParam setCubeWidgetInfos(Collection<CubeWidgetInfo> cubeWidgetInfos)
```

2.2.2.2. Render Card

This article describes the overall implementation process of rendering cards on the Android client.

The process of rendering a card is divided into four parts: the first step is to assemble the card configuration information; the second step is to request the card according to the configuration information and obtain the card instance; the third step is to use the card instance to render using the card View; the fourth step is to release the card in the `destroy` declaration cycle after the entire business is completed. Perform the following steps to upload an SSL certificate:

1. **Assemble the card configuration information.**

   Create configuration information and set various parameters. For more information, see API description.

```
/**
* Assemble card configuration information
 * @return
 */
private CubeCardConfig assembleCubeCardConfig(){
// Create a card configuration.
    CubeCardConfig cardConfig = new CubeCardConfig();
// The ID of the card created in the background.
    cardConfig.setTemplateId("hello_cube");
// The card version.
    cardConfig.setVersion("1.0.0.0");
// The width of the card. Select the width of the screen.
    cardConfig.setWidth(MFSystemInfo.getPortraitScreenWidth());
    return cardConfig;
}
```

2. **Request a card.**

   The card is requested based on the assembled card configuration information. The card engine will go to the server to obtain the card template information. You can use the `createCard` method to request one card at a time, or use the `createCards` method to request multiple cards at a time.

```
/**
* Request card information
 * @param cardConfig
 */
private void requestCubeCard(final CubeCardConfig cardConfig){
// Create card information.
    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);
        }
    });
}
```

3. **Render the card.**

   After the card information is obtained, a card view is generated and rendered in the main thread. In this step, abnormal judgment should also be made to prevent the situation that the card information is not successfully obtained.

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

4. **Release the card.**

   After the card is used, the memory resources of the card are released, usually during the `onDestroy` life cycle of the page.

```
/**
* 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();
}
```

2.2.2.3. Real Machine Preview

This topic describes how to preview cards on the Android client.

Prerequisites

- You have activated and integrated to mPaaS.
- Ant Cube Card AntCubeTool tool has been installed. For more information, see About AntCubeTool.
- The connection process is completed as described in Quick Start.

Procedure

1. **Add dependencies for real-world preview.**
   i. **Add the Ant Cube Card-Development Tool component.**
   ii. Add a third-party dependency to the `build.gradle` of the main module of the project. If there is a conflict, the version of your dependency prevails.

```
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. Start the local debugging service by using the command line. In the path of the project, run the command to open the service. The instructions for enabling the service on macOS and Windows are as follows:
   - macOS: `act prepare && act server`
   - Windows: `act prepare | act server`

   After the instruction is executed, a QR code is generated at the terminal.
3. Start the client and scan the code to establish a connection. The terminal will prompt that the device is connected.

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

> **Note**
>
> Since it is an intranet connection, if the targetVersion is greater than 27, it needs to be downgraded, or configure `android:usesCleartextTraffic="true"` `android:networkSecurityConfig="@xml/network_security_config"` and the corresponding `network_security_config.xml` in the manifest.

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

After the code scanning is completed, a toast reminder will be displayed on the client.

The terminal will also indicate that it is connected.

4. **Preview.**

   Modify the card code, and then call the `act build` to complete the compilation. The `act preview` is then called to push the compiled content to the client.

2.2.2.4. Initialize engine extensions

This topic describes how to initialize engine extensions in the Android client.

Prerequisites

- You have activated and integrate to mPaaS.
- An Ant Cube Card is published in the console.

Procedure

1. Preset local path of card.
   - **Set the properties of the card engine class `CubeEngineConfig`.**

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

   - Local ant dynamic card assets path.

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

Example: If you set a `setResourcePath("cube")`, the preset card path is the `assets/cube/`.

2. **Set exception monitoring.**

   i. **Set the `CExceptionListener` of an exception monitoring interface.**

```
public interface CExceptionListener {
    void onException(CExceptionInfo var1);
}
```

   ii. The description of the abnormal interface method.

Qualifiers and Types	Method and description
void	onException(CExceptionInfo info)

   iii. Detailed description of the exception method.

```
/**
* Exception monitoring notifications
*
* @param cExceptionInfo -The information about the exception.
*/
void onException(CExceptionInfo cExceptionInfo)
```

   iv. Integrate the `CExceptionListener` class into the `Activity` and execute the inherited method.

```
public class MainActivity extends AppCompatActivity implements CExceptionListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public void onException(CExceptionInfo cExceptionInfo) {
                                // Determine the error message of the abnormal
interface based on the data returned by CExceptionInfo.
    }
}
```

3. **Set the network image download handler.**

   i. **Network image download interface `ICKImageHandler`.**

ii. **Define `onBitmapLoaded` and `onBitmapFailed` within the `ICKImageHandler`. The following table describes how fully managed Flink processes data before fully managed Flink writes data to Lindorm.**

```
public interface ICKImageHandler {

    String loadImage(String var1, int var2, int var3, Map<String, Object> var4, ICK
ImageHandler.LoadImageListener var5);

    void cancel(String var1);

    public interface LoadImageListener {
        void onBitmapLoaded(Bitmap var1);
        void onBitmapFailed(Exception var1);
    }
}
```

iii. The description of the image download method.

Qualifiers and Types	Method and description
String	loadImage(String var1, **int** var2, **int** var3, Map<String, Object> var4, ICKImageHandler.LoadImageListener var5)
void	cancel(String var1)
void	onBitmapLoaded(Bitmap var1)
void	onBitmapFailed(Exception var1)

iv. Detailed description of the image download method.

```
/**
* Network image download monitoring
*
* @param url -The URL of the image download network.
* @param width -The width of the image.
* @param height -The height of the image.
* @param params -Parameters
* @param loadImageListener -Load the listener.
*/
String loadImage(String url, int width, int height, Map<String, Object> params, ICK
ImageHandler.LoadImageListener loadImageListener)


/**
* Network picture download cancel monitoring
*
* @param var1 - URL
*/
void cancel(String var1)


/**
* Image download success monitoring
*
* @param var1
*/
void onBitmapLoaded(Bitmap var1)


/**
* Image download failure monitoring
*
* @param var1
*/
void onBitmapFailed(Exception var1)
```

v. Integrate the `ICKImageHandler` class into the `Activity`, execute the inherited method, and rewrite the network image download function.

```
public class LocalCardsActivity extends AppCompatActivity implements
ICKImageHandler {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_local_cards);

    }

    @Override
    public String loadImage(String s, int i, int i1, Map<String, Object> map, LoadI
mageListener loadImageListener) {
        new Thread(){
            @Override
            public void run() {
                try {
                    final Bitmap bitmap = HttpUtil.get(url, null, null);
                    if (bitmap != null){
                        if (loadImageListener != null){
                            loadImageListener.onBitmapLoaded(bitmap);
                        }
                    }else {
                        if (loadImageListener != null){
                            loadImageListener.onBitmapFailed(new Exception("bitmap i
s null"));
                        }
                    }
                }catch (Exception e){
                    e.printStackTrace();
                    if (loadImageListener != null){
                        loadImageListener.onBitmapFailed(e);
                    }
                }
            }
        }.start();
        return url;
    }

    @Override
    public void cancel(String s) {

    }
}
```

2.2.2.5. Preset Card

This topic describes how to preset Ant Cube Card in the Android client.

Prerequisites

- You have activated and integrated to mPaaS.
- An Ant Cube Card is published in the console.

- The card local path is already preset in the initialization engine extension. For more information, see Initialize engine extensions.

Procedure

1. **Download the encrypted BIN file from the console.**
2. **Add and copy the downloaded file to the resource file.**

   Rename the downloaded Bin file to Card ID @ Card Version, for example, main@1_0_0_0.zip.

   > **Note**
   >
   > **Later, the console will directly provide the complete named zip package, without the need to manually modify the name, you only need to add it to the assets.**

3. Bring Assets into the project. For more information, see Initialize engine extensions.
4. Load preset cards.

```
// Create a card configuration.
CubeCardConfig cardConfig = new CubeCardConfig();
// The ID of the created card.
cardConfig.setTemplateId("main");
// The card version.
cardConfig.setVersion("1.0.0.0");
// The width of the card. Select the width of the screen.
cardConfig.setWidth(MFSystemInfo.getPortraitScreenWidth());
// The card data. This parameter is required.
JSONObject obj = new JSONObject("xxxxx");
cardConfig.setData(obj);
// Create card information.
CubeService.instance().getEngine().createCard(cardConfig, new CCardCallback() {
    @Override
    public void onLoaded(CubeCard cubeCard, CCardType cCardType, CubeCardConfig cubeC
ardConfig, CubeCardResultCode cubeCardResultCode) {
        if (cubeCardResultCode == CubeCardResultCode.CubeCardResultSucc) {
            // need to run in the main thread
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    // Create a card view.
                    CubeView view=CubeService.instance().getEngine().createView (where
your current activity is located);
                    // Add it to the outer ViewGroup.
                    local_cards.addView(view);
                    // Render the card.
                    cubeCard.renderView(view);
                }
            });
        } else {
                        MPLogger.info("cube", "fail " + cubeCardConfig.getTemplateId()
+ " style " + cCardType + " error " + cubeCardResultCode);
        }
    }
});
```

2.2.2.6. Cards call client methods

This article describes the implementation path of calling Android client methods in Ant Cube Card.

Procedure

1. Register the module on the client side.

i. **The custom module.**

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

    // The annotation. uiThread indicates whether the callback is sent to 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());
        }
    }
}
```

ii. **Register a custom module.**

The first parameter is the type, the second parameter is the full path of the custom module, and the third parameter is the name of the called method. The full path and method name do not need to be confused.

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

2. **Call on the card side.**

i. Register the module. The type must be consistent with that of the client. In this example, the `custom` is used.

ii. Call the module. The method name must be the same as the method name registered by the client.

```
<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>
```

iii. The client method can be called after the card is packaged and published to the background.

2.2.2.7. The client calls the card method

This article describes the implementation path of calling the card method on the Android client.

Procedure

1. **The JS method is implemented on the card side.**

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

        },
        methods: {
            onClick() {
                console.info('invoke on-click event');
            },
            // js method
            clientToCube(data) {
                this.message = data;
            }
        }
    }
</script>
```

2. Package the card and publish it to the card background.
3. The client calls.

   The client obtains the card instance, calls the corresponding JS method, and sends the data.

```
if (mCubeCard != null) {
    mCubeCard.callJsFunction("clientToCube", "client to cube: " +
System.currentTimeMillis());
}
```

2.2.2.8. Card Custom Label

This topic describes how to customize labels in cards.

Procedure

1. Register custom labels on the client side (custom view).
   i. Implement a custom label (custom view).

   Inherit CCardWidget and implement its abstract method, the main method is `onCreateView`, where the corresponding custom View is returned.

```
public class CustomCubeWidget extends CCardWidget {
    private static final String TAG = "CustomCubeWidget";

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

    /**
     * UI creation interface, which indicates the view UI of the current component
to be displayed on the screen.
```

```
     *
     * @param params The data declared on the card, including the style, event, and
property. The corresponding keys are DATA_KEY_STYLES, DATA_KEY_EVENTS, and DATA_KEY
_ATTRS;
     * @param width The width of the component.
     * @param height The height of the component.
     * @return The embedded View is returned.
     */
    @Override
    public View onCreateView(Map<String, Object> params, int width, int height) {
        MPLogger.debug(TAG, "onCreateView:" + width + "," + height);
        for (String key : params.keySet()) {
            MPLogger.debug(TAG, key + ":" + params.get(key));
        }
        TextView textView = new TextView(getContext());
        textView.setText("test");
        return textView;
    }

    /**
     * The UI reuse interface. If the component supports reuse (see the canReuse me
thod), it is called when the component is reused and re-launched.
     *
     * @param params The data declared on the card by the component.
     * @param width The width of the widget.
     * @param height The height of the component.
     */
    @Override
    public void onReuse(Map<String, Object> params, int width, int height) {
        MPLogger.debug(TAG, "onReuse");
    }

    /**
     * Component data update interface
     *
     * @param params The data declared on the card by the component.
     */
    @Override
    public void onUpdateData(Map<String, Object> params) {
        MPLogger.debug(TAG, "onUpdateData");
    }

    /**
     * Component reuse cleanup interfaces. If the component supports reuse (canReus
e returns true), the onRecycleAndCached method is called after the component enters
the reuse pool. This indicates that the current component has been put into the cac
he off-screen and needs to be cleared.
     */
    @Override
    public void onRecycleAndCached() {
        MPLogger.debug(TAG, "onRecycleAndCached");
    }

    /**
```

```
     * Whether the component supports reuse; To improve efficiency, extend the component to support reuse. For example, when a custom tag component is removed from the current view due to data update, if the component supports reuse, it will be put into the reuse pool. The next time the component is displayed, it will be directly obtained from the reuse pool. If the component does not support reuse, the onDestroy operation will be called directly.
     *
     * @return true: reuse. false: not reuse.
     */
    @Override
    public boolean canReuse() {
        MPLogger.debug(TAG, "canReuse");
        return false;
    }

    /**
     * The callback interface for component destruction to release resources.
     */
    @Override
    public void onDestroy() {
        MPLogger.debug(TAG, "onDestroy");
    }
}
```

ii. Register a custom label.

The first parameter is a custom label, which can be customized at will and will be written in <> on the card side. We recommend that you add a prefix to prevent label conflicts with dynamic cards. The second is the full path of the custom label implementation class, be careful not to confuse.

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

2. Call on the card side.

In Labels, write a custom label, which is `custom-widget` in this example.

> **Note**
>
> The custom label to be written must be the same as the first parameter in the Register Custom label step.

```
<template>
    <div class="root">
        ...
        <custom-widget></custom-widget>
        ...
    </div>
</template>
```

3. Publish the card package to the background or preview to render the client-defined view.

2.2.3. Interface Description

2.2.3.1. CubeService

This topic describes the methods of the Card Service class.

Public functions

Functions	Return value type	Description
`public static CubeCrystalService instance()`	static CubeCrystalService	Gets the CubeCrystalService instance.
`public void initEngine(CubeEngineConfig config, Application application)`	void	Global initialization.
`public CubeEngine getEngine()`	CubeEngine	Obtains an engine instance.
`public void destroyEngine()`	void	Global destruction.

instance

- **Statement**: `public static CubeCrystalService instance()`
- **Description**: Obtains a CubeCrystalService instance.
- **Parameter**: None.
- **Returned value**:

Parameter	Type
CubeCrystalService instance	CubeCrystalService

initEngine

- **Statement**: `public void initEngine(CubeEngineConfig config, Application application)`
- **Description**: Global initialization.
- **parameter**:

Parameter	Type	Description

config	CubeEngineConfig	Initialize configurations
application	Application	-

- **Return value**: None.

getEngine

- **Statement**: `public CubeEngine getEngine()`
- **Description**: You can obtain an engine instance.
- **Parameter**: None.
- **Returned value**:

Parameter	Type
Engine Instance	CubeEngine

destroyEngine

- **Statement**: `public void destroyEngine()`
- **Description**: Global destruction.
- **Parameter**: None.
- **Return value**: None.

2.2.3.2. CubeEngine

This topic describes the methods of the core classes of the card engine.

Public functions

Functions	Return value type	Description
createCard(final CubeCardConfig config, final CCardCallback callback)	void	Create a single card.
createCards(List<CubeCardConfig> configs, final CCardCallback callback)	void	Create cards in batches.
createView(Context context)	CubeView	Creates a rendered view.
setCustomUnit(String unitName, float unitRadio)	void	Set the custom unit.

registerModule(Collection<CubeModuleModel> models, Bundle options)	void	Register a custom module.
registerWidgets(Collection<CubeWidgetInfo> widgets)	void	Registers a set of custom extension components.
sendEvent(Map<String, Object> componentData, String eventName, @Nullable Map<String, Object> eventParams)	void	Native sends custom event channels to the JS side.
destroy()	void	Destroys a card instance.

createCard

- **Statement**: `public void createCard(final CubeCardConfig config, final CCardCallback callback)`
- **Description**: Create a card.
- **Parameter**:

Parameter	Type	Description
config	CubeCardConfig	Card configuration parameters.
callback	CCardCallback	The callback.

- **Return value**: None.

createCards

- **Statement**: `public void createCards(List<CubeCardConfig> configs, final CCardCallback callback)`
- **Description**: Create multiple cards at a time.
- **Parameter**:

Parameter	Type	Description
configs	List<CubeCardConfig>	Batch configuration.
callback	CCardCallback	The callback. The callback is returned once for each card result.

- **Return value**: None.

createView

- **Statement**: `public CubeView createView(Context context)`
- **Description**: Create a rendering view.
- **Parameter**:

Parameter	Type	Description
context	Context	The context in which the view is created.

- **Returned value**:

Parameter	Item	Description
createView	CubeView	Cube rendering container view.

setCustomUnit

- **Statement**: `public void setCustomUnit(String unitName, float unitRadio)`
- **Description**: The custom unit.
- **Parameter**:

Parameter	Type	Description
unitName	String	The name of the company, for example, sip.
unitRadio	float	Unit scale, such as 1.5.

- **Return value**: None.

registerModule

- **Statement**: `public void registerModule(Collection<CubeModuleModel> models, Bundle options)`
- **Description**: Registers a custom module.
- **Parameter**:

Parameter	Type	Description
models	Collection<CubeModuleModel>	key is the module name, such as animation;value is the class name, such as CKAnitmationModule.

options	Bundle	-

- **Return value**: None.

registerWidgets

- **Statement**: `public void registerWidgets(Collection<CubeWidgetInfo> widgets)`
- **Description**: Registers a group of custom extension components.
- **Parameter**:

Parameter	Type	Description
widgets	Collection<CubeWidgetInfo>	The information about the extension component.

- **Return value**: None.

sendEvent

- **Statement**: `public void sendEvent(Map<String, Object> componentData, String eventName, @Nullable Map<String, Object> eventParams)`
- **Note** :Native sends custom event channels to JS.
- **Parameter**:

Parameter	Type	Description
componentData	Map<String, Object>	Component data, that is, the input parameter data when the onCreateView component is created in the CCardWidget.
eventName	String	The name of the custom event.
eventParams	Map<String, Object>	The custom event parameters.

- **Return value**: None.

destroy

- **Statement**: `private void destroy()`
- **Description**: The card instance is destroyed.
- **Parameter**: None.
- **Return value**: None.

2.2.3.3. CubeEngineConfig

This topic describes how to initialize the configuration class for the card engine.

Public functions

Functions	Return value type	Description
`public String getResourcePath()`	String	Gets the local resource package path.
`public void setResourcePath(String resourcePath)`	void	Sets the local resource package path.
`public CExceptionListener getExceptionListener()`	CExceptionListener	Gets the exception listener.
`public void setExceptionListener(CExceptionListener exceptionListener)`	void	Set exception listeners.
`public ICKImageHandler getImageHandler()`	ICKImageHandler	Gets the imagehandler.
`public void setImageHandler(ICKImageHandler imageHandler)`	void	Set imageHandler.

getResourcePath

- **Statement**: `public String getResourcePath()`
- **Description**: Obtains the local resource package path.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
getResourcePath Object	String	The local resource package path.

setResourcePath

```
/**
 * Set the local resource package path.
 * @param resourcePath
 */
public void setResourcePath(String resourcePath)
```

- **Statement**: `public void setResourcePath(String resourcePath)`
- **Description**: Sets the local resource package path.
- **Parameter**:

Parameter	Type	Description
resourcePath	String	The local resource package path.

- **Return value**: None.

getExceptionListener

- **Statement**: `public CExceptionListener getExceptionListener()`
- **Description**: Obtains an exception listener.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
getExceptionListener Object	CExceptionListener	The exceptions that are monitored.

setExceptionListener

- **Statement**: `public void setExceptionListener(CExceptionListener exceptionListener)`
- **Description**: Configure an exception listener.
- **Parameter**:

Parameter	Type	Description
exceptionListener	CExceptionListener	The card exception listener.

- **Return value**: None.

getImageHandler

- **Statement**: `public ICKImageHandler getImageHandler()`
- **Description**: Obtain the imagehandler.
- **Parameter**: None.

- **Returned value**:

Parameter	Item	Description
imagehandler object	ICKImageHandler	The obtained imagehandler.

setImageHandler

- **Statement**: `public void setImageHandler(ICKImageHandler imageHandler)`
- **Description**: Set the imageHandler parameter. If this parameter is not set, the default internal implementation is used. We recommend that you use custom implementations.
- **Parameter**:

Parameter	Type	Description
imageHandler	ICKImageHandler	The imageHandler to set.

- **Return value**: None.

2.2.3.4. CExceptionListener

This topic describes how to use the card exception listener.

Public Methods

Functions	Return value type	Description
`public void onException(CExceptionInfo info);`	void	Exception listener notifications.

onException

- **Statement**: `public void onException(CExceptionInfo info);`
- **Description**: The system listens for exceptions.
- **Parameter**:

Parameter	Type	Description
info	CExceptionInfo	Exception Information

- **Return value**: None.

2.2.3.5. CExceptionType

This topic describes the exception types of cards.

- **Statement**: `public enum CExceptionType {JS_EXCEPTION, STYLE_EXCEPTION}`
- **Description**: The card exception type.
- **Enumeration Value**

Enumeration member	Description
JS_EXCEPTION	JS exception
STYLE_EXCEPTION	Style Exceptions

2.2.3.6. CExceptionInfo

This topic describes the methods of the card exception information class.

Public functions

Functions	Return value type	Description
`public String getCardUid()`	String	Obtain the card instance ID.
`public CExceptionType getErrCode()`	CExceptionType	Gets the card exception type.
`public String getTitle()`	String	Gets the title of the exception.
`public String getException()`	String	Queries the information about exceptions.
`public Map<String, Object> getExtParams()`	Map<String, Object>	Obtains the extended exception information.

getCardUid

- **Statement**: `public String getCardUid()`
- **Description**: You can call this operation to obtain the ID of a card instance.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card Instance ID Object	String	Card Instance ID

getErrCode

- **Statement**: `public CExceptionType getErrCode()`
- **Description**: You can call this operation to obtain the card exception type.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card Exception Type Object	CExceptionType	Card exception type

getTitle

- **Statement**: `public String getTitle()`
- **Description**: You can call this operation to obtain the title of the exception.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Exception Title Object	String	Exception Title

getException

- **Statement**: `public String getException()`
- **Description**: Obtains the exception information.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Exception information object	String	The obtained exception information.

getExtParams

- **Statement**: `public Map<String, Object> getExtParams()`
- **Description**: You can call this operation to obtain extended information about exceptions.

- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Exception extension information object	Map<String, Object>	The obtained exception extension information.

2.2.3.7. CubeCard

This topic describes the methods of the card core class.

Public functions

Functions	Return value type	Description
renderView(CubeView view)	void	To render a view, you need to provide a CubeView.
getSize()	Rect	Gets the width and height dimensions of the card.
updateData(JSONObject jsonData)	void	Updates the rendering data.
callJsFunction(final String methodName, final Object... params)	void	Call the JS method.
recycle()	void	Destruction, recovery of resources.
getCardUid()	String	Obtain the card instance ID.
notifyState(CCardState state)	void	The status of the notification card.
getCubeCardConfig()	CubeCardConfig	Gets the config parameters for creating a card.
getBindView()	CubeView	Gets the view that the card is temporarily bound to.

renderView

- **Statement**: `public void renderView(CubeView view)`
- **Note**: To render a view, you must provide a CubeView.
- **Parameter**:

Parameter	Type	Description
view	CubeView	Generated by CubeEngine

- **Return value**: None.

getSize

- **Statement**: `public Rect getSize()`
- **Description**: Obtain the width and height of the card.
- **Parameter**: None.
- **Returned value**:

Item	Description
Rect	Card width, height size objects

updateData

- **Statement**: `public void updateData(JSONObject jsonData)`
- **Description**: Updates the rendering data.
- **Parameter**:

Parameter	Type	Description
jsonData	JSONObject	External data model required to render the card

- **Return value**: None.

callJsFunction

- **Statement**: `public void callJsFunction(final String methodName, final Object... params)`
- **Description**: Call the JS method.
- **Parameter**:

Parameter	Type	Description
methodName	String	Method Name
params	Object	Invocation Parameters

- **Return value**: None.

recycle

- **Statement**: `public void recycle()`
- **Description**: Destroy and recycle resources.
- **Parameter**: None.
- **Return value**: None.

getCardUid

- **Statement**: `public String getCardUid()`
- **Description**: You can call this operation to obtain the ID of a card instance.
- **Parameter**: None.
- **Returned value**:

Item	Description
String	Card Instance ID Object

notifyState

- **Statement**: `public void notifyState(CCardState state)`
- **Note**: You are notified to change the card status when the card is displayed on the screen, displayed on the screen, and in the front and back offices.
- **Parameter**:

Parameter	Type	Description
state	CCardState	Card Status

- **Return value**: None.

getCubeCardConfig

- **Statement**: `public CubeCardConfig getCubeCardConfig()`
- **Description**: Obtain the config parameter used to create a card.
- **Parameter**: None.
- **Returned value**:

Item	Description
CubeCardConfig	Create a config object for the card.

getBindView

- **Statement**: `public CubeView getBindView()`

- **Description**: This function obtains the view that is temporarily bound to a card.

  > **Important**
  > The view may be reused by other cards.

- **Parameter**: None.
- **Returned value**:

Item	Description
CubeView	The view object bound to.

2.2.3.8. CubeCardConfig

This topic describes how to use the card configuration class.

Public functions

Functions	Return value type	Description
`public String getTemplateId()`	String	Gets the unique ID of the template.
`public CubeCardConfig setTemplateId(String templateId)`	CubeCardConfig	Sets the unique ID of the template.
`public String getVersion()`	String	Gets the template version number.
`public CubeCardConfig setVersion(String version)`	CubeCardConfig	Set the template version number.
`public String getCardUid()`	String	Obtain the unique ID of the card.
`public JSONObject getData()`	JSONObject	Gets the card data.
`public CubeCardConfig setData(JSONObject data)`	CubeCardConfig	Set card data.

`public int getWidth()`	int	Gets the width of the card preset.
`public CubeCardConfig setWidth(int width)`	CubeCardConfig	Sets the preset width of the card.
`public int getHeight()`	int	Gets the height of the card preset.
`public CubeCardConfig setHeight(int height)`	CubeCardConfig	Sets the height of the card preset.
`public JSONObject getExtOption()`	JSONObject	Gets the card extension parameters.
`public CubeCardConfig setExtOption(JSONObject extOption)`	CubeCardConfig	Set the card extension parameters.
`public JSONObject getEnvData()`	JSONObject	Obtains card environment variable data.
`public CubeCardConfig setEnvData(JSONObject envData)`	CubeCardConfig	Set card environment variable data.
`public CCardLayoutChangeListener getLayoutChangeListener()`	CCardLayoutChangeListener	Gets the layout change listener.
`public void setLayoutChangeListener(CCardLayoutChangeListener layoutChangeListener)`	void	Set layout change listeners.

getTemplateId

- **Statement**: `public String getTemplateId()`
- **Description**: Obtain the unique ID of a template.

- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Template ID object	String	The unique ID of the template.

setTemplateId

- **Statement**: `public CubeCardConfig setTemplateId(String templateId)`
- **Description**: The unique ID of the template.
- **Parameter**:

Parameter	Type	Description
templateId	String	The ID of the template that you want to set.

- **Returned value**:

Parameter	Item	Description
Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getVersion

- **Statement**: `public String getVersion()`
- **Description**: You can obtain the version number of the template.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Template version number object	String	Template version number

setVersion

- **Statement**: `public CubeCardConfig setVersion(String version)`
- **Description**: This function sets the template version number.
- **Parameter**:

Parameter	Type	Description

version	String	Template version number to set

- **Returned value**:

Parameter	Item	Description
Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getCardUid

- **Statement**: `public String getCardUid()`
- **Description**: Obtain the unique ID of a card.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card ID object	String	Card ID

getData

- **Statement**: `public JSONObject getData()`
- **Description**: Obtain card data.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card Data Object	JSONObject	Card Data

setData

- **Statement**: `public CubeCardConfig setData(JSONObject data)`
- **Description**: Set card data.
- **Parameter**:

Parameter	Type	Description
data	JSONObject	Card Data

- **Returned value**:

Parameter	Item	Description
Card configuration object	CubeCardConfig	Card Configuration

getWidth

- **Statement**: `public int getWidth()`
- **Description**: This function obtains the preset width of the card.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
The width object of the card	int	Card Preset Width

setWidth

- **Statement**: `public CubeCardConfig setWidth(int width)`
- **Description**: Set the default width of the card.
- **Parameter**:

Parameter	Type	Description
width	int	Card width

- **Returned value**:

Parameter	Item	Description
Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getHeight

- **Statement**: `public int getHeight()`
- **Description**: You can call this operation to obtain the preset height of a card.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
The height object of the card	int	Card preset height

setHeight

- **Statement**: `public CubeCardConfig setHeight(int height)`
- **Description**: Sets the preset height of the card.
- **Parameter**:

Parameter	Type	Description
height	int	Card height

- **Returned value**:

Parameter	Item	Description
Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getExtOption

- **Statement**: `public JSONObject getExtOption()`
- **Description**: You can call this operation to obtain extended card parameters.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card extension parameter object	JSONObject	Card extension parameters

setExtOption

- **Statement**: `public CubeCardConfig setExtOption(JSONObject extOption)`
- **Description**: Set the card extension parameter.
- **Parameter**:

Parameter	Type	Description
extOption	JSONObject	Card extension parameters

- **Returned value**:

Parameter	Item	Description

Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getEnvData

- **Statement**: `public JSONObject getEnvData()`
- **Description**: You can call this operation to obtain the card environment variable data.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Card Environment Variable Data	JSONObject	Card Environment Variable Data

setEnvData

- **Statement**: `public CubeCardConfig setEnvData(JSONObject envData)`
- **Description**: Set the card environment variable data.
- **Parameter**:

Parameter	Type	Description
envData	JSONObject	Card Environment Variable Data

- **Returned value**:

Parameter	Item	Description
Card Configuration Parameter Object	CubeCardConfig	Card configuration parameters

getLayoutChangeListener

- **Statement**: `public CCardLayoutChangeListener getLayoutChangeListener()`
- **Description**: Obtains the layout change listener.
- **Parameter**: None.
- **Returned value**:

Parameter	Item	Description
Layout change listener object	CCardLayoutChangeListener	Layout Change Listener

setLayoutChangeListener

- **Statement**: `public void setLayoutChangeListener(CCardLayoutChangeListener layoutChangeListener)`
- **Description**: You can configure a listener for layout changes.
- **Parameter**:

Parameter	Type	Description
layoutChangeListener	CCardLayoutChangeListener	Layout Change Listener

- **Return value**: None.

2.2.3.9. CCardType

This topic describes the types of card template sources.

- **Statement**: `public enum CCardType {C_CARD_TYPE_NONE, C_CARD_TYPE_CACHE, C_CARD_TYPE_BUNDLE, C_CARD_TYPE_FILE, C_CARD_TYPE_CLOUD}`
- **Description**: the source type of the card template.
- **Enumeration Value**:

Enumeration member	Description
C_CARD_TYPE_CACHE	Style Exceptions
C_CARD_TYPE_BUNDLE	Templates in Local resource package
C_CARD_TYPE_FILE	Templates in the local SD card
C_CARD_TYPE_CLOUD	Cloud Templates

2.2.3.10. CCardLayoutChangeListener

This topic describes how to notify a listener of a layout change.

Public functions

Functions	Return value type	Description
`void onLayout(Rect size)`	void	Card creation callback class.

onLayout

- **Statement**: `void onLayout(Rect size)`
- **Description**: The listener for layout change notifications.
- **Parameter**:

Parameter	Type	Description
size	Rect	The size of the card after the change.

- **Return value**: None.

2.2.3.11. CCardCallback

This topic describes how to create a callback class for cards.

Public functions

Functions	Return value type	Description
`public void onLoaded(final CubeCard card, CCardType cardType, CubeCardConfig config, CubeCardResultCode errorCode)`	void	Card creation callback class.

onLoaded

- **Statement**: `public void onLoaded(final CubeCard card, CCardType cardType, CubeCardConfig config, CubeCardResultCode errorCode)`
- **Description**: The callback class for card creation.
- **Parameter**:

Parameter	Type	Description
card	CubeCard	Card instance
cardType	CCardType	Card Template Source Type
config	CubeCardConfig	Card configuration parameters
errorCode	CubeCardResultCode	Error code

- **Return value**: None.

2.2.3.12. CubeCardResultCode

This topic describes card result codes.

- **Statement**: `public enum CubeCardResultCode {CubeCardResultSucc, CubeCardResultNetworkError, CubeCardResultParamError, CubeCardResultNotExist, CubeCardResultContentInvalid}`
- **Description**: the source type of the card template.
- **Enumeration Value**

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

2.2.3.13. CubeView

This topic describes the methods of the card rendering host class.

Public functions

Functions	Return value type	Description
`public void destroy()`	void	To release resources, you need to explicitly call.

destroy

- **Statement**: `public void destroy()`
- **Note**: You must explicitly call this operation to release resources.
- **Parameter**: None.
- **Return value**: None.

2.2.3.14. CubeModuleModel

This topic describes the description of the module.

Public Methods

Functions	Return value type	Description
`public CubeModuleModel(String type, String fullClsName, String[] methods)`	None	The constructor function.

CubeModuleModel

- **Statement**: `public CubeModuleModel(String type, String fullClsName, String[] methods)`
- **Description**: The constructor function.
- **Arguments**:

Parameter	Type	Limit
type	String	The name of the module, such as animation.
fullClsName	String	The full package path of the module class.
methods	String[]	JS methods declared in the module.

- **Return value**: None.

2.2.3.15. CubeModule

Custom modules must inherit from this class.

2.2.3.16. CubeJSCallback

This topic describes the methods of the CubeJSCallback class.

Public Methods

Functions	Return value type	Description
`public void invoke(Object data)`	void	Callback data sent to the card

invoke

When the card calls native, this class is used to send a callback back to the card.

- **Statement**: `public void invoke(Object data)`

- **Description**: Send callback data to the card.
- **Parameter**:

Parameter	Type	Description
data	Object	Callback data sent to the card

- **Return value**: None.

2.3. Integrate into iOS

2.3.1. Quick Start Guide

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

Prerequisites

- You have integrated the project into mPaaS. For more information, see the following:
  - Integrate by using CocoaPods based on the existing project
- Ant Cube Card AntCubeTool tool has been installed. For more information, see About AntCubeTool.

Procedure

1. Select Custom Baseline.

   Ant Cube Card is currently available in the Custom Baseline `cp_change_15200851` version. Select an add method based on your integration method.

   - Use the mPaaSPlugin plugin.

     This method is applicable to **mPaaS-based integration** or **existing projects that use mPaaS plug-ins**.

     a. Choose Xcode menu item **Editor** > **mPaaS** > **Edit Project** > **Upgrade Baseline** to switch the project to `cp_change_15200851` custom baseline.

     > **Note**
     >
     > If the **upgrade baseline** is not available, make sure that the project configuration has been imported and open the Edit Project page by referring to precondition.

     b. After you upgrade the baseline, choose **Edit Module** > **Modify Module**, select **Cube Card** (the current version strongly depends on the mini program, and you need to select the **Ariver mini program** component at the same time), and then click **Start Editing**.

   - Use cocoapods-mPaaS plugin.

     This method is applicable to the integration mode that uses **CocoaPods based on existing projects**.

a. In the Podfile file, perform steps:

a. Modify mPaaS_baseline to `cp_change_15200851`.

b. Add the Cube Card component dependency by using the **mPaaS_pod "mPaaS_Cube"**.

```
Podfile
# mPaaS Pods Begin
plugin "cocoapods-mPaaS"
source "https://code.aliyun.com/mpaas-public/podspecs.git"
#use_pod_for_mPaaS!
mPaaS_baseline 'cp_change_29273'
mPaaS_version_code 2   # This line is maintained by MPaaS plugin automatically. Please don't modify.
# mPaaS Pods End

platform :ios, '9.0'

target 'CubeProject' do

  mPaaS_pod "mPaaS_Cube"

end
```

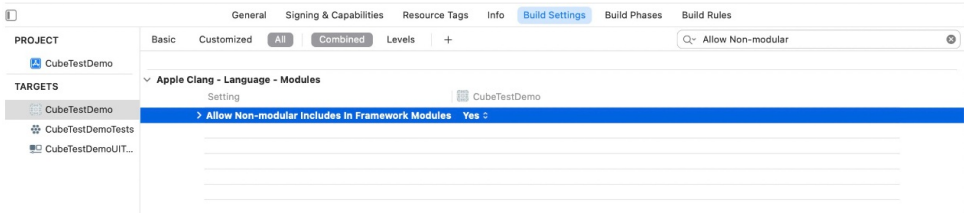
c. The current version strongly depends on mini program, and you need to use mPaaS_pod "mPaaS_Ariver" to add mini program components.

```
Podfile
# Target name must match your project!

plugin "cocoapods-mPaaS"
source 'https://gitee.com/mpaas/podspecs'
mPaaS_baseline 'cp_change_15200851'
mPaaS_version_code 30   # This line is maintained by MPaaS plugin
automatically. Please don't modify.
# mPaaS Pods End

 platform :ios, '10.0'

target "MPCubeDemo_Pod" do

 #
 mPaaS_pod "mPaaS_Cube"

 #
 mPaaS_pod "mPaaS_Ariver"

end
```

b. Execute the `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.



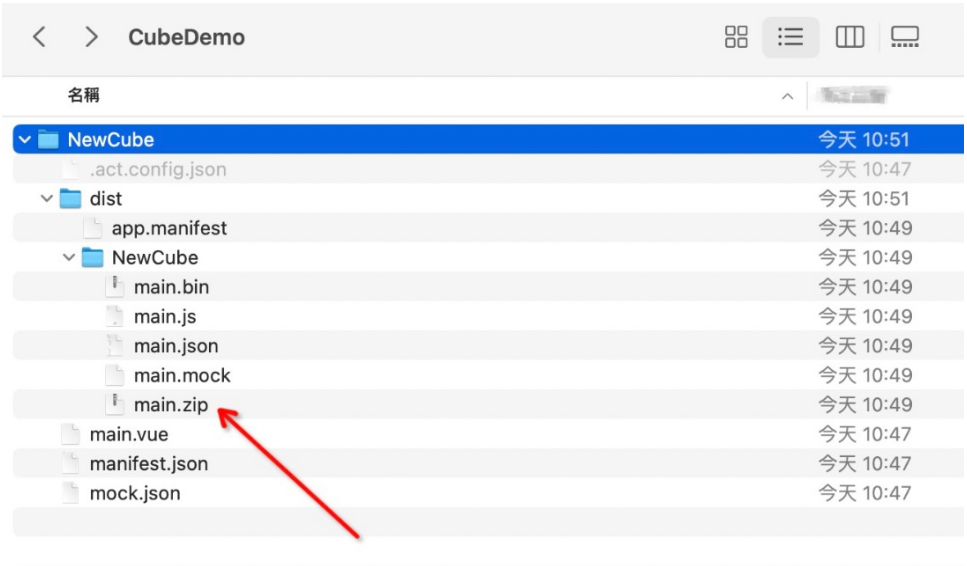
ii. Introduce libraries required for the Cube Card.

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

iii. 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.
      - Do not select "Apply to Alipay Client Card Business Scenario".
      - Select "Need to create additional project source code folder". An additional folder will be created with the application name. If you select Not Required, the system is initialized 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 build. The built product will be in the `/dist/` folder of your project.



4. Release the card.
   i. **Go to the backend of card.**
   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.**

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

The card obtains the key :"iosImage", and the value corresponding to the key :"iosImage" is obtained, as shown in the following code:

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

The proxy method after the card is created needs to follow the proxy `CCardCallback` and needs to implement the `onLoaded:cardType:config:erroCode` protocol method.

```
- (void)onLoaded:(CubeCard *)card cardType:(CCardType)cardType config:(CubeCardConfig
*)config erroCode:(CubeCardResultCode)erroCode {
    // Failed to create the card.
    if (!card) {
        NSLog(@"load card fail %@ style %d error %d", [config templteId], cardType, e
rroCode);
        return;
    }
    // The creation is successful.
    self.card=card; // You need to hold the card before you can operate it.

    NSLog(@"load card success %@ style %d error %d", [config templteId], cardType, er
roCode);
    dispatch_async(dispatch_get_main_queue(), ^{
        CubeView *view = [[[CubeService sharedInstance] getEngine] createView];
        CGSize size = [card getSize];
        [view setFrame:CGRectMake(0, 320, size.width, size.height) ];
        [self.view addSubview:view];
        [self.card renderView:view];

    });
}
```

6. **Effect preview.**

2.3.2. Advanced Guide

2.3.2.1. Render Card

This topic describes how to render cards on an iOS client.

The process of rendering a card is divided into four parts. The first step is to assemble the card configuration information; the second step is to request the card according to the configuration information and obtain the card instance; the third step is to use the card instance to render using the card `View`; the fourth step is to release the card in the `destroy` declaration cycle after the entire business is completed. Perform the following steps to upload an SSL certificate:

1. Assemble the card configuration information.

   Create configuration information and set various parameters.

```
- (void)createCubeConfig {
                // Set card parameters.
    CubeCardConfig *cardConfig = [[CubeCardConfig alloc] init];
    // The card version.
    [cardConfig setVersion:@"1.0.0.0"];
    // The ID of the card created in the background.
    [cardConfig setTempleteId:@"987654321"];
    // The preset width.
    [cardConfig setWidth:MP_Screen_Width - 40];
    // The preset height.
    [cardConfig setHeight:CGFLOAT_MAX];
}
```

2. Request a card.

   If a card is requested based on the assembled card configuration information, the card engine goes to the server to obtain the card template information. You can use the `createCard` method to request one card at a time, or you can use the `createCards` method to request multiple cards at a time.

```
- (void)requestCard {
    // Load a single card
    [[[CubeService sharedInstance] getEngine] createCard:cardConfig callback:self];

    NSMutableArray *cardArray = [NSMutableArray array];
    [cardArray addObject:cardConfig];
    [cardArray addObject:cardConfig];

    // Load multiple cards
    [[[CubeService sharedInstance] getEngine] createCards:cardArray callback:self];
}
```

3. Render the card.

   After the card information is obtained, a card View is generated and then rendered in the main thread. In this phase, abnormal judgment is required to prevent the card information from being successfully obtained.

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

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

    dispatch_async(dispatch_get_main_queue(), ^{
        // The main thread refreshes the UI.
        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];
    });
}
```

4. Release the card.

   After the card is used, the memory resources of the card need to be released, usually called or actively destroyed during the `dealloc` life cycle of the page.

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

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

2.3.2.2. Real Machine Preview

This topic describes how to preview a card on an iOS client.

Prerequisites

- You have activated and integrated to mPaaS.
- Ant Cube Card AntCubeTool tool has been installed. For more information, see About AntCubeTool.
- The connection process is completed as described in Quick Start.

Procedure

1. **Add dependencies for real-world preview. Pod is used as an example.**

   i. Add a Pod module.

   ```
   mPaaS_pod "mPaaS_Cube"
   mPaaS_pod "mPaaS_Cube_Dev"
   #The current version of the card strongly depends on the mini program and requires the pod mini program component
   mPaaS_pod "mPaaS_Ariver"
   ```

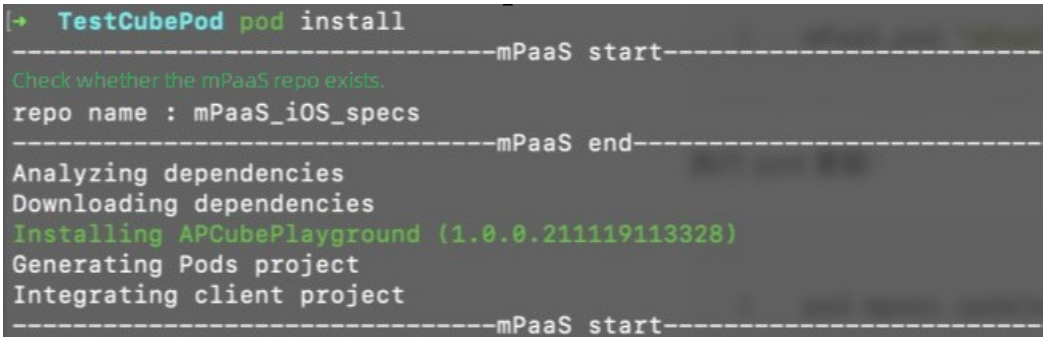
   > **Important**
   > - mPaaS_Cube and mPaaS_Cube_Dev must be used in pairs.
   > - Before releasing, be sure to delete mPaaS_Cube_Dev.

   ii. Perform a pod update.

   ```
   pod mpaas update cp_change_15200851
   ```

iii. Execute the `pod install`.

```
pod install
```



iv. Add the dependency libraries required for code scanning and preview.

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

v. Set up the preview page.

> **Important**
>
> - When using the preview function, you need to ensure that the engine has been initialized successfully.
> - If you use a custom extension module, you need to ensure that the custom module has been registered, otherwise it will be invalid.

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

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

If you define the preview page yourself, you need to actively register the `listener` and implement the `CubeCardPreviewListener Protocol`:

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

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

}
```

vi. Call the scan code preview method.

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

> **Important**
>
> The current App needs to have a `NavigationController`, otherwise no response will be received after the scan is called.

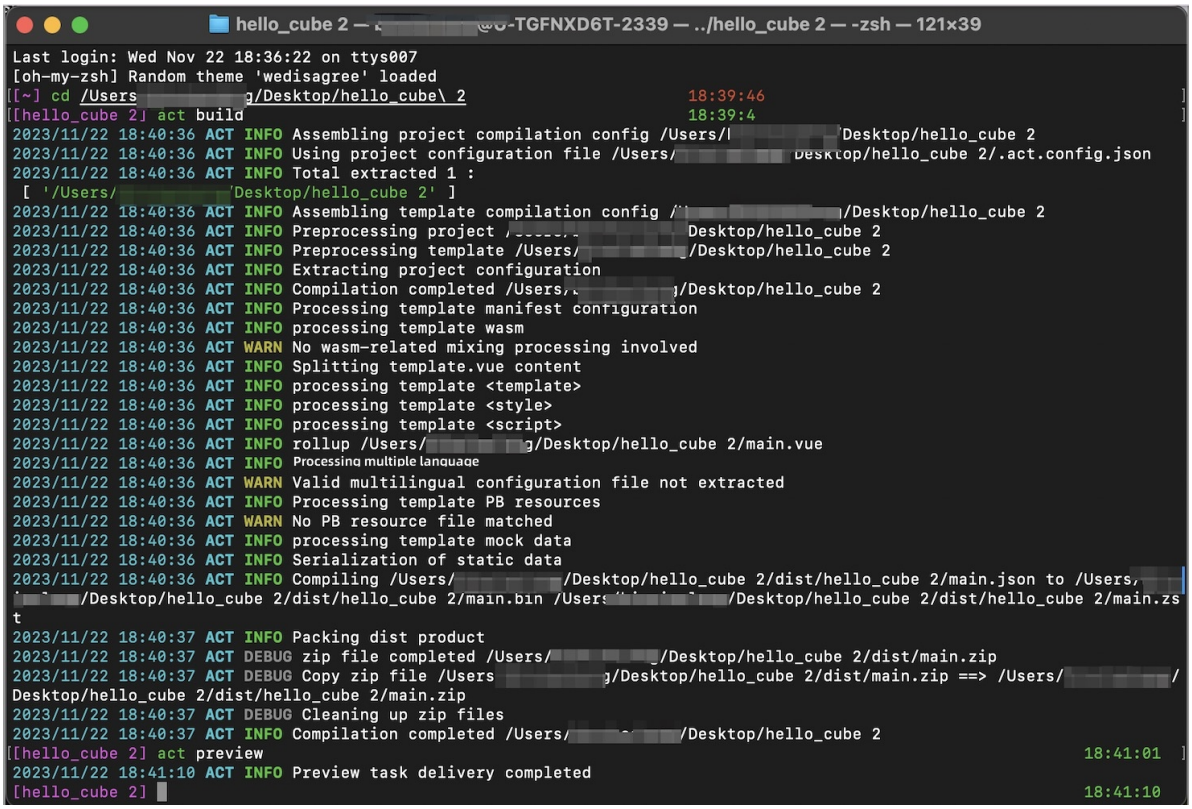
2. Use the command line to start the local debugging service. In the path of the project, run the command to open the service. The instructions for enabling the service on macOS and Windows are as follows:
   - macOS: `act prepare && act server`
   - Windows: `act prepare | act server`

   After the instruction is executed, a QR code is generated at the terminal.

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

4. Card preview.

   Create a new window in the terminal, open another window, use the cd command to switch to the project directory and execute `act build` to complete the compilation. Then call `act preview` to push the compiled content to the client.



   - If set to use the default preview page, the client can automatically jump to the card showing the preview.

- If you define the preview interface yourself, the `CubeCardPreviewListener Protocol` callback will be triggered. At this time, the business side can get the `CubeCard` instance and render it by itself.

2.3.2.3. Initialize engine extensions

This topic describes how to initialize engine extensions in an iOS client.

Prerequisites

- You have integrated the project to mPaaS. For more information, see Integration methods.
- An Ant Cube Card is published in the console. For more information, see Publish an Ant Cube Card.

Procedure

1. **Specifies the preset card local path.**
   - **Set the properties of the card engine class `CubeEngineConfig`.**

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

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

   - Set the assets path of the local Ant Cube Card.

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

2. Set exception monitoring.

   `CubeEngineConfig` supports exception monitoring to catch frontend code exceptions. The client needs to follow the `CExceptionListener` proxy and implement the monitoring method.

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

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

```
// CExceptionListener the 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 monitoring
 @ param info The error message.
 */
- (void)onException:(CExceptionInfo *)info;

@end
```

```
@interface MPCubeManager () <CExceptionListener>

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

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

3. Set the network image download handler.

   The `CubeEngineConfig` supports intercepting Image downloads and customizing image parameters; the client needs to follow the `CKImageHandler` proxy to implement the listening method.

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

/// The image handler. If this parameter is empty, the internal handler is implemente
d by default. We recommend that you customize the handler.
@property (nonatomic, strong) id<CKImageHandler> imageHandler;
```

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

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

#ifndef CKImageHandler_h
#define CKImageHandler_h

extern NSString *const CKImageAppInstanceIDKey; // The AppInstanceID corresponding to
the triggered image.
extern NSString *const CKImagePageInstanceIDKey; // The PageInstanceID that is used t
o trigger the loading of the image.
extern NSString *const CKImageInstanceOptionKey; // instance option. Add for falcon l
inks

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

@protocol CKImageHandler <NSObject>

@required
/**
 @ param url The download URL of the image.
 @ param size Image size
 @ param option The extended parameter, which depends on the requirements of the imag
e download library.
 @ param callback The callback that is called when the download is complete.
 @ return The unique ID of the download task.
 */
- (NSString *)fetchImage:(NSString *)url size:(CGSize)size option:(NSDictionary *)opt
ion callback:(CKImageHandlerCallback)callback;

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

@end
```

```
@interface MPCubeManager () <CKImageHandler>

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

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

    return @"20211202";
}
```

2.3.2.4. Preset Card

This topic describes how to preset Ant Cube Card in the iOS client.

Prerequisites

- You have integrated the project into mPaaS. For more information, see Integration methods.
- A card is published in the console.

Procedure

1. **Download the Bin file from the console.**
2. **Add the downloaded file to the bundle.**

   Create a 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 JSON file here is the data required by the card. You can use the built-in file or call the interface to dynamically request the data source according to your needs.

   > **Note**
   >
   > **Later, the console will directly provide the complete named zip package, without the need for users to manually modify the name, only need to be added to the Bundle.**

3. **Introduce the Bundle to the project. The Bundle path is specified when the engine is initialized.**

```
- (void)initEngine {
    // The bundle path of 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];
}
```

4. **Load preset cards.**

```
CubeCardConfig *cardConfig = [[CubeCardConfig alloc] init];
// Set the card version.
[cardConfig setVersion:@"1.0.0.0"];
// Set the card ID.
[cardConfig setTemplteId:@"main1"];
// The preset card width.
[cardConfig setWidth:MP_Screen_Width];
// The preset card height.
[cardConfig setHeight:CGFLOAT_MAX];

NSString *mockBundlePath = [NSString stringWithFormat:@"%@/%@", [[NSBundle mainBundle
] resourcePath], @"MPCubeBundle.bundle"];
NSString *fullTplPath = [NSString stringWithFormat:@"%@/%@", mockBundlePath, @"mock1.
json"];
NSString *strData = [NSString stringWithContentsOfFile:fullTplPath
encoding:NSUTF8StringEncoding error:nil];
// Set the card data. In this example, the data is read by using the built-in Json of
the bundle. You can also request the server interface to obtain the data. However, yo
u must convert the data to a JSON string.
[cardConfig setData:[self dictionaryWithJsonString:strData]];
// Create a card.
[[CubeEngine sharedInstance] createCard:cardConfig callback:self];
```

2.3.2.5. Cards call client methods

This topic describes the implementation path of calling iOS client methods in Ant Cube Card.

Procedure

1. **Create `NSObject` objects and introduce protocol `<CubeModuleProtocol>` in the project.**

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

NS_ASSUME_NONNULL_BEGIN

@interface MPCubeModule : NSObject <CubeModuleProtocol>

@end

NS_ASSUME_NONNULL_END
```

2. **Implement methods that are agreed with the template.**

   The method name agreed with the template is `push`, and the effect is to call the `Native` to implement the `Push Viewcontroller` operation when the card is clicked.

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

@implementation MPCubeModule

// cardUid is a property of the module. When calling module APIs, you can use cardUid
to determine which card generated the call. cardUid is the cardUid in the CubeCard.
@synthesize cardUid;

// Configure the convention method.
CK_EXPORT_METHOD(@selector(push:))

- (void)push:(CubeModuleMethodCallback)callback {
    if (callback) {
                // Push ViewController

        // The callback parameters to the template.
        callback(@"Push Page");
    }
}

@end
```

3. **Register a custom module.**

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

    // Register a custom module. The key is the module name (agreed with the server),
and the value is the client class name.
    NSDictionary *dic = @{@"crystalnavigator": @"MPCubeModule"};
    [[[CubeService sharedInstance] getEngine] registerModules:dic];
}
```

4. **Called on the card side.**

```
<script>
                // Specify the custom module ID.
    const navigator = requireModule("crystalnavigator");
    export default {
        methods: {
            onClick() {
                navigator.push(event => {console.info(event);});
                console.info('invoke on-click event');
            }
        }
    }
</script>
```

5. The client method can be called after the card is packaged and published to the background.

2.3.2.6. The client calls the card method

This topic describes how to call the card method on the iOS client.

Procedure

1. **Implement the called JS method on the card side.**

```
<script>
    const navigator = requireModule("crystalnavigator");
    export default {
        data: {
            message: 'Hello Cube',
            text:"The Cube engine is an easy-to-use cross-platform development
solution that can build high-performance with a Web-based development experience",
            string: "defaultString",
            temp:"https://gw.alicdn.com/tfs/TB1dZ4WowoQMeJjy0FnXXb8gFXa-950-1267.jpg"
        },
        methods: {
                        // The JS method called by the client.
            jsTestMethod(string) {
                    // Update page parameters.
              this.string = string;
           }
        }
    }
</script>
```

2. Pack the card and publish it to the card background.
3. The client calls.

   Obtain the card instance in the agent method of card loading, 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:@"Failed to create:
templteId=%@,style=%d, error=%d", [config templteId], cardType, erroCode];
        NSLog(@"Error message :%@", errMsg);
        return;
    }

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

    dispatch_async(dispatch_get_main_queue(), ^{
        NSString *text = @"The client calls the card JS method to pass the parameter:
Hello World";
        if (![text isEqualToString:@""]) {
            NSArray *valueArray = @[text];
            // Call the JS method agreed with the card and pass in the parameters.
            [card callJsFunction:@"jsTestMethod" arguments:valueArray];
        }
    });
}
```

2.3.2.7. Card Custom Label

This topic describes how to customize labels in cards.

1. **The client registers a custom label (custom view).**
   i. **Implement a custom label (custom view).**

   Inherit the `CCardWidget` and implement its protocol methods, where `onCreateView` and `updateData` are required, and other methods are optional.

```
#import "CustomCardView.h"

@interface CustomCardView ()

@property (nonatomic, strong) UILabel *titleLabel;

@end

@implementation CustomCardView

// You must implement the
/**
 * UI creation interface, which indicates the view UI of the current component to b
e displayed on the screen.
 *
 * @param data map type: the data declared by the widget on the card, including the
style, event, and property. The corresponding keys are DATA_KEY_STYLES, DATA_KEY_EV
ENTS, and DATA_KEY_ATTRS;
 * @param size View size, width and height
```

```
 * @return Available 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, 3
0)];
    // The data source is provided by data.
    self.titleLabel.text = @"A";
    [customView addSubview:self.titleLabel];

    return customView;
}

// You must implement the
/**
 * Component data update interface
 * @param data
 */
- (void)onUpdateData:(NSDictionary *)data {
    NSLog(@"Print data :%@", data);
}

/**
 * The existing controls obtained from the reuse pool are used before the data is
prepared.
 * @param data The initial data.
 * @param size The size of the component when it is reused.
 */
- (void)onReuse:(NSDictionary *)data size:(CGSize)size {
    NSLog(@"Print data :%@", data);
    NSLog(@"Size Print :%@", size);

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

}

/**
 * Indicates whether the component supports reuse.
 * In order to improve efficiency, the extension component can support multiplexing
. For example, when a custom label component is removed from the current view due t
o data update, if the component supports reuse, it will be put into the reuse pool.
The next time the component is displayed, it will be directly obtained from the reu
se pool.
 * @return YES: reuse. NO: not reuse.
 */
- (BOOL)canReuse {
    return YES;
}
```

```
/**
 * Component reuse cleanup interfaces. If the component supports reuse (canReuse returns true), the component calls the
 * onRecycleAndCached method, which indicates that the current component has been put into the cache off-screen and needs to clear resources.
 */
- (void)onRecycleAndCache {
    // Clean up the SubView content on the custom View
    self.titleLabel.text = @"";
}
```

ii. **Register a custom label.**

The first parameter is a custom label, which needs to be agreed with the card side and will be written in <> on the card side. We recommend that you add a prefix to prevent label conflicts with dynamic cards. The second is the class name of the custom label implementation class.

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

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

2. **Called on the card side.**

In Labels, write the custom label. Here is the `custom-widget`, which corresponds to the label registered by the client.

> **Note**
>
> The custom label to be written must be the same as the first parameter in the Register Custom Tag step.

```
<template>
    <div class="root">
        ...
        <custom-widget></custom-widget>
                                ...
    </div>
</template>
```

3. Publish the card package to the background or preview to render the custom view on the client.

2.3.3. Usage notes

2.3.3.1. 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;
```

2.3.3.2. 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;
```

2.3.3.3. CubeEngineConfig

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

Internet 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;
```

2.3.3.4. 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;
```

2.3.3.5. 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
};
```

2.3.3.6. 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;
```

2.3.3.7. CubeCard

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

Internet Properties

cardUid

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

Method

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;
```

2.3.3.8. CubeCardConfig

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

Internet 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;
```

2.3.3.9. 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

2.3.3.10. 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;
```

2.3.3.11. 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;
```

2.3.3.12. CubeModuleProtocol

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

Internet Properties

cardUid

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

CubeModuleMethodCallback

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

2.3.3.13. 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

2.3.4. Connection type demo reference

To view a demonstration of this connection type, please single click: MPCubeDemo_Pod.zip.

3.Use the console

3.1. Card Management

3.1.1. Create a card

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

Procedure

1. Log on to the mPaaS console. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Publishing** > **Ant Cube Cards** > **Card Management**. The **Card Management** page appears.
3. On the page of **Card Management**, click **Create Card** and enter the following information in the **Create Card** form:
   - **Card ID:** required. We recommend that you use a combination of English, numbers, and underscores. The card ID must be at least 8 characters in length.
   - **Card Name**: required. You can take any value.
   - **Description**: optional.
4. Click **OK**.

At this point, you have finished creating the card. The created card appears in the card list.

What to do next

After a card is created, you must upload a card resource before you can publish the card. For more information, see Add card resources.

3.1.2. Add card resources

This topic describes how to add a card resource to an Ant Cube Card in the mPaaS console.

Procedure

1. Log on to the mPaaS console. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Publishing** > **Ant Cube Cards** > **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**.
   - **Card Version Number**: required. The version number shall be four digits and the format shall refer to `0.0.0.1` .
   - **File**: the. zip file of the card template. The size of the. zip file must be less than 10MB. Please refer to the quick start document (Android, iOS) for the development of the card template.
   - **Client Version Range**: the range of client versions that can be pulled to the card. The maximum version and minimum version are included. The effective scope of the client must be differentiated by platform.

- **Minimum Version**: The minimum client version that you enter takes effect. If you want to overwrite all client versions, enter the `0.0.0.0` in the Minimum Version field.
- **Highest Version**: The card takes effect only when the client version is entered. If this parameter is not specified, all versions above the minimum version take effect.

> **Note**
>
> If you specify the highest version, the offline package may not take effect when the client is upgraded to a later version.

3.1.3. 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. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Publishing** > **Ant Cube Cards** > **Card Management**. The **Card Management** page appears.
3. In the **Card List** section of the **Card Management** page, move the pointer over the card and click.
4. In the message that appears, click **Delete**.

At this point, you have finished deleting cards.

3.1.4. Create a publishing task

After you create a dynamic ant card and add a card resource, you must publish the dynamic ant card before the client can pull the card. This topic describes how to create an Ant Cube Card Publishing task in the mPaaS console.

Prerequisites

If you want to publish a card in the whitelist mode, you must create a whitelist before you publish the card. For more information about how to create a whitelist, see Whitelist management.

Procedure

1. Log on to the mPaaS console. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Publishing** > **Ant Cube Cards** > **Card Management**. The **Card Management** page appears.
3. In the **Card List** on the **Card Management** page, click the card that you want to publish.
4. Click **Create Release**. The **Create Release** page appears.
5. On the **Create Release** page, set the following parameters and click **OK**.
   - **Release Type**: required. Grayscale release and official release. If you select phased release, you can configure the release model, whitelist, and advanced rules. If you select formal release, you do not need to configure these parameters.
   - **Publish Model**: required. You can select the release model of the whitelist or time window.

- **Whitelist**: You can publish cards based on the whitelist. Only users who are in the whitelist can pull cards.
  - **Whitelist**: If you set the Release Model parameter to Whitelist, you must select an created whitelist.

> **Note**
>
> If the number of selected whitelisted users exceeds 100000, only the first 100000 users are selected.

- **Time Window**: Within a specified time range, a phased release is performed based on the maximum number of users.
  - If you select a time window model for phased release, you need to specify the time window closing time and the number of people in the phased release.

- **Release Description**: optional. Notes the current publishing task.
- **Advanced Rules**: optional. You can publish within a specific rule scope. You can select the platform, type, operation type, and resource value.
  - **Platform**: required. iOS, Android, all.
  - **Type**: required. City, model, network, and device system version.
  - **Operation Type**: required. Included or not included. 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 publishing rule in advance and call this operation here. For more information about how to add resource values for advanced rules, see Manage publishing rules.

At this point, you have finished publishing the card. After the card is published, the task is in the **Publishing** state. For a task that is being published, you can view, pause, or end the task. If a card has a version that is in the Publishing task, you cannot create a publishing task. You must wait until the current publishing task is finished, or end the current Publishing task before creating a new task.

3.2. Card analysis

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

View card analysis

1. Log on to the mPaaS console. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Publishing** > **Ant Cube Cards** > **Card Analysis**. The **Card Analysis** page appears.

   This page displays the number of new users and the trend chart of new users, the number of active users and the trend chart of active users, the number of card visits and the trend chart of card visits, the number of card clicks, and the trend chart of card clicks. You can query card analysis by card name, card version, platform, application version, and date range.

   - New Users: the number of new users (device IDs) who access the card.
   - Number of Active Users: the total number of users (device IDs) who access the card. Multiple visits by the same user are not counted repeatedly.
   - Card Visits: the total number of visits to the card.
   - Card Clicks: The total number of clicks on the card.

Query card analysis

1. On the **Card Analysis** page, set Card Name, Card Version, Platform, Application Version, Date Range, or Time Period.
2. Click **Search** to view card analysis details.

3.3. Card performance

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

View card performance

1. Log on to the mPaaS console. In the left-side navigation pane, select an application.
2. In the left-side navigation pane, choose **Real-time Release** > **Ant Cube Card** > **Card Performance**. The **Card Performance** page appears.

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

   - The information returned for the request.
     - Active Users: the total number of users (device IDs) who access the card. Multiple visits by the same user are not counted repeatedly.
     - page view: The total number of visits to the card.
   - Package pulling exception
     - Abnormal Pull Count: the number of abnormal pull requests for the card.
     - Abnormal pull package rate: Abnormal pull package request rate=Abnormal number of pull package requests for a card /Total number of pull package requests for a card
   - Card Errors, which displays the number of card errors, number of affected users, error type exception curve, error type exception distribution, and detailed error information list.
     - Errors: the number of errors in the card (JS exceptions + style exceptions).
     - Affected Users: Number of users (device IDs) affected by the card error.

Query card performance

1. On the **Card Performance** page, set Card Name, Card Version, Platform, Application Version, Date Range, or Time Period.
2. Click **Search** to view the card performance details.

4.Development tools

4.1. About AntCubeTool

AntCubeTool (ACT) is a command-line tool used to assist Ant Cube Card in the development, debugging, and preview of application scenarios. This article provides a basic introduction to the tool.

ACT is available for both macOS and Windows platforms. The specific requirements for these two platforms are as follows:

- macOS
  - The system version is not less than 11.0.
  - Command line: You can use the terminal that comes with macOS or use other command line tools.
- Windows
  - The system version must be Windows 10 or later and must be a 64-bit system.
  - Command line: Use the PowerShell command line tool that comes with Windows.

4.2. Install AntCubeTool

This article describes the installation process of AntCubeTool, which is an Ant Cube Card development tool.

Install the SDK Credentials package

Use the `npm install xcube@en -g` to install the ACT. `act` commands can be used on the command line after xcube is successfully installed.

```
npm install xcube@en -g
```

> **Important**
>
> When installing, do not use sudo to install.

4.3. Use AntCubeTool

This article describes some of the `act` commands that need to be used in common scenarios where AntCubeTool is used.

Create a project

`act init` : New construction.

> **Note**
>
> The name field in the configuration is used as the basis for creating files or folders. Only letters, numbers, underscores, and midlines are supported.

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

Initialize a Cube application in the current directory

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

Enable Service

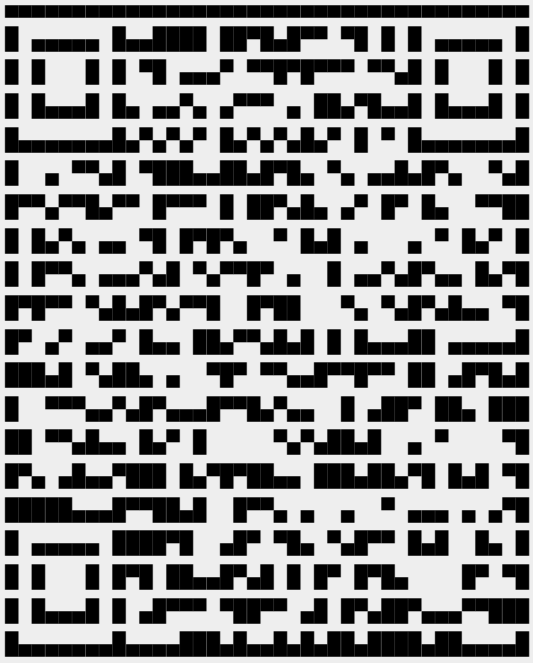
`act prepare` : clears historical ports.

`act server` : Enable connection listening.

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

Prepare the environment.

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

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

Start Communication Server

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

These two commands are usually used in combination in macOS. After these two commands are executed in the terminal, you cannot close the terminal window until the startup is successful. After the startup is successful, you can see the following information on the terminal. If you need to execute other `act` commands, you can open a new terminal window to execute.

```
➜ ~ 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] Start service [30.30.200.193:9001]
2021-1-7 16:38:26 [ACT] [INFO] The communication channel is ready. Scan the QR code or
enter the port number to connect.
2021-1-7 16:38:26 [ACT] [INFO]
```



```
2021-1-7 16:38:26 [ACT] [INFO] The routing module is started
2021-1-7 16:38:26 [ACT] [INFO] The service is started. Do not close the terminal window
.
```

Create a connection

You can use the **Ant Cube card-development tool** to establish a connection. For more information, see the following topics:

- Android real machine preview
- iOS real machine preview

Compile a project

`act build` : Compile the project.

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

Compile the card application in the specified path. If no path is specified, use the cu
rrent path.

Options:
  -p, --path [v] path to be compiled, compatible with the old version of the tool [obso
lete: please directly use build [path]]]
  -a, --all <v> batch compilation, compatible with the old version of the tool [obsolet
e: please use build [path] --batch]]
  -- Indicates whether the batch mode is used for batch processing (default: false).
  -- Specifies whether to enable real-time compilation for watch. Batch processing mode
is not supported (default: false)
  -h, --help      display help for command
```

When the `--watch` parameter is configured, that is, when real-time compilation is enabled, ACT monitors file changes in the entire project directory and automatically triggers compilation when files are saved.

> **Important**
>
> Do not close the terminal window when live compilation is enabled.

Preview project

You can use **Ant Cube Card -Development Tool** to preview a project. For more information, see the following topics:

- Android real machine preview
- iOS real machine preview

Real-time preview

You can use the **Ant Cube card-development tool** to perform real-time preview. For more information, see the following topics:

- Android real machine preview
- iOS real machine preview

View the current connection QR code

`act qrcode`: View the current connection QR code. This command is used in scenarios where you need to reconnect to a Playground device. After the act server has been running for a long time, there are too many logs and the content of the QR code output by the act server cannot be seen. In this case, you can run the `qrcode` command to view the QR code again.

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

Display the QR code available for connection

Options:
  -o, --onlyInfo only outputs the QR code content, which is disabled by default. If ena
bled, the simulated QR code image content is not output (default: false)
  -p, --pure only outputs the content and does not contain formatting information (time
, prefix, etc.) (default: false)
  -h, --help      display help for command
```

By configuring the parameters, you can output only the QR code content string. Other ACT-based tools can be considered to obtain the QR code content in this way.

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

View currently connected devices

`act device` : View the currently connected devices. This command is mainly used to check whether a Playground device is connected to the current server. If the device is not connected, the preview and alive preview functions cannot work.

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

Display a list of currently connected devices

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

View help information

- View supported instruction sets

```
act -h
```

- View instruction-related parameters

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

View information about the Internet access package

```
act info
```

4.4. Update AntCubeTool

This topic describes the commands that are used to update AntCubeTool.

Use the `act update` command to update the ACT.

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

Check and update to the latest version.

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

You can also reinstall to get the latest version by `npm install xcube@en -g` the command.

4.5. Uninstall AntCubeTool

Use the `npm uninstall xcube -g` command to uninstall ACT.

```
➜ ~ npm uninstall xcube -g
```

5.Card Syntax

5.1. Card Basics

5.1.1. Project Management

This topic describes the 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` is the configuration file of the card project. It is currently generated by the Ant Cube Card CLI tool and it does not need to be modified or concerned.

> **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 CLI 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.
}
```

Engineering example

test_cube.zip

5.1.2. 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.

5.1.3. 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 Common style. 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.

Other scenarios

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

5.1.4. Data Binding

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

Single data source binding

For more information about the Vue format, single data binding in template mode supports interpolation and instructions.

- Interpolation formats can be used only separately, and cannot be mixed, such as <text>ab{{var1}}cd</text>.
- The command form supports the shorthand format.

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>
logic controller	v-bind:name="variable"	:name="variable"	. or []	<text :value="var1[num1]"></text>

The formats supported when a text class component (such as text) is populated with 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 substituting two sets of data at the same time as the data source to be bound, mainly to solve the actual business development process in the native side to inject additional control data into the template, when such needs, to enable this function.

The data injected by the template is merged with the mock data sent by the server. The injected data has a higher priority. If there are the same fields, the corresponding fields in the mock data will be overwritten. The usage method 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>
```

Examples

Click here detailBindData.zip for the complete sample code.

5.1.5. 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 Card components.

- 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>
```

Examples

Click here detailBindEvent.zip for the complete sample code.

5.1.6. Logical rendering

This article describes the types of logical rendering and the corresponding instructions.

Conditional rendering

v-show

`v-show` instructions are used to conditionally render the content of a node. The node is rendered regardless of whether the result of the `v-show` instruction is `true` or `false`.

- If the `v-show` result is `true`, it is equivalent to adding a `display: flex` to the CSS style of the node.
- If the `v-show` result is `false`, it is equivalent to adding a `display: none` to the CSS style of the node.

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

v-if

`v-if` instructions are used to conditionally render a piece of content. This piece of content will only be rendered when the expression of the directive returns a `true`.

- You can use `v-else-if` to act as `v-if` "else-if blocks" that can be used consecutively.
- You can use `v-else` to act as "else blocks" 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**
>
> - The `v-else-if` must immediately follow the `v-if` block or it will not be recognized.
> - The `v-else` must immediately follow the `v-if` or `v-else-if` block or it will not be recognized.
> - `v-if` cannot be written on the root node of a template.

List rendering

vfor

`v-for` instructions may render a set of data based on an array `/number/` object. Currently, six methods 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**
>
> - `v-for` instructions do not support nested use, that is, `v-for` can no longer use `v-for` for rendering.
> - When rendering data based on objects, the order in which objects are displayed is not controllable. To display in a determined order, iterate through the `array/number` data.
> - `v-for` cannot be written on the root node of a template.

Examples

Click here detailLogicalRender.zip for the complete sample code.

5.2. Card components

5.2.1. div

 defines a division or section in a document. The tag can divide a document into separate and distinct parts. It can be used as a strict organizational tool and is not associated with it using any formatting.

Embedded component support

You can nest any other component.

Styles

The component supports all styles in Common Styles.

Internet Properties

None

Event

The component supports all events in Common Events.

Examples

```
<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
  }
```

Examples

Click here detailDiv.zip for the complete sample code.

5.2.2. text

<text> is a built-in component of Ant Cube Card Engine. It is used to render text in a specified style. <text> can only contain text values. You can use the {{}} tag to insert variable values as text content.

Embedded component support

You cannot nest any other components.

Styles

The <text> component supports all styles in Common Styles and other special styles:

Font-related

Parameter	Description	Value Type	Default value	Optional Value	Writing	Description
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 equals 400. iOS supports nine values, Android supports three kinds of values: normal,bold and between (normal,bold). The corresponding values are normal effect for 400, bold effect for 700 and above, and FakeBold effect for 400 and 700.	font-weight: bold;	
					font-weight: 700;	
font-style	font style	string	normal	normal, italic	font-style: normal;	

font-family	set the font	string	platform default font		font-family: PingFangSC-Regular;	This setting is not guaranteed to be consistent across different platforms and devices. If the specified font is not available on the platform, it will be downgraded to the default font of the platform.

Typography-related

Parameter	Description	Value Type	Default value	Optional Value	Writing	Description
lines	number of text lines	int	0 indicates that the number of rows is not limited.		lines: 10;	
text-align	font alignment	string	left	left, center, right	text-align: center;	
text-overflow	set the ellipsis style when the content is too long.	string	clip	clip, ellipsis	text-overflow: clip;	

line-height	set text line height	length unit + value	0		line-height: 12px;	• fontSize* value when the value is numeric. • If it is the length unit px suffix, it is the value itself. • If the value is a percentage of the length unit, the value is fontSize* value.
				• normal: White space characters are folded, and the text is automatically line-wrapping, not based on line breaks or paragraphs in the content.		

white-space	control line breaks and whitespace policies in text	string	pre-wrap	• nowrap: White space characters are collapsed, the text does not wrap, and one line is displayed, which can exceed the background frame. • pre: White space characters are not folded, and lines are changed according to the text content. Lines are not automatically changed within each paragraph (one line per paragraph can exceed the background box).	white-space: nowrap;	-

				• pre-wrap: White space characters are not folded, the paragraph breaks according to the content, and the paragraph automatically breaks. pre-line: White space characters are folded, the paragraph breaks at the end of the line, and the paragraph automatically breaks.		
word-wrap	control the folding method (word split or not)	string	break-word	• normal: The paragraph breaks at the end of the word, and the paragraph breaks automatically. • Break-word: Fold the line at the end of the word, but the length is still not enough, you can fold the line in the middle of the word. • anywhere: The line can be folded anywhere.	word-wrap:beak-word:	-

word-break	control how words are split when folding	string	None	• normal: The word does not split the line and can exceed the backgro und box. • break-all: The word can be split and folded. • keep-all: the same as normal.	word-break;	Do not distinguish between Chinese and English, and Chinese and English mixed situation.
letter-spacing	control character interval, positive, negative	string	0	normal: No extra space is available. Single Value Length Unit: positive or negative.	letter-spacing:5px;	-
text-indent	the indentation of the first line of text, which can be positive, negative, and can be a percentage (percentag e of the width of the parent element)	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 Baselines are related to fonts. Use them with caution. • The `middle` aligns the middle of the element to the baseline of the parent element plus half the x-height of the parent element.			

vertical-align	text vertical alignment style	string	baseline	• `baseline` align the baseline of the element with the baseline of the parent element. The HTML specification does not specify baselines for some of the replaceable elements, as `<textarea>`, meaning that the behavior of these elements with this value varies from browser to browser. • `sub` aligns the baseline of the element with the subscript baseline of the parent element.	vertical-align:middle	-

				• `super` aligns the baseline of the element with the superscript baseline of the parent element. • `top` aligns the top of the element and its descendant elements to the top of the entire row. • `bottom` aligns the bottom of the element and its descendant elements to the bottom of the entire row. For elements without a baseline, the lower edge of the outer margin is used instead.		

Effect-related

Parameter	Description	Value Type	Default value	Optional Value	Writing	Description
color	font color	color unit	0x000000		color:red; color:#333; color:rgb(255,0,255);	

text-decoration	font decoration	string	none	underline,none,line-through,overline	text-decoration: underline;	-
text-shadow	font shadow	length units&color units		The supported format is ${x}${y}${size}${color},x,y,size meets the unit of length, and color meets the unit of color.	text-shadow: 2px 2px 3px gray;	Color The default font color. x and y are required and optional.
text-shadow-color	font shadow color	color unit	same as color		text-shadow-color: blue;	The optional parameters.
text-shadow-offset	font shadow offset	length unit		-	text-shadow-offset: 2px 2px;	Required parameter
text-shadow-radius	font shadow radius	length unit	0		text-shadow-radius: 3px;	The optional parameters.

Internet Properties

Parameter	Description	Value Type	Default value	Writing	Description
value	the value of the component, text content	string		<text value="Text content string"></text>	
line-space	line spacing, such as 4px	length unit		<text line-space="4px"></text>	

Event

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

Examples

```
<text class="text">
                The Cube engine is an easy-to-use cross-platform development solution fo
r building high-performance, scalable native applications with a web-based development
experience. Vue is a lightweight and powerful progressive front-end framework.
</text>


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

Differences from Web

The following table describes the differences between Ant Cube Card and Web.

- **Specifies whether to distinguish between Chinese and English processing for word-break (not supported by Ant Cube Card).**
- **When a long word exceeds the background box, it does not distinguish between Chinese and English mixed rows.**

Web	Ant Cube Card
The case where the word-wrap is not written is the same as that of word-wrap:normal.	The case where the word-wrap is not written is the same as that of word-wrap:break-word.
word-wrap: Normal words are not folded and displayed beyond the background box (the principle of recognizing words is that consecutive English characters are words).	word-wrap: Normal words are not folded and are displayed beyond the background box (the principle of recognizing words is continuous English characters, including mixed Chinese characters in the middle).
word-break: Normal Chinese split line, English words do not split line, can be displayed beyond the background box.	word-break: Normal Chinese split line, English words do not split line, can be displayed beyond the background box.
word-break: Keep-all Chinese does not split the line, English words do not split the line, can be displayed beyond the background box.	word-break: Keep-all is the same as normal.
-	Letter-Spacing is supported in Android 5.0 and above.

Examples

- Click here detailFontSize.zip for complete sample code on font size.
- Click here detailFontWeight.zip for complete sample code on font weight.
- Click here detailTextAlign.zip for complete sample code on font alignment.

5.2.3. image

<image> is a built-in component of Ant Cube Card Engine. It is used to render a single image in a specified style.

Embedded component support

You cannot nest any other components.

Styles

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

Internet Properties

Parameter	Description	Value Type	Default value	Optional Value	Writing
src	Online address or local address of a component image, or Base64-encoded data	string		• URL ("https:/xxx") Alibaba Cloud Content Delivery Network address. • URL ("./xxx") The relative address of the offline package. • The URL ("data:") is Base64-encoded.	src="https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png"
resize	Image location	string	stretch	stretch | cover | contain | top | bottom | center | left | right | top left | top right | bottom left | bottom right. Beyond this range, the default cover is used.	resize: contain
placeholder	Component accounts for the online address or Base64-encoded data	string		• URL ("https:/xxx") Alibaba Cloud Content Delivery Network address. • The URL ("data:") is Base64-encoded.	placeholder="https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png"

Event

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

Examples

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

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

> **Note**
>
> - The image of the <image> component does not support the SVG format.
> - In the following cases, the original image is downloaded.
>   - The resize parameter is not cover/contain/stretch.
>   - The node does not have width or height.

Click here detailImage.zip for the complete sample code.

5.2.4. richtext

<external-richtext> is a built-in component of Ant Cube Card Engine for rendering rich text.

Embedded component support

You cannot nest any other components.

Styles

The <external-richtext> component supports all the Common Styles and some special styles.

Format

The HTML tag format is supported. Currently, the following font-related tags are included.

Tag	Description	Writing	Description
br	Line break	 To break lines in a paragraph, use the br tag. 	-

span	Used to group inline elements in a document	`<span>some text.</span>` some other text.	-
div	Split a document into separate, distinct parts	`<div style="color:#00FF00">` This is a header This is a paragraph. `</div>`	-
b	Bold text	This is plain text- This is bold text .	-
del	Remove text entries	a dozen is `<del>20</del>` 12 pieces	-
h1	Title 1	This is title 1	-
h2	Title 2	This is title 2	-
h3	Title 3	This is title 3	-
h4	Title 4	This is title 4	-
h5	Title 5	This is title 5	-
h6	Title 6	This is title 6	-
i	Italic text	Email italic cn42du@163.com or ifat3@42du.online	-

p	Define paragraph	This is some text in a very short paragraph	-
img	Definition Image	src: download link width/height: image drawing width/height 	The default value for width and height is the same as the font height. If the width and height are set to be greater than the row height, the drawing effect does not support the stretching of the row height.
a	Define Link	href: the URL of the link. <a href='https://www.alipay.com'>	-

Font-related

Parameter	Description	Value Type	Writing	Description
font-size	Font size	Length unit	<span style=\"font-size:30px;\"> Inline 30px	-

Effect-related

Parameter	Description	Value Type	Writing	Description
color	Font color	Color unit	<span style=\"color:#F00\"> Inline# F00	-
font-weight	Word weight	String	<span style=\"font-weight:100\"> Inline# F00	The value is the same as the value of the Basic Style parameter in the text tag.

font-family	Font and color	String	<span style=\"font-family:thinFont\" > Font thin body	The value is the same as the value of the Basic Style parameter in the text tag.

Internet Properties

Parameter	Description	Value Type	Default value	Writing	Description
text	The value of the component, 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	Whether to check custom emoji	1/0	0	<external-richtext text="richtextContent" detectEmotionEmoji="1" > </external-richtext>	-
linkColor	Set link font color (a label)	Color unit	0xff108ee9	<external-richtext text="richtextContent" linkColor="#FF0000"> </external-richtext>	-
highlightedColor	Set link click highlight color	Color unit	0xffa9a9a9	<external-richtext text="richtextContent" highlightedColor="#0000FF" > </external-richtext>	-

> **Note**
>
> - If the height set for the img label is greater than the row height, the drawing effect does not spread the row height. By default, the height is the same as the font height.
> - If you set the width and height for the img tag, the image will be compressed and filled.
> - The a label link font color span setting takes precedence over the attribute linkcolor setting.
> - highlightedColor control the highlight color, including a tag links. If you do not specify this parameter, the default value is gray.
> - The emoticon size is the same as the font height.

Event

The <richtext> component supports all common events.

Examples

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

data: {
    richTextContent:
      '<span style=\"font-size:30px;\"> Inline 30px</span><span style=\"font-
size:30rpx;\"> Inline 30rpx</span><span style=\"color:#F00\"> Inline# F00</span>
      <span> Global 30px,color:#F00</span>
      <span> Global 30rpx,color:#0F0</span>
      <span style=\"font-size:30px;color:#00F\"> Inline 30px,color#0F0</span><span> Glo
bal 20px,color#0F0</span>
      <div style=\"color:#F00\"><div><div> outermost# F00</div></div></div>
      <div><div><div style=\"color:#F00\"> innermost# F00</div></div></div>
      <b>b</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>i</i><p>p</p><span>span</span>'
  }
```

Examples

Click here detailRichtext.zip for the complete sample code.

5.2.5. slider

The <slider> component is used to alternately display multiple images in one view.

Embedded component support

The <slider> component supports more advanced features and can only be used by nested <cell> subcomponents. <cell> is used to define the child list items in the slider. The Ant Cube Card Engine performs efficient memory reclamation on <cell> to achieve better performance.

Styles

Some capabilities in Common Styles are not supported, including padding in the box model, flex containers in the layout, flex members, background-image-related in the background, hover, and animation.

Parameter

Parameter	Description	Value Type	Default value	Writing	Remarks
auto-play	Set whether to automatically carousel several images	boolean	true	Auto-play="true"	None
interval	The interval at which images are automatically caroused. Unit: milliseconds. This parameter takes effect when auto-play is set to true.	number	500ms	interval="500"	Set a value less than 500ms, which is expressed as 500ms
infinite	Set whether to loop	boolean	true	infinite="true"	None
show-indicators	Set whether to display the indicator	boolean	false	Show-indicators="true"	None
scrollable	Set whether you can slide to switch pages by gesture operation	boolean	true	scrollable="true"	None
index	Set the number of pages on which the slider is displayed.	Number	0	index="2"	None
previous-margin	Set the distance from the previous page	Length unit	0	previous-margin="200px"	Cannot be used with `infinite="true"`

next-margin	Set the distance of the next page	Length unit	0	next-margin="200px"	Cannot be used with `infinite="true"`

Event

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

Field	Description	Parameter
on-change	This event is triggered when the carousel index changes	index: the index of the displayed image, of the number type

Examples

```
<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>
```

5.2.6. scroller

<scroller> component is a container component that holds sub-components for horizontal or vertical scrolling. It has smooth scrolling and efficient memory management and is suitable for displaying long lists.

Embedded component support

Supports arbitrary component embedding.

Styles

Some capabilities in Common Styles are not supported, including padding in the box model, flex containers in the layout, flex members, background-image correlation in the background, hover, and animation.

Internet Properties

Parameter	**Description**	**Value Type**	**Default value**	Valid value	**Writing**	Remarks
show-scrollbar	Set whether the widget displays scroll bars	boolean	false	-	show-scrollbar="true"	-
scroll-direction	Set widget scroll direction	string	vertical	vertical horizontal	scroll-direction="vertical"	-
upper-threshhold	Set how far away the component is from the top /left to trigger the event	string	50 px	-	upper-threshhold="50px"	-
lower-threshhold	Set how far away the component is from the bottom /right to trigger the event	string	50 px	-	lower-threshhold="50px"	-
offset-accuracy	Set the component during scrolling callback The frequency of	string	10 px	-	offset-accuracy="10px"	-

allow-bounce	Whether elastic effect is allowed	boolean	false	-	allow-bounce="true"	10.2.28 Support
always-bounce	Whether to allow scrolling when the content is not satisfied (note: it takes effect only when the `allow-bounce` is true)	boolean	false	-	always-bounce="true"	10.2.28 Support

Event

Common events are not supported. The following table lists the supported events.

Field	Description	Parameter
on-scroll	Trigger in scroll	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	Triggers when scrolling ends	contentSize: the width and height of the content area. contentOffset: the offset of the display area.
on-scrolltoupper	Triggers when the scroll to the top is less than the threshold	-
on-scrolltolower	Triggers when the scroll to the bottom is less than the threshold	-

Examples

```
<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=5e3520844113908195604 2a69907f7f5"></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: console log is viewed by using act debug.
                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**
>
> Scollers in the same direction do not support nesting.

5.3. Card Style

5.3.1. Style syntax

In template mode, CSS related to page styles is placed in the <style></style> section. The styles supported in the <style></style> section are determined by the styles supported by Common Styles.

CSS Style

Class, id, and type selectors are supported. More complex combinations such as parent-child and status are not supported. The following three types are supported:

```
// class
.class {
}

// id
#id {
}

// type
div {
}
```

Inline Style

The template pattern provides run time style injection capabilities, mainly through the component's built-in properties `style` fields.

The writing specification of inline style is the same as that of popular front-end frameworks (VueJS and ReactJS), and supports both bound and unbound formats.

- Bind inline styles
  - The style fields to be bound should converge uniformly into one JSONObject.
  - The KEY of the style field to be bound should conform to the camel-peak naming convention (for example, background-color should be converted to backgroundColor).
- Unbound inline style
  - The style field to be bound should converge into a string according to the writing specifications of CSS.
  - The KEY of the style field to be bound should be spliced by the hyphen-between words according to the CSS specification (such as background-color).

The priority of the style in descending order 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"
  }
}
```

The property value that the template receives on the inline style is a JS object.

```
<div class="root" :style="{height: height}"> // The preceding example
```

Dynamic Binding Class

The CSS style of the widget. You can dynamically bind different selectors.

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

Media query @ media

The template mode introduces the media query capability in the CSS specification, which is mainly used to handle the work related to mobile UI adaptation.

Compared with the CSS specification, the media query capability supported in the template mode is limited, and there are some specific usages when using it. For more information, see the following aspects. For more information about media queries, see @media.

- Media type

  You do not need to specify this parameter. By default, all is used.

Media Type	Supported
all	Yes
screen	No
print	No
speech	No

- Media Features

  In the card, the media features are mainly designed based on firmware features, which is different from the front-end browser.

Media Features	Valid value	Description
platform	ios | android	For platform adaptation, the use is different from the CSS specification. You can directly set the platform value after @ media, for example,@media android.
support	safearea	It is suitable for the safe area of the iOS platform screen.

- Media Operators

Operator	Supported
and	Yes
not	No
only	No

The following is an example of style adaptation that combines CSS features with @ media media query capabilities.

```
<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>
```

Style Import

Before you import styles, learn about the following two different types of styles:

- Imported Styles: Styles that exist in the. css file for unified management and import.
- Qualified Styles: Styles that exist in the <style></style> section of the. vue file only apply to this template.

Syntax

The syntax for style import is as follows:

```
<style src="[.css file relative path]" />
```

- File structure

```
.
└ ── template_name // The template folder (named after the template ID)
    ├── main.vue // The template layout and style description files
    ├── manifest.json // The template configuration file.
    └ ── mock.json // The test data that can be bound to the template.
    └ ── common.css // The template common style file
```

- Template

```
<template>
        ... [Template layout related description]
</template>

<style src="./common.css" />

<style>
        ... [Only the styles used in this template]
</style>
```

Cascading Rules

When you compile the style resources in the .vue file in the template mode, only the style fields with the same selector are cascaded and integrated. Different selectors are completely retained.

Import Styles + Qualified Styles = Cascading Results, as shown in the following table.

Import a style	Qualified Style	Cascading Results
`.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; }`

Examples

Download the code sample FalconDemo.

5.3.2. Common Style

5.3.2.1. Background information

Ant Cube Card provides several background element control properties.

Background Style

There are several ways to specify the background style of an element:

- background-color, which defines the background color of the element.

Properties	Value type	Default Value	Way of writing
background-color	color unit	transparent	background-color:red;

- background-image, which defines the background image of the element.

Properties	Value type	Default Value	Optional value	Way of writing	Remarks
			`url("https:/xxx")` Alibaba Cloud Content Delivery Network address; `url("./xxx")` offline package relative address; `url("data:")` base64 encoding;	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	N/A	linear-gradient(s1,s2,...,slast), the first segement (gradient angle setting, the value is that the specific angle value ends with deg, or the direction description, including top, to top, right, to right, bottom, to bottom, left, to left), the second segement (gradient starting color setting, color unit; If there is a percentage value, it must be 0%), middle segment (gradient process color setting, color unit; Percentage value is supported and must be linearly increasing. If no percentage is provided, the color proportion will be evenly divided), and the last segment (gradient end color setting, color unit; If there is a percentage value, it must be 100%); none: clears the background.	background-image: url("https://gw-office.alipayobjects.com/basement_prod/e047f6c8-dc14-481f-a22c-8dd9012b01a3.png");	N/A

background-size	String or length unit	auto	Single description value (cover,contain,auto); ${x}px ${y}px double exact length unit + percentage; ${x}px Single Value Length Unit + Percentage;	background-size:contain; background-size:100px 200px;	If the exact value or percentage is only a single value, the other value defaults to auto.
background-position	string	0	Single description value (top,right,bottom,left,center); double description value (bottom right); ${x}px single value length unit /${y}px single value length unit + single description value; ${x}px single value length unit + percentage;${y}px single value length unit + percentage	background-position:top; background-position:bottom right; background-position:30px left; background-position:100px; background-position:50px 50px;	If you have a single value, the other value is centered by default.
				Single value: background-repeat: repeat-x;	N/A

background-repeat	string	repeat	repeat-x, repeat-y, no-repeat, repeat	Double values: background-repeat: repeat no-repeat; where the x-axis description contains (no-repeat, repeat) and the y-axis description contains (no-repeat, repeat)	N/A

- Background:
  - Background-color and background-image related styles can be combined and shorthand, regardless of the order;
  - background-image-related styles can incorporate shorthand, such as background-image and background-repeat;
  - Support the use of `none` to remove the background;

Example

```
background:url('https://img.alicdn.com/tfs/TB1uCUdfND1gK0jSZFyXXciOVXa-151-164.png') repeat-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.png') repeat-y right;
```

Examples of basic usage of backgrounds

```
div
{
 background-image:url('img_tree.png');
 background-repeat:repeat;
}
```

Important

- At the same time, set the priority relationship between background image and gradient color: gradient color (background-repeat)) > background image (background-image);
- **[v-alipay-10.2.0]** gradient background supports a variety of writing, such as: `background: linear-gradient(#FF6010, 50%, #FFD2B3, #FFF2E9, #FFFFFF);` .

Shadow

Used to set the element's shadow.

Properties	Value type	Default Value	Optional value	Way of writing	Remarks
box-shadow	Length Units&Color Units	N/A	Format `${x}${y}${size}${color}` is supported. x,y, and size meet the length unit and color meet the color unit.	box-shadow: 10px 20px 10px red;	Four values required

> **Important**
>
> - Currently, the built-in components of Ant Cube Card support this style on iOS and Android platforms.
> - You can set only one shadow for each element. Multiple shadows cannot be applied to an element at the same time.

Example of basic use of shadows

```
div
{
 width:300px;
 height:100px;
 background-color:yellow;
 box-shadow: 10px 10px 5px #888888;
}
```

Transparency

Properties	Value type	Default Value	Optional value	Way of writing
opacity	float	1	Floating-point numbers of 0-1	opacity:0.5;

5.3.2.2. filter

CSS property to apply graphic effects such as blur or color shift to an element. Filter is usually used to adjust the image, background and border rendering.

Supported functions

Functions	Description	Default value	Limit
grayscale()	The grayscale of the image. A value of 100% indicates a full conversion to grayscale.	0	0%~100%
opacity()	The degree of transparency of the image.	1	0%~100%
invert()	Inverts the image. A value of 100% indicates full inversion.	0	0%~100%
sepia()	Converts the image to dark brown. A value of 100% indicates a completely dark brown color.	0	0%~100%
saturate()	The saturation of the image. A value of 0% indicates complete desaturation; a value of 100% indicates no change in the image; other values are linear multipliers of the effect; more than 100% indicates higher saturation.	1	0% ~ + ∞
contrast()	The contrast of the image. A value of 0% means that the image will be completely black; a value of 100% means that the image will not change. The value can exceed 100%, meaning that a lower contrast will be applied.	1	0% ~ + ∞
brightness()	Apply a linear multiplier to the image to make it look more or less bright. A value of 0% creates a completely black image; a value of 100% leaves the input unchanged; other values are linear multipliers of the effect; and values greater than 100% provide brighter results.	1	0% ~ + ∞

Unsupported functions

Compound functions are not supported. Blur, drop-shadow, hue-rotate, and url 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 here detailImageFilter.zip for the complete sample code.

5.3.2.3. Box Model

The box model in Ant Cube Card is based on the CSS box model, which represents all elements as rectangular boxes. Other styles determine the size, position, and properties (such as color, background, border, etc.) of these boxes.

The box model describes the control occupied by an element. Each box has four boundaries: margin edge, border edge, padding edge, and content edge.

Margins

The outer margin refers to the blank distance between elements and elements, which is controlled by the margin attribute. Both shorthand and and non-shorthand methods are supported.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks
margin	Outer Border Distance	Length Unit	0	auto; Single Value Length Unit or Percentage	margin: 10px 10px 10px 10px;	The distances between the top, right, bottom, and left borders of the four property values in sequence
					margin: 10px 10px 10px;	The distances corresponding to the top, left, right, and bottom borders of the three property values in sequence
					margin: 10px 10px;	Corresponding to the distance between the upper and lower borders and the left and right borders in sequence
					margin: 10px;	The four sides have the same outer border distance.
margin-left		Length unit	0	auto; single value length unit or percentage	margin-left: 10px;	-

margin-right		Length unit	0	auto; single value length unit or percentage	margin-right: 10px;	-
margin-top		Length unit	0	auto; single value length unit or percentage	margin-top: 10px;	-
margin-bottom		Length unit	0	auto; single value length unit or percentage	margin-bottom: 10px;	-

Padding

The padding refers to the distance between the content and the border, which is controlled by the padding property. Both shorthand and and non-shorthand methods are supported.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks
padding		Length unit	0	auto; single value length unit or percentage	padding: 10px 10px 10px 10px;	The four property values correspond to the top, right, bottom, and left margins in sequence
					padding: 10px 10px 10px;	The three property values correspond to the top, left, right, and bottom margins in sequence.

	Padding				padding: 10px 10px;	The two property values correspond to the top, bottom, left, and right margins in sequence.
					padding: 10px;	The margins of the four sides are the same
padding-left		Length unit	0	auto; single value length unit or percentage	padding-left: 10px;	-
padding-right		Length unit	0	auto; single value length unit or percentage	padding-right: 10px;	-
padding-top		Length unit	0	auto; single value length unit or percentage	padding-top: 10px;	-
padding-bottom		Length unit	0	auto; single value length unit or percentage	padding-bottom: 10px;	-

Content margin

Content margin refers to the width or height minus the distance between the border and the inner margin, that is: content margin = width/height - border border-inner margin.

Width and height

The box-sizing property of a card only supports border-box, which means that the width and height of the border area are set.

Internet Properties	Description	Data type	Default value	Valid value	Writing

width	Element width	Length unit	0	auto; single value length unit or 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; single value length unit or percentage	height: 100px;
min-height	Minimum height limit	Length unit	-	-	min-height: 50px;
max-height	Maximum height limit	Length unit	-	-	max-height: 200px;

Border

Specifies the style of the border, including width, color, style, and rounded corners. Supports shorthand for border, border-left, border-top, border-bottom, and border-right. It also supports shorthand for border-style, border-width, border-color, and border-radius.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks
border	Border	string	none	-	border: 1px solid #f32600;	Width, line style, and color. The position of the three elements is not limited.
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 property values correspond to the styles of the top, right, bottom, and left borders in sequence
					border-style: solid dotted solid	The three property values correspond to the styles of the top, left, right, and bottom borders in sequence.
					border-style: solid dashed	The two property values correspond to the styles of the top, bottom, and left and right borders in turn.

					border-style: solid	The four-sided border style is the same.
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	Corresponds to the width of the top, right, bottom, and left border in sequence
					border-width: 1px 1px 1px	Corresponds to the width of the top, left, right, and bottom borders in sequence
					border-width: 1px 1px	Corresponds to the width of the upper and lower borders and the width of the left and right borders in sequence

					border-width: 1px	Border width on four sides
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	Border color	Color unit	0x000000	-	border-color: red #333 rgb(255, 255, 0) green	Corresponds to the colors of the top, right, bottom, and left borders in sequence
					border-color: red #333 rgb(255, 255, 0)	Corresponds to the colors of the top, left, right, and bottom borders in sequence
					border-color: red #333	Corresponds to the colors of the upper and lower borders and the left and right borders in sequence
					border-color: red	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	Border corner radius	Length unit	0	-	border-radius: 10px 10px 10px 10px;	For more information, see the description of border-radius values at the bottom of the table.
					border-radius: 10px 10px 10px;	
					border-radius: 10px 10px;	
					border-radius: 10px;	
border-top-left-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 valid values of border-radius are described as follows:

- Single value: indicates the corner radius of the four corners of the border.
- Double value: indicates the corner radius of the two corners of the border. The first value indicates topLeft/bottomRight; the second value indicates topRight/bottomLeft.
- Triple: indicates the corner radius of the three corners of the border. The first value indicates topLeft; the second value indicates topRight/bottomLeft; and the third value indicates bottomRight.
- Quad: indicates the corner radius of the four corners of the border. The first value indicates topLeft; the second value indicates topRight; the third value indicates bottomRight; and the fourth value indicates bottomLeft.

Basic usage of borders

The following is an example of the use of the border.

```
div {
    border-style:solid;
    border-color:#ff0000;
    border-width:10px;
    border-radius:5px;
}
```

Note

- When using the shorthand method of the attribute, the styles not written in the shorthand will be processed according to the default value. For example, the `border: 5px red;` has no effect because border-style defaults to none, while the `border: 5px solid;` will display a 5px solid black border because border-color defaults to black.
- If the shorthand and non-shorthand of an attribute exist at the same time, follow the principle that the latter overrides the former. Example:

```
// Display a 5px solid black border (black covers red)
border:5px red solid;
border-color:black;

// Display no border (none overrides dotted)
border-style:dotted;
border:5px red
```

Basic usage of the box model

The following is an example of the use of the box model.

```
div {
    background-color: lightgrey;
    width: 300px;
    border: 25px solid green;
    padding: 25px;
    margin: 25px;
}
```

Examples

Click here detailBoxModel.zip for the complete sample code.

5.3.2.4. Layout

This article describes the layout style in Ant Cube Card.

Flexbox

The layout model for cards is based on CSS Flexbox, which is a one-dimensional layout model that enables the layout of all page elements to be consistent and predictable, while the layout adapts to various devices or screen sizes.

Flex container

In the card, Flexbox is the only layout model. You must set display:flex. If a card element can nest to hold other elements, then it becomes a Flex container.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks
display	flex layout	string	flex	flex	display:flex	The specified node needs to be displayed as a flex layout.

The difference with the Web lies in the text node calculation. That is, in the card, the size of the text node will not be greater than the size of its parent node.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks

flex-wrap	Determines whether flex member items are distributed on one line or on multiple lines.	string	nowrap	wrap and nowrap	flex-wrap:wrap;	
flex-direction	Defines the direction in which flex member items are arranged.	string	column	column, row, row-reverse, column-reverse	flex-direction:row;	
align-items	Defines how flex member items are arranged along the vertical axis to handle white space.	string	stretch	stretch, center, flex-start, flex-end, and baseline	align-items:center;	
align-content	Defines how flex member items allocate space between and around content items along the vertical axis.	string	flex-start(web: stretch)	auto, stretch, center, flex-start, and flex-end	align-content:center;	flex-wrap: invalid when it is nowrap.
justify-content	Defines how flex member items are arranged in the primary axis direction to handle white space.	string	flex-stat	flex-start, flex-end, center, space-between, and space-around	justify-content:center;	

Flex members

The differences with the Web are:

- Text node calculation: If width and height are not set, the text node in the card is used as the node for external size calculation in yoga calculation, and the calculation result will be adjusted again according to the flex constraint. However, the text node in the web calculates the size according to the content and is not restricted by the flex condition.

- Flex-basis difference: when width and height are not set, after flex-basis is set in the card, the flex-basis value will be used as the initial value to participate in the flex style calculation, and the actual size of the content will not be considered. However, in the Web, flex constraints are not considered, and the actual size of the content is used for layout.

Internet Properties	Description	Data type	Default value	Valid value	Writing	Remarks
flex	Defines how much flex member items can occupy the remaining space in the container.	Length unit or percentage	0		flex:1; flex:1 1 30px;	The abbreviation flex: <flex-grow> | <flex-shrink> | <'flex-basis> is supported.
flex-grow	Defines the scale at which flex member items stretch when there is free space left.	Length Unit	0		flex-grow: 1;	
flex-shrink	Defines the ability to shrink flex member items.	Length Unit	0(web:1)		flex-shrink: 1;	
flex-basis	Defines the default size of flex member items before the remaining space is allocated.	Length unit or percentage	auto	auto, pixel value, percentage	flex-basis: auto; flex-basis: 50px; flex-basis: 30%;	
align-self	Allows a single flex member item to override the default alignment.	string	auto	auto, center, stretch, flex-start, and flex-end	align-self:flex-start;	

Positioning

Support position. After the position property specifies the positioning type of the element, you can set the coordinates by using the top, bottom, left, and right properties.

Internet Properties	Description	Data type	Default value	Valid value	Writing
position	Positioning type	string	relative	relative, absolute, and fixed	position: fixed;
top	Offset above distance	Length unit or percentage	0		top: 10px;
bottom	Offset below distance	Length unit or percentage	0		bottom: 10px;
left	Offset from left	Length unit or percentage	0		left: 10px;
right	Offset to the right	Length unit or percentage	0		right: 10px;

Others

Internet Properties	Description	Data type	Default value	Valid value	Writing
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 Flexbox Usage

Here is an example of a basic use of Flexbox.

Important

- When you use the shorthand mode of a property, the styles that are not written in the shorthand will be processed as the default values.
- When a shorthand for an attribute exists at the same time as a non-shorthand, the principle that the latter overrides the former is followed.

```
.flex-container {
    display: flex;
    width: 400px;
    height: 250px;
    background-color: lightgrey;
    flex-direciton: 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>
```

Examples

Click here detailFlex.zip for the complete sample code.

5.3.2.5. hover

Ant Cube Card allows you to configure the hover attribute and use the hover attribute 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 attribute. The original style is restored when the gesture leaves the node.

Styles

Currently, the following two hover styles are supported:

Internet Properties	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 attribute, 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 attribute 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 attribute 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.

Examples

Click here detailHover.zip for the complete sample code.

5.3.2.6. Animations

Ant Cube Card supports animated properties, including size, rotation, translation, and color, which can gradually change from one value to another.

Transition

Attribute Value	Data type	Default value	Valid value	Writing
				transition-property: all;

transition-property	string	No default value	background-color,opacity, transform,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 usage 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

Internet Properties	Data type	Default value	Valid value	Remarks

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>}): Translate in the X-axis direction. Unit: length or percentage. translateY({<length/percentage>}):Y-axis translation. Unit: length or percentage. translate({<length/percentage>} {<length/percentage>}): translates the data in both the X-axis and Y-axis directions. The translation is abbreviated as translateX + translateY. scaleX(<number>): scales in the X-axis direction. The value is a numeric value that indicates the scaling ratio. Percentage is not supported. scaleY(<number>): scales in the Y-axis direction. The value is a numeric value that indicates the scaling ratio. Percentage is not supported. scale(<number>): scales the data in both the X-axis and Y-axis directions. It is abbreviated as scaleX + scaleY. rotate(<degree>): A transform that rotates the element around a fixed point (specified by the transform-origin attribute) without deforming it. The specified angle defines the measure of rotation. If the angle is positive, it rotates clockwise, otherwise it rotates counterclockwise. transform-origin: specifies the origin of an element to be deformed. Only center is supported. matrix:2D transformation matrix translateZ and rotateZ take effect only when the transform-style value is preserve-3d. translateZ does not support the percent expression.

transform-origin	string	center	left, right, top, bottom, center, and numeric values	
transform-style	string	flat	preserve-3d, flat	
perspective	length	none	none | <length>	must be positive, negative, or 0 is the same as none effect.
perspective-origin	string	center	left, right, top, bottom, center, and numeric values	

Transform

```
.transform {
    align-items: center;
    transform: translate(150px, 200px) rotate(20deg);
    transform-origin: 0 -250px;
    border-color:red;
    border-width:2px;
  }
```

3D Animation Example

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 2 layers (that is, the parent and child nodes have animation at the same time). If the parent, child, and grandchild nodes have 3D animation at the same time, the final effect may be limited.
- For Android client effects, due to platform limitations, View cannot be divided, and View can only display all or be covered.
  - The Android effect: A face2 completely covers face1.
  - CSS, iOS can achieve the effect: face 1 and face 2 cross each other.

Animation

Internet Properties	Data type	Default value	Valid value	Writing
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		Value ifinite (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;

Usage example of **Animation**

```
.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 animation

```
<template>
  <div class="root">
    <div class="line">
      <div class="subline"></div>
    </div>
  </div>
</template>

<script>
          const animation=requireModule ("animation"); // Obtain 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

The node defines the event @ on-animationEnd. After the animation ends, the corresponding method is called back to pass in the parameter `{"status":"finish/interrupt"}`. finish indicates that the animation is finished. cancel indicates that the animation is interrupted or canceled.

Sample code:

```
<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 execution completed");
      } else if (param.status == "interrupt") {
         console.info("animation interrupted or canceled");
      }
    },
  },
};
</script>

<style>
  ...
</style>
```

Precautions

Note the following points when using animated properties:

- The root node does not support animation.
- Animation (input, slider, and so on) is not supported for entity components and add-ins.
- Skew animation is not supported. The effect implementation can be replaced by matrix.
- The CSS animation is not valid when submitted during the KeyFrame animation.
- The iOS platform does not support gestures during the movement and can only respond after it is finished.

Examples

Click here detailTransitionAnimation.zip for the complete sample code.

5.3.2.7. Accessibility

Ant Cube Card provides accessibility mode.

Accessibility mode supports the following properties:

Internet Properties	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>
```

Examples

Click here detailBarrierFree.zip for the complete sample code.

5.4. JS Capabilities

5.4.1. 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.

Examples

Click here detailJSCapacity.zip for the complete sample code.

5.4.2. 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 background. If you enter the background multiple times, it will be called multiple times.

onForeground

This method is called when the application enters the foreground. If you enter the foreground 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();
}
```

Examples

Click here detailLifeCycle.zip for the complete sample code.

5.4.3. 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);
```

Examples

Click here detailTimer.zip for the complete sample code.

5.4.4. JS API

5.4.4.1. 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	Limit	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

Examples

```
<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 sample code.

5.4.4.2. animation

The animation module is an interface for performing animations on subcomponents, supporting a series of simple transformations on components such as position, size, rotation angle, background color, and opacity.

Call the loadKeyframes (keyframes) method to load the key frames of the animation.

Parameter	Type	Limit
keyframes	object	Keyframe parameters. name{string} is the animation name. The values{list}_NativeStyleKeyFrameProperty is data information and is a dictionary in the format of {'p':number;'v':string}, where p represents percent and v represents value.

For more information, see the following data definition. keyframes is an object that defines multiple animation effects. The outer key (for example, fluctuate) is the animation name, which indicates one of the animation effects. Fluctuate is a list of animation properties that will change. For example, transform is one of the animation properties. The internal "p" /"v" indicates the progress of the current animation and the specific usage of the property values. For more information, see Animation.

5.4.5. keyframe animation

This article describes the implementation of a JS keyframe animation needs to operate.

This article takes the online payment effect page click animation as an example to introduce the implementation process of animation.

1. The Animation module is introduced and must be introduced outside of export default (line 19).
2. Defines the specific animation for the keyframe (line 21).
3. Loads the animation. The `loadkeyframe` that calls animation can be clipped to the defined animation content (line 135).
4. Assign the loaded animation to the node (line 159). This example is to dynamically specify the keyframe animation of a node. You can also write the relevant properties of the animation directly in the class of the node.
5. For more information about how to support the animation property in keyframe animation, see Animation.

Example:

```
<template>
    <div class="card_root">
        <div class="white_bg"  @click="clickAnimation()">
            <div class = "right_content_div">
                    <text value="Click the suspended water drop to execute the animation
"></text>
                    <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/6h2dHzKEfpVbFLzfpmkqvcz0uGjal6ZwJIQ7XJI0pFH3b1AsJKEEso2tjgpnpn92VLXb9p44r
rptDfCqeBdBXDqse2fS6Snf5BYXd8RTBS0wLWCIEYjZuKVw2bwN/tlIu6+9GyTPrMbIGL88+xYqB4p3gna3LzJ9U2
Wbqbmy68s/+HkoJ9WkPrumnrbLxp+5PJ8u/+tN4c/FL6LKYKNBlFTjX6yJh4wGkw7/cY6aPTZl1V15gSqWyOfTIS6
ZYvpufxcY8BjigCMAyIh0u4tXbpoSkenzMwxc89E5PJta7Z+ZvK0NcoqPu0ATjz5wwuTsf4fJXsTl5g6dEUCJhcsc
LRU1ueMIceuhFk2ptML0fOcuECBZm5VKhbPoff9kUJmbM6k9caBLtjcbMFyrHRQWyBC/NzJmZvuyzuXz3F9s+8Y3d
PE0wigF848dsvvJ5tndsR665tNNGq9zgKo/ZbLFlzjL/aZwd0HTM8HN5gmdlcuVwgKA3nQrcdePWKGn9lvujefYZr
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
lhjaqLHbxrc9ZdnORqC5CIA773Xi8Rig7cmu+tqRBHvhAZ4k4f1VT8ONDs3nWFSPFoPwAMTDWMIbFZZjf3iil8WWb
OjMidgJZ4akqpy5XrqqYIkEAiM1rOkwak0nfTjzMk1MkCFCGPYcXtofNHcmG0rFXbv0dYBPI2sQiAD9Ud9Mn0u3ha
oCqhNuyH2tsO/3ienxtyeBbOtalW+IF+FDA3zW6ccleBVsftMFpHqMs3b4gWFSOmNrCAiI/U7RMrmCmblRetMDhPM
5D/qRCFvLqeIiunK4pfPr79bdcAWpFqADQA8LJxPQ34u21RM1n1ErRdXNYP00cHDErL4U3y8ElLKARywVutxhUx3K
NdLBJwfvSkaa5HqpAFL6Q3KVWCJmEBzmYNetBanREdf6jPzU7NVXRlyWB+mW1MRkzuybTs2Fa6mCgD7k1+/PN0SuU
DnJVGT2I098VOPmVEw5ZX5HHVVeVXqpMvKOqxVXZKi02uxzTUpVAZDl2VR6UI4ED3RR2J83ru82Rp/YBMPY+y0gMe

Vk4Zkcevlhe98WFUwrgAwGc/+UbytlqepAnLB+6YPDuMUuWSaN+BggAtlAqEXeU4UAmPBJHrx49Ioy3wlTeks0DKJ
OGpQL0q/1vxqri6Mpd5/Wa6ePTZnZoXMY+AY56eMELG1pSkdL08BdhjGpfAPDV//nV1am60DU8wwsaqWwwYgRrvg4
uS4QKNYVuWhoFSGpYxfilYp5efId1K8yzAtQ15UldSDH/SoCEBqP6fXDD73ht9OYUZD1Qtx4JCI5H/r0M4/71U9gQ
xGf/dbqjBvswi5qNiHwW258S1k4Nq3GuJotlVCgVq5olYpYHCEjGYbLCb6ZDJKplrRrxfkdlUBdpqrhKtiobSAs7f
F047HDdlM1sxhqSbLGgEPnMQDXbu5MZ4uTRz4NyorAHpwyUSqfK2s+1hCIa0MU/vY3sOmCcdCMjYoXTRUZBZkVLsb
jnEpy0llGdXpoI8rYsMMyVd9JBUj2gysagvCpBG3DYGsHqo7G33YzuG5Chq6JRwCWOE/VIcfpaXdIIgC//5IaeRG1
7BlU90QKM/mTBbHVAJg4H2oTButvBKTjgQvbehSfBU8wlylT2mi1PlRuo1Zh5OUtObdMsecgN/RGiihrThIaIMXTg
l7NN7ZDBeIhdWDRc2nR70514fcJOIjW1k1vqm5LY8zqexxuI8W4G3TeKY6wYXOXjzBsYWGHB8nSyqUw1mAHd0aU8W
ymndjTQcVWGbq0DLGqlOr0bzLo2VwkhrMyDx4nT2Gd5Pk3UUtRIwema6Lps3Mwc2kigfiROrSSH3VfpnKwTyFJ131
jyT0C1YkpqWGSgtUroYLJZZV827+qplhFhEIWuhTZmpFdQCRMgWpYKckC0hS4P07zQ1voVTWZqAAcN3J3Y4LEATlV
zNefvpDJKPUC8XiZqPsPCjtAoXlSnZsytStAIA8J2PQuyux1ugX+eXk0byWI9YkE4EO5VsiVn65aVZGFbvyUm7lVY
phcq9dPVvEG56icb0lgXpgHgDCcSRwYbWZGIwDtDXvFCs90Lh5/422/UxRLhtbI8YZ0aAGQRD6A5+6Ya2H2doyrlE
thhUFeZvWvBjBjBIoY/NKro5FreW3kU9Cxs2rqgp+R45JvZRX5ZXuxjwXwS6L3biAZ8clHrhyaGPQGN06bIprj/3O
XhlDfYEYmFsT4hEy5VDuY8tjURnPuFE3hdQnYeKJTKbEylkmfChqXyLNI7WQG0FLBw6Ut4WWaDKysi+NHioMzSmA+
TOqoAcMfkLKwITeFk/G5f3EAibRRc6FbhwOe23HjzzRFq6NRjqisQhcDMFVDqY81kMxPgdxxgG/JvsrdolVPsGVZ1
yiI0W4+WSxl5SXcvZmyQ13qredxyTZPHphkF9Wlay8hHmtIRMzi0ENocgePkZ+as51kZ8sG5ouFQTSg/2xotmVJLI
dB4U1SDpkChCOyecHKNDyilcwqNS6Nxno4tIApVEhBZEnSZL8YdrSAsWW7pf6vFpWUYSM5jUWfZZOmoKouqTc/gQy
g5rKzLjzR7X3EO3/Ko7UHFOH+NmXA5m2mKxhLROm5hxCo2ipXbvl7EeVmIWzsxiJVQkwYaYPMEVcoWZJXLr9PhrSw
n4isNTPUoxBYzdNmtDxo2naKRfLLJ2FLl3yVlbkLF/AQ6JfHsZYV5KvA5x2kw8hAoxCpflUlDtAKRY6fxYaIbyiGG
SGEfN/ApLRBbkPXViGarEkxabG+QD4BotD9ixAIhyCKhaY2VIHn+4FJ7tIyabJEoDdLkZRl/bJCkzQtd0wvlys6Yp
QOBT1SuU5vgkvh1ykBheglZd1Cn5FpPtimFCQDIOwCAgyBSj5/JK1QIIiQS0RclWZ8jBWPkm7BSyz5WqXxA4tKEdC
wEK2f5qIRl8kSCaWb9OITYwzKOGAQTqDD4smX/HZt8uJDPZnhM5RsrGsDhM8XwsGieYwDIQblfaGmWLkaQicE3QPi
b6sXyS6hE/ztk7hrUq7fEFaeRx5AhDU5dDBWmkH8hX2WV6lBcaxDkzG8/Mmgtk42IFRn/B62PEhE09Ph0te3Wg+B6
DEAhli1xfHJawDsu8nGLla0wSCsWxdRNORu7NJEFIQCJPM4l5fyxwS2TtCvrpEUJZRya8guZdNVp00uVC4+Wy34DT
bIs4I41yF0Nu0DiaBTaFQmg9FI+PRbCF2rDg/nwGxSTZ2CiSEEtj/FnN542ExzWndN4a1Sb/0wZL5x+aVbm1eMmK3
0VwXZjJS8qt8xKVxuDPMopK/I2EdxI5Cv4bN7VIWl/lyE6sIQroM0evjmJ8zMzPQMQ1eADNhj1JkOtPWPhycI5Y/C
rB4mVUkSuiBeGRJmTzeQ/bHO1ZOQxlXpy2NDPgv5fbGBbwgL5YlDYyBTgqTZvVVx2yMNIjlLp+mEVeAR7otYyKQde
D3jAi1VJbjqL6TWCtaDfhf2xkXwAEMCWyqHBqZovT4Z/6ytfyWO3sr/iIRGNRkX8BoMgznJPKJ5jDdHKLJ/fWDV0C
FITDe8iASy119yitFvsGiP5AljWJa5qsI8uxuFWUOn3ZFFzCrSycPkQWA06OTsoXFRAt91LlQZxlLm1I5+tqmTabI
jm8+Xn+Xiv8u5Qwj+VyOB7DL93qjHVMRgrjHbz4KVHKAAak1/TGiuNeaFZPVounqW6GWsaCanfzWva8gX6kBc9VgZ
Cq6lAegTeOzs3o9RLH28/sU2oU5w5TD8ecBiicA5ufCv5mbRDcVBVTMSvCDuaWhpw0f703gI0C8T8fAsuBCQtKMcU
qE0awwLlteXayKAeKxPIunw2rbwaq24ULwaNMkvUuaSswxeU+x2tiEPV/NSMqetoEgwqu2/ZTE3nTTSZfhoa/PPAw
kC1OTuePiqlTGi5bwVKKtAXZ6ONbC+yM6AQRjFtksvxpBrdJIxErTvKjFj+StXAVNZW2ZyJOGi0HlXBny6FXV1USu
bwqqzQy1CpNKlAnsZlcJAaw/CVqMcrfXCioF7IeMWiyea9kbrVF+NVBgvgNV+7Z3hutvhs8IEKlYrbluVktqG7zYw
OT+xSqDpBoWdAfQKMdjf5YF5WqgQ0PS70ZWhnm5yFNdBprYs1wMAQHBymoa2YDu0tQu0Eiu6M4kxCNm/NCQaeLrwP
AENGuy+P9wql6NM9H/y945D2PZB9OTsX+X/eDNZ8buNZAe5A21krzeThEVPmOMmgRmgcGAgByjDPAVvLNa7gIy/5n
T1XxKXzK29UGNBPK4l6tL6S4tSAMFTB55bBxmMWQ1r8bzbw5bQZ1wJrQpy3VxrO5+Ysb6ZAxkYjJXd//YCI5fqj0I
M1HY1yrDV+cMh/4EyUtOKgAmu40qmeaTG6CkxtiCsbyFnd1WWL8vQ8W4fGyiO6HM9kObJij7VdPUt0oEiU4bnvsQN
R1eFE2ka2UrRznf+6ACgI5Pzk9H0qseFBH8BAA+fOzu/VOZwhMB6hUeZEznBevM0EsHsQaiJQhiFH447gSGkw6a5j
r9KFRqsoa3lJk8vqUp6ArryOHpAW63BsER2q18rb+hSQhRuAci6e0dvGXz9iVnxgrRzr++2wOtHuIk6o8+XYzrOe+
/apcQALh9e6g8k43+XX4cL9e4Xgh5gtq4ptNEcLQ1+hqel/KISw0MQGBFTgMUIGmoS4c+6lS5YEJiI1mGH+rRtOoJ
OyKh/wUw5XcEOtLupxdCx4HnkXrhD2/UN7++Q75Lq2Ztl9+baAB/bwYGFiEn8EI9X4UwOsoFVCACBzh0Mr/mV4ePa
UwTNERBQfnKS/AG0/NYc/NlREpKI2xc0SBW6l7Ku2Cw31jyoMzllUZZGooXgHZkg7ogLKBRj5ZTxtEZ8HLSUD7y+L
lvEvg8zNzpnx1/D57cVnVox9OnmUsduYzJqX6y/9QtB9oWWhCzNz7fU7pjOzkbuLU1nfC/XDPIKI9+fSq9pNuqvF9
aX//gyVRFwwmEGGgbRZnAYDWcsZYjzfI3u1Sv6loEtqPb5VG91o4K8EQHqwZ4NBtjX99Te00b3s5KNWHZhpmWdhFc
fzwDS++SuH6+5aufJSLJgXQoUHkjweWvfTwaOzB/khnt9YEOXuQhkU937kPJMZGjPH38CyBmslAUAMssBQSdAQ0OR
0OqG+/ml01bve7NJK/WIUBZG8Ea3FSh+16nXrQw5nH9R9uwXcVpyxCGpVJ2znReuAF/PkVXIihnnRjOSvj7C2OZ8t
i67+H2yeGxYBuPX3t08en4nflh+HF/IlSo6HYjyU4U5EcT629oqN5tCTe/B3DjBeBmtD2yjllQaiKgVG8pZnKVC1X
Rp8ZK11jBYx7/A/0QEVCF7ssvTBaW15aJNyLNp42ZiYwZeeF1vGB5gYlw3cf2osyz4HEHxrGvXNN+61J/em8RgEQ3
fj3x8dzv2KH6GIQrqyVcw9c0Nvh+k8f6157Re7gTHKuOGuBkqB1AaDTRqofBqLJ6DQdjXxWAcE0St5BwSV1Tjghx5
wK/lkvs16HeGMZEmMOpysHHnjcrN59taloa8baBfx4gbRbHwcwLjzw+F3lw+LN3/iOxqQ5LAnjudefOZ0otN4wPzm
uAM+iNYgGoZJpGvTWaYee8XXHnwGYwMA5BV4FhtLfhjtAqlp20i/gTAJbAuTiaYRQ0UlCNRZRVOgrE4CpCBVgu93Z
F8OZh7/6Hn8GLRF2mdcMq4+E7Y9rqy9LuMl4umjdDY/mMqVv79a2hkD0MQP1OWBJAlm/8/F88M5yJ3lbAhBJC13Xv
CDu7Vm60YTQ5fe9+DTPojYK3PglfGFd10aBWMCDyAAuBRIoVvayaRdsMhvxzF62yLgpJy6YY9cfjqMZxzzGHZe/8X

```
k8nenz9limhIiG8zwQiCKI1yGy0eClmhAViN6akWDJ/mN3ahW2uesj4g5FsATW+S1W11CbBaP+rgmJcrl8xrDzxlG
IsvLS843HJRnaIXXRDgII8ArouiPT0f8z2nnPXcRiuYB+d+Kw+97bV3VF9+3sXJla6yXwfAB3z/+TTMCe3Rb7Rw93
/R58zU0TGz4arNpqa+Rv6gjn+KDR7WwpOcIPLTPkF+l/4BHv46I0gIvsJMYBgYyaV5xASX5QIey9Ft8dFQZnLGHHr
SxXekwXPI9DkT884aaI5xE8nDYDvIGx4t5Q66aPrb3yqyOsZrngtGQ5FmNeuvfGD3Wlxu5vWVHT6PE9GYAmz0gIII
QtSAPvTca2YQC+1VHzzHtK3FZhwHjzKmsBYLoADCH9I0uGniwLzFQ1kW8gTGUgUkpitpAXjoKSGuErDLGDkwYI69d
0bznbtKzvNR4PBWTYUPCgA55YRm8bOV48lk2tu3rDp7a/ENS/TMI1fRkWn7z/f3/t2rbU5P9qaE0lPXyIJyDSOP2b
zohPEwc+SYeGOqBX9ADEDG8UzB01McBbEaQFaxlCUKFMuDtAOWAEdvIwOCAIoMgUEIA7gc1nADT++FV+XMqo9uxIe
G0FPH+Ika4rQxLGeUwaYzgsyIS7Prvh2h0PiZI3+VnK7GVF9v/TDf+uIzn5t/XN8UQZLxXK3WWX1k9h6ZHwxCIW4U
vcdM9o9guXOGadvQgxdJ0D7+LRcGAhnUXJVeQMrn5a8CpAnJaze15QoeysKpmAEc5tj+w2Z8X79pPqMH37ecJe8Cy
Er7/LALOCh2fA41PF2cly2+fP/Mx/+78LlZ84FTTjxGwLpYfuu/F3W2ITP65vitd63M5x0iCI2p2JDLs0tkfTR0fN
dewQPqgunACUfzqhUYNjFTyyNDIIp0KADzRKagof5/a4jNBDOt31VDuHklTCDH+wbxGW4/nnvX4Js+/NUOeh0nC45
h6B42THyRA3/Ph0aWoy1Pl76//Vbf+80No3T53I6mWl+x78ziebzNDfpRsj7WV8nIwHzP6EAhADIAkMt3qgTfYPYb
35QwvjSt6TJNqztNshZfeXLpgL8NKN6gtYlF+KFHUQdjDUwyb8tYF+vmk53cTNZMALjMwDETwyTWfv56fFyN1x65j
eFtihIn9MJvACXsmEcXAyPm2O5OJdn1/5ie8+plWdbCzmniyzy9f30C0bG73BnzbUmws8Hm8JiGiJeCLazpjeyJhj
pmhsfRpfrw1v9xw9dG6jpbsRhvMUkcXkb49wUZ2Dh7cTIQA0WV1QVQATuGiSK2kvhzyCPjZnYYb9QDiFroa97Qa1L
XA2lSFDAefN4vJGPA91gJ/LsPFs5DeF1Or/0LX1m/jG660Hse+ti/kS+x//SVtT/oU/b0zkPhdJRU0Z3TmYnauBpD
aFwlHEzOjIyZ6YERMzc2KSe9UYAcw18EidfWCLhh5EUXqpIH2fAibqvm4WnzWbx6DADxXS4+uW0y6Z4OU9vejB0Gb
HB4AnNwE40VsD4KznhbGFm8uVzGwp9TeRlo9+vXnTdXjw/fbCOwJQqsSL1qOXbbu+NjR1a6ou3O7RQzAGSuMJGjzP
Q43lGCNFz0WF5YvBYwQ/IxIq95vBFQRF7ey+PCFkL0ZoIVxWencTwpSzTUyZF7NIl1KZUSMHiTHACIx6Ea63mypKH
VcI3TYEL5yZ845kTcMfd1z5PZyu+M82aNnbCdKktyNYLTPw1I71Tbmjt8TDud+N1UTRyQCkrBdRBRspF6RYI0FRBa
UC6DwC9pLauOCYr1IhnDmK4ADALMs5OzTMsxUYSBUz5bKuW95D+UEt1/0vyRbYer1b+dfNCOtyO8WMYLTT1x8yfj5
vJqPFD5sEdikEEgD5Sx1IBEDaqiXCj5u1igmFBDcRpFESgKdpdlcw8AJ4IXgeqyvlSzh5iezMeXV/2vqxHY/4Sk/N
X71ChTLbs9L3bmk9/67YQ3/aVwuPCRCL5VBpKYCP3uGIBI1GiBWOGY4iQpVxHEw0AhXdJI8L8FjTS++0lPLs4Vy8V
NFCqPbOBy773n3XhU7NX6107XFNdemnJM0/GbU5+e0r4uXZf4/vKq6JJE2rHMDCO6T9dA8EWQtKIvhhYnGw4AlYLG
//k2qWjiUIDuWsqXh0vl8P3zJvmz+p3ff8J9CLRY6TujnFYAXdOyL97RE50bvyrs5T+Fvwx9CTYv3aEkPJNBgGD3Q
C6tNRRAsJkkAuVOuwKLBl5Tl5hnEYvrer5IXvK8STj9Rt+d6wz316f981AN1mZF69qyU1c+Q8rF+24JOpizxTOhNe
XwGjGPxGUno2OlCmI8466hXPbyKMLbTqGjAPw1eOOzWEQ+HU00vRzauB1fB7674T0BsLqJO3fujF7R8IsmbEfbwuH
rzA3QSPS4e9coIbiHAkPFcsedORcHkiGomPThsz9sDBD05cd911WL+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 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 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;
        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>
```

You can click here detailKeyFrame.zip to obtain the complete sample code.

6.Questions

How do I resolve the Permission Denied error when I install AntCubeTool?

If the Permission Denied error is reported when you perform a global installation in macOS, you can grant the administrator permissions to resolve the error. Run the following command to grant administrator permissions.

```
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}
```

How do I resolve Operation not permitted errors encountered when I use AntCubeTool?

If you encounter **shell-init: error retrieving current directory: getcwd: cannot access parent directories: Operation not permitted** errors when using in macOS, you can solve them by adding file access permissions to the terminal. To add file access permissions for terminal tools, select **Terminal** in the **System Preferences Security and Privacy> Privacy Options** section.