Ant Technology

Code Scanner

User Guide

Document Version: 20240808

Legal disclaimer

Ant Group all rights reserved©2022.

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

Trademark statement

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.Code Scanner 05
1.1. Overview 05
1.2. Integrate Android SDK 09
1.2.1. Quick start 10
1.2.2. Advanced guide 12
1.2.3. Tutorial 22
1.2.3.1. Overview 22
1.2.3.2. Create an Application in Android Studio 23
1.2.3.3. Create an application in the mPaaS Console 25
1.2.3.4. Integrate Scan to project through Native AAR 25
1.2.3.5. Use the scan feature in the standard UI 26
1.2.3.6. Use the scan feature in custom UI 31
1.3. Integrate iOS SDK 57
1.3.1. Quick Start 57
1.3.2. Advanced guide 59
1.3.3. Multi-code recognition 63
1.4. FAQ 68

1.Code Scanner

1.1. Overview

The Scan component is provided by the mPaaS and is originated from the scan feature in Alipay. This component inherits the precise and fast scan capability of Alipay. It can quickly identify barcodes and accurately obtain the information in the barcodes.

Features

The Scan component can scan two-dimensional barcodes (QR codes) and one-dimensional barcodes (barcodes).

Two-dimensional barcodes (QR codes)

- Gen0 (Common QR code):

- Gen1 (Visualead custom code):

One-dimensional barcodes (barcodes)

- EAN8:

<1234 5670>

- EAN13:

1 234567 890128>

- EAN14:

(01)00000000000017

- EAN18:

(00)000000000000000017

- EAN128:

(422)616(8008)160520185045

- ISBN:

ISBN 978-2-12-345680-3
9 782123 456803 >

- ISSN:

ISSN 2434-561X
13 >
9 772434 561006

- Code39:

12345RTEF

- Code128:

ABCDE12345abcde

- UPC-A:

9 87654 32109 8

- UPC-E:

0 071245 3

- ITF-14:

12345678901231

Benefits

Compared with similar industry-leading products, the mPaaS has advantages in the code recognition speed and recognition rate under the same conditions.

Fast recognition

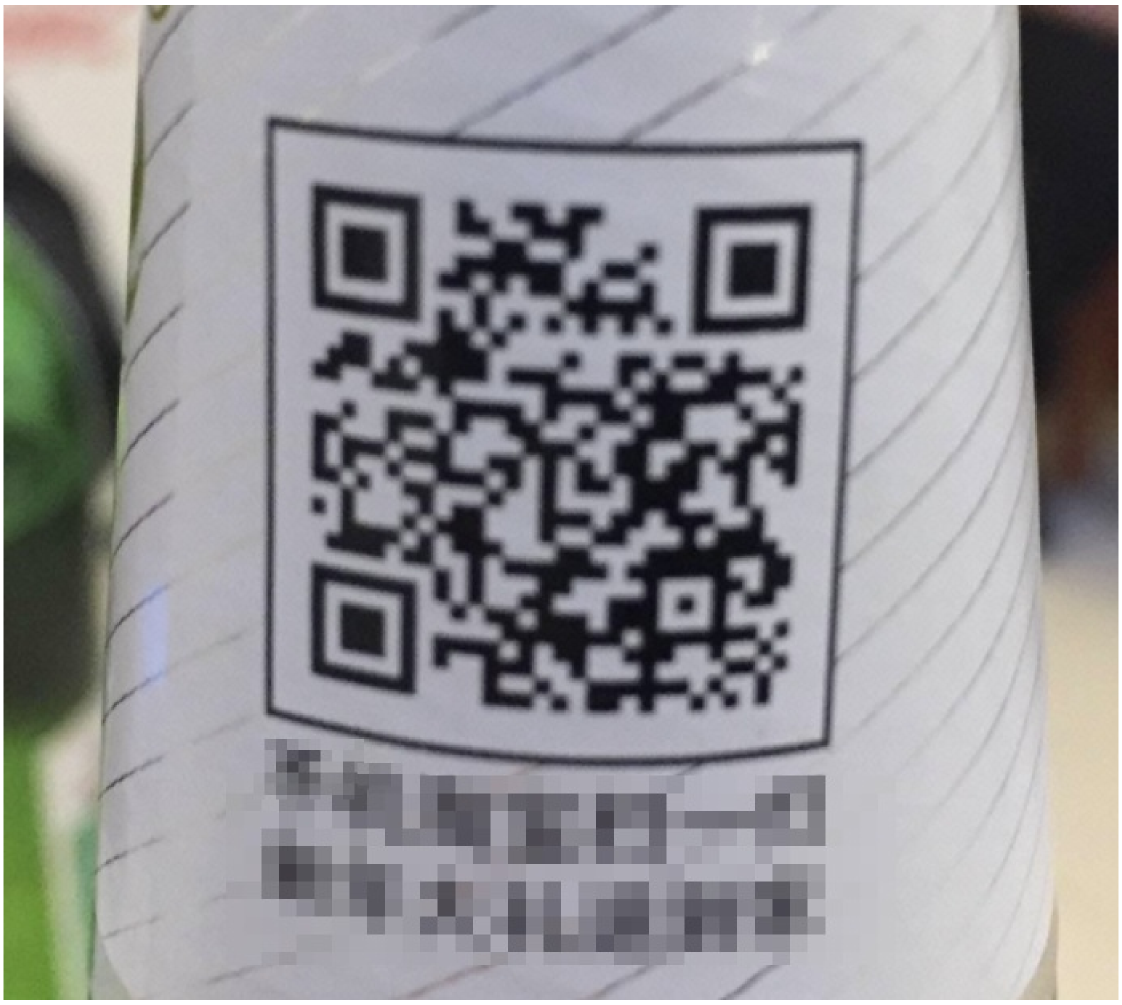
Compared with similar products, the mPaaS Scan recognizes QR codes and barcodes faster in the case of the same distance and the same light source.

High recognition rate

Based on the unique blur processing and data evaluation and correction features, mPaaS can recognize photos that cannot be recognized by the scan function in the album of similar products.

- This code can be recognized by the cameras of similar products, but cannot be recognized by the scan function in their albums.

- Codes that cannot be recognized by similar products.



1.2. Integrate Android SDK

1.2.1. Quick start

This article introduces the operation steps of accessing the Scan SDK in Android.

Note

Since June 28, 2020, Alibaba Cloud stopped maintaining the baseline 10.1.32 for mPaaS. Upgrade your baseline to version 10.1.60, 10.1.68 or 10.2.3. The Scan component supports two access modes, they are **native AAR mode** and **component-based mode (Portal & Bundle)** . This topic describes how to use the scan function of the baseline version of 10.1.68, 10.1.60 and 10.2.3. Scan supports multi-code recognition in full-screen mode starting from the mPaaS baseline version of 10.1.68.33. Scan has added the function of AI recognizing small codes and automatically zooming in starting from the mPaaS baseline version of 10.2.3.

Prerequisites

- If you want to connect the component to the mPaaS based on the native AAR mode, you need to first complete the prerequisites and the subsequent steps. For more information, see Add mPaaS to your project.
- If you want to connect the component to the mPaaS based on components, you need to first complete the Component-based access procedure.

Add the SDK

10.2.3

If you want to use AI to recognize small codes and automatically zoom in, please install the **Scan AI** component.

Native AAR mode

In your project, install the **Scan/Scan AI** component on the **Component Management (AAR)** page. For more information, see AAR component management .

Component-based mode

In your Portal and Bundle projects, install the **Scan/Scan AI** component on the **Component Management** page. For more information, see Manage component dependencies.

10.1.68/10.1.60

Native AAR mode

In your project, install the **Scan** component on the **Component Management (AAR)** page. For more information, see AAR component management .

Component-based mode

In your Portal and Bundle projects, install the **Scan** component on the **Component Management** page. For more information, see Manage component dependencies.

Use the scan function

10.2.3/10.1.68

Use the full-screen scan code function

```
ScanRequest scanRequest = new ScanRequest();
MPScan.startMPaasScanFullScreenActivity(this, scanRequest, new MPScanCallbackAdapter()
{
    @Override
    public boolean onScanFinish(final Context context, MPScanResult mpScanResult, final
MPScanStarter mpScanStarter) {
        Toast.makeText(getApplicationContext(),
                mpScanResult != null ? mpScanResult.getText() : "No code recognized", To
ast.LENGTH_SHORT).show();
        ((Activity) context).finish();
        // Returning true means that the callback has been consumed and no need to call
back again.
        return true;
    }
});
```

Use the window scan code function

Call the scan function of the baseline 10.1.68 (old standard UI). If it failed, you will directly return to the scan page. If it succeeded, you will obtain URL information of the QR code.

```
ScanRequest scanRequest = new ScanRequest();
scanRequest.setScanType(ScanRequest.ScanType.QRCODE);
MPScan.startMPaasScanActivity(this, scanRequest, new ScanCallback() {
    @Override
    public void onScanResult(final boolean isProcessed, final Intent result) {
        if (!isProcessed) {
            // In the scan page, click the physical back button or the back button in t
he upper left corner.
            return;
        }
        // Note: this callback is executed in the child thread.
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if (result == null || result.getData() == null) {
                    // Scan failed.
                    return;
                }
                // Scanned.
                String url = result.getData().toString();
            }
        });
    }
});
```

10.1.60

Call the scan function of the baseline 10.1.60. If it failed, you will directly return to the scan page. If it succeeded, you will obtain the URL information of the QR code.

```
    ScanService service = LauncherApplicationAgent
        .getInstance().getMicroApplicationContext()
        .findServiceByInterface(ScanService.class.getName());

    ScanRequest scanRequest = new ScanRequest();
    scanRequest.setScanType(ScanRequest.ScanType.QRCODE);

    service.scan(this, scanRequest, new ScanCallback() {
        @Override
        public void onScanResult(boolean isProcessed, final Intent result) {
            if (!isProcessed) {
                // In the scan page, click the physical back button or the back button in
the upper left corner.
                return;
            }
            // Note: this callback is executed in the child thread.
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if (result == null || result.getData() == null) {
                        // Scan failed.
                        return;
                    }
                    // Scanned.
                    String url = result.getData().toString();
                }
            });
        }
    });
```

1.2.2. Advanced guide

Window scan code means using the scan code function under the old standard UI. If you need to use the full-screen code scanning function that supports multi-code recognition, please upgrade the mPaaS baseline version to 10.1.68.33 or above.

Use the Scan function in the standard UI

Scan code in full screen

If you need to scan the code continuously, that is, continue to recognize without exiting after the code is scanned successfully, you can implement it according to the following code.

```
ScanRequest scanRequest = new ScanRequest();
        MPScan.startMPaasScanFullScreenActivity(this, scanRequest, new
MPScanCallbackAdapter() {
            @Override
            public boolean onScanFinish(Context context, MPScanResult mpScanResult,
final MPScanStarter mpScanStarter) {
                new android.app.AlertDialog.Builder(context)
                        .setMessage(mpScanResult != null ? mpScanResult.getText() : "No
code recognized")
                        .setPositiveButton(R.string.confirm, new
DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                                mpScanStarter.restart();
                            }
                        })
                        .create()
                        .show();
                // Returning false means that the callback is not consumed, and the
callback will continue for the next recognition.
                return false;
            }
        });
```

Override other methods of `MPScanCallbackAdapter` to monitor other events:

```
MPScan.startMPaasScanFullScreenActivity(this, scanRequest, new MPScanCallbackAdapter()
{
    @Override
    public boolean onScanFinish(final Context context, MPScanResult mpScanResult, final
MPScanStarter mpScanStarter) {
        return true;
    }

    @Override
    public boolean onScanError(Context context, MPScanError error) {
        // recognition error
        return super.onScanError(context, error);
    }

    @Override
    public boolean onScanCancel(Context context) {
        // recognition cancelled
        return super.onScanCancel(context);
    }
});
```

Before starting the full-screen code scanning function, you can set the startup parameters according to the following code.

```
ScanRequest scanRequest = new ScanRequest();

// Set the prompt text
scanRequest.setViewText("prompt text");

// Set the prompt text for turning on the torch
scanRequest.setOpenTorchText("turning on the torch");

// Set the prompt text for turning off the torch
scanRequest.setCloseTorchText("turning off the torch");

// Set the code recognition type
// This setting is only valid when a code is scanned directly. It is invalid when an al
bum picture needs to be recognized.
scanRequest.setRecognizeType(
    ScanRequest.RecognizeType.QR_CODE,    // QR code
    ScanRequest.RecognizeType.BAR_CODE,   // Barcode
    ScanRequest.RecognizeType.DM_CODE,    // DM code
    ScanRequest.RecognizeType.PDF417_Code // PDF417 code
); // If not set, the first three types are scanned by default.

// Set the Hide Album button
scanRequest.setNotSupportAlbum(true);

// Set a multi-code tagged image
scanRequest.setMultiMaMarker(R.drawable.green_arrow);

// Set a multi-code tagged text
```

Window scan

When using the window scan function, you can set the start-up parameters according to the following code snippet.

```
ScanRequest scanRequest = new ScanRequest();

// Set the UI style of the scan page.
scanRequest.setScanType(ScanRequest.ScanType.QRCODE); // The style of the QR code.
scanRequest.setScanType(ScanRequest.ScanType.BARCODE); // The default style of the barc
ode.

// Set the scan page title
scanRequest.setTitleText("Standard Scan")

// Set the prompt text under the scan window.
scanRequest.setViewText("Prompt Text");

// Set the prompt text for turning on the torch, for baseline 10.1.60 and later version
s only.
scanRequest.setOpenTorchText("Turn on the torch");

// Set the prompt text for turning off the torch, for baseline 10.1.60 and later
versions only.
scanRequest.setCloseTorchText("Turn off the torch");

// Set the code recognition type, for the baseline 10.1.60.6 and later veresions and 10
.1.68.2 and later veresions.
// This setting is only valid when a code is scanned directly. It is invalid when an al
bum picture needs to be recognized.
scanRequest.setRecognizeType(
    ScanRequest.RecognizeType.QR_CODE, // QR code
    ScanRequest.RecognizeType.BAR_CODE, // Barcode
    ScanRequest.RecognizeType.DM_CODE, // DM code
    ScanRequest.RecognizeType.PDF417_Code // PDF417 code
); // If not set, the first three types are scanned by default.

// Set the transparent status bar (valid on Android 4.4 and later versions), for the ba
seline 10.1.68.15 and later versions only.
scanRequest.setTranslucentStatusBar(true);

// Set the Hide Album button, for the baseline 10.1.68.22 and later versions only.
scanRequest.setNotSupportAlbum(true);
```

Use the Scan component in the custom UI

See Sample code.

Upgrade adaptation in the custom UI

- Since baseline 10.2.3.35, the Scan SDK adds class `MPCustomScanView` and related APIs to replace the original APIs such as `MPScanner` ,which was previously used for customizing the scan feature. Compared with `MPScanner` , the solution of using `MPCustomScanView` encapsulates the core processes of the scan service such as camera management, code recognition, multi-code recognition, screen zoom in and out, and code result analysis. You don't need to pay attention to related operations when developing, just focus on implementing your customized UI in `MPCustomScanView` . You can still continue to use `MPScanner` , but the solution will no longer be maintained and you will not be able to obtain feature updates consistent with the full-screen UI (such as multi-code recognition) in subsequent upgrades. It is recommended that you switch to the `MPCustomScanView` solution to implement custom UI when the time is right. This solution will remain consistent with the full-screen UI in subsequent feature upgrades.
- Since baseline 10.1.68.5 and 10.1.60.11, the Scan SDK adds class `MPScanner` and related APIs to replace the original APIs such as `BQCScanCallback` and `MaScanCallback` . These two APIs were previously used for customizing the scan feature. Compared with the original APIs, `MPScanner` provides complete encapsulation, easy-to-use APIs, and support for more new features such as callbacks for insufficient environmental brightness. If you are still using original APIs such as `BQCScanCallback` and `MaScanCallback` , you may need to adapt the following changes when you upgrade from an earlier version:
  - Version 10.1.68.22: Some APIs are added for the `MaScanCallback` class, `BQCScanCallback` class, and `IOnMaSDKDecodeInfo` class. You only need to handle these APIs with empty implementation, among which “false” is returned for the `MaScanCallback.onMaCodeInterceptor` method.
  - Version 10.1.60.6: Some APIs are added for the `BQCScanCallback` class. You only need to handle these APIs with empty implementation.
  - Version 10.1.60: Some APIs are added for the `BQCScanCallback` class. You only need to handle these APIs with empty implementation.
  - Version 10.1.20: For `MaScanCallback` class, the interface `void onResultMa(MaScanResult maScanResult)` changed to `void onResultMa(MultiMaScanResult multiMaScanResult)` . You can obtain `MaScanResult` in the following code snippet:

```
MaScanResult maScanResult = multiMaScanResult.maScanResults[0];
```

API description for the custom UI

MPCustomScanView

To use `MPCustomScanView` , you need to let `Activity` inherit `MPaasToolsCaptureActivity` , implement the `getCustomScanView` method and return the customized `MPCustomScanView` .

```
public class MyScanActivity extends MPaasToolsCaptureActivity {

    private MyScanView myScanView;

    @Override
    protected MPCustomScanView getCustomScanView() {
        myScanView = new MyScanView(this);
        // For details, please refer to the github code example.
        return myScanView;
    }

}
```

In `MPCustomScanView` you can implement or call the following methods:

```
/**
 * Scan start callback
 */
public void onStartScan();

/**
 * Callback for camera first frame display
 *
 * There is no guarantee which one will be executed first between this method and the s
can start callback.
 */
public void onPreviewShow();

/**
 * Scan end callback
 */
public void onStopScan();

/**
 * Grayscale value callback of camera frame
 * Each frame during the scanning process will be called back once.
 *
 * @param gray The average gray value can be used to measure the brightness of the envi
ronment
 */
public void onGetAvgGray(int gray);

/**
 * Callback for successful scanning (code recognized)
 *
 * @param context Current context
 * @param list Recognized code result
 */
public abstract void onScanFinished(Context context, List<MPScanResult> list);

/**
 * Scan failed callback
 *
 * @param context Current context
 * @param list Reason for failure
 */
public abstract void onScanFailed(Context context, MPScanError error);

/**
 * Callback for failure to open camera
 */
public void onCameraOpenFailed();

/**
 * Turn flash on or off
 *
 * @return The status of the flash after calling this method
 */
public boolean switchTorch();

/**
 * Identification code from file
 *
 * @param path File path
 * @return Recognized code result
 */
public List<MPScanResult> scanFromPath(String path);
```

MPScanResult

```
/**
 * Recognize result strings
 */
private String text;

/**
 * Recognized code type
 */
private MPRecognizeType mpRecognizeType;

/**
 * The coordinates of the center point of the recognized code
 */
private Point centerPoint;
```

MPScanner (abandoned)

The settings related to the custom UI are as follows:

```
/**
 * Set View to display the camera content.
 * It is recommended to call in the onConfiguration method of {@link MPScanListener}.
 *
 * @param textureView Customize TextureView in the scan page.
 */
public void setDisplayView(TextureView textureView);

/**
 * Set the area to be scanned.
 *
 * @param rect The recognized area.
 */
public void setScanRegion(Rect rect);

/**
 * Set the scan listener.
 */
public void setMPScanListener(MPScanListener mpScanListener);

/**
 * Set the listener for identifying the gray value of the image.
 */
public void setMPImageGrayListener(MPImageGrayListener mpImageGrayListener);

/**
 * Obtain the Camera object.
 *
 * @return Camera object.
 */
public Camera getCamera();

/**
 * Set the code type to be scanned.
 * Only valid for direct scan. Invalid for scanning a code from the bitmap.
 *
 *
 * @param recognizeTypes BAR_CODE Barcode;
 *                       QR_CODE QR code;
 *                       DM_CODE DM code;
 *                       PDF417_CODE PDF417 code;
 *                       If not set, the first three types are scanned by default.
 */
public void setRecognizeType(MPRecognizeType... recognizeTypes);
```

The scan content related to the custom UI is as follows:

```
/**
 * Open the camera and start scanning.
 *
 * Call the API when entering the page for the first time or the camera exits.
 */
public void openCameraAndStartScan();

/**
 * Open the camera and stop scanning.
 */
public void closeCameraAndStopScan();

/**
 * Start scanning.
 *
 * The camera state will not be changed. The invocation of this method takes effect onl
y when the camera is turned on.
 */
public void startScan();

/**
 * Stop scanning.
 *
 * The camera state will not be changed.
 */
public void stopScan();

/**
 * Scan the code from the bitmap.
 *
 * @param bitmap The bitmap to be scanned.
 * @return: The scan result.
 */
public MPScanResult scanFromBitmap(Bitmap bitmap);
```

Others:

```
/**
 * Turn on or off the torch.
 *
 * @return Whether the torch is turned on after the method is called.
 */
public boolean switchTorch();

/**
 * Turn on the torch.
 */
public void openTorch();

/**
 * Turn off the torch.
 */
public void closeTorch();

/**
 * Play the default "beep" sound.
 */
public void beep();

/**
 * Release the resource.
 *
 * Call in onDestroy.
 */
public void release();
```

MPScanListener (abandoned)

```
/**
 * Scanning parameter configuration completed.
 */
void onConfiguration();

/**
 * Scanning starts.
 */
void onStart();

/**
 * Scanned.
 *
 * @param result The scan result.
 */
void onSuccess(MPScanResult result);

/**
 * Scan error.
 *
 * @param error The error.
 */
void onError(MPScanError error);
```

MPImageGrayListener (abandoned)

```
/**
 * Obtain the average gray value of the recognized image.
 *
 * The normal range is from 50 to 140,
 * When the gray value is lower or higher than the normal range, it usually means that
the ambient brightness is too low or too high. The user can be prompted to turn on or o
ff the torch.
 * Note: This method will be called continuously during the scan process.
 *
 * @param gray The average gray value of the image.
 */
void onGetImageGray(int gray);
```

1.2.3. Tutorial

1.2.3.1. Overview

The Scan component is supported in the native AAR mode, the mPaaS Inside mode, and the component-based mode. If you want to connect to and use mPaaS as other SDKs, we recommend that you use the native AAR mode.

The native AAR mode uses the native Android AAR-based packaging solution, and therefore is closer to the technology stack of Android developers. This access method frees you from the need to understand mPaaS-related packaging knowledge. You can integrate mPaaS into your projects by using the mPaaS Android Studio plug-in. This method reduces your access costs and makes it easier for you to get started with mPaaS.

To help you get familiar with the native AAR mode, this tutorial describes how to add the Scan component in the native AAR mode and how to use the scan feature.

This tutorial contains the following five parts:

1. Create an application in Android Studio
2. Create an application in the mPaaS console
3. Access a project in the native AAR mode
4. Use the scan feature in a standard UI
5. Use the scan feature in a custom UI

What you will learn

- How to create an Android app that displays Toast when a button is tapped
- How to access a project in the native AAR mode.
- How to use the scan feature in a standard UI.
- How to use the scan feature in a custom UI.

Prerequisites

1. The development environment is configured. In this tutorial, the development environment in Windows is taken as an example.
2. A web browser is installed. The Chrome browser is recommended.
3. An Android phone and the supporting data cable are prepared. The operating system version is Android 4.3 or later. You can also use a simulator for debugging. This tutorial uses a simulator as an example.

1.2.3.2. Create an Application in Android Studio

In this section, you will create an app that causes a Toast to appear when a user clicks a button, and obtain the installation package in APK format.

This process mainly includes the following four steps:

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

If you already have a native Android project and the project has already been signed, skip this tutorial and directly Create an application in the mPaaS console.

Create a project

1. Start Android Studio and choose **File** > **New** > **New Project**.
2. In the Create New Project dialog box that appears, select **Empty Activity** and click **Next**.
3. Enter a name in **Name**, enter a package name in **Package name**, and specify a project location in **Save location**. Note that the default package name can be used. For example, enter **Scan Application** in the **Name** entry box. From the **Minimum SDK** drop-down list, select **API 18: Android 4.3 (Jelly Bean)** .

Note

API 18: Android 4.3 (Jelly Bean)" is the earliest version that the mPaaS supports. You can select a version according to the actual production requirements.

4. Click **Finish**. The **Create a project** procedure is complete.

Write the code

1. Open the `activity_main.xml` file and refer to the following code to add a button.

```
<Button
    android:id="@+id/button"
    android:layout_width="101dp"
    android:layout_height="50dp"
    android:layout_marginStart="142dp"
    android:layout_marginTop="153dp"
    android:layout_marginBottom="151dp"
    android:text="Button"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2. Open the `MainActivity` class and add a click event to the button.

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

3. Compile the code. After successful compilation, the **Write the code** procedure is completed.

Create a signature file and sign the project

1. In Android Studio, choose **Build** > **Generate Signed Bundle / APK**.
2. In the dialog box that appears, select **APK** and click **Next**.
3. Click **Create new**.
4. Enter all the required information and click **OK**. The signature creation is complete. You can obtain the generated signature file in the path specified in **Key store path**.
5. After the fields are automatically filled, click **Next** to start signing the project.
6. Select a build variant in **Build Variants** as needed. Remember the selected build variant. You need to select the same variant when you use the encrypted file.

   Then select the encrypted version **V1 (Jar Signature)**. Note that V1 (Jar Signature) is required while V2 (Full APK Signature) is optional.
7. Click **Finish**. After the packaging is complete, you can locate the signed APK installation package in the `debug` folder under the project directory (`~\MyHApplication\app\debug`). In this tutorial, the name of the installation package is `app-debug.apk` .

Install the application on a mobile phone

1. Connect a cell phone to the computer and enable the USB Debugging mode of the cell phone.
2. Run the project.
3. Click **BUTTON**, pop up Toast. This notification indicates that you have successfully installed the application with the expected features. Now you complete the **installation of the application on a mobile phone**.

1.2.3.3. Create an application in the mPaaS Console

1. Open the network browser and log on to the mPaaS console.
2. Create an mPaaS application.
3. Enter an application name and click **OK**.
4. Click the created application to open the application page.

   Choose **Overview** > **Configure now** > **Download the configuration file** > **Android** > **Download now**, in the pop-up dialog box, enter the package name in **Package Name**, for example, **com.mpaas.scan**. Then upload the APK installation package that has been compiled and signed. Click **Download Configuration** to download the configuration file.
5. The downloaded file is a package in compressed format. Decompress the package. Then you will obtain a configuration file and an encrypted image.

1.2.3.4. Integrate Scan to project through Native AAR

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

Procedure

1. In Android Studio, choose **mPaaS** > **Native AAR mode**.
2. In the dialog box that appears on the right side of the window, click **Start Import** under **Import App Configuration**.
3. In the **Import mPaaS Configuration File** dialog box that appears, select **I have downloaded the configuration file in the console and get ready to import the configuration file to the project**.
4. Select the Configuration file that is downloaded after you create the mPaaS application in the console. Click **Finish**.
5. A success message appears after the configuration file is imported.
6. Click **Start Configuration** under **Access/Upgrade the baseline** on the right side of the window.
7. In the **Select an mPaaS baseline version** dialog box that appears, select 10.1.68 from the drop-down list, and click **OK** to access the mPaaS SDK.

   Note

   You can click **Start Configuration** again to upgrade the baseline.
8. Click **Start Configuration** under **Configure/update components** on the right side of the window.

9. In the component list that appears, select **Code Scanner** and click **OK** to add the scan component to the project.

   Now you complete the connection of your project to mPaaS by using the native AAR mode.

What to do next

- Use the scan feature in a standard UI: Add the scan feature in a standard UI to the project and set the title of the scan screen.
- Use the scan feature in a custom UI: Add the scan feature in a custom UI to the project.

> **Note**
>
> You can configure your project to support the scan feature in both standard UIs and custom UIs, or either. Click to download the code sample to be used in the follow-up steps.

1.2.3.5. Use the scan feature in the standard UI

This topic describes how to add the scan feature in a standard UI to a project and how to set the title of the scan screen.

Use the scan feature in the standard UI

1. Open the `activity_main.xml` file in Android Studio, reset the Button layout, and set the Button ID to `standard_ui_btn` .

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
     xmlns:app="http://schemas.android.com/apk/res-auto"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     tools:context=".MainActivity">

     <Button
         android:id="@+id/standard_ui_btn"
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginTop="48dp"
         android:background="#108EE9"
         android:gravity="center"
         android:text="Scan in the standard UI"
         android:textColor="#ffffff"
         app:layout_constraintEnd_toEndOf="parent"
         app:layout_constraintHorizontal_bias="0.498"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

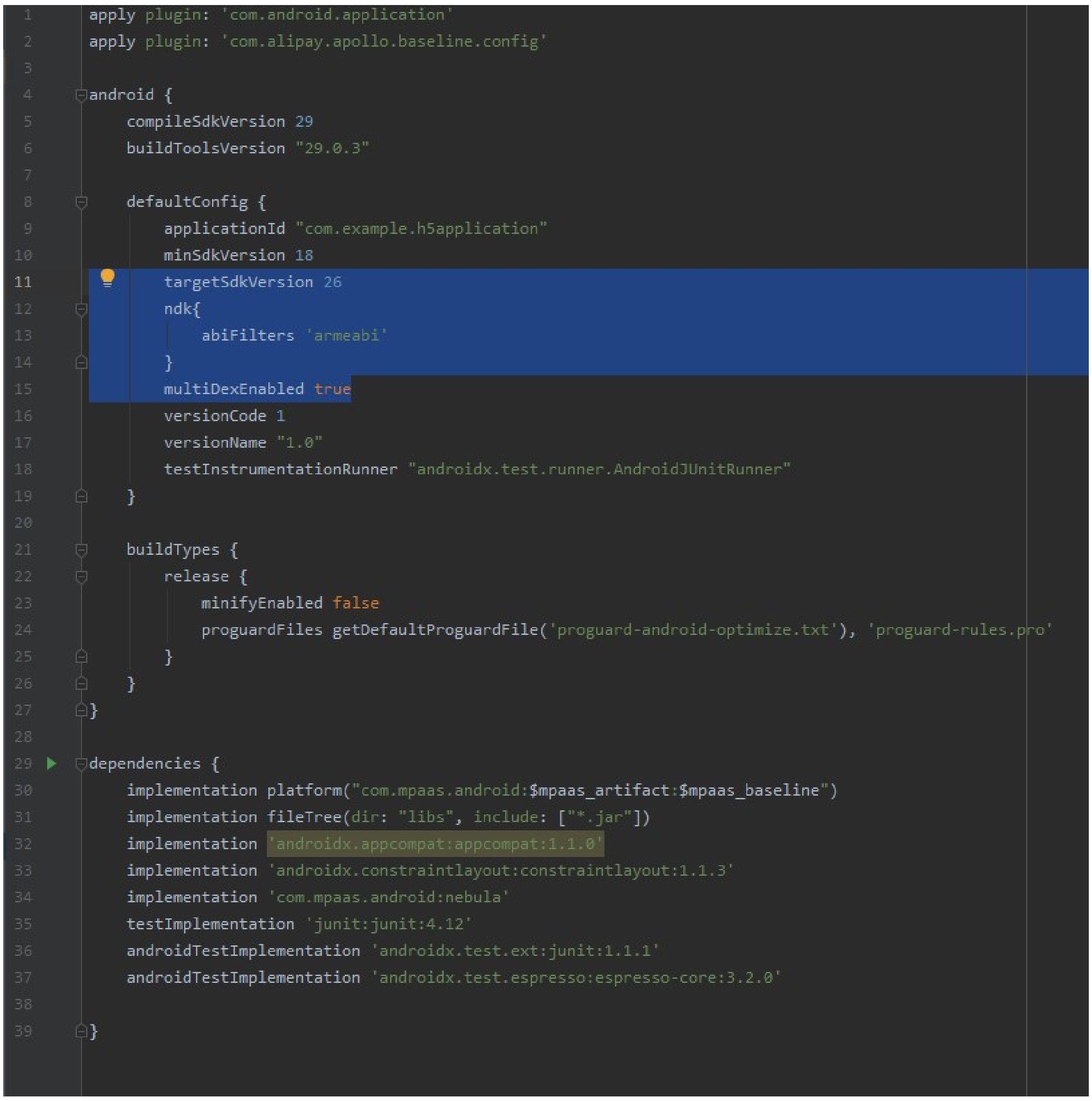
2. In the `MainActivity` class, rewrite the click event so that you can click the button to implement the scan feature. The code is as follows:

```
    private ScanRequest scanRequest = new ScanRequest();
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.standard_ui_btn).setOnClickListener(new
View.OnClickListener(){
            @Override
            public void onClick(View v) {
                MPScan.startMPaasScanActivity(MainActivity.this, scanRequest, new Sca
nCallback() {
                    @Override
                    public void onScanResult(final boolean isProcessed, final Intent
result) {
                        if (!isProcessed) {
                            // In the scan page, click the physical back button or th
e back button in the upper left corner.
                            return;
                        }
                        // Note: this callback is executed in the child thread.
                        runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                if (result == null || result.getData() == null) {
                                    // Scan failed.
                                    Toast.makeText(MainActivity.this, "Scan failed, tr
y again.", Toast.LENGTH_SHORT).show();
                                    return;
                                }
                                // Scanned.
                                new AlertDialog.Builder(MainActivity.this)
                                        .setMessage(result.getData().toString())
                                        .setPositiveButton(R.string.confirm, null)
                                        .create()
                                        .show();
                            }
                        });
                    }
                });

            }
        });
    }
```

3. In the `AndroidManifest.xml` file of the project, add the read and write permissions and the network access permission.

```
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
```

4. In the build.gradle(:app) file under the main module of the project, add the following configuration:



```
apply plugin: 'com.android.application'
apply plugin: 'com.alipay.apollo.baseline.config'

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        applicationId "com.example.h5application"
        minSdkVersion 18
        targetSdkVersion 26
        ndk{
            abiFilters 'armeabi'
        }
        multiDexEnabled true
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    implementation platform("com.mpaas.android:$mpaas_artifact:$mpaas_baseline")
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation 'androidx.appcompat:appcompat:1.1.0'
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
    implementation 'com.mpaas.android:nebula'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'androidx.test.ext:junit:1.1.1'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

}
```

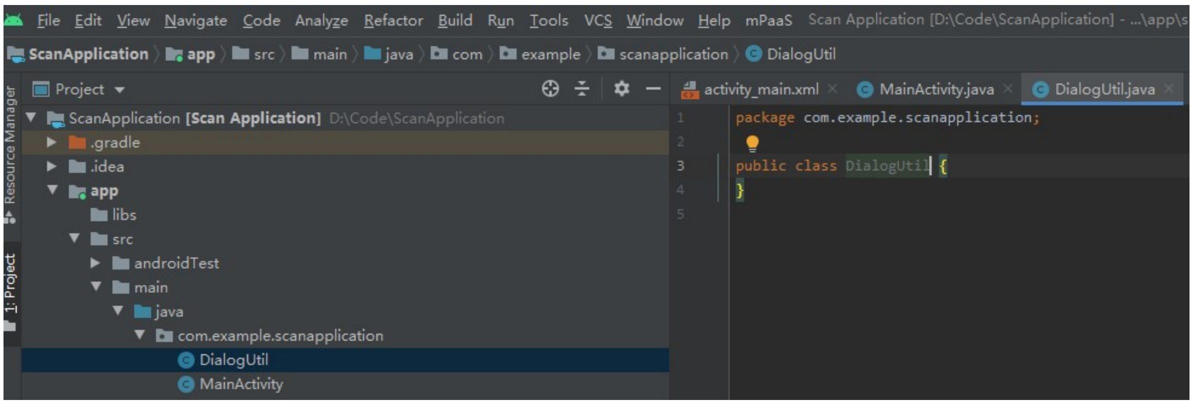
5. Compile and run the project. Then install the application on a mobile phone.
6. Click **Scan in the standard UI**. Then you can use the scan feature in a standard UI.
7. Scan the following QR code. The information of this QR code appears on the user interface.

Set the title of the scan screen

1. In the `activity_main.xml` file, add a button and set the button ID to `btn_title`.

```
<Button
    android:id="@+id/btn_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="128dp"
    android:background="#108EE9"
    android:gravity="center"
    android:text="Set the title of the scan screen on a standard UI"
    android:textColor="#ffffff"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

2. Create a `DialogUtil` class in the `com.example.scanapplication` package.


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help mPaaS Scan Application [D:\Code\ScanApplication] - ...\app\s
ScanApplication app src main java com example scanapplication DialogUtil
Project
activity_main.xml MainActivity.java DialogUtil.java
ScanApplication [Scan Application] D:\Code\ScanApplication
.gradle
.idea
app
libs
src
androidTest
main
java
com.example.scanapplication
DialogUtil
MainActivity
Resource Manager
1: Project

```
package com.example.scanapplication;

public class DialogUtil {
}
```

3. Set the layout of the scan screen in the `DialogUtil` class.

```
public interface PromptCallback {
    void onConfirm(String msg);
}

public static void prompt(Activity activity, final PromptCallback callback) {
    final EditText edit = new EditText(activity);
    new AlertDialog.Builder(activity)
            .setTitle("Enter text")
            .setView(edit)
            .setPositiveButton("OK"
                    , new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int which) {
                            if (callback != null) {
                                String text = edit.getText().toString().trim();
                                callback.onConfirm(text);
                            }
                            dialog.dismiss();
                        }
                    })
            .setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    dialog.dismiss();
                }
            })
            .create()
            .show();
}
```

4. Write code in the `MainActivity` class. The code allows you to set the title of the scan screen after you click the `btn_title` button. The code is as follows:

```
    findViewById(R.id.btn_title).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            DialogUtil.prompt(MainActivity.this, new DialogUtil.PromptCallback() {
                @Override
                public void onConfirm(String msg) {
                    scanRequest.setTitleText(msg);
                }
            });
        }
    });
```

5. Compile the project and then install the application on a mobile phone.
6. Click **Set the scan screen title on a standard UI** and enter a title to be displayed, for example, `mPaaS`. Then click **OK**.
7. Click **Scan in the standard UI**. The title information entered in step 6 is displayed in the upper left corner of the scan screen. The scan screen title is successfully set in the standard UI.

1.2.3.6. Use the scan feature in custom UI

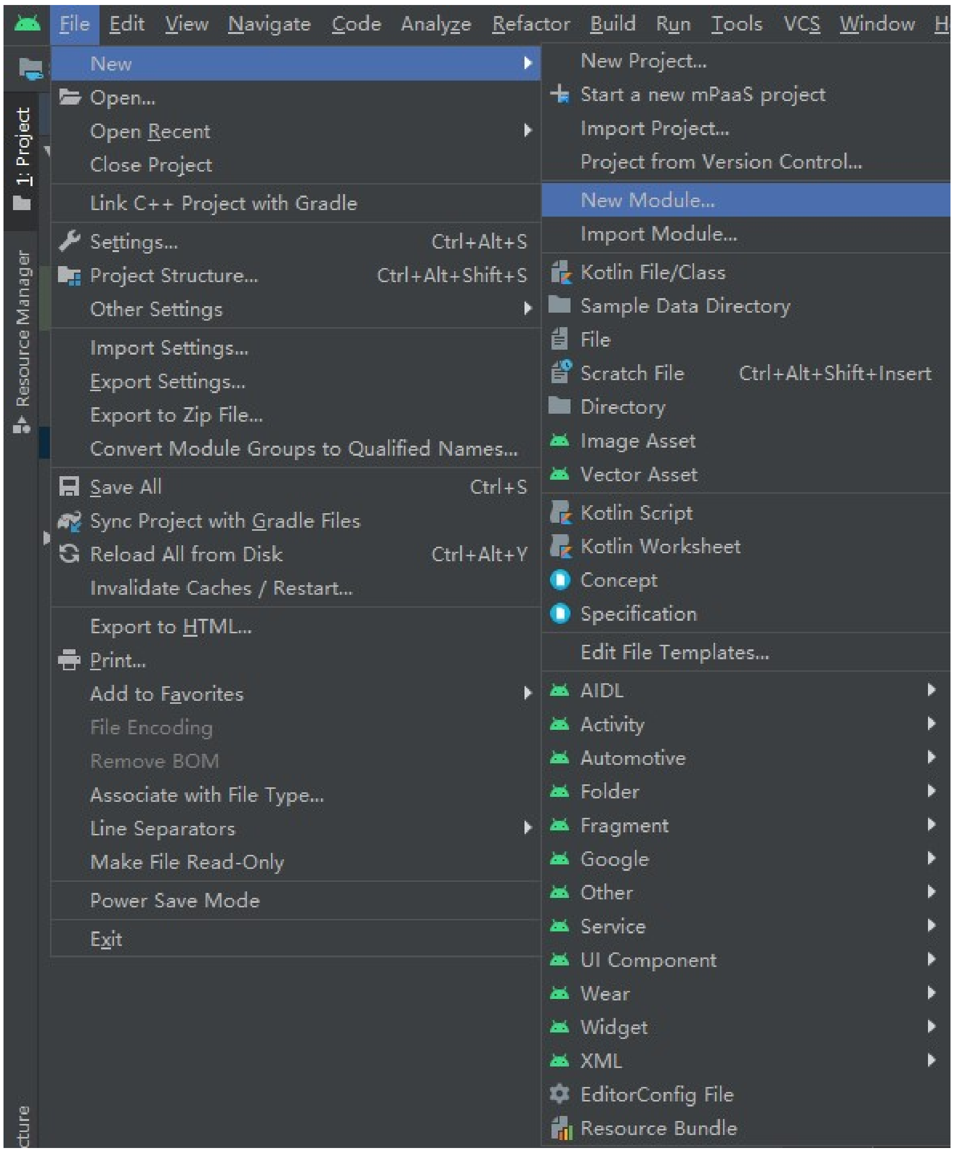
This topic describes how to customize a UI and add the scan capability in the custom UI to a project. This process involves the following four steps:

1. Create a dependency project
2. Create and customize a UI in the dependency project
3. Use the scan feature in the dependency project
4. Call the scan feature in your custom UI in the main project

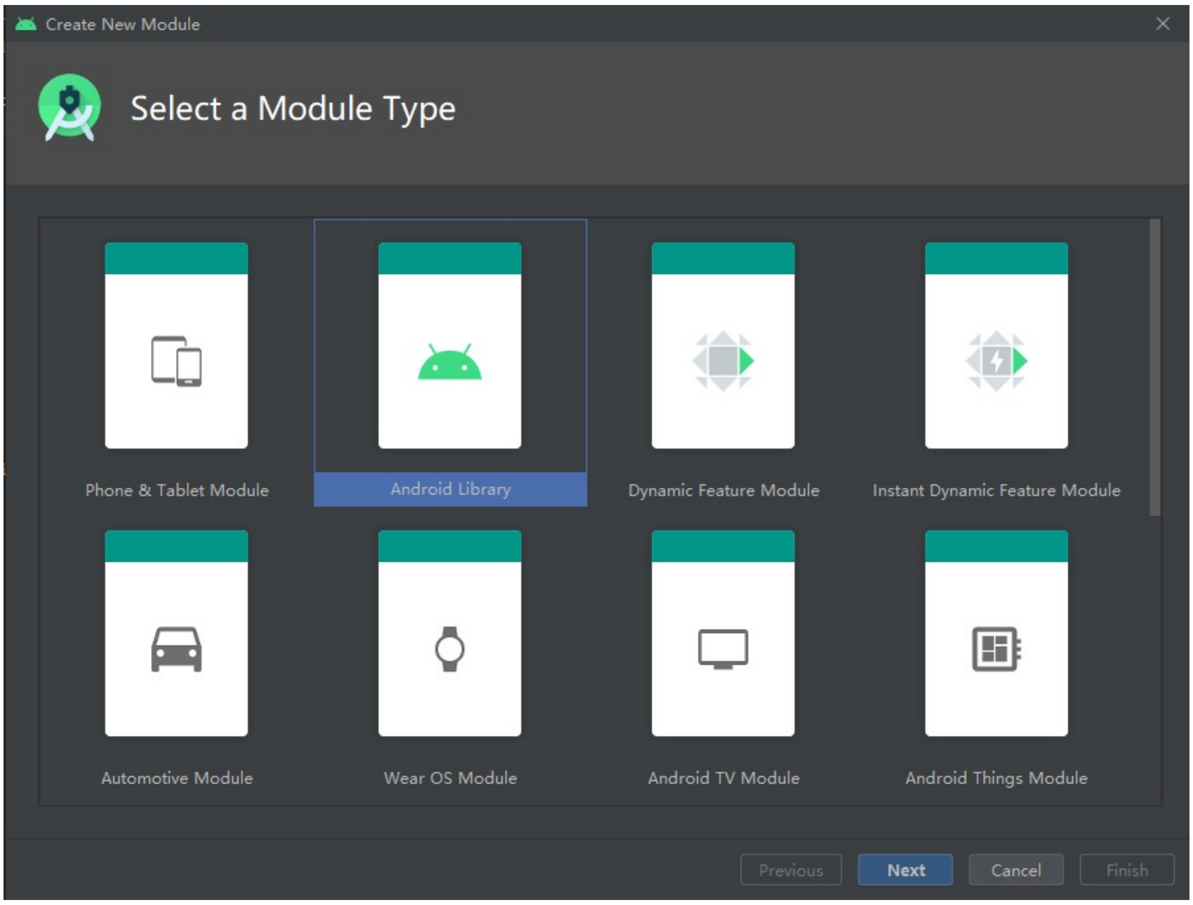
Procedure

Create a dependency project

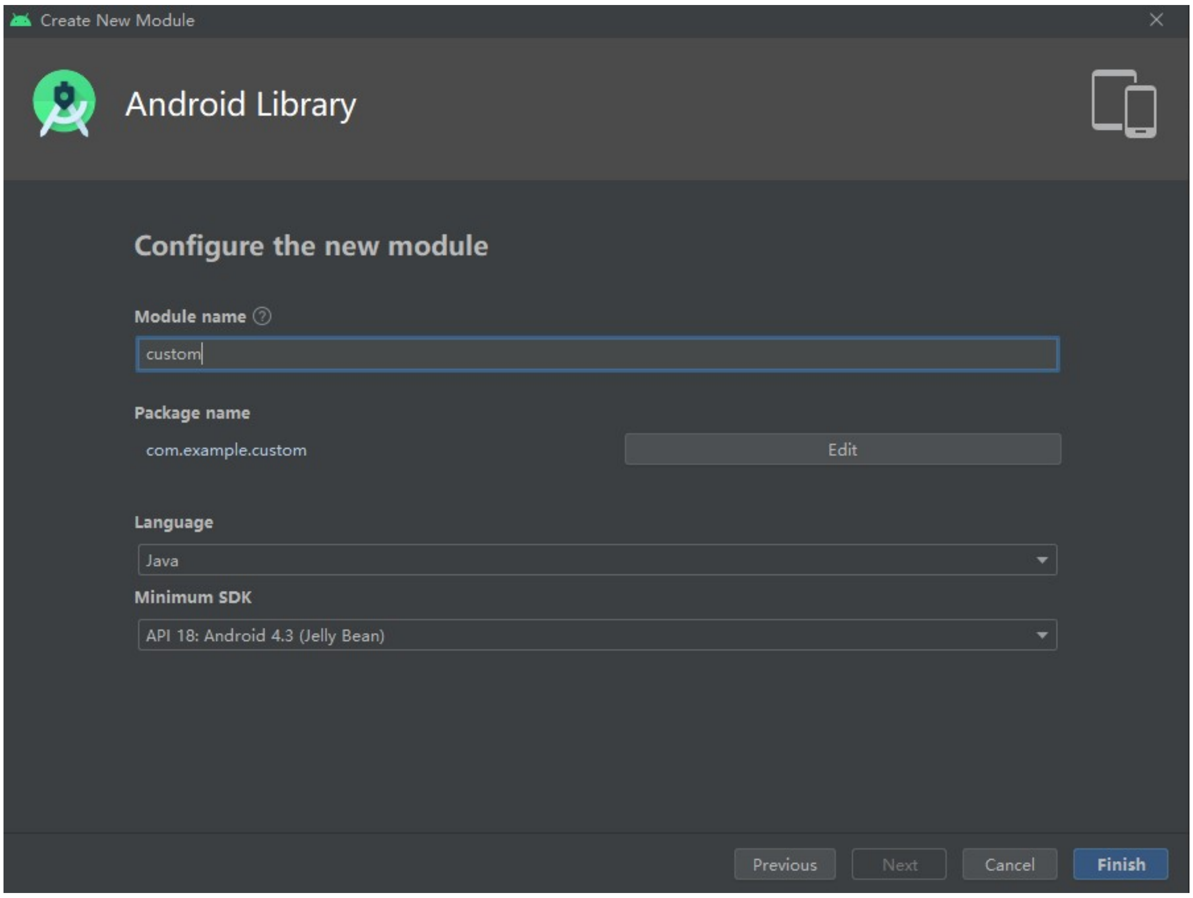
1. Choose **File** > **New** > **New Module**.



2. Select **Android Library** and click **Next**.



3. Enter **custom** in **Module name** and click **Finish**.



Configure the new module

Module name: custom

Package name: com.example.custom

Language: Java

Minimum SDK: API 18: Android 4.3 (Jelly Bean)

Create and customize a UI in the dependency project

1. Create a `widget` package in the `com.example.custom` package of the `custom` module. In the `widget` package, add the `APSurfaceTexture` class that inherits from the `SurfaceTexture` class to capture image streams.

```
public class APSurfaceTexture extends SurfaceTexture {

 private static final String TAG = "APSurfaceTexture";

 public SurfaceTexture mSurface;

 public APSurfaceTexture() {
     super(0);
 }

 @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
 @Override
 public void attachToGLContext(int texName) {
     mSurface.attachToGLContext(texName);
 }

 @TargetApi(Build.VERSION_CODES.JELLY_BEAN)
 @Override
 public void detachFromGLContext() {
     try {
         mSurface.detachFromGLContext();
     } catch (Exception ex) {
         try {
             Method nativeMethod =
SurfaceTexture.class.getDeclaredMethod("nativeDetachFromGLContext");
             nativeMethod.setAccessible(true);
             int retCode = (Integer) nativeMethod.invoke(mSurface);
             LoggerFactory.getTraceLogger().debug(TAG, "nativeDetachFromGLContext
invoke retCode:" + retCode);
         } catch (Exception e) {
             LoggerFactory.getTraceLogger().error(TAG, "nativeDetachFromGLContext
invoke exception:" + e.getMessage());
         }
         LoggerFactory.getTraceLogger().error(TAG, "mSurface.detachFromGLContext() ex
ception:" + ex.getMessage());
     }
 }

 @Override
 public boolean equals(Object o) {
     return mSurface.equals(o);
 }

 @Override
 public long getTimestamp() {
     return mSurface.getTimestamp();
 }

 @Override
 public void getTransformMatrix(float[] mtx) {
     mSurface.getTransformMatrix(mtx);
 }

 @Override
 public void release() {
     super.release();
     mSurface.release();
 }

 @Override
 public int hashCode() {
     return mSurface.hashCode();
 }

 @TargetApi(Build.VERSION_CODES.KITKAT)
 @Override
 public void releaseTexImage() {
     mSurface.releaseTexImage();
 }

 @TargetApi(Build.VERSION_CODES.ICE_CREAM_SANDWICH_MR1)
 @Override
 public void setDefaultBufferSize(int width, int height) {
     mSurface.setDefaultBufferSize(width, height);
```

```
 }

 @Override
 public void setOnFrameAvailableListener(OnFrameAvailableListener listener) {
     mSurface.setOnFrameAvailableListener(listener);
 }

 @Override
 public String toString() {
     return mSurface.toString();
 }

 @Override
 public void updateTexImage() {
     mSurface.updateTexImage();
 }
}
```

2. In the `widget` package of the `custom` module, add the `APTextureView` class that inherits from the `TextureView` class for the display of image streams.

```
public class APTextureView extends TextureView {

 private static final String TAG = "APTextureView";

 private Field mSurfaceField;

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

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

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

 @Override
 protected void onDetachedFromWindow() {
     try {
         super.onDetachedFromWindow();
     } catch (Exception ex) {
         LoggerFactory.getTraceLogger().error(TAG, "onDetachedFromWindow exception:"
+ ex.getMessage());
     }
 }

 @Override
 public void setSurfaceTexture(SurfaceTexture surfaceTexture) {
     super.setSurfaceTexture(surfaceTexture);
     afterSetSurfaceTexture();
 }
 private void afterSetSurfaceTexture() {
     LoggerFactory.getTraceLogger().debug(TAG, "afterSetSurfaceTexture
Build.VERSION.SDK_INT:" + Build.VERSION.SDK_INT);
     if (Build.VERSION.SDK_INT < 16 || Build.VERSION.SDK_INT > 20) {
         return;
     }

     try {
         if (mSurfaceField == null) {
             mSurfaceField = TextureView.class.getDeclaredField("mSurface");
             mSurfaceField.setAccessible(true);
         }

         SurfaceTexture innerSurface = (SurfaceTexture) mSurfaceField.get(this);
         if (innerSurface != null) {
             if (!(innerSurface instanceof APSurfaceTexture)) {
                 APSurfaceTexture wrapSurface = new APSurfaceTexture();
                 wrapSurface.mSurface = innerSurface;
                 mSurfaceField.set(this, wrapSurface);
                 LoggerFactory.getTraceLogger().debug(TAG, "afterSetSurfaceTexture wra
p mSurface");
             }
         }
     } catch (Exception ex) {
         LoggerFactory.getTraceLogger().error(TAG, "afterSetSurfaceTexture
exception:" + ex.getMessage());
     }
 }
}
```

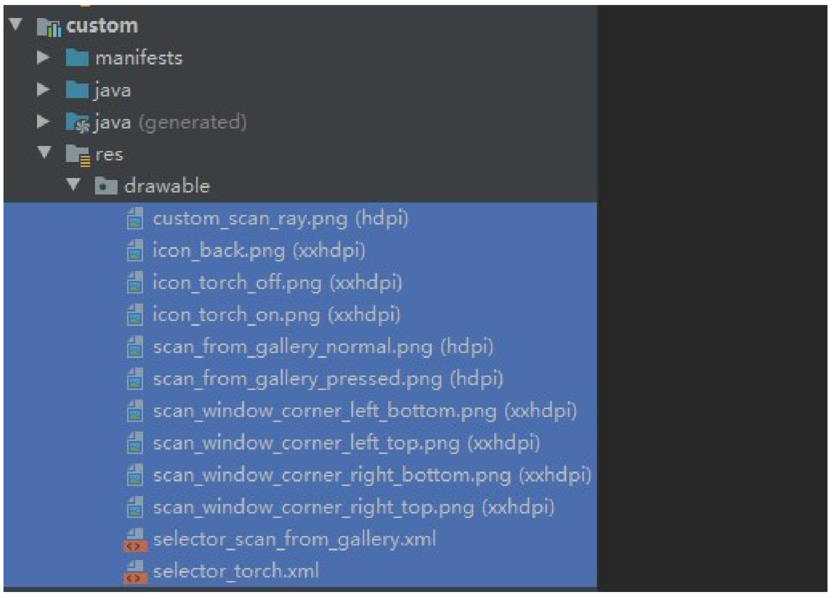
3. In the `com.example.custom` package, create a `Utils` class for the conversion of images.

```
public class Utils {

 private static String TAG = "Utils";

 public static void toast(Context context, String msg) {
     Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
 }

 public static Bitmap changeBitmapColor(Bitmap bitmap, int color) {
     int bitmap_w = bitmap.getWidth();
     int bitmap_h = bitmap.getHeight();
     int[] arrayColor = new int[bitmap_w * bitmap_h];

     int count = 0;
     for (int i = 0; i < bitmap_h; i++) {
         for (int j = 0; j < bitmap_w; j++) {

             int originColor = bitmap.getPixel(j, i);
             // Non-transparent area
             if (originColor != 0) {
                 originColor = color;
             }

             arrayColor[count] = originColor;
             count++;
         }
     }
     return Bitmap.createBitmap(arrayColor, bitmap_w, bitmap_h,
Bitmap.Config.ARGB_8888);
 }

 public static Bitmap uri2Bitmap(Context context, Uri uri) {
     Bitmap bitmap = null;
     InputStream in;
     try {
         in = context.getContentResolver().openInputStream(uri);
         if (in != null) {
             bitmap = BitmapFactory.decodeStream(in);
             in.close();
         }
     } catch (Exception e) {
         LoggerFactory.getTraceLogger().error(TAG, "uri2Bitmap: Exception " +
e.getMessage());
     }
     return bitmap;
 }
}
```

4. In the `custom` module, create an `attrs.xml` file under the `res > values` directory, and add the following code to the file.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
 <declare-styleable name="scan">
     <attr name="shadowColor" format="color" />
 </declare-styleable>
</resources>
```

5. Create a `drawable` folder under the `res` directory of the `custom` module, and copy the resource files to the `drawable` folder, as shown in the figure below.



```
▼ custom
  ▶ manifests
  ▶ java
  ▶ java (generated)
  ▼ res
    ▼ drawable
        custom_scan_ray.png (hdpi)
        icon_back.png (xxhdpi)
        icon_torch_off.png (xxhdpi)
        icon_torch_on.png (xxhdpi)
        scan_from_gallery_normal.png (hdpi)
        scan_from_gallery_pressed.png (hdpi)
        scan_window_corner_left_bottom.png (xxhdpi)
        scan_window_corner_left_top.png (xxhdpi)
        scan_window_corner_right_bottom.png (xxhdpi)
        scan_window_corner_right_top.png (xxhdpi)
        selector_scan_from_gallery.xml
        selector_torch.xml
```

6. In the `widget` package of the `custom` module, add the `FinderView` class that inherits from the `View` class. Add the following code to implement the drawing of the scan window, corners, and border shadows.

```
public class FinderView extends View {

 private static final int DEFAULT_SHADOW_COLOR = 0x96000000;

 private int scanWindowLeft, scanWindowTop, scanWindowRight, scanWindowBottom;
 private Bitmap leftTopCorner, rightTopCorner, leftBottomCorner, rightBottomCorner;
 private Paint paint;
 private int shadowColor;

 public FinderView(Context context, AttributeSet attrs, int defStyle) {
     super(context, attrs, defStyle);
     init(context, attrs);
 }

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

 private void init(Context context, AttributeSet attrs) {
     applyConfig(context, attrs);
     setVisibility(INVISIBLE);
     initCornerBitmap(context);
     initCornerBitmap(context);

     paint = new Paint();
     paint.setAntiAlias(true);
 }

 private void applyConfig(Context context, AttributeSet attrs) {
     if (attrs != null) {
         TypedArray typedArray = context.obtainStyledAttributes(attrs,
R.styleable.scan);
         shadowColor = typedArray.getColor(R.styleable.scan_shadowColor,
DEFAULT_SHADOW_COLOR);
         typedArray.recycle();
     }
 }
 //Initialize the corner style of the scan window.
 private void initCornerBitmap(Context context) {
     Resources res = context.getResources();
     leftTopCorner = BitmapFactory.decodeResource(res,
R.drawable.scan_window_corner_left_top);
     rightTopCorner = BitmapFactory.decodeResource(res,
R.drawable.scan_window_corner_right_top);
     leftBottomCorner = BitmapFactory.decodeResource(res,
R.drawable.scan_window_corner_left_bottom);
     rightBottomCorner = BitmapFactory.decodeResource(res,
R.drawable.scan_window_corner_right_bottom);
 }

 @Override
 public void draw(Canvas canvas) {
     super.draw(canvas);
     drawShadow(canvas);
     drawCorner(canvas);
 }
 //Draw the corner styles of the scan window.
 private void drawCorner(Canvas canvas) {
     paint.setAlpha(255);
     canvas.drawBitmap(leftTopCorner, scanWindowLeft, scanWindowTop, paint);
     canvas.drawBitmap(rightTopCorner, scanWindowRight - rightTopCorner.getWidth(), s
canWindowTop, paint);
     canvas.drawBitmap(leftBottomCorner, scanWindowLeft, scanWindowBottom -
leftBottomCorner.getHeight(), paint);
     canvas.drawBitmap(rightBottomCorner, scanWindowRight -
rightBottomCorner.getWidth(), scanWindowBottom - rightBottomCorner.getHeight(), paint
);
 }
 //Draw the border shadows of the scan window.
 private void drawShadow(Canvas canvas) {
     paint.setColor(shadowColor);
     canvas.drawRect(0, 0, getWidth(), scanWindowTop, paint);
     canvas.drawRect(0, scanWindowTop, scanWindowLeft, scanWindowBottom, paint);
     canvas.drawRect(scanWindowRight, scanWindowTop, getWidth(), scanWindowBottom, pa
int);
     canvas.drawRect(0, scanWindowBottom, getWidth(), getHeight(), paint);
 }
```

```
    /**
     * Determine the location of the scan window according to the position of RayView.
     */
    public void setScanWindowLocation(int left, int top, int right, int bottom) {
        scanWindowLeft = left;
        scanWindowTop = top;
        scanWindowRight = right;
        scanWindowBottom = bottom;
        invalidate();
        setVisibility(VISIBLE);
    }

    public void setShadowColor(int shadowColor) {
        this.shadowColor = shadowColor;
    }
    //Set the corner colors of the scan window.
    public void setCornerColor(int angleColor) {
        leftTopCorner = Utils.changeBitmapColor(leftTopCorner, angleColor);
        rightTopCorner = Utils.changeBitmapColor(rightTopCorner, angleColor);
        leftBottomCorner = Utils.changeBitmapColor(leftBottomCorner, angleColor);
        rightBottomCorner = Utils.changeBitmapColor(rightBottomCorner, angleColor);
    }
}
```

7. In the `widget` package, add the `RayView` class that inherits from the `ImageView` class. Add the following code to implement the drawing of the scan lines.

```
public class RayView extends ImageView {

 private FinderView mFinderView;
 private ScaleAnimation scanAnimation;
 private int[] location = new int[2];

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

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

 @Override
 protected void onLayout(boolean changed, int left, int top, int right, int bottom) {
     super.onLayout(changed, left, top, right, bottom);

     //Set the position of the scan window in FinderView.
     getLocationOnScreen(location);
     if (mFinderView != null) {
         mFinderView.setScanWindowLocation(location[0], location[1], location[0] + ge
tWidth(), location[1] + getHeight());
     }
 }

 public void startScanAnimation() {
     setVisibility(VISIBLE);
     if (scanAnimation == null) {
         scanAnimation = new ScaleAnimation(1.0f, 1.0f, 0.0f, 1.0f);
         scanAnimation.setDuration(3000L);
         scanAnimation.setFillAfter(true);
         scanAnimation.setRepeatCount(Animation.INFINITE);
         scanAnimation.setInterpolator(new AccelerateDecelerateInterpolator());
     }
     startAnimation(scanAnimation);
 }

 public void stopScanAnimation() {
     setVisibility(INVISIBLE);
     if (scanAnimation != null) {
         this.clearAnimation();
         scanAnimation = null;
     }
 }

 public void setFinderView(FinderView FinderView) {
     mFinderView = FinderView;
 }
}
```

8. Create a `res > layout > File` folder, then create a `view_scan.xml` file under the folder, and add the following code to draw the layout of the scan page.

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android">

    <com.example.custom.widget.FinderView
        android:id="@+id/finder_view"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="20dp"
        android:gravity="center_vertical"
        android:orientation="horizontal">

        <ImageView
            android:id="@+id/back"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:scaleType="center"
            android:src="@drawable/icon_back" />

        <TextView
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="center"
            android:text="@string/custom_title"
            android:textColor="#ffffff"
            android:textSize="16sp" />

        <ImageView
            android:id="@+id/gallery"
            android:layout_width="34dp"
            android:layout_height="34dp"
            android:layout_marginEnd="10dp"
            android:layout_marginRight="10dp"
            android:scaleType="fitXY"
            android:src="@drawable/selector_scan_from_gallery" />

        <ImageView
            android:id="@+id/torch"
            android:layout_width="34dp"
            android:layout_height="34dp"
            android:layout_marginEnd="10dp"
            android:layout_marginRight="10dp"
            android:scaleType="fitXY"
            android:src="@drawable/selector_torch" />
    </LinearLayout>

    <com.example.custom.widget.RayView
        android:id="@+id/ray_view"
        android:layout_width="270dp"
        android:layout_height="280dp"
        android:layout_centerInParent="true"
        android:background="@drawable/custom_scan_ray" />

    <TextView
        android:id="@+id/tip_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/ray_view"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="10dp"
        android:includeFontPadding="false"
        android:text="@string/scan_tip"
        android:textColor="#7fffffff"
        android:textSize="14sp" />

</merge>
```

9. In the `widget` package, add the `ScanView` class that inherits from the `RelativeLayout` class. Add the following code. This code implements the interaction between the scan-related view and the scan engine.

```
public class ScanView extends RelativeLayout {

    private RayView mRayView;

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

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

    public ScanView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        init(context);
    }

    private void init(Context ctx) {
        LayoutInflater.from(ctx).inflate(R.layout.view_scan, this, true);
        FinderView finderView = (FinderView) findViewById(R.id.finder_view);
        mRayView = (RayView) findViewById(R.id.ray_view);
        mRayView.setFinderView(finderView);
    }

    public void onStartScan() {
        mRayView.startScanAnimation();
    }

    public void onStopScan() {
        mRayView.stopScanAnimation();
    }
```

```
public float getCropWidth() {
    return mRayView.getWidth() * 1.1f;
}

public Rect getScanRect(Camera camera, int previewWidth, int previewHeight) {
    if (camera == null) {
        return null;
    }
    int[] location = new int[2];
    mRayView.getLocationOnScreen(location);
    Rect r = new Rect(location[0], location[1],
            location[0] + mRayView.getWidth(), location[1] + mRayView.getHeight());
    Camera.Size size;
    try {
        size = camera.getParameters().getPreviewSize();
    } catch (Exception e) {
        return null;
    }
    if (size == null) {
        return null;
    }
    double rateX = (double) size.height / (double) previewWidth;
    double rateY = (double) size.width / (double) previewHeight;
    // The size of the crop box = The size of the grid animation box × 1.1
    int expandX = (int) (mRayView.getWidth() * 0.05);
    int expandY = (int) (mRayView.getHeight() * 0.05);
    Rect resRect = new Rect(
            (int) ((r.top - expandY) * rateY),
            (int) ((r.left - expandX) * rateX),
            (int) ((r.bottom + expandY) * rateY),
            (int) ((r.right + expandX) * rateX));

    Rect finalRect = new Rect(
            resRect.left < 0 ? 0 : resRect.left,
            resRect.top < 0 ? 0 : resRect.top,
            resRect.width() > size.width ? size.width : resRect.width(),
            resRect.height() > size.height ? size.height : resRect.height());

    Rect rect1 = new Rect(
            finalRect.left / 4 * 4,
            finalRect.top / 4 * 4,
            finalRect.right / 4 * 4,
            finalRect.bottom / 4 * 4);

    int max = Math.max(rect1.right, rect1.bottom);
    int diff = Math.abs(rect1.right - rect1.bottom) / 8 * 4;

    Rect rect2;
    if (rect1.right > rect1.bottom) {
        rect2 = new Rect(rect1.left, rect1.top - diff, max, max);
    } else {
        rect2 = new Rect(rect1.left - diff, rect1.top, max, max);
    }
    return rect2;
```

```
  }
}
```

0. Create an `activity_custom_scan.xml` file under the `res > layout` folder, and add the following code to the file. This code draws the main page of the custom scan feature.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <com.mpaas.aar.demo.custom.widget.APTextureView
       android:id="@+id/surface_view"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />

   <com.mpaas.aar.demo.custom.widget.ScanView
       android:id="@+id/scan_view"
       android:layout_width="match_parent"
       android:layout_height="match_parent" />

</FrameLayout>
```

Use the scan feature in the dependency project

1. In the `com.example.custom` package of the `custom` module, add a `ScanHelper` class and add the following code. This code calls the scan feature and obtains the callback result of a scan result.

```
public class ScanHelper {

 private static class Holder {
     private static ScanHelper instance = new ScanHelper();
 }

 private ScanCallback scanCallback;

 private ScanHelper() {
 }

 public static ScanHelper getInstance() {
     return Holder.instance;
 }

 public void scan(Context context, ScanCallback scanCallback) {
     if (context == null) {
         return;
     }
     this.scanCallback = scanCallback;
     context.startActivity(new Intent(context, CustomScanActivity.class));
 }

 void notifyScanResult(boolean isProcessed, Intent resultData) {
     if (scanCallback != null) {
         scanCallback.onScanResult(isProcessed, resultData);
         scanCallback = null;
     }
 }

 public interface ScanCallback {
     void onScanResult(boolean isProcessed, Intent result);
 }
}
```

2. In the `com.example.custom` package of the `custom` module, add the `CustomScanActivity` class that inherits from the `Activity` class. Set the UI immersive mode and create the `View` and `Button` corresponding to the resource file.

```
public class CustomScanActivity extends Activity {
 private final String TAG = CustomScanActivity.class.getSimpleName();
 private static final int REQUEST_CODE_PERMISSION = 1;
 private static final int REQUEST_CODE_PHOTO = 2;
 private ImageView mTorchBtn;
 private APTextureView mTextureView;
 private ScanView mScanView;
 private boolean isFirstStart = true;
 private boolean isPermissionGranted;
 private boolean isScanning;
 private boolean isPaused;
 private Rect scanRect;
 private MPScanner mpScanner;
 @Override
 protected void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_custom_scan);

     // Sets the immersive mode.
     if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
         getWindow().setFlags(
                 WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS,
                 WindowManager.LayoutParams.FLAG_TRANSLUCENT_STATUS);
     }

     mTextureView = findViewById(R.id.surface_view);
     mScanView = findViewById(R.id.scan_view);
     mTorchBtn = findViewById(R.id.torch);

 }

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

 }

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

 }

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

 }

 @Override
 public void onRequestPermissionsResult(int requestCode, @NonNull String[] permission
s, @NonNull int[] grantResults) {
     super.onRequestPermissionsResult(requestCode, permissions, grantResults);

 }
  @Override
 public void onBackPressed() {
     super.onBackPressed();

 }

 @Override
 public void onActivityResult(int requestCode, int resultCode, Intent data) {
     super.onActivityResult(requestCode, resultCode, data);

 }
}
```

3. Follow the operations below to realize the function of opening mobile phone gallery.

   i. Create a `pickImageFromGallery` method in `CustomScanActivity` .

```
private void pickImageFromGallery() {
 Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
 intent.setType("image/*");
 startActivityForResult(intent, REQUEST_CODE_PHOTO);
}
```

   ii. In the `onCreate` method in `CustomScanActivity` , add the click event of `gallery` , and call the `pickImageFromGallery` method.

```
     findViewById(R.id.gallery).setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             pickImageFromGallery();
         }
     });
```

4. Follow the operations below to realize the function of turning on or off torch.
   i. Create a `switchTorch` method in `CustomScanActivity` .

```
 private void switchTorch() {
     boolean torchOn = mpScanner.switchTorch();
     mTorchBtn.setSelected(torchOn);
 }
```

   ii. In the `onCreate` method in `CustomScanActivity` , add the click event of `mTorchBtn` and call the `switchTorch` method.

```
  mTorchBtn.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             switchTorch();
         }
     });
```

5. In CustomScanActivity, create a `notifyScanResult` method, and call the `notifyScanResult` method in the `onBackPressed` method.

```
private void notifyScanResult(boolean isProcessed, Intent resultData) {
    ScanHelper.getInstance().notifyScanResult(isProcessed, resultData);
}

@Override
public void onBackPressed() {
    super.onBackPressed();
    notifyScanResult(false, null);
}
```

6. In the `onCreate` method in `CustomScanActivity` , add the click event of `back` and call the `onBackPressed` method.

```
    findViewById(R.id.back).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            onBackPressed();
        }
    });
```

7. Create an `initMPScanner` method in `CustomScanActivity`, and use the `setRecognizeType` method in the `mpScanner` object to set the type of the identification code.

```
private void initMPScanner() {
     mpScanner = new MPScanner(this);
     mpScanner.setRecognizeType(
             MPRecognizeType.QR_CODE,
             MPRecognizeType.BAR_CODE,
             MPRecognizeType.DM_CODE,
             MPRecognizeType.PDF417_CODE
     );
}
```

8. Create an `onScanSuccess` method in `CustomScanActivity` and implement the following code:

```
 private void onScanSuccess(final MPScanResult result) {
     runOnUiThread(new Runnable() {
         @Override
         public void run() {
             if (result == null) {
                 notifyScanResult(true, null);
             } else {
                 Intent intent = new Intent();
                 intent.setData(Uri.parse(result.getText()));
                 notifyScanResult(true, intent);
             }
             CustomScanActivity.this.finish();
         }
     });
 }
```

9. Create an `initScanRect` method in `CustomScanActivity` to initialize the scan feature.

i. Call the `getCamera` method in the `mpScanner` object to get the `Camera` object. Call the `setScanRegion` method in the `mpScanner` object to set scan area.

```
private void initScanRect() {
 if (scanRect == null) {
     scanRect = mScanView.getScanRect(
             mpScanner.getCamera(), mTextureView.getWidth(),
mTextureView.getHeight());

     float cropWidth = mScanView.getCropWidth();
     LoggerFactory.getTraceLogger().debug(TAG, "cropWidth: " + cropWidth);
     if (cropWidth > 0) {
         // Maximum preview window width = Screen width/Crop box width
         WindowManager wm = (WindowManager)
getSystemService(Context.WINDOW_SERVICE);
         float screenWith = wm.getDefaultDisplay().getWidth();
         float screenHeight = wm.getDefaultDisplay().getHeight();
         float previewScale = screenWith / cropWidth;
         if (previewScale < 1.0f) {
             previewScale = 1.0f;
         }
         if (previewScale > 1.5f) {
             previewScale = 1.5f;
         }
         LoggerFactory.getTraceLogger().debug(TAG, "previewScale: " +
previewScale);
         Matrix transform = new Matrix();
         transform.setScale(previewScale, previewScale, screenWith / 2,
screenHeight / 2);
         mTextureView.setTransform(transform);
     }
 }
 mpScanner.setScanRegion(scanRect);
}
```

ii. In the `initMPScanner` method, call the `setMPScanListener` method in the `mpScanner` object to scan the listener.

```
mpScanner.setMPScanListener(new MPScanListener() {
     @Override
     public void onConfiguration() {
         mpScanner.setDisplayView(mTextureView);
     }

     @Override
     public void onStart() {
         if (!isPaused) {
             runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                     if (!isFinishing()) {
                         initScanRect();
                         mScanView.onStartScan();
                     }
                 }
             });
         }
     }

     @Override
     public void onSuccess(MPScanResult mpScanResult) {
         mpScanner.beep();
         onScanSuccess(mpScanResult);
     }

     @Override
     public void onError(MPScanError mpScanError) {
         if (!isPaused) {
             runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                     Utils.toast(CustomScanActivity.this,
getString(R.string.camera_open_error));
                 }
             });
         }
     }
 });
```

iii. In the `initMPScanner` method, call the `setMPImageGrayListener` method in the `mpScanner` object to listen to the gray value of a recognized image.

```
 mpScanner.setMPImageGrayListener(new MPImageGrayListener() {
     @Override
     public void onGetImageGray(int gray) {
         // Note: This callback may be executed multiple consecutive times in dark
environments.
         if (gray < MPImageGrayListener.LOW_IMAGE_GRAY) {
             runOnUiThread(new Runnable() {
                 @Override
                 public void run() {
                     Utils.toast(CustomScanActivity.this, "The light is too dark, pl
ease turn on the flashlight");
                 }
             });
         }
     }
 });
}
```

0. In `CustomScanActivity`, create a `startScan` method to enable the scan feature of the camera, and create a `stopScan` method to disable the scan feature of the camera.

```
private void startScan() {
    try {
        mpScanner.openCameraAndStartScan();
        isScanning = true;
    } catch (Exception e) {
        isScanning = false;
        LoggerFactory.getTraceLogger().error(TAG, "startScan: Exception " +
e.getMessage());
    }
}

private void stopScan() {
    mpScanner.closeCameraAndStopScan();
    mScanView.onStopScan();
    isScanning = false;
    if (isFirstStart) {
        isFirstStart = false;
    }
}
```

1. In `CustomScanActivity`, create `onPermissionGranted`, `checkCameraPermission` and `scanFromUri` methods.

```
private void onPermissionGranted() {
    isPermissionGranted = true;
    startScan();
}

private void checkCameraPermission() {
    if (PermissionChecker.checkSelfPermission(
            this, Manifest.permission.CAMERA) !=
PermissionChecker.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.CAMERA}, REQUEST_CODE_PERMISSION);
    } else {
        onPermissionGranted();
    }
}

private void scanFromUri(Uri uri) {
    final Bitmap bitmap = Utils.uri2Bitmap(this, uri);
    if (bitmap == null) {
        notifyScanResult(true, null);
        finish();
    } else {
        new Thread(new Runnable() {
            @Override
            public void run() {
                MPScanResult mpScanResult = mpScanner.scanFromBitmap(bitmap);
                mpScanner.beep();
                onScanSuccess(mpScanResult);
            }
        }, "scanFromUri").start();
    }
}
```

2. Call the `checkCameraPermission` method in the `onCreate` method in `CustomScanActivity` to check camera permissions.

```
checkCameraPermission();
```

3. Separately add the following content to the `onPause` , `onResume` , `onDestroy` , `onRequestPermissionsResult` , and `onActivityResult` methods in `CustomScanActivity` :

```
@Override
public void onPause() {
    super.onPause();
    isPaused = true;
    if (isScanning) {
        stopScan();
    }
}

@Override
public void onResume() {
    super.onResume();
    isPaused = false;
    if (!isFirstStart && isPermissionGranted) {
        startScan();
    }
}

@Override
public void onDestroy() {
    super.onDestroy();
    mpScanner.release();
}
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions
, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == REQUEST_CODE_PERMISSION) {
        int length = Math.min(permissions.length, grantResults.length);
        for (int i = 0; i < length; i++) {
            if (TextUtils.equals(permissions[i], Manifest.permission.CAMERA)) {
                if (grantResults[i] != PackageManager.PERMISSION_GRANTED) {
                    Utils.toast(this, getString(R.string.camera_no_permission));
                } else {
                    onPermissionGranted();
                }
                break;
            }
        }
    }
      @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (data == null) {
        return;
    }
    if (requestCode == REQUEST_CODE_PHOTO) {
        scanFromUri(data.getData());
    }
}
}
```

4. In the `AndroidManifest.xml` file in the `custom` module, set `CustomScanActivity` as the main entry of `custom`.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.mpaas.aar.demo.custom">
       <activity
           android:name=".CustomScanActivity"
           android:configChanges="orientation|keyboardHidden|navigation"
           android:exported="false"
           android:launchMode="singleTask"
           android:screenOrientation="portrait"
           android:theme="@android:style/Theme.NoTitleBar"
           android:windowSoftInputMode="adjustResize|stateHidden" />
   </application>

</manifest>
```

Call the scan feature in your custom UI in the main project

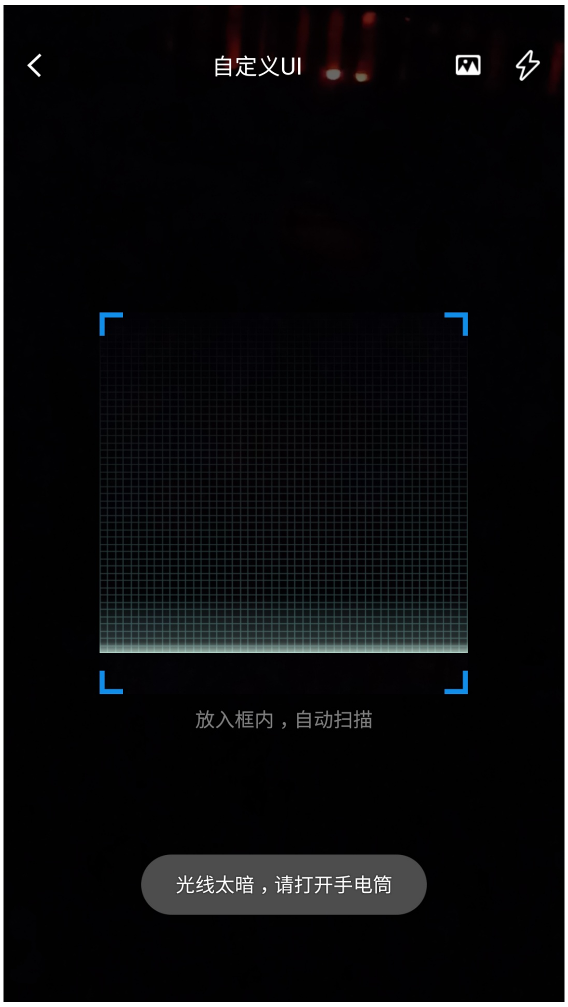
1. In the `activity_main.xml` file, add a button and set the ID of the button to `custom_ui_btn`.

```
 <Button
     android:id="@+id/custom_ui_btn"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:layout_marginTop="208dp"
     android:background="#108EE9"
     android:gravity="center"
     android:text="Use Scan in the Custom UI"
     android:textColor="#ffffff"
     app:layout_constraintEnd_toEndOf="parent"
     app:layout_constraintHorizontal_bias="0.0"
     app:layout_constraintStart_toStartOf="parent"
     app:layout_constraintTop_toTopOf="parent" />
```

2. Edit code in the `MainActivity` class. Add a click event to the `custom_ui_btn` button. Obtain the custom UI and then use the scan feature in the custom UI. The code is as follows:

```
findViewById(R.id.custom_ui_btn).setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
             ScanHelper.getInstance().scan(MainActivity.this, new
ScanHelper.ScanCallback() {
                 @Override
                 public void onScanResult(boolean isProcessed, Intent result) {
                     if (!isProcessed) {
                         // In the scan page, click the physical back button or the ba
ck button in the upper left corner.
                         return;
                     }

                     if (result == null || result.getData() == null) {
                         Toast.makeText(MainActivity.this, "Scan failed, try again.",
Toast.LENGTH_SHORT).show();
                         return;
                     }
                     new AlertDialog.Builder(MainActivity.this)
                             .setMessage(result.getData().toString())
                             .setPositiveButton(R.string.confirm, null)
                             .create()
                             .show();
                 }
             });
         }
     });
```

3. After you compile and run the project, click **Use Scan in the Custom UI** to use the scan feature in the custom UI.



4. Scan the QR code below, then the information about the QR code will be displayed.

1.3. Integrate iOS SDK

1.3.1. Quick Start

The Scan SDK is currently used by Alipay to scan QR codes, barcodes, and other functions. This topic describes how to use the Scan SDK.

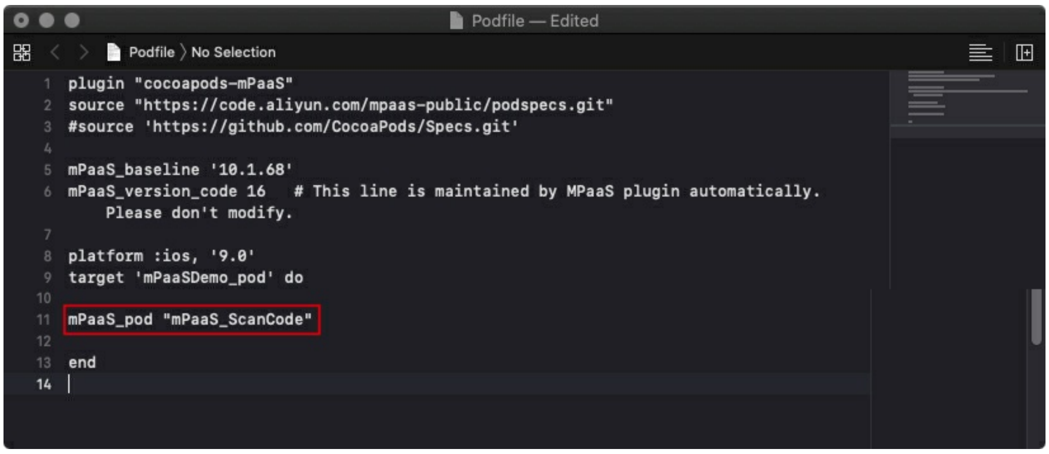
Prerequisites

You have added the Scan SDK to the project according to your access method. For more information, see the following content: Connect to mPaaS based on an existing project and CocoaPods.

Add the SDK

Use CocoaPods plugin to add the Scan SDK. Complete the following steps:

1. In the Podfile file, use `mPaaS_pod "mPaaS_ScanCode"` to add mobile gateway component dependencies.



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

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

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

mPaaS_pod "mPaaS_ScanCode"

end

```

2. In the terminal, run `pod install` to complete access.

Using SDK version 10.1.68.17 and later

Procedure

1. Trigger the default scan page and process the scan results.

```
 @interface MPScanDemoVC()<TBScanViewControllerDelegate>
 @property(nonatomic, strong) TBScanViewController *scanVC;
 @end
 - (void)defaultScan {
    TBScanViewController *vc = [[MPScanCodeAdapterInterface sharedInstance]
createDefaultScanPageWithallback:^(id  _Nonnull result, BOOL keepAlive) {
        // Process scan results.
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
message:result[@"resp_result"] delegate:self cancelButtonTitle:@"OK"
otherButtonTitles:nil, nil];
         alert.tag = 1999;
         [alert show];
     }];
     [self.navigationController pushViewController:vc animated:YES];
     self.scanVC =  vc;
 }
```

2. Continue to scan the code.

```
 - (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
(NSInteger)buttonIndex {
     // Continue to scan the code.
     [self.scanVC resumeScan];
 }
```

Use SDK version 10.1.68.17 and earlier

This topic describes how to use the Scan SDK in baseline version 10.1.68.17 and earlier based on the official demo of Scan.

Procedure

1. Trigger the scan page.

```
@interface MPScanDemoVC()<TBScanViewControllerDelegate>
@property(nonatomic, strong) TBScanViewController *scanVC;
@end
- (void)startDefauleScanViewController
{
    TBScanViewController *vc = [[TBScanViewController alloc] init];
    vc.scanType = ScanType_All_Code;
    vc.delegate = self;
    [self.navigationController pushViewController:vc animated:YES];
    self.scanVC = vc;
}
```

2. Process scan results.

```
#pragma mark Process scan results.
-(void)didFind:(NSArray<TBScanResult*>*)resultArray
{
    if([resultArray count] > 0) {
        TBScanResult *result = resultArray.firstObject;
        NSString* content = result.data;

        dispatch_async(dispatch_get_main_queue(), ^{
            // Note: The scan results are in the child thread. If there are UI-
related operations, switch to the main thread.
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
message:content delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
            [alert show];
        });
    }
}
```

3. Continue to scan the code.

```
#pragma mark alert
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
(NSInteger)buttonIndex {
    [self.scanVC resumeScan];
}
```

1.3.2. Advanced guide

This article will introduce how to use the scan feature in conjunction with the official scan demo.

- To use the scan feature in baseline version 10.1.60 and above and before 10.2.3.5, please refer to Use scan in the default UI and Use Scan in custom UI.
- To use the scan feature in baseline version 10.2.3.5 and above, please refer to Multi-code recognition.

Use scan in the default UI

Modify the parameters on the scan page in the default UI.

```
    - (void)custoDefaultScan {
       TBScanViewController *vc = [[MPScanCodeAdapterInterface sharedInstance]
createDefaultScanPageWithallback:^(id  _Nonnull result, BOOL keepAlive) {
            UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
message:result[@"resp_result"] delegate:self cancelButtonTitle:@"OK"
otherButtonTitles:nil, nil];
            alert.tag = 1001;
            [alert show];
        }];
        [self.navigationController pushViewController:vc animated:YES];
        self.scanVC =  vc;

        // Set the scan page title
        vc.title = @"Standard scan";

        // Set the tooltip "Turn on the torch."
        vc.torchStateNormalTitle = @"Turn on the torch";

        // Set the tooltip "Turn off the torch."
        vc.torchStateSelectedTitle = @"Turn off the torch";

        // Set the type of the object that you want to scan.
        vc.scanType =  ScanType_QRCode;

        // Set a Select Album button.
        vc.navigationItem.rightBarButtonItem = [[UIBarButtonItem alloc]
initWithImage:APCommonUILoadImage(@"camera") style:UIBarButtonItemStylePlain target:sel
f action:@selector(selectPhotos)];

    }

    - (void)selectPhotos
    {
        [self.scanVC scanPhotoLibrary];
    }
```

Use Scan in custom UI

If you want to customize the entire scan UI, you can customize the scan page and make it inherit TBScanViewController.

- Create a scan page and customize the scan area.

```
  @interface MPScanCodeViewController : TBScanViewController
<TBScanViewControllerDelegate>

  @end

  @implementation MPScanCodeViewController

  - (instancetype)init
  {
      if (self = [super init])
      {
          self.delegate = self;
          self.delegate = self;
          self.scanType = ScanType_All_Code;
      }
      return self;
  }

  - (void)viewDidLoad {
      [super viewDidLoad];
      // Do any additional setup after loading the view.
      self.title = @"Scan";

      // Customize the size of the scan page.
      CGRect rect = [MPScanCodeViewController constructScanAnimationRect];
      self.rectOfInterest = rect;

      // Customize the Select Album button.
      self.navigationItem.rightBarButtonItem = [[UIBarButtonItem alloc]
initWithTitle:@"Select Album" style:UIBarButtonItemStylePlain target:self action:@sel
ector(selectPhoto)];
  }

  + (CGRect)constructScanAnimationRect
  {
      CGSize screenXY = [UIScreen mainScreen].bounds.size;
      NSInteger focusFrameWH = screenXY.width / 320 * 220;//as wx
      int offet = 10;
      if (screenXY.height == 568)
          offet = 19;

      return CGRectMake((screenXY.width - focusFrameWH) / 2,
                        (screenXY.height - 64 - focusFrameWH - 83 - 50 - offet) / 2 +
64,
                        focusFrameWH,
                        focusFrameWH);
  }

  -(void)buildContainerView:(UIView*)containerView
  {
      // Customize the view of the scan box.
      UIView* bg = [[UIView alloc] initWithFrame:containerView.bounds];
      [containerView addSubview:bg];
      CGRect rect = [MPScanCodeViewController constructScanAnimationRect];
      UIView* view = [[UIView alloc] initWithFrame:rect];
      view.backgroundColor = [UIColor orangeColor];
      view.alpha = 0.5;
      [bg addSubview:view];
  }

  - (void)selectPhoto
  {
      [self scanPhotoLibrary];
  }
```

- Process scan results.

```
  -(void)didFind:(NSArray<TBScanResult*>*)resultArray
  {
      TBScanResult *result = resultArray.firstObject;
      NSString* content = result.data;
      if (result.resultType == TBScanResultTypeQRCode) {
          content = [NSString stringWithFormat:@"qrcode:%@, hiddenData:%@,
TBScanQRCodeResultType:%@", result.data, result.hiddenData, [result.extData objectFor
Key:TBScanResultTypeQRCode]];
          NSLog(@"subType is %@, ScanType_QRCode is %@", @(result.subType),
@(ScanType_QRCode));
      } else if (result.resultType == TBScanResultTypeVLGen3Code) {
          content = [NSString stringWithFormat:@"gen3:%@", result.data];
          NSLog(@"subType is %@, ScanType_GEN3 is %@", @(result.subType),
@(ScanType_GEN3));
      } else if (result.resultType == TBScanResultTypeGoodsBarcode) {
          content = [NSString stringWithFormat:@"barcode:%@", result.data];
          NSLog(@"subType is %@, EAN13 is %@", @(result.subType), @(EAN13));
      } else if (result.resultType == TBScanResultTypeDataMatrixCode) {
          content = [NSString stringWithFormat:@"dm:%@", result.data];
          NSLog(@"subType is %@, ScanType_DATAMATRIX is %@", @(result.subType),
@(ScanType_DATAMATRIX));
      } else if (result.resultType == TBScanResultTypeExpressCode) {
          content = [NSString stringWithFormat:@"express:%@", result.data];
          NSLog(@"subType is %@, ScanType_FASTMAIL is %@", @(result.subType),
@(ScanType_FASTMAIL));
      }
      dispatch_async(dispatch_get_main_queue(), ^{
          UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"" message:content
delegate:self cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
          alert.tag = 9999;
          [alert show];
      });
  }
```

- Continue to scan the code.

```
     - (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
(NSInteger)buttonIndex{
      // Continue to scan the code.
      [self resumeScan];
     }
```

- Set a callback that is triggered when the local album fails to be identified.

```
  - (void)scanPhotoFailed
  {
      // The callback that is triggered when the album fails to be identified
      NSLog(@"scanPhotoFailed");
  }
```

- Other callback processing

```
- (void)cameraPermissionDenied
{
    [self.navigationController popViewControllerAnimated:YES];
}

- (void)cameraDidStart
{
    NSLog(@"started!!");
}

-(void)setTorchState:(TorchState)bState
{
    NSLog(@"TorchState:%lu", (unsigned long)bState);
}

-(void)userTrack:(NSString*)name
{
    NSLog(@"userTrack:%@", name);
}

-(void)userTrack:(NSString*)name args:(NSDictionary*)data
{
    NSLog(@"userTrack:%@, args:%@", name, data);
}

- (void)scanPhotoFailed
{
    // The callback that is triggered when the album fails to be identified
    NSLog(@"scanPhotoFailed");
}
```

1.3.3. Multi-code recognition

This topic describes how to use the multi-code recognition feature of the scan SDK in the custom baseline `cp_change_28238` or baseline version 10.2.3.5 or later. You can integrate multi-code recognition SDK to iOS client based on native project with CocoaPods.

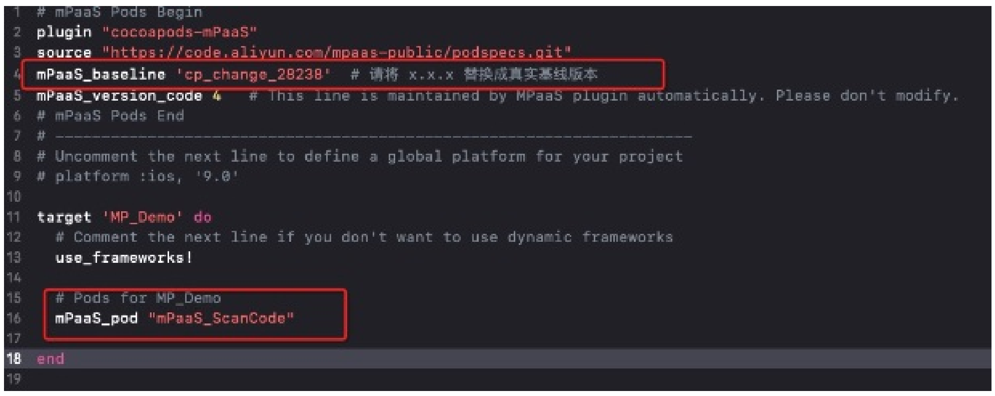
Prerequisites

You have connected your project to mPaaS. For more information, see Access based on native framework and using Cocoapods.

Add the SDK

Use CocoaPods plugin to add the multi-code recognition SDK. Complete the following steps:

1. Open the `Podfile` file, change mPaaS_baseline to `cp_change_28238` or baseline version 10.2.3.5 or later.
2. Run `mPaaS_pod "mPaaS_ScanCode"` to add dependencies for the scan component.



```
# mPaaS Pods Begin
plugin "cocoapods-mPaaS"
source "https://code.aliyun.com/mpaas-public/podspecs.git"
mPaaS_baseline 'cp_change_28238'  # 请将 x.x.x 替换成真实基线版本
mPaaS_version_code 4   # This line is maintained by MPaaS plugin automatically. Please don't modify.
# mPaaS Pods End
# ---------------------------------------------------------------------
# Uncomment the next line to define a global platform for your project
# platform :ios, '9.0'

target 'MP_Demo' do
  # Comment the next line if you don't want to use dynamic frameworks
  use_frameworks!

  # Pods for MP_Demo
  mPaaS_pod "mPaaS_ScanCode"

end
```

3. Click here to learn how to use CocoaPods. In the CLI, run `pod install` or `pod update` to add the SDK.

Use the SDK

This section describes how to use the multi-code recognition feature of the scan SDK in the customized baseline `cp_change_28238` or baseline version 10.2.3.5 or later. The official demo of the scan component is used for reference.

Open the default scan page

> **Note**
>
> The multi-code recognition feature is available only in a standard UI.

- Trigger the default scan page and process the scan results.

```
 #import <TBScanSDK/TBScanSDK.h>

 @interface MPScanDemoVC()

 @property(nonatomic, strong) TBScanViewController *scanVC;

 @end

- (void)defaultScan {

   // Define whether to display the entry to the album.
   [MPScanCodeAdapterInterface sharedInstance].shoulShowAlbum = NO;

   TBScanViewController *vc = [[MPScanCodeAdapterInterface sharedInstance]
createDefaultScanPageWithallback:^(id  _Nonnull result, BOOL keepAlive) {
       // Process the scan results.
       UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@""
message:result[@"resp_result"] delegate:self cancelButtonTitle:@"OK"
otherButtonTitles:nil, nil];
        alert.tag = 1999;
        [alert show];
    }];

     // Set the type of the scan.
     vc.scanType =  ScanType_Default_Code;

    [self.navigationController pushViewController:vc animated:YES];
    self.scanVC =  vc;
}
```

- Perform multi-code recognition and continuous code scan.

```
- (void)alertView:(UIAlertView *)alertView clickedButtonAtIndex:
(NSInteger)buttonIndex {
    // Enable continuous code scan.
    [self.scanVC resumeCaptureSession];
}
```

How to use the custom UI

This article will introduce how to use the multi-code recognition feature SDK under the custom UI in conjunction with the scan official demo.

Customize the ViewController that inherits TBScanViewController

Plaintext

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

NS_ASSUME_NONNULL_BEGIN

@interface MPScanCodeViewController :
TBScanViewController<TBScanViewControllerDelegate>

@end

NS_ASSUME_NONNULL_END
```

Initialize custom scan ViewController

Plaintext

```
//Custom scan entrance
- (void)customScanAction
{
    MPScanCodeViewController *vc = [[MPScanCodeViewController alloc]
initWithConfig:@{}];
    [self.navigationController pushViewController:vc animated:YES];
}
```

Plaintext

```
@implementation MPScanCodeViewController

- (instancetype)initWithConfig:(NSDictionary *)config
{
    if (self = [super initWithConfig:config])
    {
        self.delegate = self;
        self.scanType = ScanType_All_Code;
    }
    return self;
}
```

Important

The ViewController that initializes the custom scan code can only use the `- (instancetype)initWithConfig:(NSDictionary *)config;` method.

Customize the scan box

Plaintext

```
- (void)buildContainerView:(UIView*)containerView
{
    // Customize the scan box view
    UIView* bg = [[UIView alloc] initWithFrame:containerView.bounds];
    [containerView addSubview:bg];
    CGRect rect = [MPScanCodeViewController constructScanAnimationRect];
    UIView* view = [[UIView alloc] initWithFrame:rect];
    view.backgroundColor = [UIColor orangeColor];
    view.alpha = 0.5;
    [bg addSubview:view];
}
```

Handle scan results

Users handle it according to their own business scenarios.

Plaintext

```
#pragma mark TBScanViewControllerDelegate

-(void)didFind:(NSArray<TBScanResult*>*)resultArray
{
    TBScanResult *result = resultArray.firstObject;
    NSString* content = result.data;
    if (result.resultType == TBScanResultTypeQRCode) {
        content = [NSString stringWithFormat:@"qrcode:%@, hiddenData:%@, 
TBScanQRCodeResultType:%@", result.data, result.hiddenData, [result.extData 
objectForKey:TBScanResultTypeQRCode]];
        NSLog(@"subType is %@, ScanType_QRCode is %@", @(result.subType), 
@(ScanType_QRCode));
    } else if (result.resultType == TBScanResultTypeVLGen3Code) {
        content = [NSString stringWithFormat:@"gen3:%@", result.data];
        NSLog(@"subType is %@, ScanType_GEN3 is %@", @(result.subType), 
@(ScanType_GEN3));
    } else if (result.resultType == TBScanResultTypeGoodsBarcode) {
        content = [NSString stringWithFormat:@"barcode:%@", result.data];
        NSLog(@"subType is %@, EAN13 is %@", @(result.subType), @(EAN13));
    } else if (result.resultType == TBScanResultTypeDataMatrixCode) {
        content = [NSString stringWithFormat:@"dm:%@", result.data];
        NSLog(@"subType is %@, ScanType_DATAMATRIX is %@", @(result.subType), 
@(ScanType_DATAMATRIX));
    } else if (result.resultType == TBScanResultTypeExpressCode) {
        content = [NSString stringWithFormat:@"express:%@", result.data];
        NSLog(@"subType is %@, ScanType_FASTMAIL is %@", @(result.subType), 
@(ScanType_FASTMAIL));
    }
    dispatch_async(dispatch_get_main_queue(), ^{
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"" message:content del
egate:self cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
        alert.tag = 9999;
        [alert show];
    });
}
```

1.4. FAQ

Is there a charge to use the Scan component?

There is no charge for the accessing the Scan component. However, in the process of using the Scan component, logs will be collected to obtain the number total scans, successful scans, failed scans, and other information, so as to realize monitoring and analysis on the scan performance. The log collection process relies on the Mobile Analysis Service which is chargable, so certain fees may arise.

Log tracking of the Scan component is configured at the initial stage. By default, log reporting is enabled. If you need to disable log reporting, see Upload logs.

How to initialize mPaaS in case of integrating Scan to project through native AAR or mPaaS Inside?

You need to add the following code to the Application class.

```
public class MyApplication extends Application {

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

For more details, see Initialize mPaaS.

How to process the lag (ANR) that occurs when code scanning starts in the Android baseline 10.1.68?

In the native AAR and the mPaaS Inside modes, if you have referenced other components in addition to the code scanning component, you must initialize mPaaS. Otherwise, ANR may occur on the main thread.