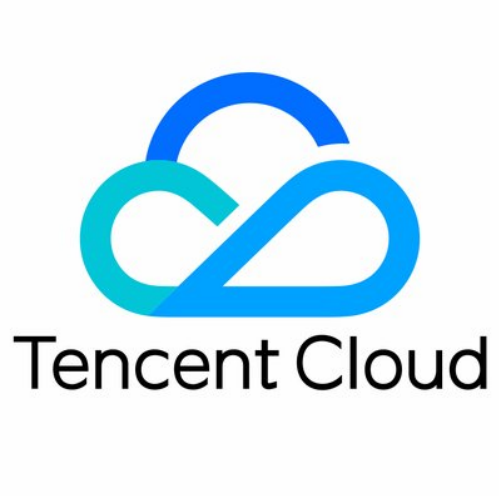
Tencent Effect SDK

SDK Integration Guide（Including UI）

Product Documentation



Contents

SDK Integration Guide（Including UI）
General Integration of Tencent Special Effects
iOS

Last updated：2025-04-27 17:31:55

Features

TEBeautyKit is the UI panel library for the Tencent Effect module, used for quick and convenient usage and management of the effect features. The effect is as shown in the figure below:

Integration Steps

1. Download and unzip TEBeautyKit.
2. Copy the TEBeautyKit folder into your project and make it the same level as the podfile directory.
3. Edit the podfile, adding the following code:

```
pod 'TEBeautyKit',:path => 'TEBeautyKit/TEBeautyKit.podspec'
```

User Guide

1. Effect Authentication

After the app is launched, effect authentication needs to be performed once to use the effect feature normally.

API:

```
TEBeautyKit.h
+ (void)setTELicense:(NSString *)url key:(NSString *)key completion:(callback _Null
```

Example:

```
[TEBeautyKit setTELicense:@"your license" key:@"your key" completion:^(NSInteger au
  NSLog(@"----------result: %zd  %@",authresult,errorMsg);
}];
```

2. Set the Effect Material Path

The effect data on the effect panel are all parsed from the json file in the material path set here.

API:

```
#import "TEUIConfig.h"
/// Configure the beauty panel settings according to the beauty package options.
/// - Parameter panelLevel: Beauty Package
/// Beauty Package Types: A1_00,A1_01,A1_02,A1_03,A1_04,A1_05,A1_06,S1_00,S1_01,S1_
-(void)setPanelLevel:(TEPanelLevel)panelLevel;
```

Example:

```
[[TEUIConfig shareInstance] setPanelLevel:S1_07];
```

3. Initialize and add TEPanelView

```
-(TEPanelView *)tePanelView{
    if (!_tePanelView) {
        _tePanelView = [[TEPanelView alloc] init:nil comboType:nil];
        _tePanelView.delegate = self;
    }
    return _tePanelView;
}
[self.view addSubview:self.tePanelView];
[self.tePanelView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.width.mas_equalTo(self.view);
    make.centerX.mas_equalTo(self.view);
    make.height.mas_equalTo(250);
    make.bottom.mas_equalTo(self.view.mas_bottom);
}];
```

4. Create an effect object

API:

```
//Create a TEBeautyKit object with effectMode options: EFFECT_MODE_NORMAL (High-Per
+ (void)createXMagic:(EffectMode)effectMode onInitListener:(OnInitListener _Nullabl
//Create the TEBeautyKit object, do not enable the High Performance Mode
+ (void)create:(OnInitListener _Nullable )onInitListener;
```

Example:

```
- (void)initXMagic {
    __weak __typeof(self)weakSelf = self;
    [TEBeautyKit createXMagic:EFFECT_MODE_PRO onInitListener:^(TEBeautyKit * _Nulla
        __strong typeof(self) strongSelf = weakSelf;
        strongSelf.teBeautyKit = api;
        strongSelf.tePanelView.teBeautyKit = strongSelf.teBeautyKit;
        [strongSelf.tePanelView setDefaultBeauty];
        [strongSelf.teBeautyKit setLogLevel:YT_SDK_ERROR_LEVEL];
```

```
        //  [strongSelf.teBeautyKit registerSDKEventListener:strongSelf];
    }];
}
```

5. Processing video data

API:

```
/**
textureId: Texture ID
textureWidth: Texture width
textureHeight: Texture height
origin: Enumeration value (YtLightImageOriginTopLeft, YtLightImageOriginBottomLeft)
orientation: Enumeration value for image rotation angle
*/
- (YTProcessOutput *)processTexture:(int)textureId
         textureWidth:(int)textureWidth
        textureHeight:(int)textureHeight
           withOrigin:(YtLightImageOrigin)origin
      withOrientation:(YtLightDeviceCameraOrientation)orientation
```

Example:

```
#pragma mark - TRTCVideoFrameDelegate
- (uint32_t)onProcessVideoFrame:(TRTCVideoFrame *_Nonnull)srcFrame dstFrame:(TRTCVi
    if(!_teBeautyKit){
        [self initXMagic];
    }
    YTProcessOutput *output = [self.teBeautyKit processTexture:srcFrame.textureId
    textureWidth:srcFrame.width textureHeight:srcFrame.height
    withOrigin:YtLightImageOriginTopLeft withOrientation:YtLightCameraRotation0];
    dstFrame.textureId = output.textureData.texture;
    return 0;
}
```

6. Destroy effect

```
- (void)destroyXMagic{
   [self.teBeautyKit onDestroy];
   self.teBeautyKit=nil;
}
```

Appendix

Panel JSON File Description

Beauty, Body Shaping.

Field	Description
displayName	Chinese Name
displayNameEn	English Name
icon	Image address, supports setting local images and network images. Local images support assets resources and SD resources. Assets images are as shown in the image above. For SD card images, set the full path of the image. For network images, set the corresponding HTTP link.
sdkParam	The effect SDK requires four properties. Refer to the Effect Parameters table
effectName	Effect attribute key, refer to the Effect Parameters table
effectValue	Setting the attribute intensity, refer to the Effect Parameters table
resourcePath	Setting the resource path, refer to the Effect Parameters table
extraInfo	Setting other information, refer to the Effect Parameters table

Filters, Animated Stickers and Segmentation.

Since the configuration of **Filters, Animated Stickers and Segmentation** is primarily identical, the JSON for filters is used here for illustration. The fields downloadPath and resourceUri are added here.

Field	Description
downloadPath	If your filter material is downloaded from the network, then the configuration here is the location of your material stored locally after download, which is a **relative path**, and the full path is set in `TEDownloader.h` using `basicPath` **+the path set here**
resourceUri	If your material needs to be downloaded via network, configure the network address here, as in the third red box in the image above. However, if your filter material is local, configure the corresponding local address according to the figure above.

Makeup

In the Makeup, the `makeupLutStrength` field is added under `extraInfo` . This field is used to adjust the strength of the filter in the makeup material (if this makeup material supports adjusting the filter strength, configure it accordingly). This field can be referenced in the Effect Parameters table.

TEBeautyKit Method Descriptions

```
//Create a TEBeautyKit object with effectMode options: EFFECT_MODE_NORMAL (High-Per
+ (void)createXMagic:(EffectMode)effectMode onInitListener:(OnInitListener _Nullabl
//Create TEBeautyKit object, do not enable the High Performance Mode
+ (void)create:(OnInitListener _Nullable )onInitListener;
/**
 Create TEBeautyKit object
 isEnableHighPerformance: Whether to enable the High Performance Mode
 When the High Performance Mode is enabled, the effect feature consumes fewer syste
 Note: After enabling the High Performance Mode, the following effect options will
  1. Eyes: Eye width, eye height, eye bag removal
  2. Eyebrows: Angle, distance, height, length, thickness, eyebrow peak
  3. Lips: Smile lip
  4. Face: Slim face (natural, woman, man), chin tuck, wrinkle removal, smile lines
 */
+ (void)create:(BOOL)isEnableHighPerformance onInitListener:(OnInitListener _Nullab
// Effect Authentication
+ (void)setTELicense:(NSString *)url key:(NSString *)key completion:(callback _Null
// Setting effect object
- (void)setXMagicApi:(XMagic *_Nullable)xmagicApi;
// Mute Effect
- (void)setMute:(BOOL)isMute;
/**
 *  Set the toggle of a certain feature
 *
 * @param featureName Values see XmagicConstant.FeatureName
 * @param enable      true indicates enable, false indicates disable
 */
- (void)setFeatureEnableDisable:(NSString *_Nullable)featureName enable:(BOOL)enabl
/// Configure Frame Synchronization Mode
/// @isSync Indicates whether synchronization is enabled
/// @syncFrameCount The number of frames to synchronize. A value of -1 denotes no l
- (void)setSyncMode:(BOOL)isSync syncFrameCount:(int)syncFrameCount;
// Processing image beautification
- (UIImage *_Nullable)processUIImage:(UIImage *_Nullable)inputImage
                  imageWidth:(int)imageWidth
                 imageHeight:(int)imageHeight
                   needReset:(bool)needReset;
// Processing texture
- (YTProcessOutput *_Nullable)processTexture:(int)textureId
          textureWidth:(int)textureWidth
         textureHeight:(int)textureHeight
            withOrigin:(YtLightImageOrigin)origin
       withOrientation:(YtLightDeviceCameraOrientation)orientation;
// Processing CVPixelBufferRef
- (YTProcessOutput * _Nullable)processPixelData:(CVPixelBufferRef _Nullable )pixelD
```

```
                      pixelDataWidth:(int)pixelDataWidth
                     pixelDataHeight:(int)pixelDataHeight
                          withOrigin:(YtLightImageOrigin)origin
                     withOrientation:(YtLightDeviceCameraOrientation)orientation;
// Set effect
- (void)setEffect:(TESDKParam *_Nullable)sdkParam;
// Setting effect list
- (void)setEffectList:(NSArray<TESDKParam *>*_Nullable)sdkParamList;
//Is the Beauty Enhancement Mode Enabled
- (BOOL)isEnableEnhancedMode;
// Whether to enable the Enhanced Mode
- (void)enableEnhancedMode:(BOOL)enable;
// Access the current effect parameters
- (NSString *_Nullable)exportInUseSDKParam;
// Access saved effect parameters. Upon re-entering the Effect mode, calling setEff
- (NSMutableArray<TESDKParam *> *_Nonnull)getInUseSDKParamList;
// Restore the effect
- (void)onResume;
// Pause the effect
- (void)onPause;
// Destroy the effect
- (void)onDestroy;
// Access current texture image
- (void)exportCurrentTexture:(void (^_Nullable)(UIImage * _Nullable image))callback
// Set the log
- (void)setLogLevel:(YtSDKLoggerLevel)level;
// Set AIDataListener
- (void)setAIDataListener:(id<TEBeautyKitAIDataListener> _Nullable)listener;
// Set TipsListener
- (void)setTipsListener:(id<TEBeautyKitTipsListener> _Nullable)listener;
// Save the effect parameters
- (void)saveEffectParam:(TESDKParam *_Nonnull)sdkParam;
// Delete a saved effect data
- (void)deleteEffectParam:(TESDKParam *_Nonnull)sdkParam;
// Clear saved effect parameters
- (void)clearEffectParam;
```

TEUIConfig Descriptions.

```
// Color of the following properties can be modified externally
// Background color of the effect panel
@property(nonatomic,strong)UIColor *panelBackgroundColor;
// Dividing line color
@property(nonatomic,strong)UIColor *panelDividerColor;
// Selected item color
@property(nonatomic,strong)UIColor *panelItemCheckedColor;
```

```
// Text color
@property(nonatomic,strong)UIColor *textColor;
// Selected text color
@property(nonatomic,strong)UIColor *textCheckedColor;
// Progress bar color
@property(nonatomic,strong)UIColor *seekBarProgressColor;

/// Configure the beauty panel settings according to the beauty package options.
/// - Parameter panelLevel: Beauty Package
/// Beauty Package Types: A1_00,A1_01,A1_02,A1_03,A1_04,A1_05,A1_06,S1_00,S1_01,S1_
-(void)setPanelLevel:(TEPanelLevel)panelLevel;

/**
 configure the effect parameters
 beauty: Effect json path
 beautyBody: Body shaping json path
 lut: Filter json path
 motion: Animated effects json path
 makeup: Makeup json path
 segmentation: Background segmentation json path
 lightMakeup: The Path to Light and Elegant Makeup
 */
-(void)setTEPanelViewRes:(NSString *)beauty
beautyBody:(NSString *)beautyBody
lut:(NSString *)lut
motion:(NSString *)motion
makeup:(NSString *)makeup
segmentation:(NSString *)segmentation
lightMakeup:(NSString *)lightMakeup;
```

TEPanelView Description.

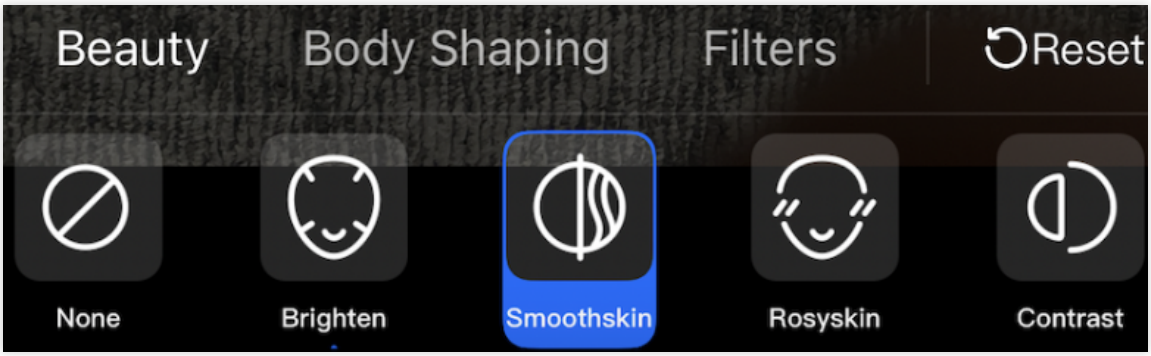
```
// Initialize the effect panel, fill in nil for abilityType and comboType.
- (instancetype)init:(NSString *)abilityType comboType:(NSString *)comboType;

@protocol TEPanelViewDelegate <NSObject>
// Callback after setting the effect
- (void)setEffect;
@end
```

Android

Last updated：2024-11-22 13:23:19

Feature Overview

To facilitate customers' quick integration of the beauty effect and simplify the development of UI panels, we provide the beauty effect UI component: TEBeautyKit. It includes further encapsulation of the SDK and customizable UI panels, as shown in the figure below. If you do not need to use this UI, refer to No UI integration of Tencent effects.



Demo Project: **TEBeautyDemo**

Clone the demo project from GitHub, follow the instructions in the TEBeautyDemo/README document to run TEBeautyDemo, and then refer to this document for detailed steps on integrating the SDK with UI.

Integrate TEBeautyKit

Note:

This library only supports **Tencent Effect SDK V3.5.0** and later versions.

Step 1: Add TencentEffectSDK dependency

Integrate TencentEffect SDK

Integrate TencentEffcet Resources

Step 2: Add TEBeautyKit dependency

Source Code Integration (Recommended)

Maven Integration

Downloading AAR Integration

Copy the TEBeautyKit module from the demo project into your project, and modify the SDK package type and version number in `tebeautykit/build.gradle` to match the SDK package type and version number in Step 1.

```
implementation 'com.tencent.mediacloud:TencentEffect_S1-04:SDK version number'
```

Then add the dependency on the TEBeautyKit module in the relevant module of your app:

```
implementation project(':tebeautykit')
```

Add the dependency on the TEBeautyKit library in your app's dependencies. The SDK version number should match the version number in Step 1.

```
dependencies{
    ...
    implementation 'com.tencent.mediacloud:TEBeautyKit:SDK version number'
}
```

Since TEBeautyKit also depends on components like Gson and OkHttp, you need to add the following dependencies:

```
dependencies{
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:okhttp:3.9.0'
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    implementation 'androidx.appcompat:appcompat:1.0.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
}
```

Download TEBeautyKit (The downloaded file will be a zip file. Extract it to get the AAR file.)

Copy the `tebeautykit-xxxx.aar` file to the app project's `libs` directory.

Open the app module's `build.gradle` and add the dependency reference:

```
dependencies{
    ...
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])
}
```

Since TEBeautyKit also depends on components like Gson and OkHttp, you need to add the following dependencies:

```
dependencies{
    implementation 'com.google.code.gson:gson:2.8.2'
    implementation 'com.squareup.okhttp3:okhttp:3.9.0'
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    implementation 'androidx.appcompat:appcompat:1.0.0'
```

```
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'androidx.recyclerview:recyclerview:1.2.1'
}
```

Step 3: Add the panel JSON file

Obtain the panel configuration file from the demo project's demo/src/main/assets/beauty_panel, or click here to download and unzip it. The file includes configurations for beauty, body beauty, filters, dynamic effect stickers, and segmentation properties. Select a set of JSON files according to your package type and place them in the `assets/beauty_panel` folder in your project (the **panel_icon** folder also needs to be placed in this directory).

The downloaded archive contains the files as shown above. Each package name corresponds to several JSON files. The table below explains each JSON file:

File	Description
beauty.json	Configuration file for beauty effect, shaping, image adjustment, etc.
beauty_body..json	Configuration file for body beauty

lut.json	Filter configuration file. Note: Since different customers use different filter materials, customers can configure according to the JSON structure after downloading (refer to JSON File Structure Description) and delete the default configuration.
makeup.json	Configuration file for full-face makeup style. Note: Since different customers use different materials for full-face makeup style, customers can configure according to the JSON structure after downloading (refer to JSON File Structure Description).
motions.json	Dynamic effect sticker configuration file. Note: Since different customers use different dynamic effect sticker materials, customers can configure according to the JSON structure after downloading (refer to JSON File Structure Description).
segmentation.json	Background segmentation (virtual background) configuration file. Note: Since different customers use different segmentation materials, customers can configure according to the JSON structure after downloading (refer to JSON File Structure Description).
panel_icon	This folder is used for storing images configured in the JSON file, and must be added.

How to Use TEBeautyKit

We strongly recommend that you refer to the `TEMenuActivity.java` and `TECameraBaseActivity.java` of the demo project's to understand how to integrate TEBeautyKit.

Step 1: Authenticate

1. Apply for authorization to obtain the license URL and license key. Please refer to the License Guide.
2. Set the URL and key in the initialization code of the related business module to **trigger license download to avoid downloading temporarily before use**. For example, in our demo project, the download is triggered in the onCreate method of the application. However, we do not recommend triggering it here in your project because network permission might not be available or there could be a high rate of network failure at this time. Please choose a more appropriate time to trigger the license download.

```
// If you only want to trigger the download or update the license without caring ab
//TEApplication.java
TEBeautyKit.setTELicense(context, LicenseConstant.mXMagicLicenceUrl, LicenseConstan
```

3. Perform authentication before actually using the beauty feature (e.g., before launching the camera):

```
//TEMenuActivity.java
TEBeautyKit.setTELicense(context, LicenseConstant.mXMagicLicenceUrl, LicenseConstan

        @Override
        public void onLicenseCheckFinish(int errorCode, String msg) {
            // Note: This callback may not be on the calling thread.
```

```
            if (errorCode == TELicenseCheck.ERROR_OK) {
                // Authentication succeeded.
            } else {
                // Authentication failed.
            }
        }
    });
```

Authentication error code description:

Error Code	Description
0	Success
-1	Invalid input parameters, such as empty URL or KEY
-3	Download failure. Please check your network settings.
-4	TE authorization information read locally is empty, possibly due to I/O failure.
-5	Content of the VCUBE TEMP license file is empty, possibly due to I/O failure.
-6	JSON fields in the v_cube.license file are incorrect. Please contact the Tencent Cloud team to deal with it.
-7	Signature verification failed. Please contact the Tencent Cloud team to deal with it.
-8	Decryption failed. Please contact the Tencent Cloud team to deal with it.
-9	JSON fields in the TELicense field are incorrect. Please contact the Tencent Cloud team to deal with it.
-10	The TE authorization information parsed from the network is empty. Please contact the Tencent Cloud team to deal with it.
-11	Failed to write TE authorization information to a local file, possibly due to I/O failure
-12	Download failed, and parsing local assets also failed.
-13	Authentication failed. Please check if the .so file is in the package or if the .so path is set correctly.
3004/3005	Invalid authorization. Please contact the Tencent Cloud team to deal with it.
3015	Bundle ID/Package Name mismatch. Check if the Bundle ID/Package Name used by your app is consistent with the applied one, and ensure you are using the correct license file.
3018	The license file has expired. You need to apply for a renewal from Tencent Cloud.

Others	Please contact the Tencent Cloud team to deal with it.

Step 2: Set the path

```
    //TEMenuActivity.java
  String resPath = new File(getFilesDir(), AppConfig.getInstance().getBeautyFileDir
  TEBeautyKit.setResPath(resPath);
```

Step 3: Copy resources

The resource files mentioned here consist of two parts:

The SDK model files, located in the `assets` directory of the SDK's AAR package.

Filter and dynamic effect resource files, located in the `assets` directory of the demo project, named `lut` and `MotionRes` respectively.

Before using the beauty effect, you need to copy the above resources to the resPath set in Step 2. If the SDK version is not updated, you only need to copy it once. After a successful copy, you can record it in the SharedPreference of the app, so that you do not need to copy it again next time. For details, refer to the `copyRes` method of `TEMenuActivity.java` in the demo project.

```
//TEMenuActivity.java
private void copyRes() {
    if (!isNeedCopyRes()) {
        return;
    }
    new Thread(() -> {
        TEBeautyKit.copyRes(getApplicationContext());
    }).start();
}
```

Step 4: Initialize TEBeautyKit and add the view to the page

```
//TECameraBaseActivity.java
TEBeautyKit.create(this.getApplicationContext(), beautyKit -> {
    mBeautyKit = beautyKit;
    initBeautyView(beautyKit);
});

public void initBeautyView(TEBeautyKit beautyKit){

    TEPanelViewResModel resModel = new TEPanelViewResModel();
    String combo = "S1_07";   //Set according to your package. If your package does
    resModel.beauty = "beauty_panel/"+combo+"/beauty.json";
    resModel.lut = "beauty_panel/"+combo+"/lut.json";
```

```
    resModel.beautyBody = "beauty_panel/"+combo+"/beauty_body.json";
    resModel.motion = "beauty_panel/"+combo+"/motions.json";
    resModel.lightMakeup = "beauty_panel/"+combo+"/light_makeup.json";
    resModel.segmentation = "beauty_panel/"+combo+"/segmentation.json";
    TEUIConfig.getInstance().setTEPanelViewRes(resModel);

    mTEPanelView = new TEPanelView(this);
    mTEPanelView.setTEPanelViewCallback(this);
    mTEPanelView.setupWithTEBeautyKit(beautyKit);
    mTEPanelView.showView(this);
    this.mPanelLayout.addView(mTEPanelView, new LinearLayout.LayoutParams(ViewGroup
}
```

The parameter in the `TEUIConfig.getInstance().setTEPanelViewRes` method is the JSON file in the src/main/assets/beauty_panel mentioned above. If you do not need a particular item, you can input null at the corresponding position.

Step 5: Use beauty effect

We assume you have already implemented the camera application, can start the camera normally, and can call back the camera's SurfaceTexture texture information to Activity for beauty effect processing, as shown below:

```
//TECameraBaseActivity.java
@Override
public int onCustomProcessTexture(int textureId, int textureWidth, int textureHeigh
   // The beauty effect SDK processes the textureId here, adds beauty effects and s
}
```

If you have not yet implemented the camera application, you can refer to `TECameraBaseActivity.java` in the demo project and use the `GLCameraXView` component to add it to your Activity's layout for quick camera preview:

```
<com.tencent.demo.camera.camerax.GLCameraXView
   android:id="@+id/te_camera_layout_camerax_view"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   app:back_camera="false"
   app:surface_view="false"
   app:transparent="true" />
```

The beauty effect SDK processes each frame of data and returns the corresponding processing results. For detailed information on the process method, see the API Documentation.

```
//TECameraBaseActivity.java
@Override
public int onCustomProcessTexture(int textureId, int textureWidth, int textureHeigh
```

```
    return mBeautyKit.process(textureId, textureWidth, textureHeight);
}
```

Step 6: Manage the lifecycle

Lifecycle method onResume: It is recommended to call it in the `Activity` 's `onResume()` method. When called, it will resume the sound in the effects.

```
mBeautyKit.onResume();
```

Lifecycle method onPause: It is recommended to call it in the Activity's `onPause()` method. When called, it will pause the sound in the effects.

```
mBeautyKit.onPause();
```

Beauty effect release SDK: It is called when the OpenGL environment is being terminated. **It needs to be called in the GL thread, and cannot be called in the main thread (Activity's onDestroy);** otherwise, it may cause resource leaks and result in a white or black screen after multiple entries and exits.

```
@Override
public void onGLContextDestroy() {
    mBeautyKit.onDestroy();
}
```

Step 7: Export and import beauty effect parameters

Export the current beauty effect parameters and save them in SharedPreference:

```
String json = mBeautyKit.exportInUseSDKParam();
if (json != null) {
    getSharedPreferences("demo_settings", Context.MODE_PRIVATE).edit().
      putString("current_beauty_params", json).commit();
}
```

Next time when initializing TEBeautyKit, import this string to modify the default beauty effect:

```
public void initBeautyView(TEBeautyKit beautyKit){
    TEUIConfig.getInstance().setTEPanelViewRes( beauty: "beauty_panel/S1_07/beauty.json",  beautyBody: "beauty_
            lut: "beauty_panel/S1_07/lut.json",  motion: "beauty_panel/S1_07/motions.json",
            makeup: "beauty_panel/S1_07/makeup.json",  segmentation: "beauty_panel/S1_07/segmentation.json");
    mTEPanelView = new TEPanelView( context: this);
    mTEPanelView.setTEPanelViewCallback(this);
    mTEPanelView.setupWithTEBeautyKit(beautyKit);

    SharedPreferences sp = getSharedPreferences( name: "demo_settings", Context.MODE_PRIVATE);
    String savedParams = sp.getString( s: "current_beauty_params",  s1: "");
    if (!savedParams.isEmpty()) {
        mTEPanelView.setLastParamList(savedParams);
    }

    mTEPanelView.showView( tePanelViewCallback: this);
    this.mPanelLayout.addView(mTEPanelView, new LinearLayout.LayoutParams(ViewGroup.LayoutParams.MATCH_PAR
}
```

Appendix

Panel JSON File Description

Beauty effect and body beauty

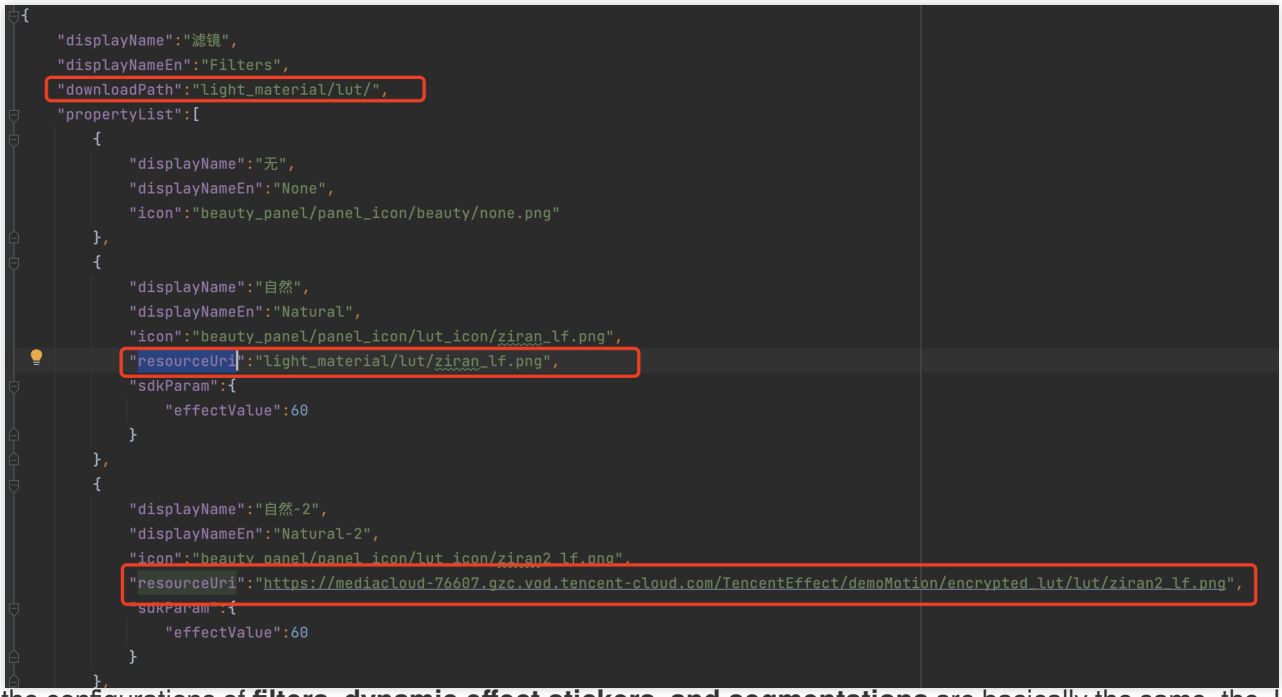
ld.gradle (:tebeautykit) × beauty.json ×

```
{
    "displayName":"美颜",
    "displayNameEn":"Beauty effects",
    "propertyList":[
        {
            "displayName":"关闭",
            "displayNameEn":"None",
            "icon":"beauty_panel/panel_icon/beauty/none.png"
        },
        {
            "displayName":"美白",
            "displayNameEn":"Brighten",
            "icon":"beauty_panel/panel_icon/beauty/beauty_whiten.png",
            "propertyList":[...],
            "uiState":2
        },
        {
            "displayName":"磨皮",
            "displayNameEn":"Smoothskin",
            "icon":"beauty_panel/panel_icon/beauty/beauty_smooth.png",
            "sdkParam":{
                "effectName":"smooth.smooth",
                "effectValue":40
            },
            "uiState":1
        },
        {
            "displayName":"红润",
            "displayNameEn":"Rosyskin",
            "icon":"beauty_panel/panel_icon/beauty/beauty_ruddy.png",
            "sdkParam":{
                "effectName":"smooth.rosy",
                "effectValue":0
            }
        }
    ],
    "uiState":2
}
```

Field	Description
displayName	Chinese name
displayNameEn	English name
icon	Image address, supports local images and network images. Local images support assets resources and SD resources. For asset images, refer to the image above. For SD card images, set the full paths of the images. For network images, set the corresponding HTTP links.
sdkParam	Properties required by the beauty effect SDK, including four properties. Refer to the beauty effect parameter table.
effectName	Beauty property key. Refer to the Property Parameter Table.
effectValue	Sets property strength. Refer to the Property Parameter Table.
resourcePath	Sets the resource path. Refer to the Property Parameter Table.

extraInfo	Sets other information. Refer to the Property Parameter Table.

Filters, dynamic effect stickers, and segmentations



Since the configurations of **filters, dynamic effect stickers, and segmentations** are basically the same, the JSON for filters is explained here. The `downloadPath` and `resourceUri` fields are added.

Field	Description
downloadPath	If your filter materials are downloaded from the network, the local storage location for the download is configured here. This is a **relative path**, with the full path being the one set in `TEBeautyKit.setResPath` + **the path set here**.
resourceUri	If your material needs to be downloaded via a network, the network address is configured here, as in the third red box in the image above. However, if your filter material is local, configure the corresponding local address according to the image above.

Full-face makeup style

An additional field `makeupLutStrength` under `extraInfo` has been included in the full-face makeup style. This field is used to moderate the **intensity of filters** in the full-face makeup style (configure this if the full-face makeup style supports filter intensity modulation). Refer to the beauty effect parameter table for this field.

TEBeautyKit Method Description

```
//Asynchronously create the TEBeautyKit object.
public static void create(@NonNull Context context, @NonNull OnInitListener initLis

//isEnableHighPerformance: Whether to enable high-performance mode
//After the high-performance mode is enabled, the system CPU/GPU resources occupied
//However, please note: after the high-performance mode is enabled, the following b
//1. Eye: eye width, eye height, and elimination of eye bags
//2.Eyebrow: angle, distance, height, length, thickness, and eyebrow peak
//3. Mouth: smile lip
//4. Face: slim face (natural, goddess, and handsome), tighten lower jaw, remove wr
public static void create(@NonNull Context context, boolean isEnableHighPerformance

//Constructor method of TEBeautyKit, used for synchronously creating the TEBeautyKi
public TEBeautyKit(Context context)

/**
 * @param context                 Application context
 * @param isEnableHighPerformance: Whether to enable the high-performance mode
 */
public TEBeautyKit(Context context, boolean isEnableHighPerformance)

/**
 * Set the mute state.
```

```
     *
     * @param isMute: true indicates the mute state.
     */
    public void setMute(boolean isMute)

    /**
     * If you are carrying out beauty processing on pictures, you need to call this met
     * Camera data source: XmagicApi.PROCESS_TYPE_CAMERA_STREAM   Picture data source:
     * @param type: The default type is a video stream.
     */
    public void setBeautyStreamType(int type)

    /**
     * Enable or disable a specific feature.
     *
     * @param featureName: Refer to XmagicConstant.FeatureName for values.
     * @param enable      true indicates to enable; false indicates to disable.
     */
    public void setFeatureEnableDisable(String featureName, boolean enable)

    /**
     * Perform beauty processing on the image.
     *
     * @param bitmap
     * @param needReset
     * @return
     */
    public Bitmap process(Bitmap bitmap, boolean needReset)

    /**
     * Process each frame of video/camera data.
     *
     * @param textureId: The ID of the texture. This texture requires a texture type of
     * @param width     Texture width
     * @param height    Texture height
     * @return: processed texture ID
     */
    public int process(int textureId, int width, int height)

    /**
     * Update beauty properties.
     *
     * @param paramList
     */
    public void setEffectList(List<TEUIProperty.TESDKParam> paramList)
```

```
/**
 * Update beauty properties.
 *
 * @param teParam
 */
public void setEffect(TEUIProperty.TESDKParam teParam)

/**
 * Enable or disable the enhanced mode.
 *
 * @param enableEnhancedMode: true indicates to enable the enhanced mode; false ind
 * @return: A returned value of true indicates a state change has occurred; and a r
 */
public boolean enableEnhancedMode(boolean enableEnhancedMode)

/**
 * Obtain the currently active beauty properties list string.
 * Clients can save the exported string locally and call the `setLastParamList` met
 * @return
 */
public String exportInUseSDKParam()

/**
 * Used to restore the sound in stickers.
 * Restore the gyroscope sensor.
 */
public void onResume()

/**
 * Used to pause the sound in stickers.
 * Pause the gyroscope sensor.
 */
public void onPause()

/**
 * Terminate the beauty effect.
 * Note: This method must be called in the GL thread.
 */
public void onDestroy()

/**
 * Set event listener for listening to smartphone direction events for the adapter.
 * @param listener Event listener callback
 */
public void setEventListener(EventListener listener)
```

```
/**
 * Set setAIDataListener callback.
 *
 * @param aiDataListener
 */
public void setAIDataListener(XmagicApi.XmagicAIDataListener aiDataListener)

/**
 * Set dynamic effect tips callback function to display tips on the front-end page.
 *
 * @param tipsListener
 */
public void setTipsListener(XmagicApi.XmagicTipsListener tipsListener)

/**
 * Capture the current texture screen.
 *
 * @param callback
 */
public void exportCurrentTexture(XmagicApi.ExportTextureCallback callback)

/**
 * Set the degree of anti-clockwise rotation of the texture.
 * Primary feature: Used for texture rotation inside the SDK. After rotation of a c
 * By default, inside the SDK, a sensor is used to obtain the required degree of ro
 *
 * @param orientation: Its value can only be 0, 90, 180, or 270.
 */
public void setImageOrientation(TEImageOrientation orientation)

/**
 * Check whether the current device supports this material.
 *
 * @param motionResPath: the path to the material file
 * @return
 */
public boolean isDeviceSupport(String motionResPath)

/**
 * Obtain the on-and-off state of the beauty effect.
 *
 * @return EffectState
 */
public EffectState getEffectState()

/**
 * Set whether to enable the beauty effect.
```

```
     *
     * @param effectState ENABLED indicates enabled; DISABLED indicates disabled.
     */
    public void setEffectState(EffectState effectState)

    /**
     * Set the policy implementation class for the enhanced mode. If not set, the defau
     *
     * @param teParamEnhancingStrategy: Enhanced mode processing class.
     */
    public void setParamEnhancingStrategy(TEParamEnhancingStrategy teParamEnhancingStra



    //The following are the static methods.

    /**
     * Set the path for storing beauty resources.
     *
     * @param resPath
     */
    public static void assignResPath(String resPath)

    /**
     * Unpack resource files from the APK's assets to a specified path. First, you need
     * This should be called once after the first installation of the app, or after an
     * Copy xmagic resource from assets to a local path.
     */
    public static boolean copyRes(Context context)

    /**
     * Perform beauty authorization check.
     * Note: When this method is used, if the callback interface is not set, authorizat
     * Therefore, you can refer to the demo and call it in the application without sett
     * @param context                Application context
     * @param licenseKey             LicenseKey applied for on the platform
     * @param licenseUrl             License URL applied for on the platform
     * @param teLicenseCheckListener: Authorization callback interface
     */
    public static void setTELicense(Context context, String licenseUrl, String licenseK
```

TEUIConfig Description

Modify panel colors

Global modification: After obtaining the object through `TEUIConfig.getInstance()` , you can adjust the panel colors by modifying the following fields.

Partial modification: Using the `new TEUIConfig` object, adjust the panel colors by modifying the following fields, and update the panel style using the `TEPanelView.updateUIConfig` method.

```
@ColorInt
public int panelBackgroundColor = 0x66000000;  //Default background color
@ColorInt
public int panelDividerColor = 0x19FFFFFF;  //Divider color
@ColorInt
public int panelItemCheckedColor = 0xFF006EFF;  //Selected item color
@ColorInt
public int textColor = 0x99FFFFFF; //Text color
@ColorInt
public int textCheckedColor = 0xFFFFFFFF; //Selected text color
@ColorInt
public int seekBarProgressColor = 0xFF006EFF; //Progress bar color
```

Configure the JSON file of the panel

```
/**
 * Set the JSON file path of the beauty panel.
 * @param beauty: JSON file path for beauty properties. Set to null if not availabl
 * @param beautyBody: JSON file path for body beauty properties. Set to null if not
 * @param lut: JSON file path for filter properties. Set to null if not available.
 * @param motion: JSON file path for dynamic effect sticker properties. Set to null
 * @param makeup: JSON file path for full-face makeup style properties. Set to null
 * @param segmentation: JSON file path for segmentation properties. Set to null if
 */
public void setTEPanelViewRes(String beauty, String beautyBody, String lut, String
```

Update panel language

```
//When the client program detects a modification to the system's font, it may call
public void setSystemLocal(Locale locale)
```

TEPanelView Description

```
/**
 * Used to set the data of the last beauty effect, with the intent to restore the b
 * Note: This method needs to be used prior to the {@link #showView(TEPanelViewCall
 *
 * @param lastParamList: Beauty data, which can be obtained through the {@link TEBe
 */
void setLastParamList(String lastParamList);

/**
 * Present the beauty panel.
```

```
     * @param tePanelViewCallback: Panel event callback interface
     */
    void showView(TEPanelViewCallback tePanelViewCallback);


    /**
     * Bind the TEBeautyKit object. When the user clicks an item, the panel will direct
     * @param beautyKit The TEBeautyKit object
     */
    void setupWithTEBeautyKit(TEBeautyKit beautyKit);

    /**
     * Select the selected items for the custom segmentation or green screen, because a
     * Only after the user has selected a picture or video, the corresponding item can
     *
     * @param uiProperty
     */
    void validatePanelViewItem(TEUIProperty uiProperty);

    /**
     * Set the UI configuration of the current panel.
     * @param uiConfig
     */
    void updateUIConfig(TEUIConfig uiConfig);
```

Integrating Tencent Effect into MLVB SDK iOS

Last updated：2025-04-27 17:31:55

SDK Integration

1. For integrating Tencent Effect SDK, see Integrating Tencent Effect into MLVB SDK.
2. This documentation explains integrating and using the TEBeautyKit library in the Mobile Live Video BroadcastingStreaming (MLVB) SDK project.
3. Refer to demo.

Utilization of SDK

Step 1: Integrate TEBeautyKit

1. Download and extract TEBeautyKit.
2. Download and extract TELiveAdapter.
3. Copy the TEBeautyKit folder into your project and make it the same level as the podfile directory.
4. Edit the podfile and add the following code:

```
  pod 'TEBeautyKit',:path => 'TEBeautyKit/TEBeautyKit.podspec'
```

5.Incorporate the `TELiveAdapter.framework` into your project.

6.Execute the command `pod install`.

Step 2: Authenticate

```
[TEBeautyKit setTELicense:@"your license" key:@"your key" completion:^(NSInteger au
  NSLog(@"----------result: %zd  %@",authresult,errorMsg);
}];
```

Step 3: Configure the beauty material path

If the materials in the JSON file are local, you need to add the beauty materials to the project.

```
- (void)initBeautyJson{
    [[TEUIConfig shareInstance] setPanelLevel:S1_07];
```

```
}
```

Step 4: Initialize and add TEPanelView

```
-(TEPanelView *)tePanelView{
    if (!_tePanelView) {
        _tePanelView = [[TEPanelView alloc] init:nil comboType:nil];
        _tePanelView.delegate = self;
    }
    return _tePanelView;
}
[self.view addSubview:self.tePanelView];
[self.tePanelView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.width.mas_equalTo(self.view);
    make.centerX.mas_equalTo(self.view);
    make.height.mas_equalTo(250);
    make.bottom.mas_equalTo(self.view.mas_bottom);
}];
```

Step 5:Adapter Binding for Beautification

```
-(TEBeautyLiveAdapter *)liveAdapter{
    if (!_liveAdapter) {
        _liveAdapter = [[TEBeautyLiveAdapter alloc] init];
    }
    return _liveAdapter;
}
__weak __typeof(self)weakSelf = self;
[self.liveAdapter bind:self.livePusher onCreatedXmagicApi:^(XMagic * _Nullable xmag
    __strong typeof(self) strongSelf = weakSelf;
    strongSelf.teBeautyKit.xmagicApi = xmagicApi;
    [strongSelf.teBeautyKit setLogLevel:YT_SDK_ERROR_LEVEL];
    strongSelf.tePanelView.teBeautyKit = strongSelf.teBeautyKit;
    [strongSelf.tePanelView setDefaultBeauty];
  } onDestroyXmagicApi:^{
     __strong typeof(self) strongSelf = weakSelf;
     [strongSelf.teBeautyKit onDestroy];
     strongSelf.teBeautyKit = nil;
}];
```

Step 6: Parameter Change Notification Adapter

```
//Notify the Adapter of the Front and Rear Cameras: Whether to Encode a Mirror Imag
```

```
[self.liveAdapter notifyCameraChanged:self.isFrontCamera isEncoderMirror:self.isEnc
//Notify the Adapter of Screen Orientation Changes
[self.liveAdapter setDeviceOrientation:orientation];
```

Step 7: Unbinding the Adapter

```
 [self.liveAdapter unbind];
 self.liveAdapter=nil;
```

Android

Last updated：2025-06-05 10:51:27

SDK Integration

1. **Integration TencentEffect**

Integration TencentEffect SDK

Integration TencentEffcet Resources

2. **Integration TEBeautyKit**

Add TEBeautyKit dependencies

Add panel JSON file（optional if not using default panel）

3. **Integration** `te_adapter_live`

Maven Dependencies

AAR Dependencies

Add the dependency for `te_adapter_live` in dependencies

```
dependencies{
    ...
    implementation 'com.tencent.mediacloud:te_adapter_live:Version number'
}
```

Download AAR file (a zip file will be downloaded; extract it to get the AAR file).

Add the downloaded te_adapter_live_xxxx `.aar` file to the app project `libs` directory.

Open `build.gradle` in the app module and add a dependency reference:

```
dependencies{
    ...
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])   //add *.aar
}
```

Refer to MLVB Demo **project.**

Note:

To run the code, you need to add your applied License information in the file `com.tencent.thirdbeauty.xmagic.LicenseConstant.java`, and modify the `applicationId` in the `build.gradle` file under the App module to the package name you provided when applying for the License.

Due to the dependency on MLVB capability, you need to find the file `com.tencent.mlvb.debug.GenerateTestUserSig.java` under the Debug module and add the corresponding `LICENSEURL, LICENSEURLKEY, SDKAPPID, and SECRETKEY`.

For the main code integration of beauty effects, please refer to the file `com.tencent.mlvb.thirdbeauty.ThirdBeautyActivity.java`.

SDK Usage Steps

Step 1: Set Panel's JSON File

Please append the path of the JSON file added to your project in the second step of the 'How to Integrate' section in the TEBeautyKit Integration Document. If there is no JSON file, then the path Setting should be null.

```
    TEPanelViewResModel resModel = new TEPanelViewResModel();
    String combo = "S1_07";   //Set according to your package. If your package does
    resModel.beauty = "beauty_panel/"+combo+"/beauty.json";
    resModel.lut = "beauty_panel/"+combo+"/lut.json";
    resModel.beautyBody = "beauty_panel/"+combo+"/beauty_body.json";
    resModel.motion = "beauty_panel/"+combo+"/motions.json";
    resModel.lightMakeup = "beauty_panel/"+combo+"/light_makeup.json";
    resModel.segmentation = "beauty_panel/"+combo+"/segmentation.json";
    TEUIConfig.getInstance().setTEPanelViewRes(resModel);
```

Note: If you do not use the provided beauty panel, please ignore this step.

Step 2: Authenticate and copy resource

```
TEBeautyKit.setupSDK(this.getApplicationContext(),LicenseConstant.mXMagicLicenceUrl
    if (i == LicenseConstant.AUTH_STATE_SUCCEED) {
        runOnUiThread(() -> {
            Intent intent = new Intent(MainActivity.this, ThirdBeautyActivity.class
            startActivity(intent);
       }
    } else {
        Log.e(TAG, "te license check is failed,please checke  ");
    }
});
```

Note：

Resource copying is based on the SDK version number, so resources will only be successfully copied once for the same SDK version number.

Step 3: Initialize the adapter and the panel

```
this.beautyLiveAdapter = new TEBeautyLiveAdapter(XmagicConstant.EffectMode.PRO, TEB
// Set the phone orientation
this.beautyLiveAdapter.notifyScreenOrientationChanged(ActivityInfo.SCREEN_ORIENTATI
```

```
// Set the camera to front-facing or rear-facing, and whether to use encoding mirro
this.beautyLiveAdapter.notifyCameraChanged(isFront, this.isEncoderMirror);


private void initBeautyPanelView() {
    RelativeLayout panelLayout = findViewById(R.id.live_pusher_bp_beauty_panel);
    this.tePanelView = new TEPanelView(this);
    if (lastParamList != null) {  // Used to restore the last beauty effects
        this.tePanelView.setLastParamList(lastParamList);
    }
  this.tePanelView.showView(this);
    panelLayout.addView(this.tePanelView);
}
```

Note: If you do not want to use the provided panel, you can skip creating `TEPanelView` and configure the beauty attributes yourself by calling `TEBeautyKit` 's `setEffect` to set beauty attributes.

Step 4. Create Beauty

After V3.9.0

V3.9.0 and earlier

```
this.beautyLiveAdapter.bind(this, mLivePusher, new ITEBeautyAdapter.CallBack() {
    @Override
    public void onCreatedTEBeautyApi(XmagicApi xmagicApi) {
        mBeautyKit = new TEBeautyKit(xmagicApi);
        tePanelView.setupWithTEBeautyKit(mBeautyKit);
        setTipListener(mBeautyKit);
        setLastParam(mBeautyKit);
        Log.e("beautyLiveAdapter", "onCreatedTEBeautyKit");
    }

    @Override
    public void onDestroyTEBeautyApi() {
        mBeautyKit = null;
        Log.e("beautyLiveAdapter", "onDestroyTEBeautyKit");
    }
});
```

```
this.beautyLiveAdapter.bind(this, mLivePusher, new ITEBeautyAdapter.CallBack() {
    @Override
    public void onCreatedTEBeautyKit(TEBeautyKit beautyKit) {
        mBeautyKit = beautyKit;
        tePanelView.setupWithTEBeautyKit(mBeautyKit);
        setTipListener(mBeautyKit);
        setLastParam(mBeautyKit);
```

```
        Log.e("beautyLiveAdapter", "onCreatedTEBeautyKit");
    }

    @Override
    public void onDestroyTEBeautyKit() {
        mBeautyKit = null;
        Log.e("beautyLiveAdapter", "onDestroyTEBeautyKit");
    }
});
```

Step 5. Notify adapter while Parameter Changed

```
//Notify adapter of the latest status when the camera or screen mirror changes
 this.beautyLiveAdapter.notifyCameraChanged(isFront, this.isEncoderMirror);

//When the screen orientation changes, you need to call the notifyScreenOrientation
 this.beautyLiveAdapter.notifyScreenOrientationChanged(orientation);
```

Step 6: Destroy Effect

```
 //You can call the unbind method to unbind when the beauty filter is no longer nee
 this.beautyLiveAdapter.unbind();
```

Step 7: Resume Audio

```
/**
 * Used to restore sound in stickers
 * Restore gyroscope sensor, usually called in the onResume method of Activity
 */
this.mBeautyKit.onResume()
```

Step 8: Pause Audio

```
/**
 * Used to pause sound in stickers
 * Pause gyroscope sensor, usually called in the onPause method of Activity
 */
this.mBeautyKit.onPause()
```

Flutter

Last updated：2025-04-17 15:16:46

Beauty Flutter SDK requires dependence on the Beauty SDK of the Android/iOS end. Through the Plugins provided by Flutter, the native end features can be exposed to the Flutter client. Therefore, when integrating the beauty filter features, you need to manually integrate the SDK of the native end.

Running Demo

Download the demo project, modify the `demo/lib` under the `main.dart` file, add your `licenseUrl` and `licenseKey` in this file. The sample code for using beauty features in MLVB is primarily located in `demo/lib/page/live_page.dart` and `demo/lib/main.dart` .

Android

iOS

1. In demo/app, find the build.gradle file, open the file, and change the value of `applicationId` to the package name you used when applying for the license.
2. In demo/lib, execute `flutter pub get` .
3. Use Android Studio to open the demo project and run it.

1. In the demo directory, execute `flutter pub get` .
2. In the demo/ios directory, execute `pod install` .
3. Use Xcode to open Runner.xcworkspace.
4. Change the project's `bundle ID` (it needs to be the same as the bundle ID you provided when applying for the license).

SDK Integration

Native Integration

Android

iOS

1. In the app module, find the build.gradle file and add the Maven repository URL for your corresponding package. For example, if you choose the S1-04 package, add the following:

```
dependencies {
   implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
}
```

For the Maven URL corresponding to each package, please refer to the documentation. The latest version number of the SDK can be found in theRelease History.

2. In your project, find the `android/app` module and locate the `src/main/assets` folder. Copy the `demo/android/app/src/main/assetslut` and `MotionRes` folders from the demo project to your own project's `android/app/src/main/assets` . If your project doesn't have an assets folder, you can create one manually.

3. Locate the AndroidManifest.xml file under the app module, and add the following Tag in the application section:

```
    <uses-native-library
            android:name="libOpenCL.so"
            android:required="false" />
        //true indicates that libOpenCL is essential for the current app. Without t
        //false indicates that libOpenCL is not essential for the current app. The
        //For information about uses-native-library, please refer to the Android of
```

After adding, it looks as follows:

4. Obfuscation configuration

If you enable compile optimization (setting minifyEnabled to true) when packaging the release, it will trim some code that is not called in the Java layer. This code may possibly be invoked by the native layer, thus causing the `no xxx method` exception.

If you enabled such compile optimization, you should add these keep rules to avoid trimming xmagic's code:

```
-keep class com.tencent.xmagic.** { *;}
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class com.tencent.cloud.iai.lib.** { *;}
-keep class com.tencent.beacon.** { *;}
-keep class com.tencent.qimei.** { *;}
-keep class androidx.exifinterface.** { *;}
-keep class com.tencent.effect.** { *;}
```

Copy the xmagic folder from the ios/Runner directory of the demo project to the ios/Runner directory of your project.

After adding, it looks as follows:

Note:

The materials copied from the demo project are **test materials**. Official materials need to be obtained from us by contacting staff after purchasing a package, and then re-added.

Flutter Integration

Method 1:

Remote Dependencies, add the following reference in your project's pubspec.yaml file:

```
  tencent_effect_flutter:
   git:
     url: https://github.com/Tencent-RTC/TencentEffect_Flutter
```

Method 2:

Local Dependencies, download the latest version of tencent_effect_flutter, then add the folders android, ios, lib and the files pubspec.yaml, tencent_effect_flutter.iml to your project directory, and then add the following reference in your project's pubspec.yaml file: (refer to demo)

```
tencent_effect_flutter:
    path: ../
```

Run the following command:

```
flutter pub get
```

Note:

tencent_effect_flutter only provides a bridging, the beauty feature relies on beauty SDKs provided by each platform, so if you need to update the beauty SDK, you can upgrade the SDK using the following steps:

Android

iOS

Find the build.gradle file under the app module in your project, and change the `implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'` field's `latest.release` to the latest version number.

If you need to change the package, you need to modify the `TencentEffect_S1-04` field to your package field.

Need to change package type and SDK version:

1. The dependency for the Beauty Flutter SDK needs to be changed to a local dependency.
2. Find the `ios/tencent_effect_flutter.podspec` file under the Beauty Flutter SDK, open it and change `TencentEffect_All` to the package you need, and you can also change the version number to the version you need.
3. Execute the pod update command in the ios directory of your project.

SDK usage

1. Associated with MLVB

Android

iOS

Add the following code in the onCreate method of the application class (or the onCreate method of FlutterActivity):

```
TXLivePluginManager.register(new XmagicProcesserFactory());
```

Add the following code in the didFinishLaunchingWithOptions method of the AppDelegate class of your application:

```
XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
[TXLivePluginManager registerWithCustomBeautyProcesserFactory:instance];
```

After adding, it looks as follows:

2. Call the resource initialization API

v0.3.5.0 and later

Before v0.3.1.1

```
void _initSettings(InitXmagicCallBack callBack) async {
  String resourceDir = await ResPathManager.getResManager().getResPath();
  TXLog.printlog('$TAG method is _initResource ,xmagic resource dir is $resourceDir
  TencentEffectApi.getApi()?.setResourcePath(resourceDir);
  /// Resource copying only needs to be done once. In the current version, if copie
  /// Copying the resource only needs to be done once. Once it has been successfull
  if (await isCopiedRes()) {
    callBack.call(true);
    return;
  } else {
    _copyRes(callBack);
  }
}


void _copyRes(InitXmagicCallBack callBack) {
  _showDialog(context);
  TencentEffectApi.getApi()?.initXmagic((result) {
    if (result) {
      saveResCopied();
    }
    _dismissDialog(context);
    callBack.call(result);
```

```
    if (!result) {
      Fluttertoast.showToast(msg: "initialization failed");
    }
  });
}
```

```
String dir =  await BeautyDataManager.getInstance().getResDir();
 TXLog.printlog('The file path: $dir');
 TencentEffectApi.getApi()?.initXmagic(dir,(reslut) {
     _isInitResource = reslut;
     callBack.call(reslut);
     if (!reslut) {
         Fluttertoast.showToast(msg: "Failed to initialize the resources");
     }
 });
```

3. Perform beauty authorization

```
TencentEffectApi.getApi()?.setLicense(licenseKey, licenseUrl,
           (errorCode, msg) {
         TXLog.printlog("Print the authentication result errorCode = $errorCode   m
         if (errorCode == 0) {
            // Authentication succeeded
         }
       });
```

4. Enable/Disable Beauty Filter

```
/// Enable beauty filter operation
 var enableCustomVideo = await _livePusher.enableCustomVideoProcess(open);
```

Note: When enabling the beauty filter on the page, you must disable the beauty filter first before closing the camera. The enable and disable operations should be paired to ensure proper functionality.

5. Set Beauty Attributes

v0.3.5.0 and later

Before v0.3.1.1

For specific beauty attributes, refer to Beauty Attributes Table.

```
TencentEffectApi.getApi()?.setEffect(sdkParam.effectName!,
    sdkParam.effectValue, sdkParam.resourcePath, sdkParam.extraInfo)
```

```
TencentEffectApi.getApi()?.updateProperty(_xmagicProperty!);
```

```
/// You can call `BeautyDataManager.getInstance().getAllPannelData()` to get all th
```

6. Set other properties

Pause Beauty Sound Effects

```
 TencentEffectApi.getApi()?.onPause();
```

Resume Beauty Sound Effects

```
TencentEffectApi.getApi()?.onResume();
```

Monitor Beauty Events

```
TencentEffectApi.getApi()
     ?.setOnCreateXmagicApiErrorListener((errorMsg, code) {
       TXLog.printlog("Error creating an effect object errorMsg = $errorMsg , code
 });   /// Needs to be set before creating the beauty filter
```

Set Callback for Face, Gesture, and Body Detection Status

```
TencentEffectApi.getApi()?.setAIDataListener(XmagicAIDataListenerImp());
```

Set Callback Function for Dynamic Prompt Messages

```
TencentEffectApi.getApi()?.setTipsListener(XmagicTipsListenerImp());
```

Configure the Callback of Facial Keypoints and Other Data (only available in S1-05 and S1-06)

```
TencentEffectApi.getApi()?.setYTDataListener((data) {
   TXLog.printlog("setYTDataListener  $data");
 });
```

Remove All Callbacks. You need to remove all callbacks when terminating the page:

```
 TencentEffectApi.getApi()?.setOnCreateXmagicApiErrorListener(null);
 TencentEffectApi.getApi()?.setAIDataListener(null);
 TencentEffectApi.getApi()?.setYTDataListener(null);
 TencentEffectApi.getApi()?.setTipsListener(null);
```

Note:

For more information on the APIs, see API Documentation. For others, refer to the Demo Project.

7. Add Data to and Remove Data from the Effect Panel

Add effect resources:

Add your resource file to the corresponding folder as described in step 1. For example, to add a 2D animated effect resource:
Android
iOS
You should put the resource in `android/xmagic/src.mian/assets/MotionRes/2dMotionRes` of your project:

Also add the resource to `ios/Runner/xmagic/2dMotionRes.bundle` of your project.

Beauty Panel Configuration:

V0.3.5.0 and later
V0.3.1.1 and earlier
The demo provides a simple beauty panel UI. The panel's properties are configured through JSON files, located as shown in the following diagram. The implementation of the panel can be referred from the demo project.

JSON Files
In the classes BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid, and BeautyPropertyProducerIOS, you can independently configure the beauty panel data.

Delete Beauty Resources

For some licenses that do not authorize certain beauty and body shaping features, these features should not be displayed on the beauty panel. You need to remove these features from the beauty panel data configuration.
For example, to delete the Lipstick Effect, remove the following code from the getBeautyData method in both the BeautyPropertyProducerAndroid and BeautyPropertyProducerIOS classes.

Integrating Tencent Effect into TRTC SDK
iOS

Last updated：2025-04-27 17:31:55

SDK Integration

1. For integrating Tencent Effect SDK, see Integrating Tencent Effect into TRTC SDK.
2. This documentation explains integrating and using the TEBeautyKit library in the TRTC SDK project.
3. Refer to demo.

Utilization of SDK

Step 1: Integrate TEBeautyKit

1. Download and extract TEBeautyKit.
2. Download and extract TRTCAdapter.
3. Copy the TEBeautyKit folder into your project and make it the same level as the podfile directory.
4. Incorporate the `TRTCAdapter.framework` into your project.

5. Edit the podfile and add the following code:

```
  pod 'TEBeautyKit',:path => 'TEBeautyKit/TEBeautyKit.podspec'
```

6. Execute the command `pod install`.

Step 2: Authenticate

```
[TEBeautyKit setTELicense:@"your license" key:@"your key" completion:^(NSInteger au
  NSLog(@"----------result: %zd  %@",authresult,errorMsg);
}];
```

Step 3: Configure the beauty material path

If the materials configured in the JSON file are local, you need to add the beauty materials to your project.

```
- (void)initBeautyJson{
    [[TEUIConfig shareInstance] setPanelLevel:S1_07];
}
```

Step 4: Initialize and add TEPanelView

```
-(TEPanelView *)tePanelView{
    if (!_tePanelView) {
        _tePanelView = [[TEPanelView alloc] init:nil comboType:nil];
        _tePanelView.delegate = self;
    }
    return _tePanelView;
}
[self.view addSubview:self.tePanelView];
[self.tePanelView mas_makeConstraints:^(MASConstraintMaker *make) {
    make.width.mas_equalTo(self.view);
    make.centerX.mas_equalTo(self.view);
    make.height.mas_equalTo(250);
    make.bottom.mas_equalTo(self.view.mas_bottom);
}];
```

Step 5: Adapter Binding for Beautification

```
-(TEBeautyTRTCAdapter *)trtcAdapter{
    if (!_trtcAdapter) {
        _trtcAdapter = [[TEBeautyTRTCAdapter alloc] init];
    }
    return  _trtcAdapter;
}
__weak __typeof(self)weakSelf = self;
[self.trtcAdapter bind:self.trtcCloud onCreatedXmaicApi:^(XMagic * _Nullable xmagic
   __strong typeof(self) strongSelf = weakSelf;
   strongSelf.teBeautyKit.xmagicApi = xmagicApi;
   [strongSelf.teBeautyKit setLogLevel:YT_SDK_ERROR_LEVEL];
   strongSelf.tePanelView.teBeautyKit = strongSelf.teBeautyKit;
   [strongSelf.tePanelView setDefaultBeauty];
 }onDestroyXmaicApi:^{
    __strong typeof(self) strongSelf = weakSelf;
    [strongSelf.teBeautyKit onDestroy];
    strongSelf.teBeautyKit = nil;
 }];
```

Step 6: Parameter Change Notification Adapter

```
//Notify the Adapter of the Front and Rear Cameras: Whether to Encode a Mirror Imag
[self.trtcAdapter notifyCameraChanged:self.isFrontCamera isEncoderMirror:self.isEnc
//Notify the Adapter of Screen Orientation Changes
[self.trtcAdapter setDeviceOrientation:currOrientation];
```

Step 7: Unlink the adapter and terminate the beauty enhancement feature

```
[self.trtcAdapter unbind];
self.trtcAdapter = nil;
```

Android

Last updated：2025-06-05 10:52:05

SDK Integration

1. **Integration TencentEffect**

Integration TencentEffect SDK

Integration TencentEffcet Resources

2. **Integration TEBeautyKit**

Add TEBeautyKit dependencies

Add panel JSON file（optional if not using default panel）

3. **Integration** `te_adapter_trtc`

Maven Dependencies

AAR Dependencies

Add the dependency for `te_adapter_trtc` in dependencies

```
dependencies{
    ...
    implementation 'com.tencent.mediacloud:te_adapter_trtc:Version number'
}
```

Download AAR file (a zip file will be downloaded; extract it to get the AAR file).

Add the downloaded te_adapter_live_xxxx `.aar` file to the app project `libs` directory.

Open `build.gradle` in the app module and add a dependency reference:

```
dependencies{
    ...
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])   //add *.aar
}
```

Refer to TRTC Demo **project.**

Note：

To run the code, you need to add your applied License information in the file `com.tencent.thirdbeauty.xmagic.LicenseConstant.java`, and modify the applicationId in the `build.gradle` file under the App module to the package name you provided when applying for the License.

Due to the dependency on TRTC capability, you need to find the file `com.tencent.trtc.debug.GenerateTestUserSig.java` under the Debug module and add the corresponding `APPID, SDKAPPID, and SECRETKEY`.

For the main code integration of beauty effects, please refer to the file `com.tencent.trtc.thirdbeauty.ThirdBeautyActivity.java`.

SDK Usage Steps

Step 1: Set Panel's JSON File

Please append the path of the JSON file added to your project in the second step of the 'How to Integrate' section in the TEBeautyKit Integration Document. If there is no JSON file, then the path Setting should be null.

```
TEPanelViewResModel resModel = new TEPanelViewResModel();
   String combo = "S1_07";   //Set according to your package. If your package does
   resModel.beauty = "beauty_panel/"+combo+"/beauty.json";
   resModel.lut = "beauty_panel/"+combo+"/lut.json";
   resModel.beautyBody = "beauty_panel/"+combo+"/beauty_body.json";
   resModel.motion = "beauty_panel/"+combo+"/motions.json";
   resModel.lightMakeup = "beauty_panel/"+combo+"/light_makeup.json";
   resModel.segmentation = "beauty_panel/"+combo+"/segmentation.json";
   TEUIConfig.getInstance().setTEPanelViewRes(resModel);
```

Note: If you do not use the provided beauty panel, please ignore this step.

Step 2: Authenticate and copy resource

```
TEBeautyKit.setupSDK(this.getApplicationContext(),LicenseConstant.mXMagicLicenceUrl
    if (i == LicenseConstant.AUTH_STATE_SUCCEED) {
        runOnUiThread(() -> {
            Intent intent = new Intent(MainActivity.this, ThirdBeautyActivity.class
            startActivity(intent);
       }
    } else {
        Log.e(TAG, "te license check is failed,please checke  ");
    }
});
```

Note：

Resource copying is based on the SDK version number, so resources will only be successfully copied once for the same SDK version number.

Step 3: Initialize the adapter and the panel

```
this.beautyAdapter = new TEBeautyTRTCAdapter(XmagicConstant.EffectMode.PRO, TEBeaut
// Set phone orientation
this.beautyAdapter.notifyScreenOrientationChanged(ActivityInfo.SCREEN_ORIENTATION_P
```

```
// Set whether the camera is front-facing or rear-facing, and whether encoding mirr
this.beautyAdapter.notifyCameraChanged(isFront, this.isEncoderMirror);
// Initialize panel code
private void initializeBeautyPanelView() {
    RelativeLayout panelLayout = findViewById(R.id.live_pusher_bp_beauty_panel);
    this.elegantPanelView = new TEPanelView(this);
    if (previousParamList != null) {  //To restore the previous beauty effect
        this.elegantPanelView.setPreviousParamList(previousParamList);
    }
    this.elegantPanelView.displayView(this);
    panelLayout.addView(this.elegantPanelView);
}
```

Note: If you do not use the provided panel, you do not need to create `TEPanelView`. Beauty attributes can be set using `TEBeautyKit`'s `setEffect` method.

Step 4. Create Beauty

After V3.9.0

V3.9.0 and earlier

```
this.beautyAdapter.bind(this, this.mTRTCCloud, new ITEBeautyAdapter.CallBack() {
    @Override
    public void onCreatedTEBeautyApi(XmagicApi xmagicApi) {
        Log.e(TAG, "onCreatedTEBeautyApi  ");
        mBeautyKit = new TEBeautyKit(xmagicApi);
        mPanelView.setupWithTEBeautyKit(mBeautyKit);
        setTipListener(mBeautyKit);
        setLastParam(mBeautyKit);
    }

    @Override
    public void onDestroyTEBeautyApi() {
        Log.e(TAG, "onDestroyTEBeautyApi  ");
        mBeautyKit = null;
    }
});
```

```
this.beautyAdapter.bind(this, this.mTRTCCloud, new ITEBeautyAdapter.CallBack() {
    @Override
    public void onCreatedTEBeautyKit(TEBeautyKit beautyKit) {
        mBeautyKit = beautyKit;
        // Bind mBeautyKit with the beauty panel
        tePanelView.setupWithTEBeautyKit(mBeautyKit);
        setTipListener(mBeautyKit);
        setLastParam(mBeautyKit);
```

```
        Log.e("beautyLiveAdapter", "onCreatedTEBeautyKit");
    }

    @Override
    public void onDestroyTEBeautyKit() {
        mBeautyKit = null;
        Log.e("beautyLiveAdapter", "onDestroyTEBeautyKit");
    }
});
```

Step 5. Notify adapter while Parameter Changed

```
// Call notifyCameraChanged to inform the adapter about changes in camera or screen
 this.beautyAdapter.notifyCameraChanged(isFront, this.isEncoderMirror);

//When the screen orientation changes, you need to call the notifyScreenOrientation
 this.beautyAdapter.notifyScreenOrientationChanged(orientation);
```

Step 6: Destroy Beauty

```
// Unbind method can be called to release the binding when beauty enhancement is no
 this.beautyAdapter.unbind();
```

Step 7: Resume Audio

```
/**
 * Used to restore the sound in stickers
 * Restores the gyroscope sensor, typically called in the Activity's onResume metho
 */
 this.mBeautyKit.onResume()
```

Step 8: Pause Audio

```
/**
 * Used to pause sound in stickers
 * Pause gyroscope sensor, usually called in the onPause method of Activity
 */
this.mBeautyKit.onPause()
```

Flutter

Last updated：2025-03-03 17:30:04

Beauty Flutter SDK requires dependence on the Beauty SDK of the Android/iOS end. Through the Plugins provided by Flutter, the native end features can be exposed to the Flutter client. Therefore, when integrating the beauty filter features, you need to manually integrate the SDK of the native end.

Running Demo

Download the demo project, modify the `demo/lib` under the `main.dart` file, add your `licenseUrl` and `licenseKey` in this file. The sample code for using beauty features in TRTC is primarily located in `demo/lib/page/trtc_page.dart` and `demo/lib/main.dart` .

Android

iOS

1. In demo/app, find the build.gradle file, open the file, and change the value of `applicationId` to the package name you used when applying for the license.
2. In demo/lib, execute `flutter pub get` .
3. Use Android Studio to open the demo project and run it.

1. In the demo directory, execute `flutter pub get` .
2. In the demo/ios directory, execute `pod install` .
3. Use Xcode to open Runner.xcworkspace.
4. Change the project's `bundle ID` (it needs to be the same as the bundle ID you provided when applying for the license).

SDK Integration

Native Integration

Android

iOS

1. In the app module, find the build.gradle file and add the Maven repository URL for your corresponding package. For example, if you choose the S1-04 package, add the following:

```
dependencies {
   implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
}
```

For the Maven URL corresponding to each package, please refer to the documentation. The latest version number of the SDK can be found in theRelease History.

2. In your project, find the `android/app` module and locate the `src/main/assets` folder. Copy the `demo/android/app/src/main/assetslut` and `MotionRes` folders from the demo project to your own project's `android/app/src/main/assets` . If your project doesn't have an assets folder, you can create one manually.

3. **If you are using an Android of the Tencent Effect SDK that is lower than 3.9**, you will need to locate the AndroidManifest.xml file under the app module, and add the following Tag in the application section:

```
    <uses-native-library
            android:name="libOpenCL.so"
            android:required="false" />
        //true indicates that libOpenCL is essential for the current app. Without t
        //false indicates that libOpenCL is not essential for the current app. The
        //For information about uses-native-library, please refer to the Android of
```

After adding, it looks as follows:

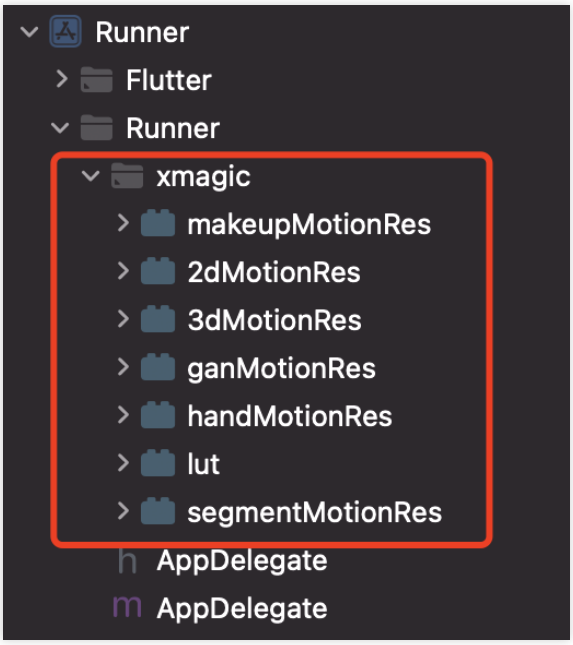
4. Obfuscation configuration

If you enable compile optimization (setting minifyEnabled to true) when packaging the release, it will trim some code that is not called in the Java layer. This code may possibly be invoked by the native layer, thus causing the `no xxx method` exception.

If you enabled such compile optimization, you should add these keep rules to avoid trimming xmagic's code:

```
-keep class com.tencent.xmagic.** { *;}
-keep class org.light.** { *;}
-keep class org.libpag.** { *;}
-keep class org.extra.** { *;}
-keep class com.gyailib.**{ *;}
-keep class com.tencent.cloud.iai.lib.** { *;}
-keep class com.tencent.beacon.** { *;}
-keep class com.tencent.qimei.** { *;}
-keep class androidx.exifinterface.** { *;}
-keep class com.tencent.effect.** { *;}
```

Copy the xmagic folder from the ios/Runner directory of the demo project to the ios/Runner directory of your project. After adding, it looks as follows:



Note:

The materials copied from the demo project are **test materials**. Official materials need to be obtained from us by contacting staff after purchasing a package, and then re-added.

Flutter Integration

Method 1:

Remote Dependencies, add the following reference in your project's pubspec.yaml file:

```
  tencent_effect_flutter:
   git:
     url: https://github.com/Tencent-RTC/TencentEffect_Flutter
```

Method 2:

Local Dependencies, download the latest version of tencent_effect_flutter, then add the folders android, ios, lib and the files pubspec.yaml, tencent_effect_flutter.iml to your project directory, and then add the following reference in your project's pubspec.yaml file: (refer to demo)

```
tencent_effect_flutter:
    path: ../
```

Run the following command:

```
flutter pub get
```

Note:

tencent_effect_flutter only provides a bridging, the beauty feature relies on beauty SDKs provided by each platform, so if you need to update the beauty SDK, you can upgrade the SDK using the following steps:

Android

iOS

Find the build.gradle file under the app module in your project, and change the `implementation 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'` field's `latest.release` to the latest version number.

If you need to change the package, you need to modify the `TencentEffect_S1-04` field to your package field.

Need to change package type and SDK version:

1. The dependency for the Beauty Flutter SDK needs to be changed to a local dependency.
2. Find the `ios/tencent_effect_flutter.podspec` file under the Beauty Flutter SDK, open it and change `TencentEffect_All` to the package you need, and you can also change the version number to the version you need.
3. Execute the pod update command in the ios directory of your project.

SDK usage

1. Associated with TRTC

Android

iOS

Add the following code in the onCreate method of the application class (or the onCreate method of FlutterActivity):

```
TRTCCloudPlugin.register(new XmagicProcesserFactory());
```

Add the following code in the didFinishLaunchingWithOptions method of the AppDelegate class of your application:

```
XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
[TencentTRTCCloud registerWithCustomBeautyProcesserFactory:instance];
```

After adding, it looks as follows:

```objc
#import "AppDelegate.h"
#import "GeneratedPluginRegistrant.h"
@import live_flutter_plugin;
@import tencent_effect_flutter;
@import tencent_trtc_cloud;

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
  [GeneratedPluginRegistrant registerWithRegistry:self];
  // Override point for customization after application launch.
  XmagicProcesserFactory *instance = [[XmagicProcesserFactory alloc] init];
  [TXLivePluginManager registerWithCustomBeautyProcesserFactory:instance];
  [TencentTRTCCloud registerWithCustomBeautyProcesserFactory:instance];
  return [super application:application didFinishLaunchingWithOptions:launchOption
}

@end
```

2. Call the resource initialization API

v0.3.5.0 and later

Before v0.3.1.1

```
void _initSettings(InitXmagicCallBack callBack) async {
  String resourceDir = await ResPathManager.getResManager().getResPath();
  TXLog.printlog('$TAG method is _initResource ,xmagic resource dir is $resourceDir
  TencentEffectApi.getApi()?.setResourcePath(resourceDir);
  /// Resource copying only needs to be done once. In the current version, if copie
  /// Copying the resource only needs to be done once. Once it has been successfull
  if (await isCopiedRes()) {
    callBack.call(true);
    return;
  } else {
    _copyRes(callBack);
  }
}


void _copyRes(InitXmagicCallBack callBack) {
  _showDialog(context);
  TencentEffectApi.getApi()?.initXmagic((result) {
    if (result) {
      saveResCopied();
```

```
    }
    _dismissDialog(context);
    callBack.call(result);
    if (!result) {
      Fluttertoast.showToast(msg: "initialization failed");
    }
  });
}
```

```
String dir =  await BeautyDataManager.getInstance().getResDir();
 TXLog.printlog('The file path: $dir');
 TencentEffectApi.getApi()?.initXmagic(dir,(reslut) {
     _isInitResource = reslut;
     callBack.call(reslut);
     if (!reslut) {
         Fluttertoast.showToast(msg: "Failed to initialize the resources");
     }
 });
```

3. Perform beauty authorization

```
TencentEffectApi.getApi()?.setLicense(licenseKey, licenseUrl,
           (errorCode, msg) {
         TXLog.printlog("Print the authentication result errorCode = $errorCode   m
         if (errorCode == 0) {
            // Authentication succeeded
         }
       });
```

4. Enable/Disable Beauty Filter

```
/// Enable beauty filter operation
/// Setting it to true enables the beauty filter, while setting it to false disable
 var enableCustomVideo = await trtcCloud.enableCustomVideoProcess(open);
```

Note: When enabling the beauty filter on the page, you must disable the beauty filter first before closing the camera. The enable and disable operations should be paired to ensure proper functionality.

5. Set Beauty Attributes

v0.3.5.0 and later

Before v0.3.1.1

For specific beauty attributes, refer to Beauty Attributes Table.

```
TencentEffectApi.getApi()?.setEffect(sdkParam.effectName!,
    sdkParam.effectValue, sdkParam.resourcePath, sdkParam.extraInfo)
```

```
TencentEffectApi.getApi()?.updateProperty(_xmagicProperty!);
/// You can call `BeautyDataManager.getInstance().getAllPannelData()` to get all th
```

6. Set other properties

Pause Beauty Sound Effects

```
 TencentEffectApi.getApi()?.onPause();
```

Resume Beauty Sound Effects

```
TencentEffectApi.getApi()?.onResume();
```

Monitor Beauty Events

```
TencentEffectApi.getApi()
     ?.setOnCreateXmagicApiErrorListener((errorMsg, code) {
       TXLog.printlog("Error creating an effect object errorMsg = $errorMsg , code
 });   /// Needs to be set before creating the beauty filter
```

Set Callback for Face, Gesture, and Body Detection Status

```
TencentEffectApi.getApi()?.setAIDataListener(XmagicAIDataListenerImp());
```

Set Callback Function for Dynamic Prompt Messages

```
TencentEffectApi.getApi()?.setTipsListener(XmagicTipsListenerImp());
```

Configure the Callback of Facial Keypoints and Other Data (only available in S1-05 and S1-06)

```
TencentEffectApi.getApi()?.setYTDataListener((data) {
   TXLog.printlog("setYTDataListener  $data");
 });
```

Remove All Callbacks. You need to remove all callbacks when terminating the page:

```
 TencentEffectApi.getApi()?.setOnCreateXmagicApiErrorListener(null);
 TencentEffectApi.getApi()?.setAIDataListener(null);
 TencentEffectApi.getApi()?.setYTDataListener(null);
 TencentEffectApi.getApi()?.setTipsListener(null);
```

Note

For more information on the APIs, see API Documentation. For others, refer to the Demo Project.

7. Add Data to and Remove Data from the Effect Panel

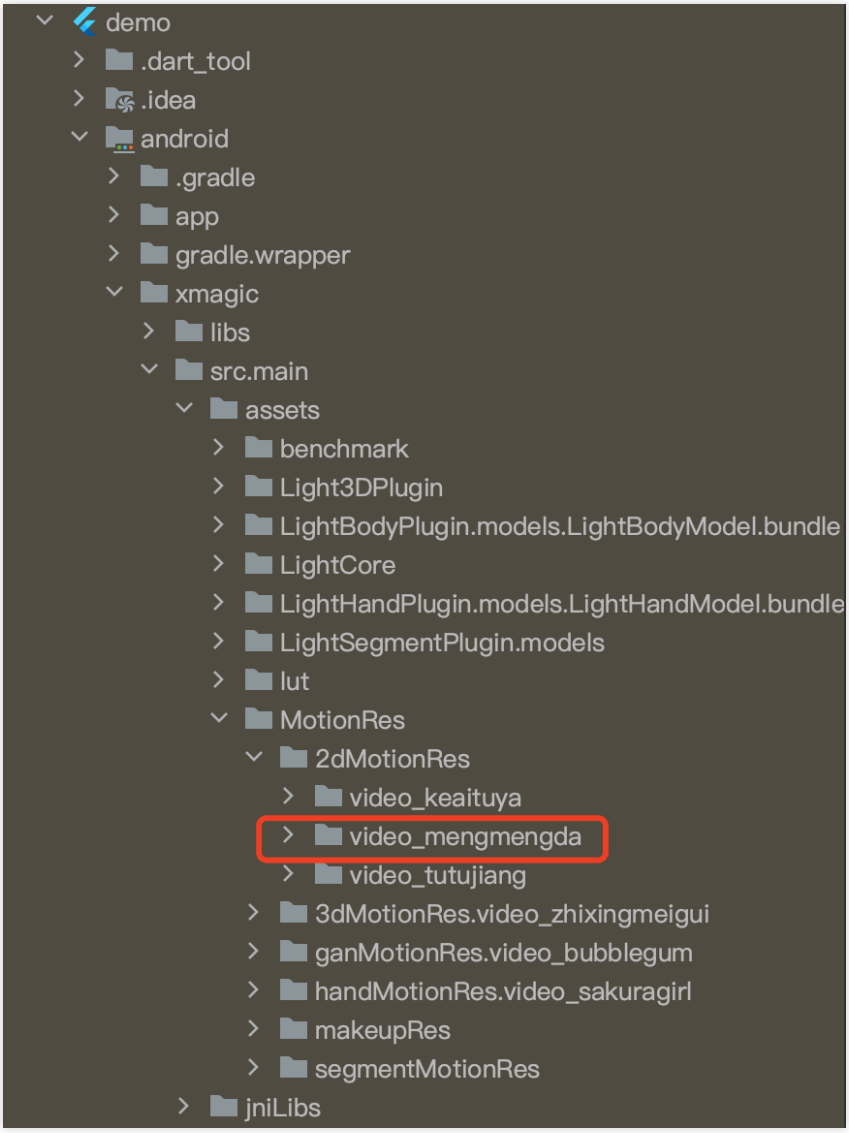
Add effect resources:

Add your resource file to the corresponding folder as described in step 1. For example, to add a 2D animated effect resource:

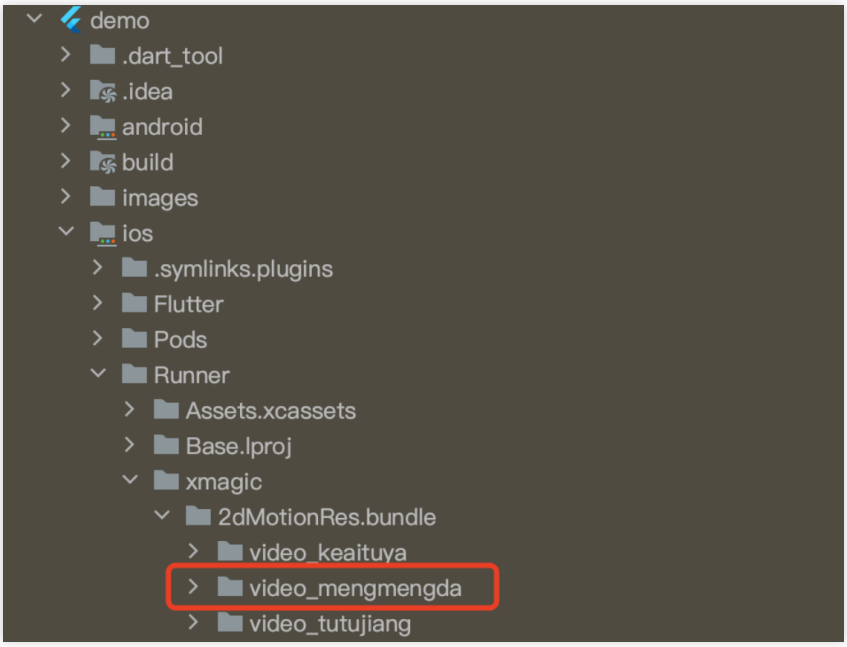
Android

iOS

You should put the resource in `android/xmagic/src.mian/assets/MotionRes/2dMotionRes` of your project:



Also add the resource to `ios/Runner/xmagic/2dMotionRes.bundle` of your project.

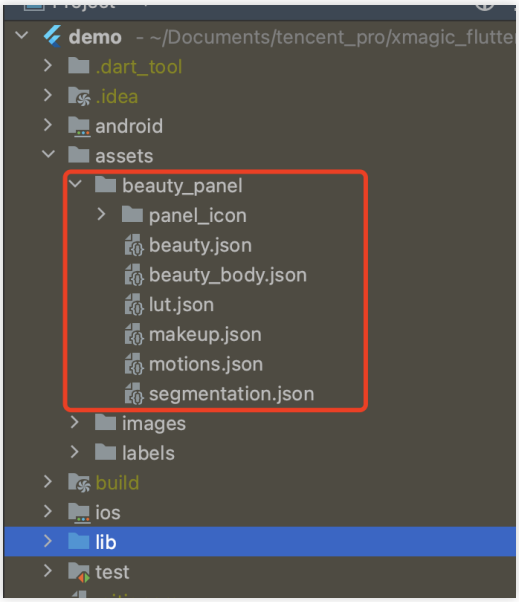


Beauty Panel Configuration:

V0.3.5.0 and later

V0.3.1.1 and earlier

The demo provides a simple beauty panel UI. The panel's properties are configured through JSON files, located as shown in the following diagram. The implementation of the panel can be referred from the demo project.



JSON Files

In the classes BeautyDataManager, BeautyPropertyProducer, BeautyPropertyProducerAndroid, and BeautyPropertyProducerIOS, you can independently configure the beauty panel data.

Delete Beauty Resources

For some licenses that do not authorize certain beauty and body shaping features, these features should not be displayed on the beauty panel. You need to remove these features from the beauty panel data configuration.
For example, to delete the Lipstick Effect, remove the following code from the getBeautyData method in both the BeautyPropertyProducerAndroid and BeautyPropertyProducerIOS classes.

```
// Map<String, String> lipsResPathNames = {};
// lipsResPathNames["lips_fuguhong.png"] = "复古红";
// lipsResPathNames["lips_mitaose.png"] = "蜜桃色";
// lipsResPathNames["lips_shanhuju.png"] = "珊瑚橘";
// lipsResPathNames["lips_wenroufen.png"] = "温柔粉";
// lipsResPathNames["lips_huolicheng.png"] = "活力橙";
// List<XmagicUIProperty> itemLipsPropertys = [];
// String lipId = "beauty.lips.lipsMask";
// for (String ids in lipsResPathNames.keys) {
//   itemLipsPropertys.add(XmagicUIProperty(
//       uiCategory: Category.BEAUTY,
//       displayName: lipsResPathNames[ids]!,
//       id: lipId,
//       resPath: resPaths + ids,
//       thumbDrawableName: "beauty_lips",
//       effKey: BeautyConstant.BEAUTY_MOUTH_LIPSTICK,
//       effValue: XmagicPropertyValues(0, 100, 50, 0, 1),
//       rootDisplayName: "口红"));
// }
// XmagicUIProperty itemLips = XmagicUIProperty(
//     displayName: "口红",
//     thumbDrawableName: "beauty_lips",
//     uiCategory: Category.BEAUTY);
// itemLips.xmagicUIPropertyList = itemLipsPropertys;
// beautyList.add(itemLips);
```

Integrating Tencent Effect into UGSV SDK iOS

Last updated：2023-02-27 14:27:24

Prerequisites

1. Download and decompress the demo package, and copy the `xmagickit` folder in the `demo/XiaoShiPin/` directory to the directory of the Podfile in your project.
2. Add the following dependencies to your `Podfile` and run `pod install`.

```
pod 'xmagickit', :path => 'xmagickit/xmagickit.podspec'
```

3. Set `Bundle ID` to the bundle ID bound to your license.

Environment requirements

Xcode 11 or later (download from App Store or here)
Recommended runtime environment:
Device requirements: iPhone 5 or later. iPhone 6 and older models support up to 720p for the front camera.
System requirements: iOS 12.0 or later.

SDK API Integration

Step 1. Authenticate

Add the following code to `didFinishLaunchingWithOptions` of `AppDelegate` (`LicenseURL` and `LicenseKey` are the authorization information you obtain from Tencent Cloud). If your SDK version is earlier than 2.5.1, you can find `TELicenseCheck.h` in `XMagic.framework`; if your SDK version is 2.5.1 or later, `TELicenseCheck.h` is in `YTCommonXMagic.framework`.

```
[TXUGCBase setLicenceURL:LicenseURL key:LicenseKey];

[TELicenseCheck setTELicense:LicenseURL key:LicenseKey completion:^(NSInteger authr
              if (authresult == TELicenseCheckOk) {
                  NSLog(@"Authentication successful");
               } else {
                  NSLog(@"Authentication failed");
               }
       }];
```

Authentication error codes:

Error Code	Description
0	Successful.
-1	The input parameter is invalid; for example, the `URL` or `KEY` is empty.
-3	Download failed. Check the network settings.
-4	Unable to obtain any Tencent Effect authentication information from the local system, which may be caused by an I/O failure.
-5	The VCUBE TEMP license file is empty, which may be caused by an I/O failure.
-6	The JSON field in the `v_cube.license` file is incorrect. Please contact Tencent Cloud team for help.
-7	Signature verification failed. Please contact Tencent Cloud team for help.
-8	Decryption failed. Please contact Tencent Cloud team for help.
-9	The JSON field in `TELicense` is incorrect. Please contact Tencent Cloud team for help.
-10	The Tencent Effect authentication information parsed online is empty. Please contact Tencent Cloud team for help.
-11	Failed to write Tencent Effect SDK authentication information to the local file, which may be caused by an I/O failure.
-12	Download failed, and failed to parse local assets.
-13	Authentication failed.
Others	Please contact Tencent Cloud team for help.

Step 2. Set the SDK material path

```
CGSize previewSize = [self getPreviewSizeByResolution:self.currentPreviewResolution
NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirecto
beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_config
NSFileManager *localFileManager=[[NSFileManager alloc] init];
BOOL isDir = YES;
NSDictionary * beautyConfigJson = @{};
if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] && !isD
    NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beautyConfig
```

```
    NSError *jsonError;
    NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8StringEncodin
    beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData
                                            options:NSJSONReadingMutableContainers
                                            error:&jsonError];
}
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                             @"tnn_"
                             @"beauty_config":beautyConfigJson
};
// Init beauty kit
self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:assetsDi
```

Step 3. Add log and event listeners

```
// Register log
[self.beautyKit registerSDKEventListener:self];
[self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL];
```

Step 4. Configure effects

```
- (int)configPropertyWithType:(NSString *_Nonnull)propertyType withName:(NSString *
```

Step 5. Render the video

In the preprocessing frame callback, construct `YTProcessInput` and pass `textureId` to the SDK for rendering.

```
 [self.xMagicKit process:inputCPU withOrigin:YtLightImageOriginTopLeft
withOrientation:YtLightCameraRotation0]
```

Step 6. Pause/Resume the SDK

```
[self.beautyKit onPause];
[self.beautyKit onResume];
```

Step 7. Add an effect panel to the layout

```
UIEdgeInsets gSafeInset;
#if __IPHONE_11_0 && __IPHONE_OS_VERSION_MAX_ALLOWED >= __IPHONE_11_0
if(gSafeInset.bottom > 0){
```

```
}
if (@available(iOS 11.0, *)) {
    gSafeInset = [UIApplication sharedApplication].keyWindow.safeAreaInsets;
} else
#endif
    {
        gSafeInset = UIEdgeInsetsZero;
    }

dispatch_async(dispatch_get_main_queue(), ^{
    // Effect option UI
    _vBeauty = [[BeautyView alloc] init];
    [self.view addSubview:_vBeauty];
    [_vBeauty mas_makeConstraints:^(MASConstraintMaker *make) {
        make.width.mas_equalTo(self.view);
        make.centerX.mas_equalTo(self.view);
        make.height.mas_equalTo(254);
        if(gSafeInset.bottom > 0.0){  // Adapt to full-view screen
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(0);
        } else {
            make.bottom.mas_equalTo(self.view.mas_bottom).mas_offset(-10);
        }
    }];
        _vBeauty.hidden = YES;
});
```

Android

Last updated：2022-12-01 09:33:11

Step 1. Replace resources

1. Download the UGSV demo which has integrated the Tencent Effect SDK S1 - 04.
2. Replace the SDK files in the demo with the files for the SDK you actually use. Specifically, follow the steps below:
In the `build.gradle` file of the `xmagickit` module, find the following:

```
api 'com.tencent.mediacloud:TencentEffect_S1-04:latest.release'
```

Replace it with the SDK edition you purchased as described in Integrating the Tencent Effect SDK (Android).
If your SDK edition includes animated effects and filters, you need to download the corresponding SDK package and add the resources for animated effects and filters to the following directories of `xmagickit`:
Animated effects: `../assets/MotionRes`.
Filters: `../assets/lut`.
3. Import `xmagickit` in the demo project into your own project.

Step 2. Modify the package name

Open `build.gradle` in `app` and set `applicationId` to the package name bound to your trial license.

Step 3. Integrate the SDK APIs

You can refer to the `UGCKitVideoRecord` class of the demo.
1. **Set the license:**

```
// For details about authentication and the error codes, see https://www.tencentcl
XMagicImpl.checkAuth(new TELicenseCheck.TELicenseCheckListener() {
        @Override
        public void onLicenseCheckFinish(int errorCode, String msg) {
            if (errorCode == TELicenseCheck.ERROR_OK) {
                loadXmagicRes();
            } else {
                Log.e("TAG", "auth fail , please check auth url and key" + errorCo
            }
        }
    });
```

2. **Initialize the resources:**

```
private void loadXmagicRes() {
    if (XMagicImpl.isLoadedRes) {
        XmagicResParser.parseRes(mActivity.getApplicationContext());
        initXMagic();
        return;
    }
    new Thread(new Runnable() {
        @Override
        public void run() {
            XmagicResParser.copyRes(mActivity.getApplicationContext());
            XmagicResParser.parseRes(mActivity.getApplicationContext());
            XMagicImpl.isLoadedRes = true;
            new Handler(Looper.getMainLooper()).post(new Runnable() {
                @Override
                public void run() {
                    initXMagic();
                }
            });
        }
    }).start();
}
```

3. **Bind UGSV and Tencent Effect:**

```
private void initBeauty() {
    TXUGCRecord instance = TXUGCRecord.getInstance(UGCKit.getAppContext());
    instance.setVideoProcessListener(new TXUGCRecord.VideoCustomProcessListener()
        @Override
        public int onTextureCustomProcess(int textureId, int width, int height) {
            if (xmagicState == XMagicImpl.XmagicState.STARTED && mXMagic != null)
                return mXMagic.process(textureId, width, height);
            }
            return textureId;
        }

        @Override
        public void onDetectFacePoints(float[] floats) {
        }

        @Override
        public void onTextureDestroyed() {
            if (Looper.getMainLooper() != Looper.myLooper()) {  // Not the main th
                boolean stopped = xmagicState == XMagicImpl.XmagicState.STOPPED;
                if (stopped || xmagicState == XMagicImpl.XmagicState.DESTROYED) {
```

```
                    if (mXMagic != null) {
                        mXMagic.onDestroy();
                    }
                }
                if (xmagicState == XMagicImpl.XmagicState.DESTROYED) {
                    TXUGCRecord.getInstance(UGCKit.getAppContext()).setVideoProces
                }
            }
        }
    });
}
```

4. **Pause/Terminate the SDK**:

`onPause()` is used to pause effects, which can be implemented in an Activity/Fragment lifecycle method. The `onDestroy` method needs to be called in an OpenGL thread (you can call `onDestroy()` of the `XMagicImpl` object in `onTextureDestroyed`). For more information, see `onTextureDestroyed` in the demo.

```
        @Override
        public void onTextureDestroyed() {
            if (Looper.getMainLooper() != Looper.myLooper()) {  // Not the main th
                boolean stopped = xmagicState == XMagicImpl.XmagicState.STOPPED;
                if (stopped || xmagicState == XMagicImpl.XmagicState.DESTROYED) {
                    if (mXMagic != null) {
                        mXMagic.onDestroy();
                    }
                }
                if (xmagicState == XMagicImpl.XmagicState.DESTROYED) {
                    TXUGCRecord.getInstance(UGCKit.getAppContext()).setVideoProces
                }
            }
        }
```

5. **Add layout for the effect panel:**

```
<RelativeLayout
    android:id="@+id/panel_layout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:visibility="gone"/>
```

6. **Create an effect object and add the effect panel:**

```
private void initXMagic() {
    if (mXMagic == null) {
```

```
            mXMagic = new XMagicImpl(mActivity, getBeautyPanel());
        } else {
            mXMagic.onResume();
        }
    }
```

For detailed directions, see the `UGCKitVideoRecord` class of the demo.

Virtual Avatars
iOS
Integrating Avatars

Last updated：2023-02-27 14:18:15

Avatars are a capability of the Tencent Effect SDK. To use it, you need to integrate the SDK first and then add the avatar materials. For how to integrate the SDK, see Integrating Tencent Effect SDK.

Step 1. Copy the Avatar Materials

1. Integrate the Tencent Effect SDK.
2. Download the demo project from our website and decompress it.
3. Copy `BeautyDemo/bundle/avatarMotionRes.bundle` in the demo to your project.

Step 2. Integrate the Demo UI

Directions

1. Do the following to use the same avatar UI in your project as that in the demo:
2. Copy all the classes in the `BeautyDemo/Avatar` folder of the demo to your project and add the following code:

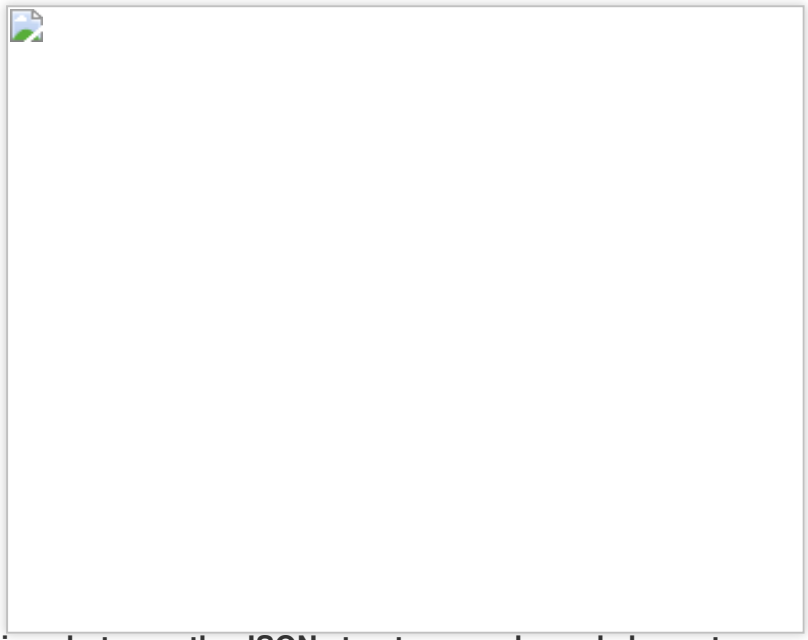
```
AvatarViewController *avatarVC = [[AvatarViewController alloc] init];
avatarVC.modalPresentationStyle = UIModalPresentationFullScreen;
avatarVC.currentDebugProcessType = AvatarPixelData; // Image or texture ID
[self presentViewController:avatarVC animated:YES completion:nil];
```

Demo UI

1. Demo UI view

2. Using the demo UI

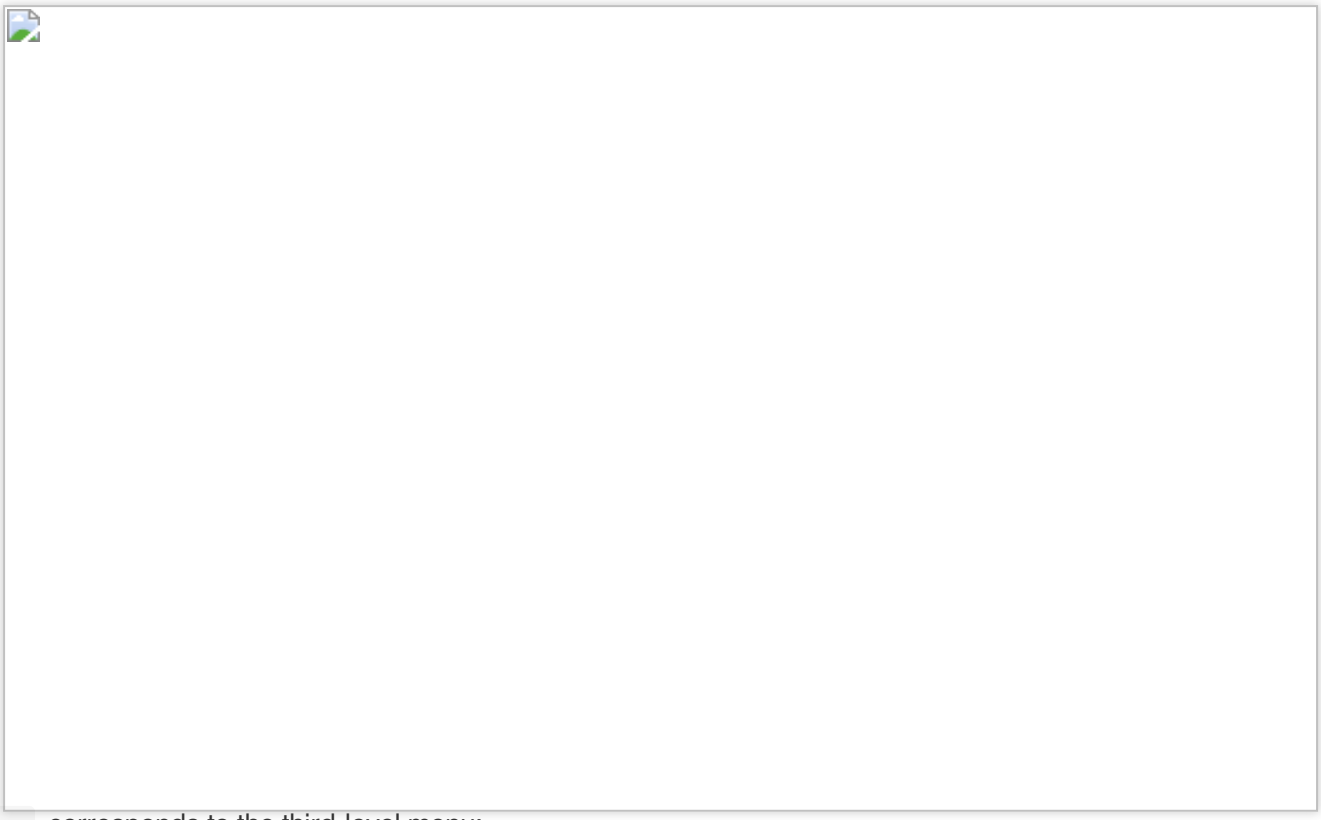
The data of the operation panel is obtained by parsing a JSON file. You can find this file in the `BeautyDemo/Avatar/` directory of the demo.



Mappings between the JSON structures and panel elements

`head` corresponds to the first icon of the top-level menu:

`subTabs` corresponds to the second-level menu:

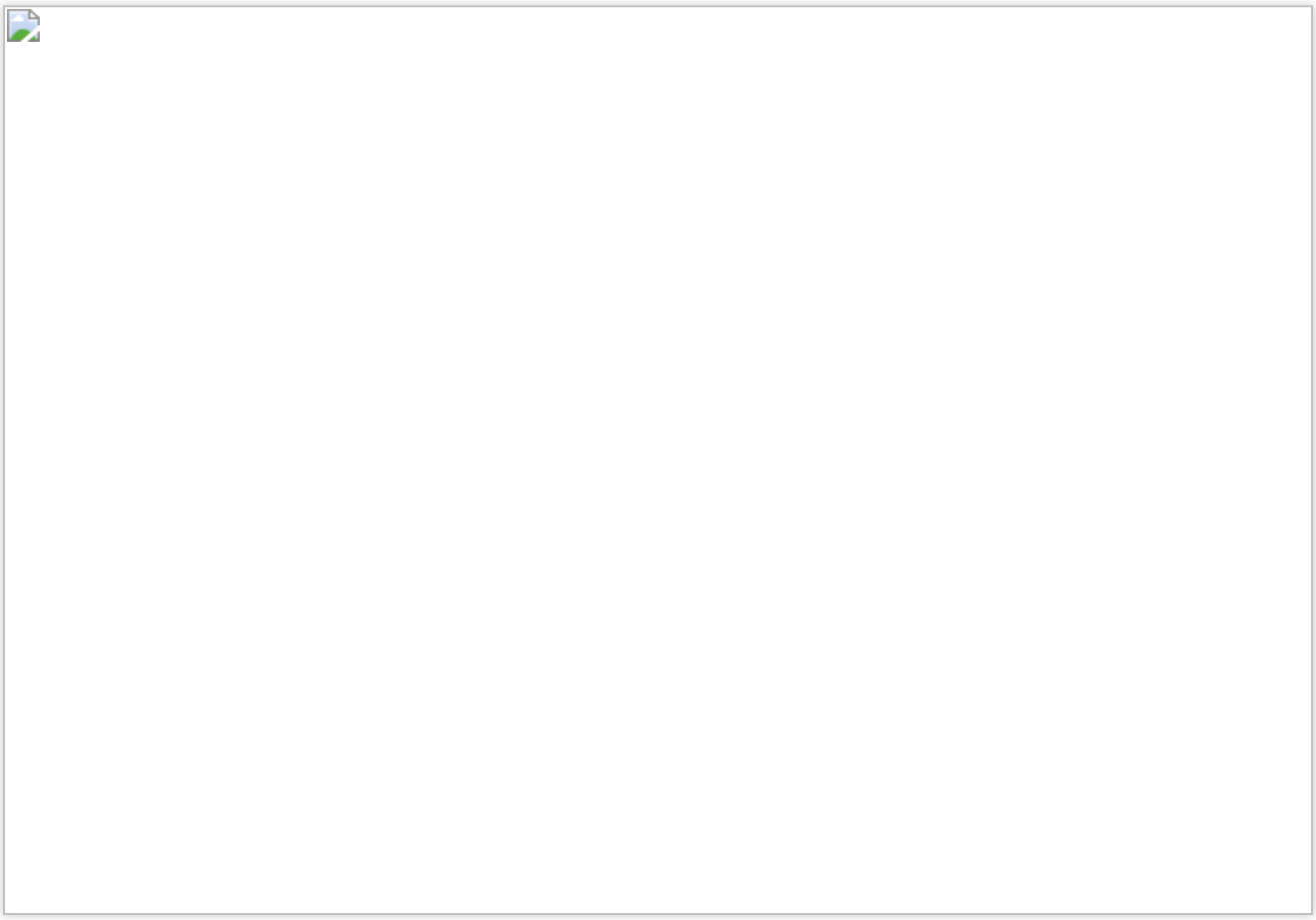


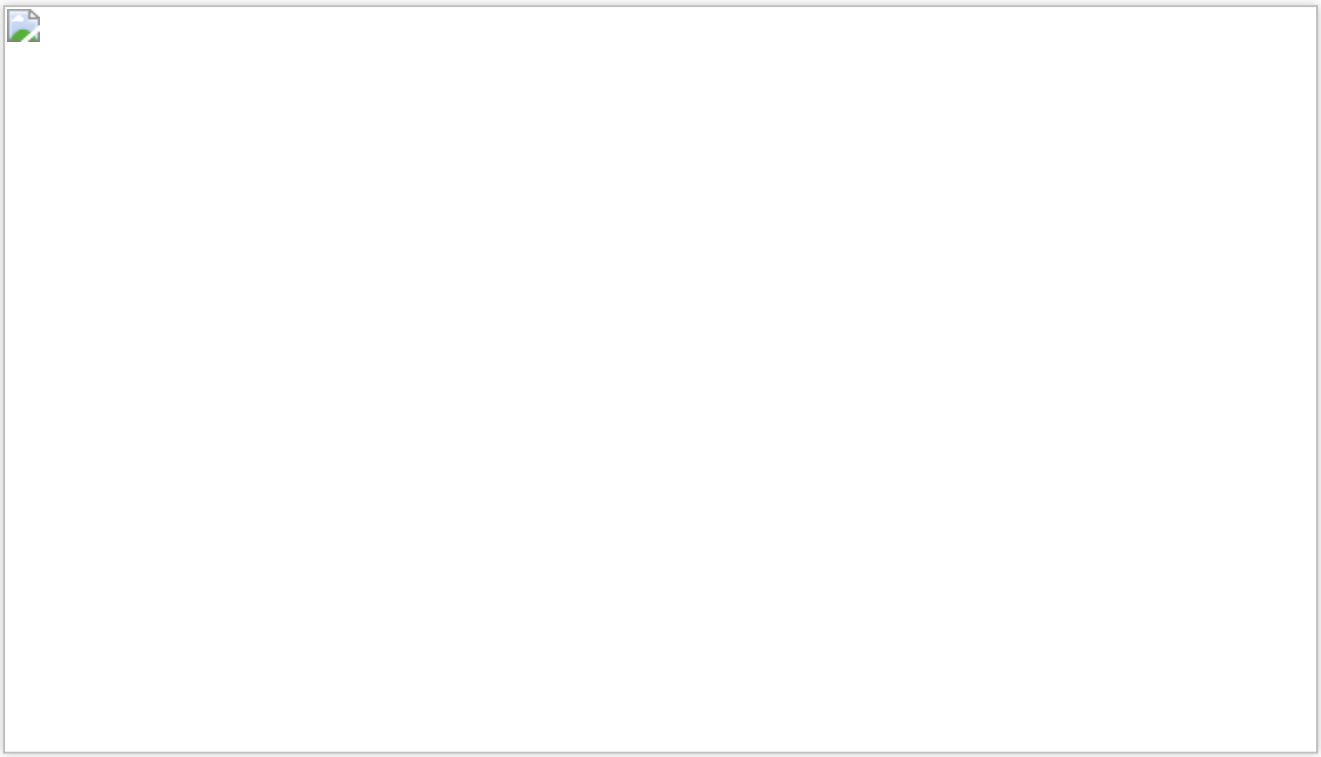
`items` corresponds to the third-level menu:

Using the panel data to determine the avatar object data passed in to the SDK API

The first screenshot below is the avatar dictionary obtained by the SDK (`key` indicates the category and `value` is an avatar data array). The second screenshot is the panel data. When the user taps an icon on the panel, get the

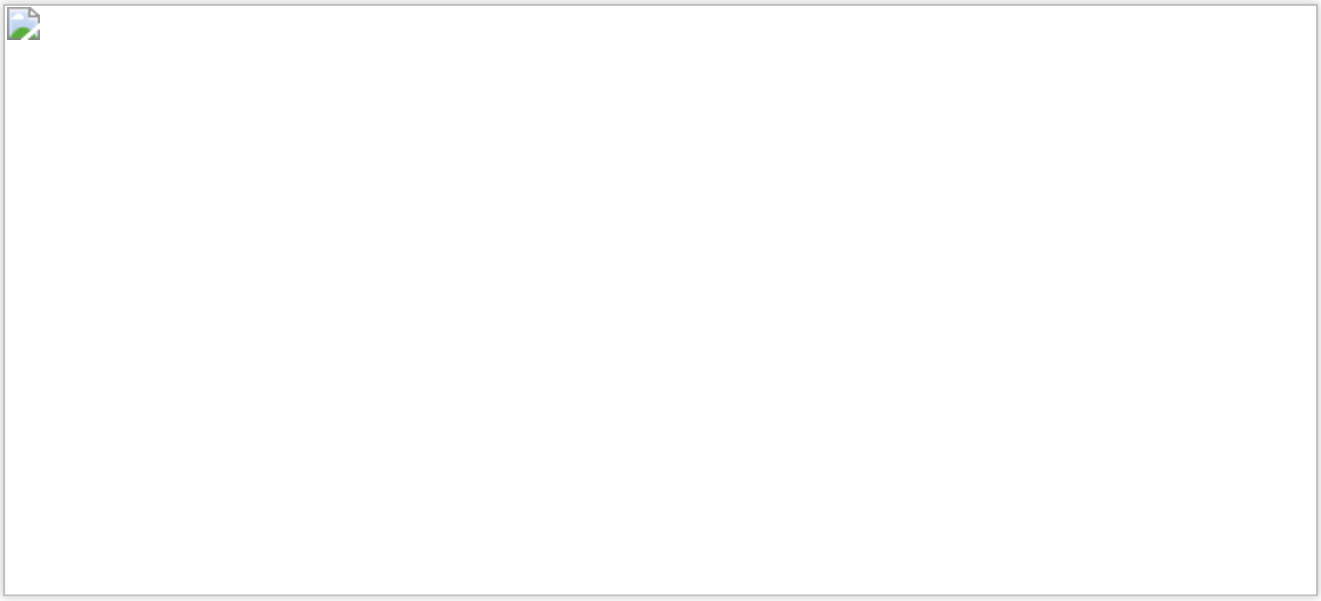
category from the second-level heading (**red box**), and in the avatar dictionary returned by the SDK, get the avatar data array of the category. Get the ID from the third-level heading (**blue box**), and then find the avatar object in the avatar data array of the category. Pass the object to the `updateAvatar` API of the SDK to edit the avatar.





3. Changing icons/headings

You can modify the above JSON file of the demo to change an icon or heading of the UI. For example, to change the **head** icon in the top-level menu, just modify the value of `iconUrl` or `checkedIconUrl`.



Step 3. Customize Avatar Features

You can refer to the **AvatarViewController** code in `BeautyDemo/Avatar/Controller`.

Note

For details about avatar APIs, see Avatar APIs.

1. Create an `xmagic` object and configure the default avatar template.

```
- (void)buildBeautySDK {

    CGSize previewSize = CGSizeMake(kPreviewWidth, kPreviewHeight);
    NSString *beautyConfigPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDir
    beautyConfigPath = [beautyConfigPath stringByAppendingPathComponent:@"beauty_co
    NSFileManager *localFileManager=[[NSFileManager alloc] init];
    BOOL isDir = YES;
    NSDictionary * beautyConfigJson = @{};
    if ([localFileManager fileExistsAtPath:beautyConfigPath isDirectory:&isDir] &&
            NSString *beautyConfigJsonStr = [NSString stringWithContentsOfFile:beau
            NSError *jsonError;
            NSData *objectData = [beautyConfigJsonStr dataUsingEncoding:NSUTF8Strin
            beautyConfigJson = [NSJSONSerialization JSONObjectWithData:objectData

    }
    NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                                                                @"root_path":[[NSBundl
                                                                @"tnn_"
                                                                @"beauty_config":beaut
    };
    // Init beauty kit
    self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:asse
    // Register log
    [self.beautyKit registerSDKEventListener:self];
    [self.beautyKit registerLoggerListener:self withDefaultLevel:YT_SDK_ERROR_LEVEL

    // Pass in the path of the avatar materials to load the default avatar
    AvatarGender gender = self.genderBtn.isSelected ? AvatarGenderFemale : AvatarGe
    NSString *bundlePath = [self.resManager avatarResPath:gender];
    [self.beautyKit loadAvatar:bundlePath exportedAvatar:nil];

}
```

2. Get the source data of the avatar materials

```
@implementation AvatarViewController
_resManager = [[AvatarResManager alloc] init];
NSDictionary *avatarDict = self.resManager.getMaleAvatarData;
@end


@implementation AvatarResManager

- (NSDictionary *)getMaleAvatarData
{
```

```
    if (!_maleAvatarDict) {
            NSString *resDir = [self avatarResPath:AvatarGenderFemale];
            NSString *savedConfig = [self getSavedAvatarConfigs:AvatarGenderMale];
            // Call an API of the SDK to parse the source data
            _maleAvatarDict = [XMagic getAvatarConfig:resDir exportedAvatar:savedCo
    }
    return _maleAvatarDict;
}
@end
```

3. Edit the avatar

```
// Get the desired avatar object from the material source data parsed by the SDK AP
NSMutableArray *avatars = [NSMutableArray array];
// `avatarConfig` is an avatar object obtained by the `getAvatarConfig:exportedAvat
[avatars addObject:avatarConfig];
// Call the following API to edit the avatar (face editing, dress up) in real time
[self.beautyKit updateAvatar:avatars];
```

4. Export the avatar object as a string

Export the configured avatar object as a string. You can store it in a custom location.

```
- (BOOL)saveSelectedAvatarConfigs:(AvatarGender)gender
{
    NSMutableArray *avatarArr = [NSMutableArray array];
    NSDictionary *avatarDict = gender == AvatarGenderMale ? _maleAvatarDict : _fema
    // 1. Traverse to find the selected avatar object.
    for (NSArray *arr in avatarDict.allValues) {
            for (AvatarData *config in arr) {
                    if (config.type == AvatarDataTypeSelector) {
                            if (config.isSelected) {
                                    [avatarArr addObject:config];
                            }
                    } else {
                            [avatarArr addObject:config];
                    }
            }
    }
    // 2. Call an SDK API to export the selected avatar object as a string.
    NSString *savedConfig = [XMagic exportAvatar:avatarArr.copy];
    if (savedConfig.length <= 0) {
            return NO;
    }
    NSError *error;
    NSString *fileName = [self getSaveNameWithGender:gender];
    NSString *savePath = [_saveDir stringByAppendingPathComponent:fileName];
    // Check whether the directory exists; if not, create one.
```

```
    BOOL isDir;
    if (![[NSFileManager defaultManager] fileExistsAtPath:_saveDir isDirectory:&isD
            [[NSFileManager defaultManager] createDirectoryAtPath:_saveDir withInte
    }
    // 3. Write the exported string to the sandbox for subsequent use.
    [savedConfig writeToFile:savePath atomically:YES encoding:NSUTF8StringEncoding
    if (error){
            return NO;
    }
    return YES;
}
```

5. Use a virtual background

```
- (void)bgExchangeClick:(UIButton *)btn
{
    btn.selected = !btn.isSelected;
    NSDictionary *avatarDict = self.resManager.getFemaleAvatarData;
    NSArray *array = avatarDict[@"background_plane"];
    AvatarData *selConfig;
    // A background is also an avatar object (its category is `background_plane`).
    for (AvatarData *config in array) {
            if ([config.Id isEqual:@"none"]) {
                    if (btn.selected) {
                            selConfig = config;
                            break;
                    }
            } else {
                    selConfig = config;
            }
    }
    [self.beautyKit updateAvatar:@[selConfig]];
}
```

Avatar APIs

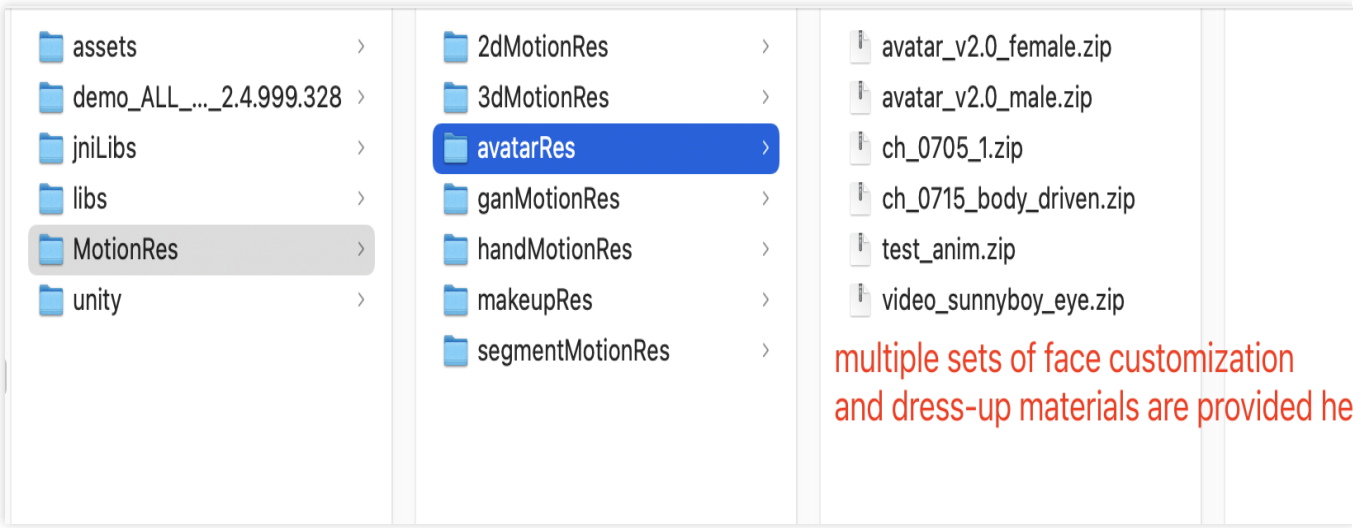
Last updated：2023-04-21 17:45:08

SDK Integration

For how to download and integrate the SDK, how to set the license, as well as how to run a demo project, see Integrating Tencent Effect SDK (iOS).

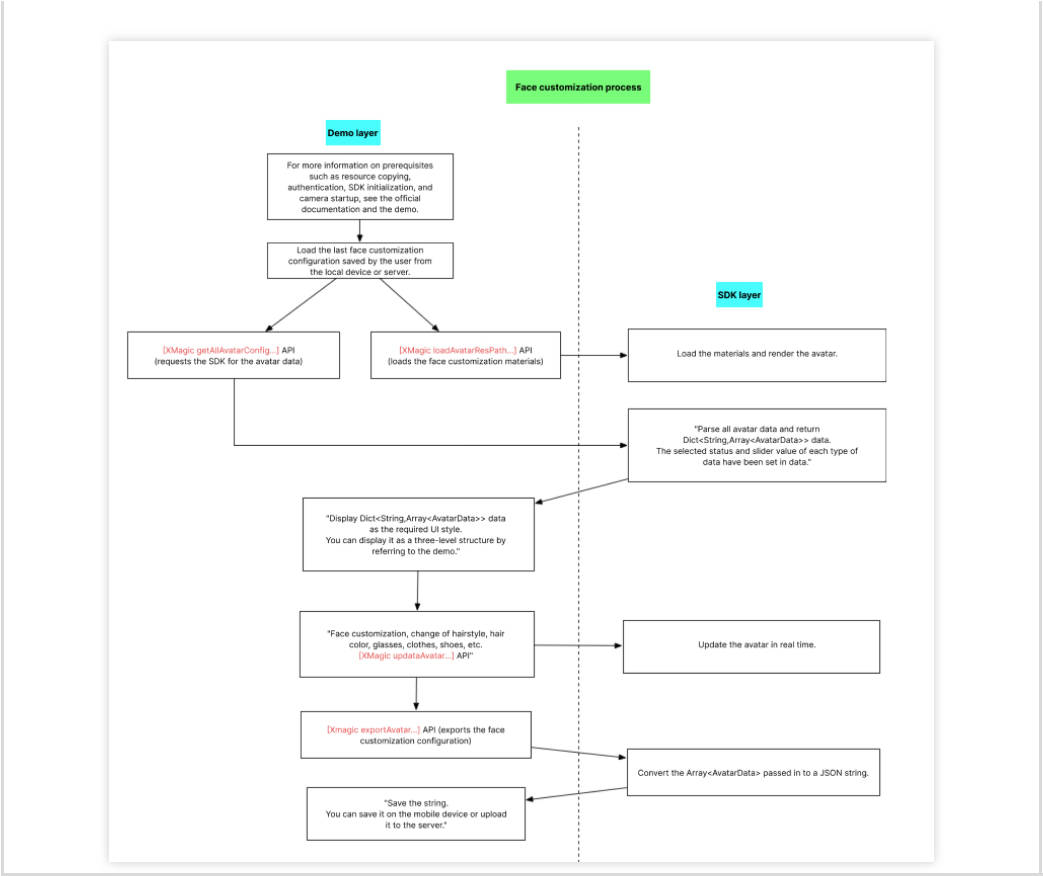
Copying the Avatar Materials

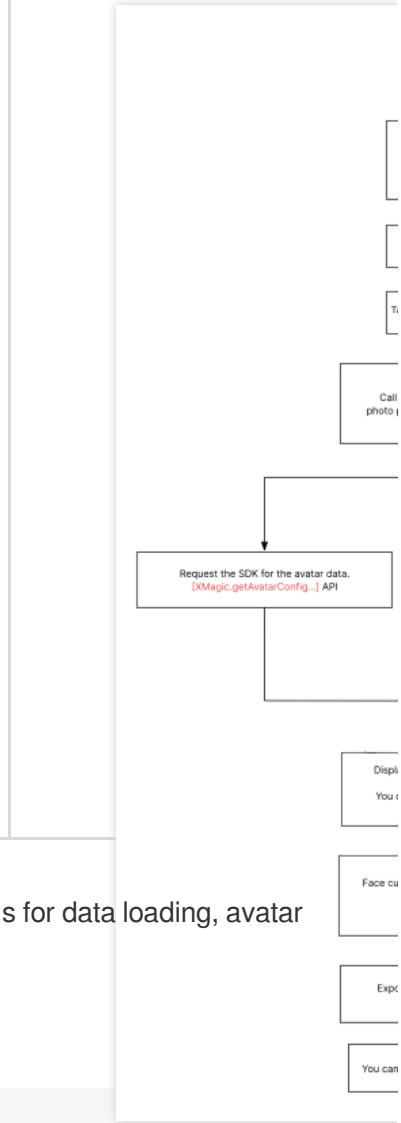
We offer multiple sets of face customization and dress-up materials, which can be found in the `MotionRes/avatarRes` directory in the SDK package after decompression. Like other animated effect materials, you need to copy them to the `assets` directory of your project.



Avatar Customization and SDK APIs

Face customization process	Photo-based face customizati





The section below offers descriptions of the APIs of `XMagicApi`, including the APIs for data loading, avatar customization, and photo-based avatar customization.

1. Get the avatar source data (`getAvatarConfig`)

```
+ (NSDictionary <NSString *, NSArray *>* _Nullable)getAvatarConfig:(NSString * _Nul
```

Input parameters:

`resPath` : The absolute path of avatar materials on the user's mobile device, such as `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` .

`exportedAvatar` : The avatar configuration data (a JSON string) saved from the last time the user customized an avatar. If an avatar is customized for the first time or no configuration data is saved, pass in `nil` .

Output parameters:

This API returns an `NSDictionary` object, in which `key` is the category (for details, see the `TEDefine` class below) and `value` is the data of the category. After the application layer gets the dictionary data, display it on the UI as needed.

2. Load the avatar materials (`loadAvatar`)

```
- (void)loadAvatar:(NSString * _Nullable)resPath exportedAvatar:(NSString * _Nullab
```

Input parameters:

`resPath` : The absolute path of avatar materials on the user's mobile device, such as `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` .

`exportedAvatar` : The data saved after the user's last face customization, which is a JSON string. If the user uses the face customization feature for the first time or no data has been saved before, the parameter value will be `nil` .

Output parameters:

This API returns an `NSDictionary` object, in which `key` is the category (for details, see the `TEDefine` class below) and `value` is the data of the category. After the application layer gets the dictionary data, display it on the UI as needed.

3. Customize the face and dress up (`updateAvatar`)

```
- (void)updateAvatar:(NSArray<AvatarData *> *_Nonnull)avatarDataList;
```

When this API is called, the avatar preview will be updated in real time. Each `AvatarData` object corresponds to a configuration (such as changing the hairstyle). You can pass multiple `AvatarData` objects to an API call to edit multiple aspects of an avatar, for example, change the hairstyle and hair color. The API will validate the `AvatarData` objects passed in. If an object is valid, it will be passed to the SDK; if not, a callback will be sent. For example, if an `AvatarData` object is passed in to change the hairstyle, but the hair model file (specified by `value` in `AvatarData`) cannot be found on the local device, the `AvatarData` object will be considered invalid.

Also, if an `AvatarData` object is passed in to change the iris image, but the image file (specified by `value` in `AvatarData`) cannot be found on the local device, the `AvatarData` object will be considered invalid.

4. Export avatar settings (`exportAvatar`)

```
+ (NSString *_Nullable)exportAvatar:(NSArray <AvatarData *>*_Nullable)avatarDataLis
```

When the user edits their avatar, the value of `selected` or shape values in `AvatarData` will change. After editing, a new `AvatarData` list will be generated and can be exported as a JSON string. You can save it locally or upload it to the server.

The string is exported for the following purposes:

The next time you call **loadAvatar** of Xmagic to load the avatar materials, you need to set **exportedAvatar** to the JSON string so that the preview will remember the avatar settings from the last time.

Also, you need to pass in this JSON string when you call `getAllAvatarData` to update `selected` and the shape values in the avatar source data.

5. Customize an avatar based on a photo (`createAvatarByPhoto`)

For this API to work, the SDK must be connected to the internet.

```
+ (void)createAvatarByPhoto:(NSString * _Nullable)photoPath avatarResPaths:(NSArray
```

photoPath: The photo path. Make sure the face is in the center of the photo. Ideally, the photo should include only one face. If there are multiple, the SDK will select one randomly. To ensure the recognition results, the short side of the photo should preferably be longer than 500 px.
avatarResPaths: You can pass in multiple sets of avatar materials, and the SDK will select the most suitable set based on the photo analysis result.
Note
Currently, only one set of materials is supported. If multiple sets are passed in, the SDK will use the first one.
isMale: Whether the person is male. Currently, this property is not used. However, it may be in the future. We recommend that you pass in a correct value.
success: The callback for successful configuration, in which `matchedResPath` indicates the path of the matched materials and `srcData` indicates the matching result (same as the return value of `exportAvatar`).
failure: The callback when configuration fails, in which `code` indicates the error code and `msg` indicates the error message.

6. Save the downloaded configuration file (`addAvatarResource`)

```
+ (void)addAvatarResource:(NSString * _Nullable)rootPath category:(NSString * _Null
```

This API is used if avatar materials are downloaded dynamically. For example, suppose you offer 10 hairstyles and one of them is dynamically downloaded to a device. You need to call this API to pass the path of the downloaded ZIP file to the SDK. The SDK will parse the file and save it to the folder of the corresponding category. When you call `getAllAvatarData` the next time, the SDK will return the newly added data.
Parameters:
rootPath: The root directory of avatar materials, such as `/var/mobile/Containers/Data/Application/C82F1F7A-01A1-4404-8CF6-131B26B4DA1A/Library/Caches/avatarMotionRes.bundle/avatar_v2.0_male` .
category: The category of the downloaded material.
zipFilePath: The local path of the downloaded ZIP file.
completion: The result callback, in which `error` indicates the error message, and `avatarList` indicates the avatar data array obtained after parsing.

7. Send a custom event (`sendCustomEvent`)

This API is used to send a custom event, for example, to display the idle status when no face is detected.

```
- (void)sendCustomEvent:(NSString * _Nullable)eventKey eventValue:(NSString * _Null
```

eventKey: The key of the custom event. For details, see `AvatarCustomEventKey` of `TEDefine`.

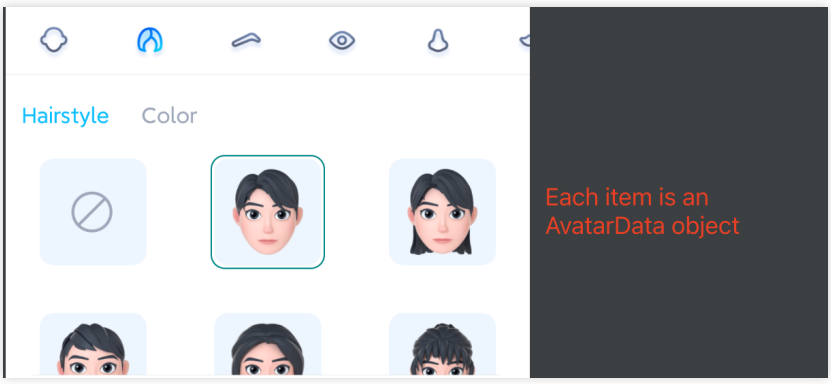
eventValue: The value of the custom event, which is a JSON string. For example, you can convert `@{@"enable" : @(YES)}` to a JSON string or directly enter `@"{\\\"enable\\\" : true}"`.

8. Call `AvatarData`

The `AvatarData` class is at the core of the above APIs. `AvatarData` contains the following fields:

```
/// @brief The configuration type.
@interface AvatarData : NSObject
// Such as face shape changing or eye adjustment. If the standard categories define
@property (nonatomic, copy) NSString * _Nonnull category;
// The ID of an avatar configuration item. For example, each type of glasses has a
@property (nonatomic, copy) NSString * _Nonnull Id;
// The type, which can be `selector` or `AvatarDataTypeSlider`.
@property (nonatomic, assign) AvatarDataType type;
/// If `type` is `selector`, this field indicates whether the item is selected.
@property (nonatomic, assign) BOOL isSelected;

/// The part of an avatar to be edited, for example, face, eyes, hair, shirt, or sh
@property (nonatomic, copy) NSString * _Nonnull entityName;
/// The action to be performed on `entityName`. For details, see our documentation.
@property (nonatomic, copy) NSString * _Nonnull action;
/// The details of the action to be performed on `entityName`. For details, see our
@property (nonatomic, copy) NSDictionary * _Nonnull value;
@end
```

An `AvatarData` object is the smallest unit of configuration, for example, hairstyle change or cheek adjustment.

Hairstyle change	Cheek adjustment
	

If an item is the selector type, configure it by changing the value of `selected` in `AvatarData`. For example, suppose there are four types of glasses: A, B, C, and D. If the user selects A, set `selected` of A to `true` and that of B, C, and D to `false`. If the user selects B, set `selected` of B to `true` and that of A, C, and D to `false`.
If an item is the slider type, configure it by changing `value` in `AvatarData`. The `value` field is a JSON object that includes multiple key-value pairs. Set the `value` of a key-value pair to the slider value.

More About `AvatarData`

The SDK gets `AvatarData` by parsing the `custom_configs` directory of the material root directory and sends it to the application layer. Generally, you don't need to manually construct `AvatarData`.
The three key fields of `AvatarData` are entityName, action, and value, whose values are automatically entered when the SDK parses the configuration data. In most cases, you won't need to deal with the details of these three fields. However, for slider data, you need to parse the key-value pairs in the `value` field and configure them based on the slider values set on the UI.

`entityName`

`entityName` specifies which part of an avatar is to be edited, for example, face, eyes, hair, shirt, or shoes.

`action` and `value`

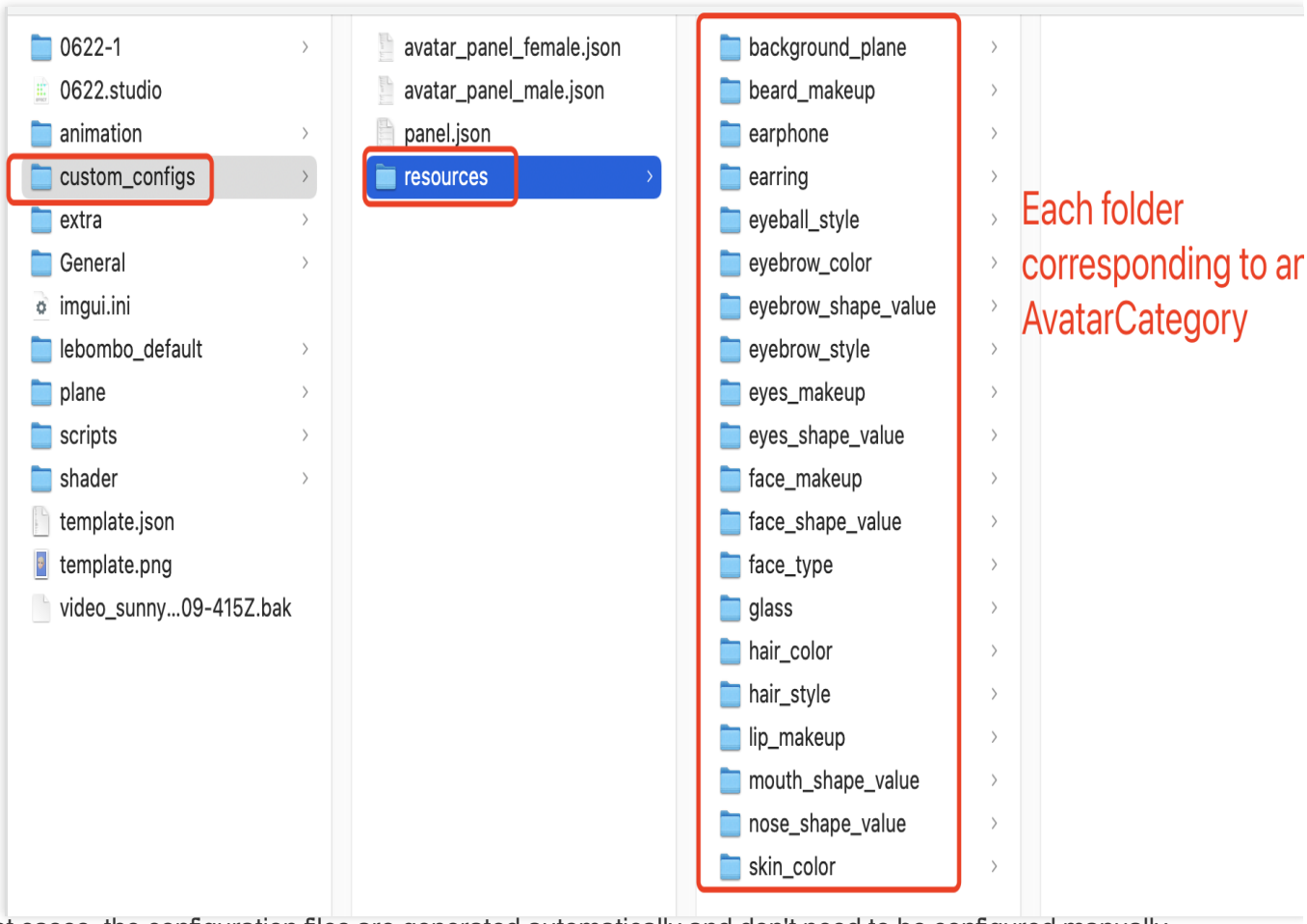
The `action` field indicates the action to be performed on `entityName`. Five `action` options are defined in the SDK, which are detailed below:

action	Description	Requirements for `value`
changeColor	Changes the color of the current material. Color attributes include basic color and emission color.	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.
changeTexture	Modifies the maps of the current material, including color texture map, metal/roughness texture map, ambient occlusion (AO) map, normal map, and emission map.	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.
shapeValue	Modifies the shape change value of `blendShape`. This parameter is generally used for slight adjustment of detailed facial features.	This field is of `JsonObject` type and is required. Its `key` is the shape change name and `value` is of `float` type. It is generated automatically by the material customization tool.

replace	Replaces a submodel, for example, glasses, hairstyles, or clothes.	`value` must be a JSON object that describes the 3D transformation information, model path, and material path of the new submodel. It is generated automatically by the material customization tool. To hide a submodel, set it to `null` .
basicTransform	Adjusts the position, rotation, and scaling settings. This field is generally used to adjust the camera distance and angle so as to switch between the full and half-length views of the model.	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.

Configuring Avatar Customization Data

Avatar configurations are stored in the `resources` folder (in `material/custom_configs/resources`):



In most cases, the configuration files are generated automatically and don't need to be configured manually.

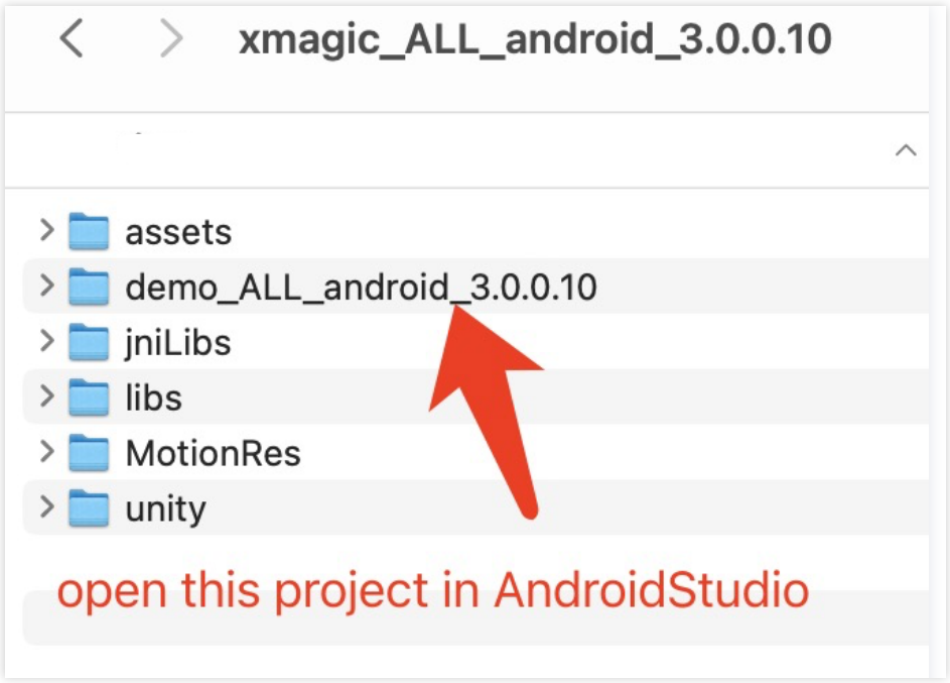
To automatically generate the configurations, design your materials using Tencent Effect Studio and run the "resource_generator_gui" app we provide (only available on macOS currently). For details, see Design Specifications.

Android

Run demo

Last updated：2024-02-19 17:15:02

1. Open the project in Android Studio.。

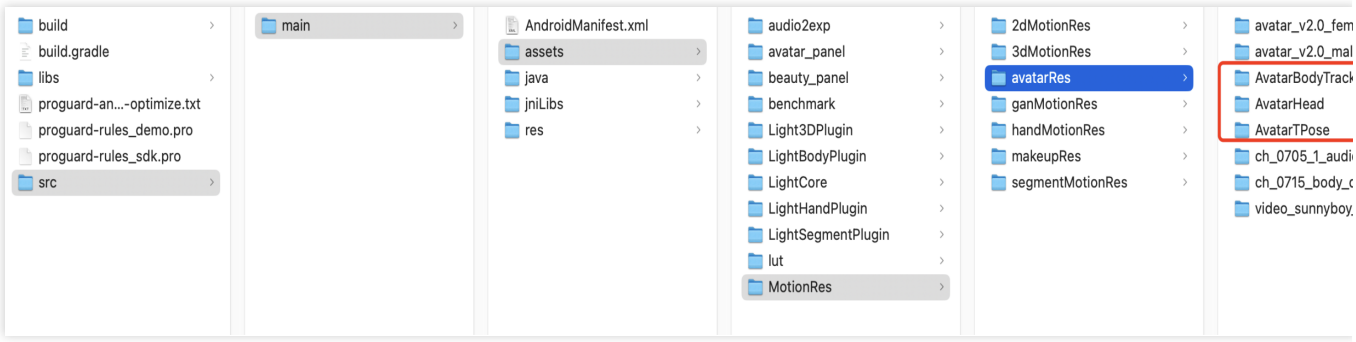


2. Set your own license information in the `com.tencent.demo.constant.LicenseConstant.java` class of the project.

3. Modify the package name to your own package name in the build.gradle file under the project.

```
cal.properties  build.gradle (:demo)
    dependencies{}
can use the Project Structure dialog to view and edit your project configuration
plugins {
    id 'com.android.application'
}

android {
    compileSdkVersion 29
    buildToolsVersion "29.0.3"

    defaultConfig {
        long time = System.currentTimeMillis()
        applicationId "com.tencent.pitumotiondemo.effects"
        minSdkVersion 21
        targetSdkVersion 29
        versionCode 1
        versionName String.valueOf(time)
        ndk {
            abiFilters "armeabi-v7a" , "arm64-v8a"
        }
```

4. Replace the authorized materials: Some Avatar materials in the demo project (the three materials circled in red in the figure below) require authorization to use. Please contact us to obtain authorized materials.



5. Complete the above steps to run the demo。

Integrating Avatars

Last updated：2023-05-18 10:33:24

Avatars are a capability of the Tencent Effect SDK. When integrating it, please refer to the Integrating Tencent Effect SDK to integrate the SDK into your project. Then, follow the method below to add Avatar UI and load Avatar materials.
1. Integrate the SDK. For detailed directions, see Integrating Tencent Effect SDK.
2. Follow the steps below to load the avatar materials.

Directions

Step 1. Copy demo files

1. Download the demo project from our website and decompress it.
2. Add Avatar resources: Contact us to obtain the Avatar resource authorization, and then copy the authorized resources to your project (in the same location as the Demo: demo/app/assets/MotionRes/avatarRes). As the Avatar resources in demo/app/assets/MotionRes/avatarRes of the Demo are encrypted and authorized, customers cannot use them directly.
3. Copy all the classes in the `com.tencent.demo.avatar` folder of the demo to your project.

Step 2. Add code

Add the following code (you can refer to the `com.tencent.demo.avatar.AvatarEditActivity` class of the demo):
1. Configure panel information in the XML file of your UI.

```
<com.tencent.demo.avatar.view.AvatarPanel
        android:id="@+id/avatar_panel"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        app:layout_constraintBottom_toBottomOf="parent" />
```

2. Get the panel object on your UI and configure the callbacks:

```
avatarPanel.setAvatarPanelCallBack(new AvatarPanelCallBack() {

            @Override
            public void onReceiverBindData(List<AvatarData> avatarData) {
               mXMagicApi.updateAvatar(avatarData, AvatarEditActivity.this);
            }

            @Override
            public void onItemChecked(MainTab mainTab, AvatarItem avatarItem) {
```

```
                if (avatarItem.avatarData == null && URLUtil.isNetworkUrl(avatarIte
                    downloadAvatarData(avatarItem, () -> updateConfig(avatarItem));
                } else {
                    updateConfig(avatarItem);
                    List<AvatarData> bindAvatarData = AvatarResManager.getAvatarDat
                    mXmagicApi.updateAvatar(bindAvatarData, AvatarActivity.this);
                }
            }

            @Override
            public void onItemValueChange(AvatarItem avatarItem) {
                updateConfig(avatarItem);
            }

            @Override
            public boolean onShowPage(AvatarPageInf avatarPageInf, SubTab subTab) {
                if (subTab != null && subTab.items != null && subTab.items.size() >
                    AvatarItem avatarItem = subTab.items.get(0);
                    if (avatarItem.type == AvatarData.TYPE_SLIDER && avatarItem.ava
                        downloadAvatarData(avatarItem, () -> {
                            if (avatarPageInf != null) {
                                avatarPageInf.refresh();
                            }
                        });
                        return false;
                    }
                }
                return true;
            }

            private void updateConfig(AvatarItem avatarItem) {
                if (mXmagicApi != null && avatarItem != null) {
                    List<AvatarData> avatarConfigList = new ArrayList<>();
                    avatarConfigList.add(avatarItem.avatarData);
                    mXmagicApi.updateAvatar(avatarConfigList, AvatarActivity.this);
                }
            }
        });
```

3. Get and configure the panel data.

```
AvatarResManager.getInstance().getAvatarData(avatarResName, getAvatarConfig(), allD
      avatarPanel.initView(allData);
 });
```

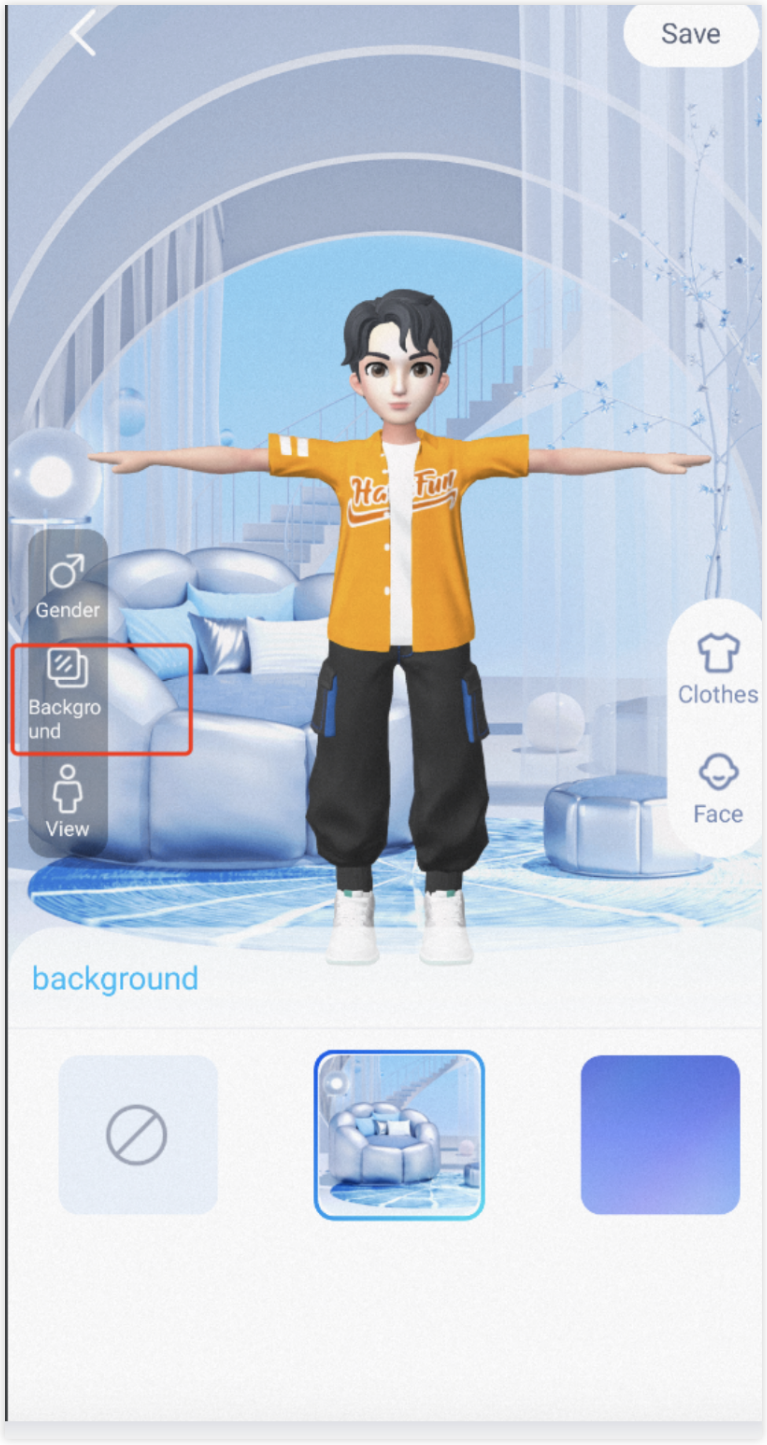
4. Create an `xmagicApi` object and load the avatar resources:

```
protected void initXMagicAndLoadAvatar(String avatarConfig, UpdatePropertyListene
    if (mXMagicApi == null && !isFinishing() && !isDestroyed()) {
        WorkThread.getInstance().run(() - > {
            synchronized(lock) {
                if (isXMagicApiDestroyed) {
                    return;
                }
                mXMagicApi = XmagicApiUtils.createXMagicApi(getApplicationContext(
                AvatarResManager.getInstance().loadAvatarRes(mXMagicApi, avatarRes
                setAvatarPlaneType();
            }
        }, this.hashCode());
    }
}
```

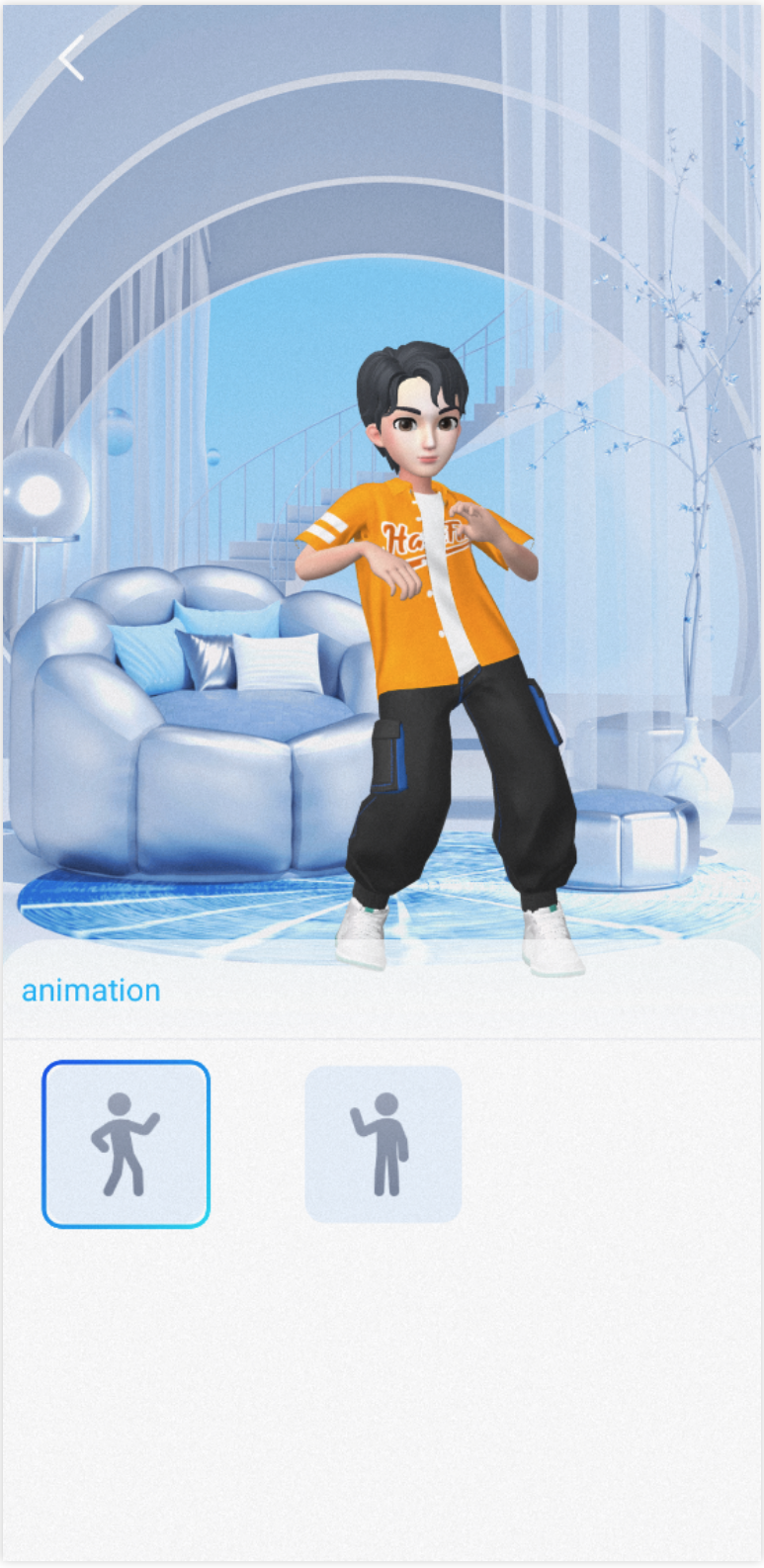
5. Save the avatar properties (you can refer to `saveAvatarConfigs` in the demo):

```
/**
    * Save the property values set by the user or the default property values.
*/
  public void onSaveBtnClick() {
    //Get the user-configured properties through getUsedAvatarData of AvatarResMana
        List < AvatarData > avatarDataList = AvatarResManager.getUsedAvatarData(ava
       //Convert the configured properties to a string through the XmagicApi.export
         String content = XmagicApi.exportAvatar(avatarDataList);
         //Save this string, and when downloading and using it, set this string to
        if (mXMagicApi != null) {
            mXMagicApi.exportCurrentTexture(bitmap - > saveAvatarModes(bitmap, cont
        }
    }
```

6. Change the background (you can refer to demo):



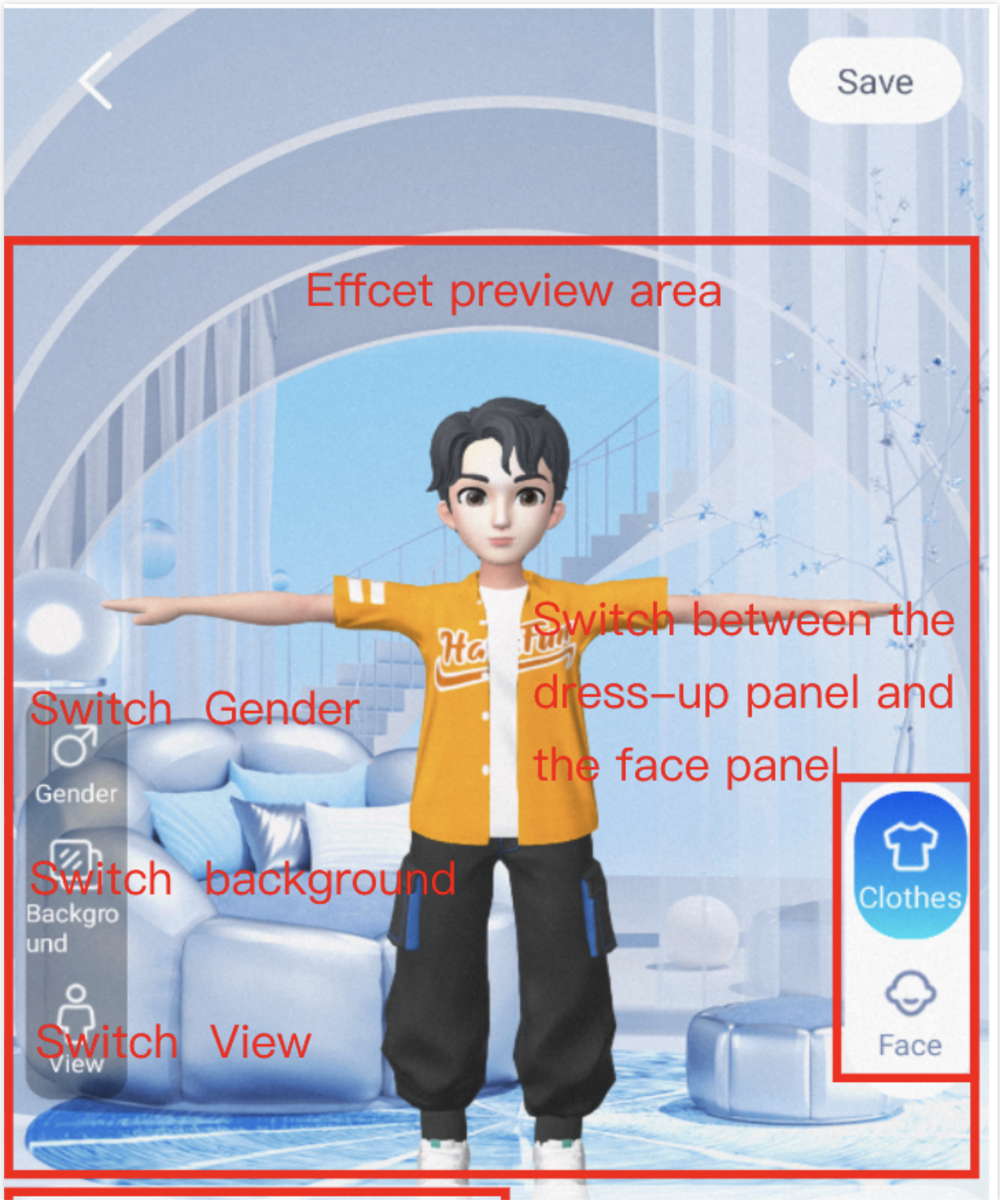
7. Refer to the interactive page in the Demo to set the model animation

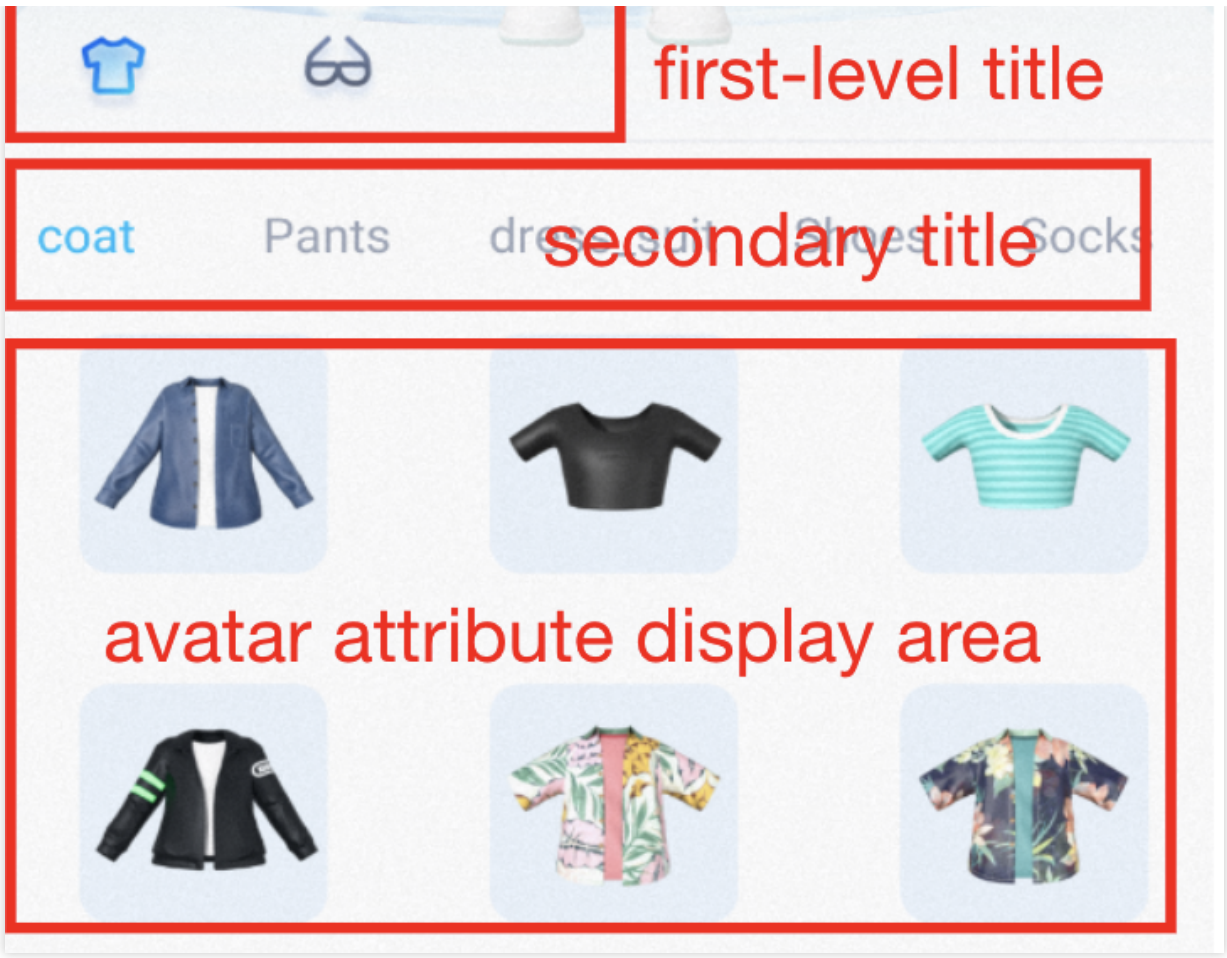


Custom Avatar UI

Last updated：2023-05-18 09:54:04

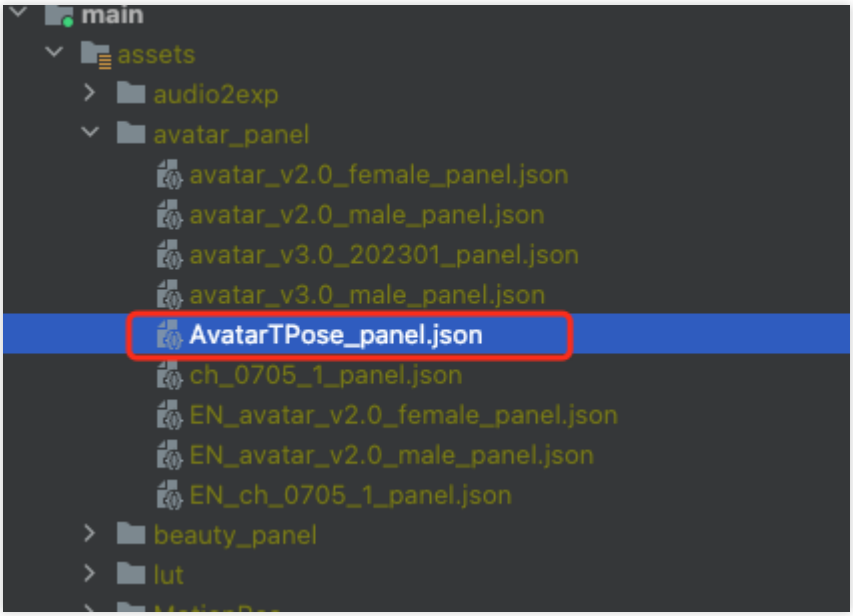
Demo UI





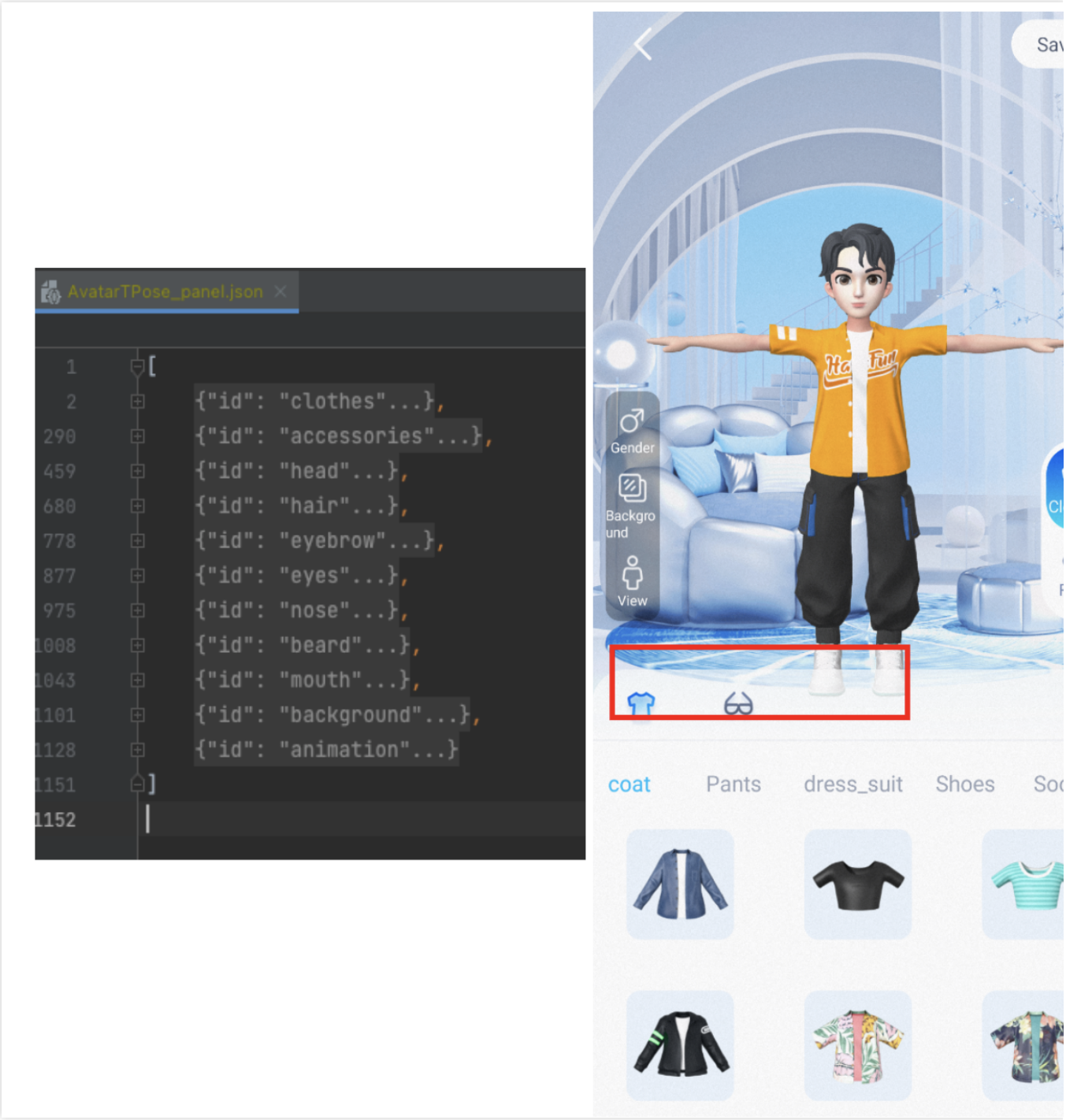
Using the demo UI

The panel configuration data can be stored anywhere. In the demo, it is in `assets`. When the panel file is used for the first time, it will be copied to the installation directory.

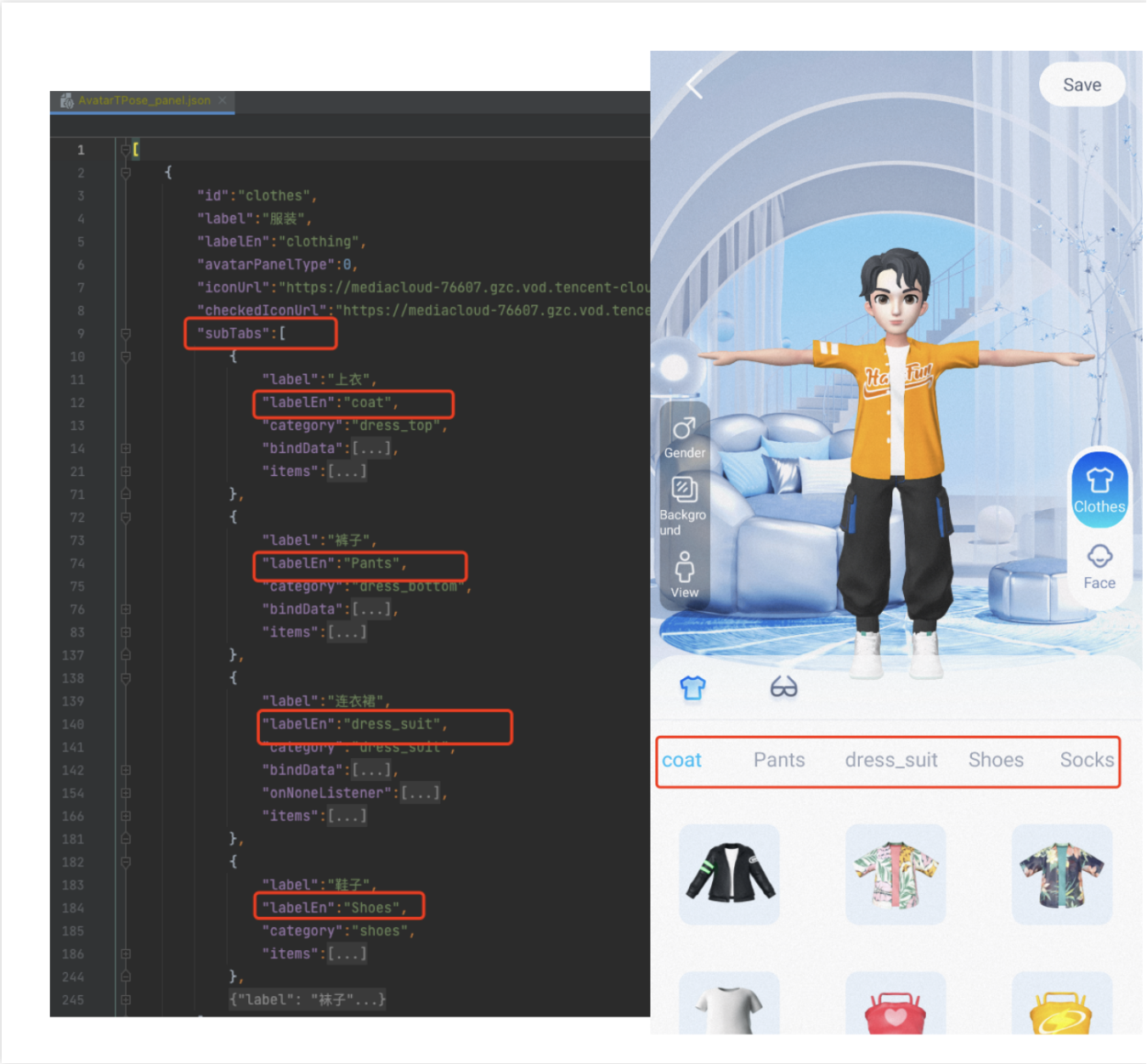


Mappings between the JSON structures and panel elements:

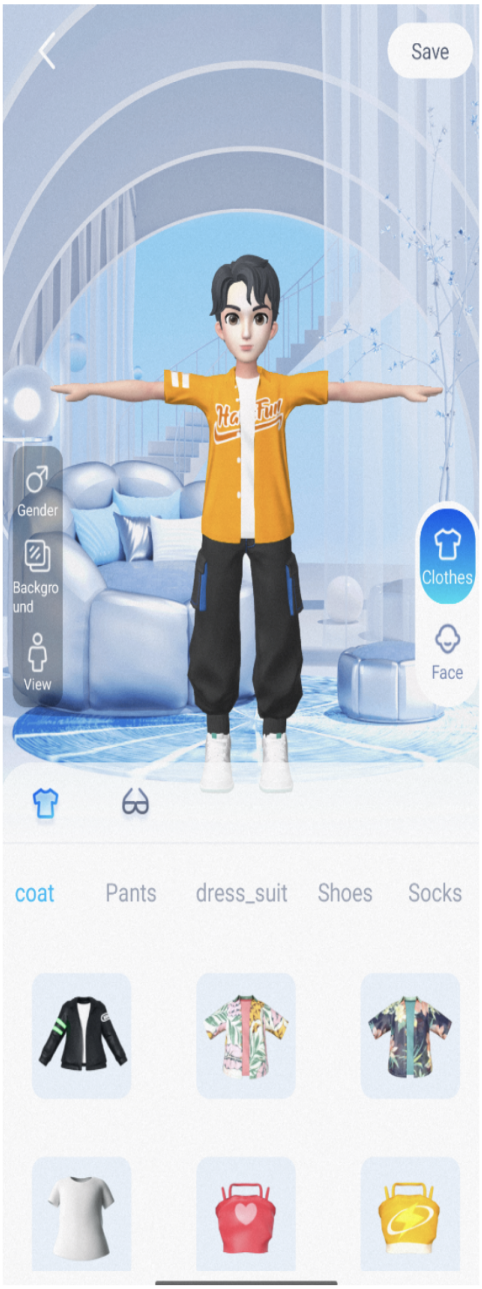
The items in the JSON file on the left correspond to the top-level menu on the right. `clothes` corresponds to the first item of the menu.



`subTabs` on the left corresponds to the second-level menu:



The AvatarData corresponding to the left icon is stored in the resources folder under the material, and the configuration data of the panel displayed on the right is associated with it through the category in the panel data. The SDK will parse the data in the resources folder and put it into the corresponding map, with the key of the map being the value of the category. Therefore, after parsing the panel.json file in the Demo, you can use the method provided by the SDK to obtain the data for association. You can refer to the getAvatarData method in AvatarResManager in the demo, which will parse the panel file and associate it with the attributes returned by the SDK.



Key Classes of the Demo

Path: `com.tencent.demo.avater.AvatarResManager.java`

1. Load the avatar resources

```
/**
```

```
     * Used to load the avatar resources
     *
     * @param xmagicApi: The `XmagicApi` object.
     * @param avatarResName: The name.
     * @param avatarSaveData: The default configuration of the loading model. If th
     */
    public void loadAvatarRes(XmagicApi xmagicApi, String avatarResName, String ava
```

2. Get the panel data

```
/**
     * Get the avatar panel data
     *
     * @param avatarResName: The avatar material name.
     * @param avatarDataCallBack: This API accesses files. The operation is perform
     *                            The returned data already contains the data in the
     */
    public void getAvatarData(String avatarResName, String avatarSaveData, LoadAvat
```

3 Get the user's settings or default settings from the panel data

```
// Get the user's settings or the default settings from the panel file
public static List<AvatarData> getUsedAvatarData(List<MainTab> mainTabList)
```

4. Parse the corresponding avatarData from bindData.

```
/**
     * Parse the corresponding avatarData from bindData.
     *
     * @param bindDataList
     * @return Return the corresponding list of avatarData.
     */
    public static List < AvatarData > getAvatarDataByBindData(Map < String, MainTab
```

Appendix

This section provides descriptions of fields in MainTab.java,SubTab.java,AvatarItem.java and BindData.java.

MainTab

Field	Type	Required	Description

id	String	Yes	The globally unique identifier of a main menu option.
label	String	No	The Chinese name of a main menu option (this is not displayed in the demo).
labelEn	String	No	The English name of a main menu option (this is not displayed in the demo).
avatarPanelType	Int	Yes	Panel type: 0: Dress-up panel 1: Face customization panel 2: Background panel 3: Action panel
iconUrl	String	Yes	The URL of the icon when not selected.
checkedIconUrl	String	Yes	The URL of the icon when selected.
subTabs	A list of SubTab	Yes	The options of the second-level menu.

SubTab

Field	Type	Required	Description
label	String	Yes	The Chinese name of a second-level menu option.
labelEn	String	Yes	The English name of a second-level menu option.
category	String	Yes	The category of a second-level menu option, which is defined in the `com.tencent.xmagic.avatar.AvatarCategory` class of the SDK.
type	int	Yes	UI type: 0: Represents the icon type, default value. 1: Represents the slider adjustment type.
bindData	A list of `BindData`	No	Configuration field for dependent properties. This field can be configured under the Subtab node or the AvatarItem node. Configuring it under the Subtab node means that all items under the Subtab node depend on the properties configured in this binData. Configuring it under the AvatarItem node means that only this item will depend on the configuration data in the binData. For example: There is a dependency between hairstyle and hair color.

			When modifying the hairstyle, the hair color that the user previously set needs to be used. In this case, this field needs to be set in the hairstyle. For the relationship between dresses and tops/pants, when setting a dress, the tops and pants need to be set to none, otherwise the page display will be abnormal. Therefore, this field can be configured under the dress node, please refer to the AvatarTPose_panel.json in the demo for details. There is a dependency between glasses and lenses, so the corresponding bindData field is configured under each glasses item to associate lens information.
onNoneListener	A list of `BindData`	No	Field explanation: Used when there is no selection in the items, the configuration information in this field is used. **For example:** for pants, tops, and dresses, this property is configured in the dress. When the user clicks on the dress, no item in the tops and pants is selected. However, when the customer clicks on the top again, at this time, a default pair of pants needs to be set for the avatar (otherwise the avatar will not have pants), and the default pants configured in this field can be parsed and set. Please refer to AvatarTPose_panel.json in the demo for details.
items	A list of AvatarItem	Yes	A list of AvatarIcon.

AvatarItem

Field	Type	Required	Description
id	String	Yes	The property ID, which corresponds to the ID in `AvatarData` returned by the SDK.
icon	String	Yes	The URL of the icon or the color in ARGB format (“#FF0085CF”).
type	Int	Yes	The display type. Valid values: `AvatarData.TYPE_SLIDER` (slider), `AvatarData.TYPE_SELECTOR` (icon).
selected	boolean	Yes	If `type` is `AvatarData.TYPE_SELECTOR`, this field indicates whether the item is selected.
downloadUrl	String	No	The dynamic download address of the configuration file.
category	String	Yes	Same as `category` in `SubTab`.

labels	Map<String, String>	No	The Chinese labels on the left of the panel. This field is not empty if `type` is `AvatarData.TYPE_SLIDER`.
enLabels	Map<String, String>	No	The English labels on the left of the panel. This field is not empty if `type` is `AvatarData.TYPE_SLIDER`.
bindData	A list of BindData	No	Configuration field for dependent properties. This field can be configured under the Subtab node or the AvatarItem node. Configuring it under the Subtab node means that all items under the Subtab node depend on the properties configured in this binData. Configuring it under the AvatarItem node means that only this item will depend on the configuration data in the binData. For example: There is a dependency between hairstyle and hair color. When modifying the hairstyle, the hair color that the user previously set needs to be used. In this case, this field needs to be set in the hairstyle. For the relationship between dresses and tops/pants, when setting a dress, the tops and pants need to be set to none, otherwise the page display will be abnormal. Therefore, this field can be configured under the dress node, please refer to the AvatarTPose_panel.json in the demo for details. There is a dependency between glasses and lenses, so the corresponding bindData field is configured under each glasses item to associate lens information.
avatarData	AvatarData	No	The property operation class defined in the SDK.
animation	AvatarAnimation	No	The property operation class defined in the SDK.

BindData

Field	Type	Required	Description
category	String	Yes	Same as category in SubTab.
id	String	Yes	The ID of each property corresponds to the ID in the AvatarData data returned by the SDK.
firstLevelId	String	Yes	The ID of the top-level category to which this data belongs.

avatarData	AvatarData	Yes	The property operation class defined in the SDK.
isTraverseOnSave	Boolean	Yes	When saving, whether to traverse the data in this bindData depends on the specific situation. In general, it needs to be traversed, except for the hairstyle and hair color configured in the hat, and the hat and hair color configured in the hairstyle, which do not need to be traversed.

Avatar APIs

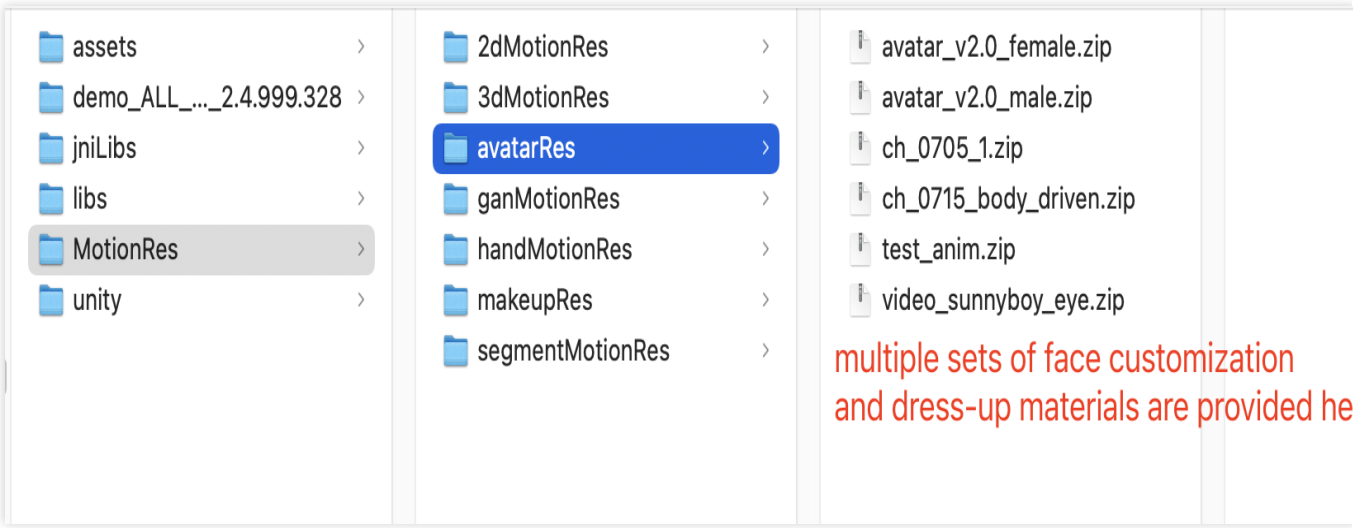
Last updated：2023-05-18 09:41:49

SDK Integration

For how to download and integrate the SDK, how to set the license, as well as how to run a demo project, see Integrating Tencent Effect SDK (Android).

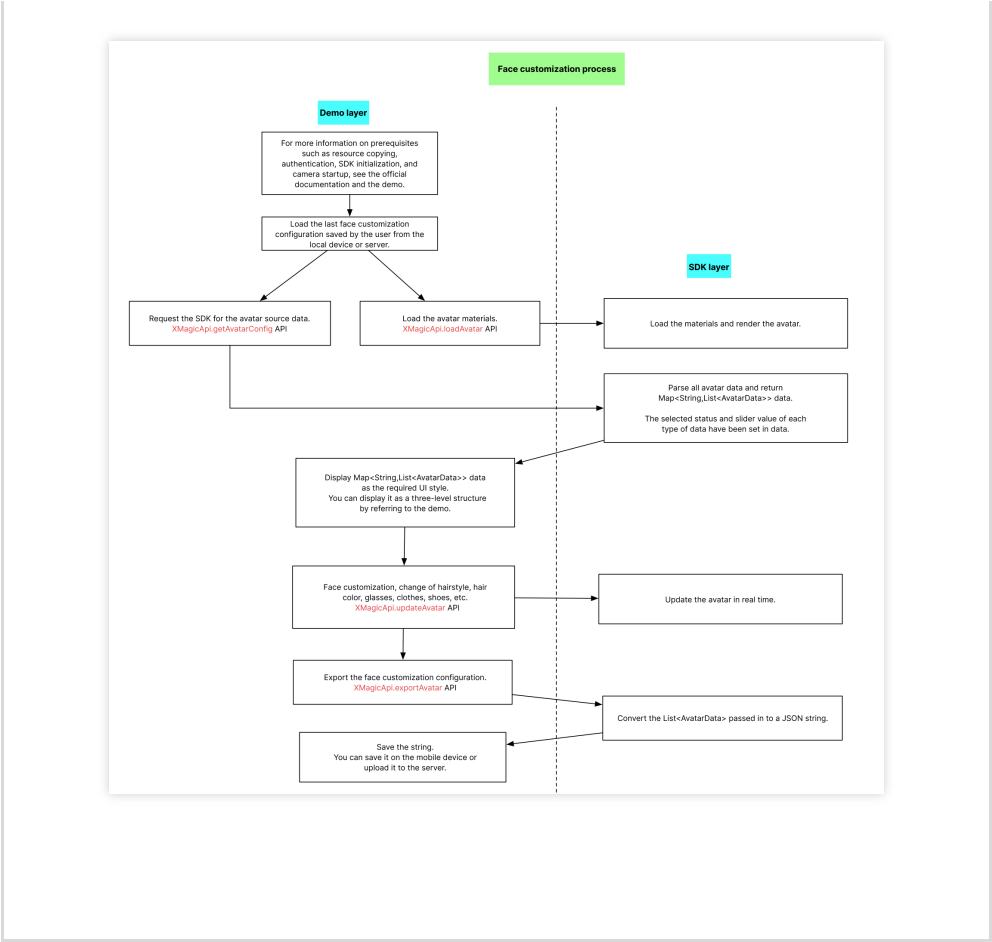
Copying the Avatar Materials

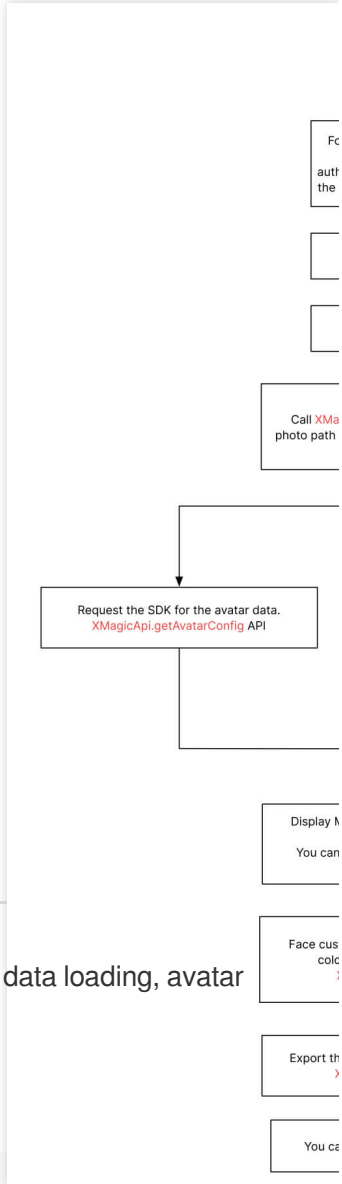
We offer multiple sets of face customization and dress-up materials, which can be found in the `MotionRes/avatarRes` directory in the SDK package after decompression. Like other animated effect materials, you need to copy them to the `assets` directory of your project.



Avatar Customization and SDK APIs

Face customization process	Photo-based face customization p





The section below offers descriptions of the APIs of `XMagicApi`, including the APIs for data loading, avatar customization, and photo-based avatar customization.

1. Load the avatar materials (`loadAvatar`)

```
public void loadAvatar(XmagicProperty<?> property, UpdatePropertyListener updatePro
```

The avatar materials are loaded in the same way as other animated effect materials. The `loadAvatar` API is equivalent to the updateProperty API.

2. Load the avatar source data (`getAvatarConfig`)

```
public static Map<String,List<AvatarData>> getAvatarConfig(String avatarResPath, St
```

Input parameters:

avatarResPath: The absolute path of the avatar materials on the user's mobile device, such as `/data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/animoji_0624`.

savedAvatarConfigs: The avatar configuration data (a JSON string) saved from the last time the user customized an avatar. If an avatar is customized for the first time or no configuration data is saved, pass in `null`.

Output parameters:
This API returns a map, in which `key` corresponds to `category` and `value` corresponds to the data of the category. For details, see the `AvatarCategory` class. After the application layer gets the map, display the data on the UI as needed.

3. Customize the face and dress up (`updateAvatar`)

```
public void updateAvatar(List<AvatarData> avatarDataList, UpdateAvatarConfigListene
```

When this API is called, the avatar preview will be updated in real time. Each `AvatarData` object corresponds to a configuration (such as changing the hairstyle). You can pass multiple `AvatarData` objects to an API call to edit multiple aspects of an avatar, for example, change the hairstyle and hair color. The API will validate the `AvatarData` objects passed in. If an object is valid, it will be passed to the SDK; if not, a callback will be sent.
For example, if an `AvatarData` object is passed in to change the hairstyle, but the hair model file (specified by `value` in `AvatarData`) cannot be found on the local device, the `AvatarData` object will be considered invalid.
Also, if an `AvatarData` object is passed in to change the iris image, but the image file (specified by `value` in `AvatarData`) cannot be found on the local device, the `AvatarData` object will be considered invalid.

4. Export avatar settings (`exportAvatar`)

```
public static String exportAvatar(List<AvatarData> avatarDataList)
```

When the user edits their avatar, the value of `selected` or shape change values in `AvatarData` will change. After editing, a new `AvatarData` list will be generated and can be exported as a JSON string. You can save it locally or upload it to the server.
The string is exported for the following purposes:
The next time you call `loadAvatar` of `XMagicApi` to load the avatar materials, you need to set `customConfigs` of `XMagicProperty` to the JSON string so that the preview will remember the avatar settings from the last time.
Also, you need to pass in this JSON string when you call `getAllAvatarData` to update `selected` and the shape values in the avatar source data.

5. Customize an avatar based on a photo (`createAvatarByPhoto`)

For this API to work, the SDK must be connected to the internet.

```
public void createAvatarByPhoto(String photoPath, List<String> avatarResPaths, bool
            final FaceAttributeListener faceAttributeListener)
```

photoPath: The photo path. Make sure the face is in the center of the photo. Ideally, the photo should include only one face. If there are multiple, the SDK will select one randomly. To ensure the recognition results, the short side of the photo should preferably be longer than 500 px.
avatarResPaths: You can pass in multiple sets of avatar materials, and the SDK will select the most suitable set based on the photo analysis result.
Note:
Currently, only one set of materials is supported. If multiple sets are passed in, the SDK will use the first one.
isMale: Whether the person is male. Set to true for male and false for female.
faceAttributeListener: If customization fails, `void onError(int errCode, String msg)` will be returned; if it succeeds, `void onFinish(String matchedResPath, String srcData)` will be returned, in which the first parameter indicates the path of the matched avatar materials, and the second is the matching result (same as the return value of `exportAvatar`).
The error codes of `onError` are defined in `FaceAttributeHelper.java`:

```
  public static final int ERROR_NO_AUTH = 1;// Insufficient permission.
  public static final int ERROR_RES_INVALID = 5;// The avatar material path
passed in is invalid.
  public static final int ERROR_PHOTO_INVALID = 10;// Failed to read the photo.
  public static final int ERROR_NETWORK_REQUEST_FAILED = 20;// Failed to
connect to the internet.
  public static final int ERROR_DATA_PARSE_FAILED = 30;// Failed to parse the
data returned from the network.
  public static final int ERROR_ANALYZE_FAILED = 40;// Failed to recognize the
face.
  public static final int ERROR_AVATAR_SOURCE_DATA_EMPTY = 50;// Failed to load
the avatar source data.
```

6. Save the downloaded configuration file (`addAvatarResource`)

```
public static Pair<Integer, List<AvatarData>> addAvatarResource(String
resourceRootPath, String category, String zipFilePath)
```

This API is used if avatar materials are downloaded dynamically. For example, suppose you offer 10 hairstyles and one of them is dynamically downloaded to a device. You need to call this API to pass the path of the downloaded ZIP file to the SDK. The SDK will parse the file and save it to the folder of the corresponding category. When you call `getAvatarConfig` the next time, the SDK will return the newly added data.
Parameters:
resourceRootPath: The root directory of the avatar materials, such as `/data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/animoji_0624`.
category: The category of the downloaded material.

zipFilePath: The path of the downloaded ZIP file.

The API returns `Pair<Integer, List<AvatarData>>`, where `pair.first` is the error code, and `pair.second` is the newly added data.

The error codes are as follows:

```
public interface AvatarActionErrorCode {
    int OK = 0;
    int ADD_AVATAR_RES_INVALID_PARAMS = 1000;
    int ADD_AVATAR_RES_ROOT_RESOURCE_NOT_EXIST = 1001;
    int ADD_AVATAR_RES_ZIP_FILE_NOT_EXIST = 1002;
    int ADD_AVATAR_RES_UNZIP_FILE_FAILED = 1003;
    int ADD_AVATAR_RES_COPY_FILE_FAILED = 1004;
}
```

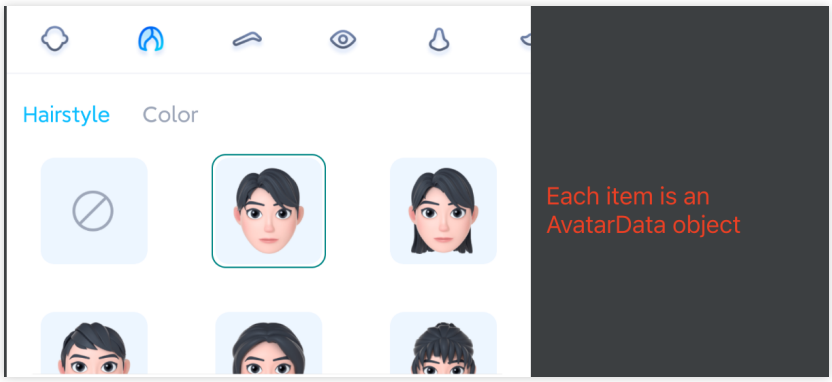
7. Call `AvatarData`

The `AvatarData` class is at the core of the above APIs. `AvatarData` contains the following fields:

```
public class AvatarData {
    /**
     * The selector data. For example, for glasses, different types of glasses
are offered, and only one can be selected.
     */
    public static final int TYPE_SELECTOR = 0;

    /**
     * The slider data, such as cheek width.
     */
    public static final int TYPE_SLIDER = 1;

    // The category, such as face shape and eye adjustment. Standard categories
are defined in `AvatarCategory.java`. If they cannot meet your requirements,
you can customize a category (make sure it's not the same as an existing one).
    // This field cannot be empty.
    public String category;

    // The ID of an avatar configuration item.
    // For example, each type of glasses has a unique ID. So does each
adjustment item.
    // This field cannot be empty.
    public String id;

    // `TYPE_SELECTOR` or `TYPE_SLIDER`
    public int type;
```

```
    // If `type` is `TYPE_SELECTOR`, this field indicates whether the item is
selected.
    public boolean selected = false;

    // Each icon or adjustment item has three aspects of configuration details:
    public String entityName;
    public String action;
    public JsonObject value;
}
```

An `AvatarData` object is the smallest unit of configuration, for example, hairstyle change or cheek adjustment.

Hairstyle change	Cheek adjustment
	

If an item is the selector type, configure it by changing the value of `selected` in `AvatarData`. For example, suppose there are four types of glasses: A, B, C, and D. If the user selects A, set `selected` of A to `true` and that of B, C, and D to `false`. If the user selects B, set `selected` of B to `true` and that of A, C, and D to `false`.

If an item is the slider type, configure it by changing `value` in `AvatarData`. The `value` field is a JSON object that includes multiple key-value pairs. Set the `value` of a key-value pair to the slider value.

8. Get the avatar animation data (`getAvatarAnimations`)

```
public static List<AvatarAnimation> getAvatarAnimations(String avatarResPath)
```

Input parameter: avatarResPath

The absolute path of the avatar materials on the user's mobile device, such as `/data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/anim oji_0624`.

Output parameter: All animation resource data is returned in a list. For more information, see the `AvatarAnimation` class.

9. Save the downloaded animation configuration file (`addAvatarAnimation`)

```
public static Pair<Integer, List<AvatarAnimation>> addAvatarAnimation(String
avatarResPath, String zipFilePath)
```

This API is used if avatar animation materials are downloaded dynamically. For example, suppose you offer one animation and it is dynamically downloaded to a device. You need to call this API to pass the path of the downloaded ZIP file to the SDK. The SDK will parse the file and save it to the folder of the corresponding animation resource. When you call `getAvatarAnimations` the next time, the SDK will return the newly added data.

Parameters:

avatarResPath: The root directory of the avatar materials, such as `/data/data/com.tencent.pitumotiondemo.effects/files/xmagic/MotionRes/avatarRes/animoji_0624` .

zipFilePath: The path of the downloaded ZIP file.

The API returns `Pair<Integer, List<` AvatarAnimation `>>` , where `pair.first` is the error code, and `pair.second` is the newly added data.

The error codes are as follows:

```
   public interface AvatarActionErrorCode {
       int OK = 0;
       int ADD_AVATAR_RES_INVALID_PARAMS = 1000;
       int ADD_AVATAR_RES_ROOT_RESOURCE_NOT_EXIST = 1001;
       int ADD_AVATAR_RES_ZIP_FILE_NOT_EXIST = 1002;
       int ADD_AVATAR_RES_UNZIP_FILE_FAILED = 1003;
       int ADD_AVATAR_RES_COPY_FILE_FAILED = 1004;
       int ADD_AVATAR_RES_PARSE_JSON_FILE_FAILED = 1005;
   }
```

10. Play back/Pause the avatar animation (`playAvatarAnimation`)

```
public void playAvatarAnimation(AnimationPlayConfig animationConfig)
```

Input parameters:

AnimationPlayConfig: Animation playback information class. This class mainly contains the following information.

```
public class AnimationPlayConfig {
     // The action description. It specifies whether to play back or pause the anim
     public static final String ACTION_PLAY = "play";
     public static final String ACTION_PAUSE = "pause";
```

```
    public static final String ACTION_RESUME = "resume";
    public static final String ACTION_STOP = "stop";

    public String entityName;
    /** * The path of the animation folder. It can be the relative path to the mat
    /** It can also be the relative path on the mobile phone, such as `/data/data/
    public String animPath;
    // The value can be `ACTION_PLAY`, `ACTION_PAUSE`, `ACTION_RESUME`, or `ACTION
    public String action;
    /** * The animation name */
    public String animName;
    /** * The number of loops. `-1` indicates an infinite loop.*/
    public int loopCount = -1;
    /** * The playback start position of the animation in μs */
    public long startPositionUs = 0;
}
```

More About `AvatarData`

The three key fields of `AvatarData` are `entityName`, `action`, and `value`, whose values are automatically entered when the SDK parses the configuration data. In most cases, you won't need to deal with the details of these three fields. However, for slider data, you need to parse the key-value pairs in the `value` field and configure them based on the slider values set on the UI.
Here, the `AvatarData` element consists of the entityName, action, and value fields.

`entityName`

`entityName` specifies which part of an avatar is to be edited, for example, face, eyes, hair, shirt, or shoes.

`action` and `value`

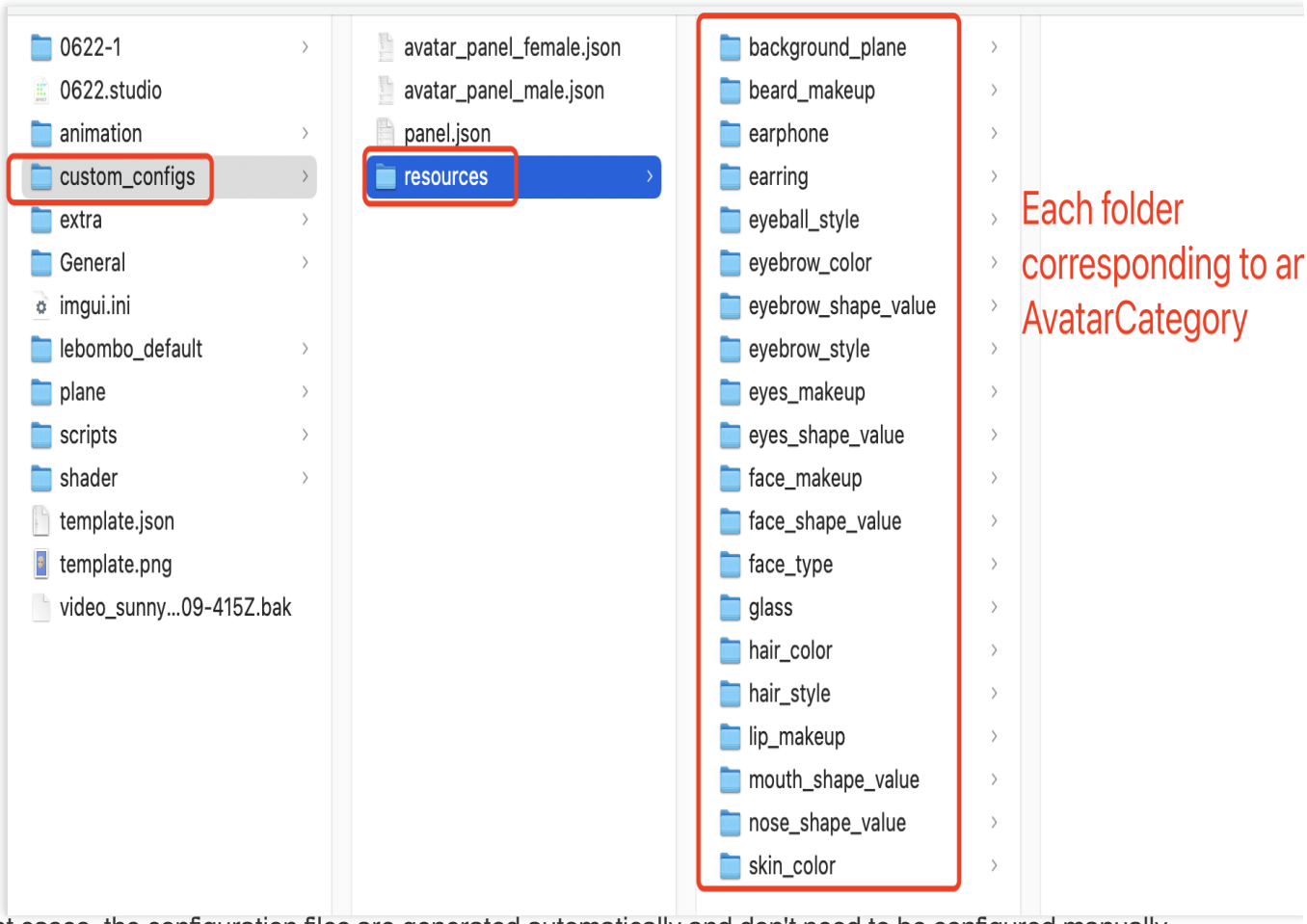
The `action` field indicates the action to be performed on `entityName`. Five `action` options are defined in `AvatarAction.java` in the SDK, which are detailed below:

action	Description	Requirements for `value`
changeColor	Changes the color of the current material. Color attributes include basic color and emission color.	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.
changeTexture	Modifies the maps of the current material, including color texture map, metal/roughness texture map, ambient	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.

	occlusion (AO) map, normal map, and emission map.	
shapeValue	Modifies the shape change value of `blendShape`. This parameter is generally used for slight adjustment of detailed facial features.	This field is of `JsonObject` type and is required. Its `key` is the shape change name and `value` is of `float` type. It is generated automatically by the material customization tool.
replace	Replaces a submodel, for example, glasses, hairstyles, or clothes.	`value` must be a JSON object that describes the 3D transformation information, model path, and material path of the new submodel. It is generated automatically by the material customization tool. To hide a submodel, set it to `null`.
basicTransform	Adjusts the position, rotation, and scaling settings. This field is generally used to adjust the camera distance and angle so as to switch between the full and half-length views of the model.	This field is of `JsonObject` type and is required. It is generated automatically by the material customization tool.

Configuring Avatar Customization Data

Avatar configurations are stored in the `resources` folder (in `material/custom_configs/resources`):



In most cases, the configuration files are generated automatically and don't need to be configured manually.

To automatically generate the configurations, design your materials using Tencent Effect Studio according to our standards and run the "resource_generator_gui" app we provide (only available on macOS currently). For details, see Design Specifications.