Tencent Effect SDK

API Documentation

Product Documentation

Contents

API Documentation
- iOS
- Android
- Flutter
- Web

API Documentation
iOS

Last updated：2024-03-19 15:50:12

XMagic.h, the core interface class of the Tencent Effect SDK, is used to initialize the SDK, update the beautification value, call the animation effect and other functions.

Public Member APIs

API	Description
initWithRenderSize	Initialization API
initWithGlTexture	Initialization API
configPropertyWithType	Configures effects.
setEffect	Configuration of various beauty filter effects (Added in 3.5.0.2)
emitBlurStrengthEvent	Setting post-processing blur strength (Applies to all blur components)
setRenderSize	Sets the render size.
deinit	Releases resources.
process:	Image data processing interface: inputs image before beautification, returns image after beautification.
process:withOrigin:withOrientation:	Image data processing interface: inputs image before beautification, returns image after beautification. This interface has two additional parameters compared to the previous one.
process	Processes data.
processUIImage	Processes an image.
getConfigPropertyWithName	Gets effect information.
registerLoggerListener	Registers a log listener.
registerSDKEventListener	Register a listener for SDK events.

clearListeners	Removes listeners.
getCurrentGlContext	Gets the current OpenGL context.
onPause	Pauses the SDK.
onResume	Resumes the SDK.
setAudioMute	Whether to turn on mute when the dynamic effect material is used (new in V2.5.0)Parameters: YES means mute, NO means no mute
enableEnhancedMode	Enable enhanced beauty filter pattern (Added in V2.5.1). By default, it is not activated. When it's disabled, the application layer can set the intensity range of each beauty option from 0 to 1 or -1 to 1. If it exceeds this range, the SDK will take the boundary value. For example, if the application layer sets the face-slimming component to 1.2, SDK will limit it to the maximum value of 1.0, then it corrects the face-slimming value to 1.0 internally, and then it corrects the face-slimming value to 1.0 internally. After enabling the enhanced pattern, the application layer can set a larger range of values. For example, if you want greater face slimming, you can set the face slimming value to 1.2. The SDK will accept and use this value and will not correct it to 1.0. **Note:** After enabling the enhanced pattern, the application layer needs to manage the maximum value that each beauty item can set, and let users adjust the value within this range. We provide a set of reference values, which you can adjust according to product requirements. However, we do not recommend exceeding our recommended values, otherwise the beauty effect may worsen.
Overlaying of materials	If you want to overlay a certain animation/beauty/segmentation material on the current material, when setting up the material, set 'mergeWithCurrentMotion' to true in the dictionary of 'withExtraInfo'
High performance mode	During SDK initialization, add @"setDowngradePerformance":@(YES) to the dictionary. With high performance mode enabled, the beauty feature will consume less system CPU/GPU resources, reducing overheating and lagging on the phone, making it more suitable for long-term use on low-end devices.

initWithRenderSize

This API is used to configure effects.

```
- (instancetype _Nonnull)initWithRenderSize:(CGSize)renderSize
                        assetsDict:(NSDictionary* _Nullable)assetsDict;
```

Parameters

Parameter	Description
renderSize	The render size.
assetsDict	The resource dictionary.

initWithGlTexture

This API is used to configure effects.

```
- (instancetype _Nonnull)initWithGlTexture:(unsigned)textureID
                        width:(int)width
                        height:(int)height
                        flipY:(bool)flipY
                        assetsDict:(NSDictionary* _Nullable)assetsDict;
```

Parameters

Parameter	Description
textureID	The texture ID.
width	The render size.
height	The render size.
flipY	Whether to flip the image.
assetsDict	The resource dictionary.

configPropertyWithType

This API is used to configure effects.

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

Parameters

Parameter	Description
propertyType	The effect type.

propertyName	The effect name.
propertyValue	The effect value.
extraInfo	A reserved parameter, which can be used for dictionary configuration.

Examples

Beautification: Configuring the skin brightening effect

```
NSString *propertyType = @"beauty";        //Set the effect type, take beauty as an
NSString *propertyName = @"beauty.whiten"; //Specify the effect name, take the skin
NSString *propertyValue = @"60";           //Set the effect value.
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
traInfo:nil];
```

Filter: Configuring the Allure filter

```
NSString *propertyType = @"lut";        //Set the effect type, take filter as an ex
NSString *propertyName = [@"lut.bundle/" stringByAppendingPathComponent:@"xindong_l
NSString *propertyValue = @"60";           //Set the effect value.
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

Body retouch: Configuring the long leg effect

```
NSString *propertyType = @"body";        //Set the effect type, take body retouch a
NSString *propertyName = @"body.legStretch"; //Specify the effect name, take the lo
NSString *propertyValue = @"60";           //Set the effect value.
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

Animated effect: Configuring the animated 2D cute effect

```
 NSString *motion2dResPath = [[NSBundle mainBundle] pathForResource:@"2dMotionRes"
NSString *propertyType = @"motion";        //Set the effect type, take animated eff
NSString *propertyName = @"video_keaituya"; //Specify the effect name, take animate
NSString *propertyValue = motion2dResPath;  //Set the path of the animated effect.
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

Makeup: Configuring the girl group makeup effect

```
NSString *motionMakeupResPath = [[NSBundle mainBundle] pathForResource:@"makeupMoti
NSString *propertyType = @"motion";        //Set the effect type, take makeup as an
NSString *propertyName = @"video_nvtuanzhuang"; //Specify the effect name, take the
NSString *propertyValue = motionMakeupResPath;  //Set the path of the animated effe
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
//Below are settings for the makeup effect (you only need to configure the above pa
NSString *propertyTypeMakeup = @"custom";         //Set the effect type, take makeu
NSString *propertyNameMakeup = @"makeup.strength"; //Specify the effect name, take
```

```
NSString *propertyValueMakeup = @"60";              //Set the effect value.
[self.xmagicApi configPropertyWithType:propertyTypeMakeup withName:propertyNameMake
```

Keying: Configuring the background blurring effect (strong)

```
NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotion
NSString *propertyType = @"motion";         //Set the effect type, take keying as a
NSString *propertyName = @"video_segmentation_blur_75"; //Specify the effect name,
NSString *propertyValue = motionSegResPath;  //Set the path of the animated effect.
NSDictionary *dic = @{@"bgName":@"BgSegmentation.bg.png", @"bgType":@0, @"timeOffse
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

Custom background:

```
NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotion
NSString *propertyType = @"motion";         //Set the effect type, take keying as a
NSString *propertyName = @"video_empty_segmentation"; //Specify the effect name, ta
NSString *propertyValue = motionSegResPath;  //Set the path of the animated effect
NSString *imagePath = @"/var/mobile/Containers/Data/Application/06B00BBC-9060-450F-
int bgType = 0;//The background type. 0: image; 1: video.
int timeOffset = 0;//The duration. If an image is used as the background, its value
NSDictionary *dic = @{@"bgName":imagePath, @"bgType":@(bgType), @"timeOffset": @(ti
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

setEffect (Added in 3.5.0.2)

For the configuration of various beauty effects, see Effect Parameters for specific usage examples.

```
- (void)setEffect:(NSString * _Nullable)effectName
      effectValue:(int)effectValue
     resourcePath:(NSString * _Nullable)resourcePath
        extraInfo:(NSDictionary * _Nullable)extraInfo;
```

Parameter

Parameter	Meaning
effectName	Effect type.
effectValue	Effect value.
resourcePath	Material path.
extraInfo	Reserved for expansion and additional configuration.

emitBlurStrengthEvent

Setting post-processing blur intensity (which applies to all blur components).

```
- (void)emitBlurStrengthEvent:(int)strength;
```

Parameter

Parameter	Meaning
strength	Effect value.

setRenderSize

This API is used to set the render size.

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

Parameters

Parameter	Description
size	The render size.

deinit

This API is used to release resources.

```
- (void)deinit;
```

process

Processing data interface, the input data formats include `YTImagePixelData,YTTextureData,YTImageRawData,YTUIImageData`, outputting the corresponding data formats. The pixel format in `YTImagePixelData` is `RGBA`, and the texture format in `YTTextureData` is `OpenGL 2D`.

```
/// @brief Process input 4 choose 1
@interface YTProcessInput : NSObject
/// Camera data object
@property (nonatomic, strong) YTImagePixelData * _Nullable pixelData;
/// Texture object
@property (nonatomic, strong) YTTextureData * _Nullable textureData;
/// Raw data object
@property (nonatomic, strong) YTImageRawData * _Nullable rawData;
/// UIImage object
@property (nonatomic, strong) YTUIImageData * _Nullable UIImageData;
/// Input data type
@property (nonatomic) enum YTProcessDataType dataType;
```

```
@end
/// @brief Process output
@interface YTProcessOutput : NSObject
/// Texture output object (always guaranteed)
@property (nonatomic, strong) YTTextureData * _Nullable textureData;
/// Camera output object (if the input is camera acquisition data)
@property (nonatomic, strong) YTImagePixelData * _Nullable pixelData;
/// Raw output object (if the input is raw data)
@property (nonatomic, strong) YTImageRawData * _Nullable rawData;
/// UIImage output object (if the input is a UIImage object)
@property (nonatomic, strong) YTUIImageData * _Nullable UIImageData;
/// Output data type
@property (nonatomic) enum YTProcessDataType dataType;
@end

- (YTProcessOutput* _Nonnull)process:(YTProcessInput * _Nonnull)input;
```

Parameter

Parameter	Meaning
input	Input data processing information, one of four input formats can be chosen from (YTImagePixelData, YTTextureData, YTImageRawData, YTUIImageData).

YTProcessInput Input data types and descriptions

When invoking the process interface, the output data type matches the input data type, YTTextureData type is always output.

Type	Meaning
YTImagePixelData	Camera data object, with the pixel format, is RGBA.
YTTextureData	Texture object, with the texture format, is OpenGL 2D.
YTImageRawData	Raw data object.
YTUIImageData	UIImage object.

process:withOrigin:withOrientation:

Data processing interface. The input and output data formats match process. withOrigin: Setting whether to flip the image vertically. withOrientation: Setting the image rotation direction.

```
- (YTProcessOutput* _Nonnull)process:(YTProcessInput* _Nonnull)input withOrigin:(Yt
```

Parameter

Parameter	Meaning
input	Input data processing information.
withOrigin	Enumeration value (YtLightImageOriginTopLeft and YtLightImageOriginBottomLeft), when set to YtLightImageOriginBottomLeft, the image is flipped vertically.
withOrientation	Enumeration value: Image rotation angle, setting the angle will change the output image angle.

TEImageTransform tool class

Image Process tool class, input/output data formats include CVPixelBufferref and texture id. It supports mutually converting CVPixelBufferref data's bgra<-->yuv format, rotation, and vertical/horizontal mirroring. It also supports rotation and vertical/horizontal mirroring of the input in texture id format.

```
/// @param context If you are using the OpenGL interface of this class, we recommen
- (instancetype)initWithEAGLContext:(EAGLContext *)context;
```

Parameter	Meaning
context	Using the context environment of OpenGLES, you can pass [xMagic getCurrentGlContext].

```
/// @brief CVPixelBufferRef yuv/rgb interconversion interface, currently, only supp
/// @param pixelBuffer Input pixelBuffer     input pixelBuffer
/// @param outputFormat Specify the type of output pixelBuffer    output pixelBuffe
- (CVPixelBufferRef)transformCVPixelBufferToBuffer:(CVPixelBufferRef)pixelBuffer ou
```

Parameter	Meaning
pixelBuffer	Input pixelBuffer data
outputFormat	Output pixelBuffer format, supports BGRA, NV12F(kCVPixelFormatType_420YpCbCr8BiPlanarFullRange), and NV12V(kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange).

```
/// Convert yuv/rgb pixelBuffer to bgra format texture id
/// @param pixelBuffer Input pixelBuffer
- (GLuint)transformPixelBufferToBGRATexture:(CVPixelBufferRef)pixelBuffer;
```

Parameter	Meaning
pixelBuffer	Input pixelBuffer data, supports BGRA, NV12F(kCVPixelFormatType_420YpCbCr8BiPlanarFullRange), and NV12V(kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange).

```
/// Rotate the CVPixelBufferRef and flip it. If you pass rotation and flipping at t
- (CVPixelBufferRef)convertCVPixelBuffer:(CVPixelBufferRef)pixelBuffer rotaion:(YtL
```

Parameter	Meaning
pixelBuffer	Input pixelBuffer data
rotation	The counterclockwise rotation angle, supports 0 degree, 90 degrees, 180 degrees, and 270 degrees.
flipType	Mirror type, horizontal mirroring, or vertical mirroring. If both rotation and flipping are passed, the processing logic is to mirror first and then rotate.

```
/// Rotate/flip the texture Id, if both rotation and flipping are passed, the proce

- (GLuint)convert:(GLuint)srcId width:(int)width height:(int)height rotaion:(YtLigh
```

Parameter	Meaning
srcId	Input Texture ID.
width	Texture width.
height	Texture height.
rotation	The Counterclockwise rotation angle, supports 0 degree, 90 degrees, 180 degrees, and 270 degrees.
flipType	Mirror type, horizontal mirroring, or vertical mirroring. If both rotation and flipping are passed, the processing logic is to mirror first and then rotate.

processUIImage

This API is used to process an image.

```
- (UIImage* _Nullable)processUIImage:(UIImage* _Nonnull)inputImage needReset:(bool)
```

Parameters

Parameter	Description
inputImage	The input image. If your image is larger than 2160 x 4096, we recommend you reduce its size before passing it in; otherwise, face recognition may fail or may be inaccurate. It may also cause an OOM error.
needReset	This parameter must be set to true in the following cases: The image processed is changed The first time a keying effect is used The first time an animated effect is used The first time a makeup effect is used

getConfigPropertyWithName

This API is used to get effect information.

```
- (YTBeautyPropertyInfo * _Nullable)getConfigPropertyWithName:(NSString *_Nonnull)p
```

Parameters

Parameter	Description
propertyName	The effect name.

registerLoggerListener

This API is used to register a log listener.

```
- (void)registerLoggerListener:(id<YTSDKLogListener> _Nullable)listener withDefault
```

Parameters

Parameter	Description
listener	The log callback.
level	The log output level, which is ERROR by default.

registerSDKEventListener

This API is used to register a listener for SDK events.

```
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;
```

Parameters

Parameter	Description
listener	The listener for SDK events, including AI events, tips, and resource events.

clearListeners

This API is used to remove listeners.

```
- (void)clearListeners;
```

getCurrentGlContext

This API is used to get the current OpenGL context.

```
- (nullable EAGLContext*)getCurrentGlContext;
```

onPause

This API is used to pause the SDK.

```
/// @brief When your app is switched to the background, you need to call this API t
- (void)onPause;
```

onResume

This API is used to resume the SDK.

```
/// @brief When your app is switched back to the foreground, you need to call this
- (void)onResume;
```

setAudioMute

Whether to turn on mute when the dynamic effect material is used (new in V2.5.0)

```
/// @brief set mute
- (void)setAudioMute:(BOOL)isMute;
```

enableEnhancedMode

```
/// @brief Activate the beautification enhancing pattern
- (void)enableEnhancedMode;
```

Enable enhanced beauty filter pattern (Added in V2.5.1). By default, it is not activated.
When it's disabled, the application layer can set the intensity range of each beauty option from 0 to 1 or -1 to 1. If it exceeds this range, the SDK will take the boundary value. For example, if the application layer sets the face-slimming component to 1.2, SDK will limit it to the maximum value of 1.0, then it corrects the face-slimming value to 1.0 internally.
After enabling the enhanced pattern, the application layer can set a larger range of values. For example, if you want greater face slimming, you can set the face slimming value to 1.2. The SDK will accept and use this value and will not correct it to 1.0.
After enabling the enhanced pattern, the application layer needs to manage the maximum value that each beauty item can set, and let users adjust the value within this range. We provide a set of reference values, which you can adjust according to product requirements. However, we do not recommend exceeding our recommended values, otherwise the beauty effect may worsen. See the reference values below:

Beauty Item Name	In enhanced pattern, recommended maximum value (magnification factor)
Whitening, shortening the face, V-face, eye distance, nose position, removal of laugh lines, lipstick, three-dimensional appearance	1.3
Eye lightening	1.5
Blush	1.8
Other	1.2

High performance mode（V3.1.0.1）

With high performance mode enabled, the beauty feature will consume less system CPU/GPU resources, reducing overheating and lagging on the phone, making it more suitable for long-term use on low-end devices.
Note that the following beauty options will be unavailable when high performance mode is enabled:
1.Eye: Eye width, Eye height, Eye bags.
2.Eyebrow: Eyebrow angle, Eyebrow distance, Eyebrow Height, Eyebrow Length, Eyebrow Thickness, Eyebrow ridge.
3.Mouth: Smile face.

4.Facial: Thin face (Natural, Woman, Man), Slim jaw, Wrinkles, Smile lines. It is recommended to use Face shape to achieve a comprehensive effect of big eyes and thin "face".

```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                                @"setDowngradePerformance":@(YES) //Enable high pe
    };
    self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:asse
```

Static APIs

API	Description
isBeautyAuthorized	Gets the authorization information of an effect parameter.

isBeautyAuthorized

This API is used to get the authorization information of the effect parameter (only supports the parameter of beauty and body retouch).

```
/// @param featureId: The effect parameter.
/// @return: The authorization information of the effect parameter.
+ (BOOL)isBeautyAuthorized:(NSString * _Nullable)featureId;
```

Callback

API	Description
YTSDKEventListener	The SDK event callback.
YTSDKLogListener	The log callback.

YTSDKEventListener

The callback for the internal events of the SDK.

```
@protocol YTSDKEventListener <NSObject>
```

Member callback APIs

Return Type	Callback

void	onAIEvent
void	onTipsEvent
void	onAssetEvent

Callback description

onAIEvent

The YTDataUpdate event callback.

```
/// @param event Callback in dict format
- (void)onAIEvent:(id _Nonnull)event;
```

The information of up to five faces is returned as JSON strings.

```
{
 "face_info":[{
  "trace_id":5,
  "face_256_point":[
    180.0,
    112.2,
    ...
  ],
  "face_256_visible":[
    0.85,
    ...
  ],
  "out_of_screen":true,
  "left_eye_high_vis_ratio":1.0,
  "right_eye_high_vis_ratio":1.0,
  "left_eyebrow_high_vis_ratio":1.0,
  "right_eyebrow_high_vis_ratio":1.0,
  "mouth_high_vis_ratio":1.0
 },
 ...
 ]
}
```

Field Description

Field	Type	Value Range	Remarks
trace_id	int	[1,INF)	The face ID. If the faces obtained continuously from a video stream have the

			same face ID, they belong to the same person.
face_256_point	float	[0,screenWidth] or [0,screenHeight]	512 values in total for 256 facial keypoints. (0,0) is the top-left corner of the screen.
face_256_visible	float	[0,1]	The visibility of the 256 facial keypoints.
out_of_screen	bool	true/false	Whether only part of the face is captured.
left_eye_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the left eye.
right_eye_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the right eye.
left_eyebrow_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the left eyebrow.
right_eyebrow_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the right eyebrow.
mouth_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the mouth.

onAIEvent

The callback for AI events.

```
/// @param event: Callback in dict format
- (void)onAIEvent:(id _Nonnull)event;
```

onTipsEvent

The callback for tips.

```
/// @param event: Callback in dict format
- (void)onTipsEvent:(id _Nonnull)event;
```

onAssetEvent

The callback for resource events.

```
/// @param event: Callback in string format
- (void)onAssetEvent:(id _Nonnull)event;
```

YTSDKLogListener

The log callback.

```
@protocol YTSDKLogListener <NSObject>
```

Member callback APIs

Return Type	API
void	onLog

Callback description

onLog

The log callback.

```
/// @param loggerLevel: The current log level.
/// @param logInfo: The log information.
- (void)onLog:(YtSDKLoggerLevel) loggerLevel withInfo:(NSString * _Nonnull) logInfo
```

Material overlay (added in version 3.0.1.5)

If you want to overlay a certain animation/beauty/segmentation material on the current material, when setting up the material, set 'mergeWithCurrentMotion' to true in the dictionary of 'withExtraInfo', An example is shown below:

```
NSString *key = _xmagicUIProperty.property.Id;
NSString *value = [[NSBundle mainBundle] pathForResource:@"makeupMotionRes" ofType:
NSDictionary* extraInfo = @{@"mergeWithCurrentMotion":@(true)};
 [self.beautyKitRef configPropertyWithType:@"motion" withName:key withData:[NSStrin
```

The precautions for material overlay are：

1. The client needs to manage whether the materials are suitable for overlaying. Here are two examples:

Example 1: Effect A is to turn into a noblewoman's face, and effect B is to turn into a fairytale face. Overlaying these two effects may result in a very awkward image.

Example 2: Effect A is a pair of rabbit ears, and effect B is a pair of pig ears. Overlaying these two effects will result in two types of ears.

These two cases are not suitable for overlaying. If effect A is a pair of rabbit ears and effect B is blowing a kiss, these two effects will not conflict and are suitable for overlaying.

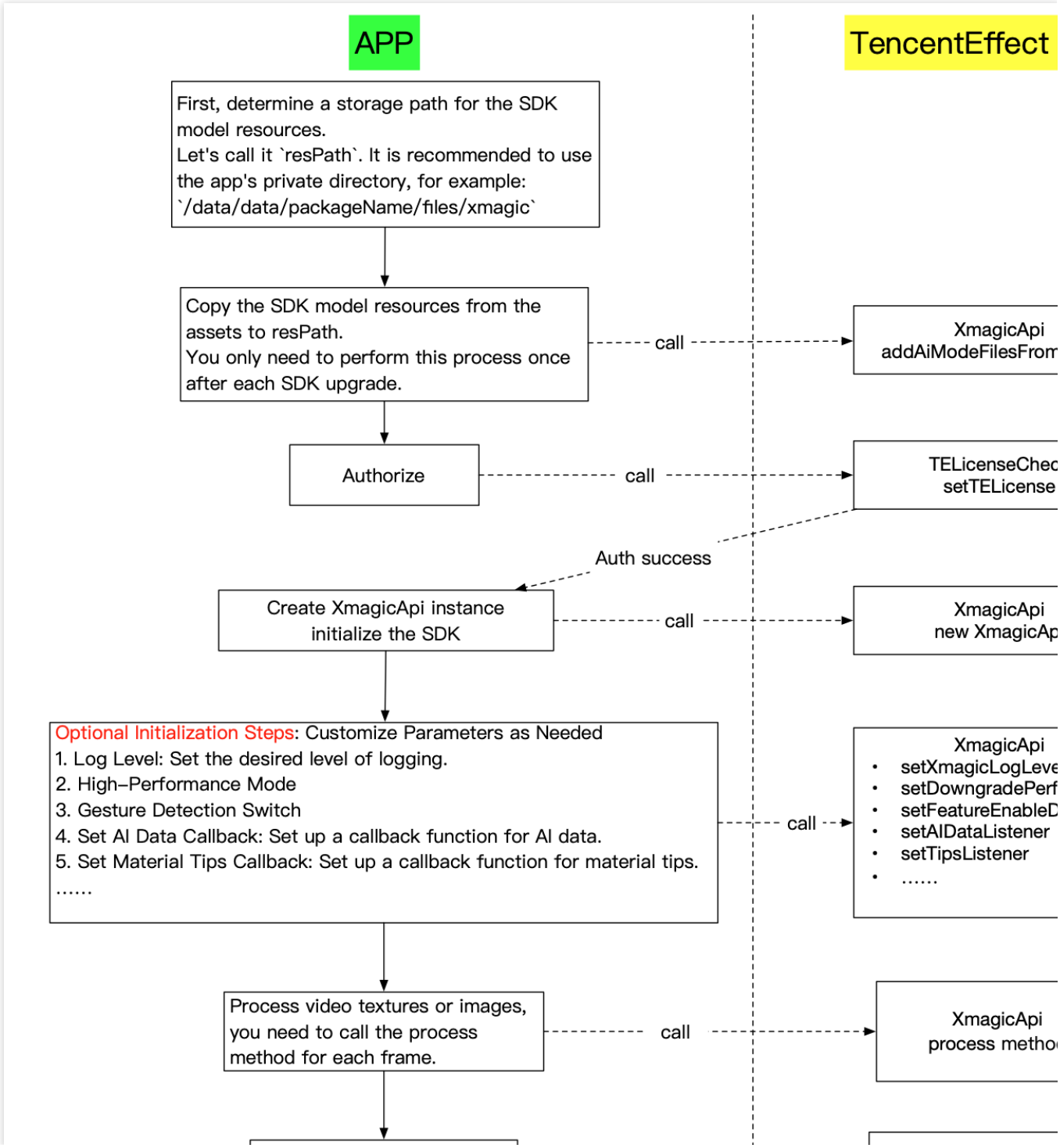
2. Only simple materials can be overlaid. Simple materials refer to those with only one animation effect, makeup effect, or background removal effect, while complex materials refer to those that contain multiple effects. There is no clear boundary between simple and complex materials, so it is recommended that clients fully test and manage which materials can be overlaid and which cannot.

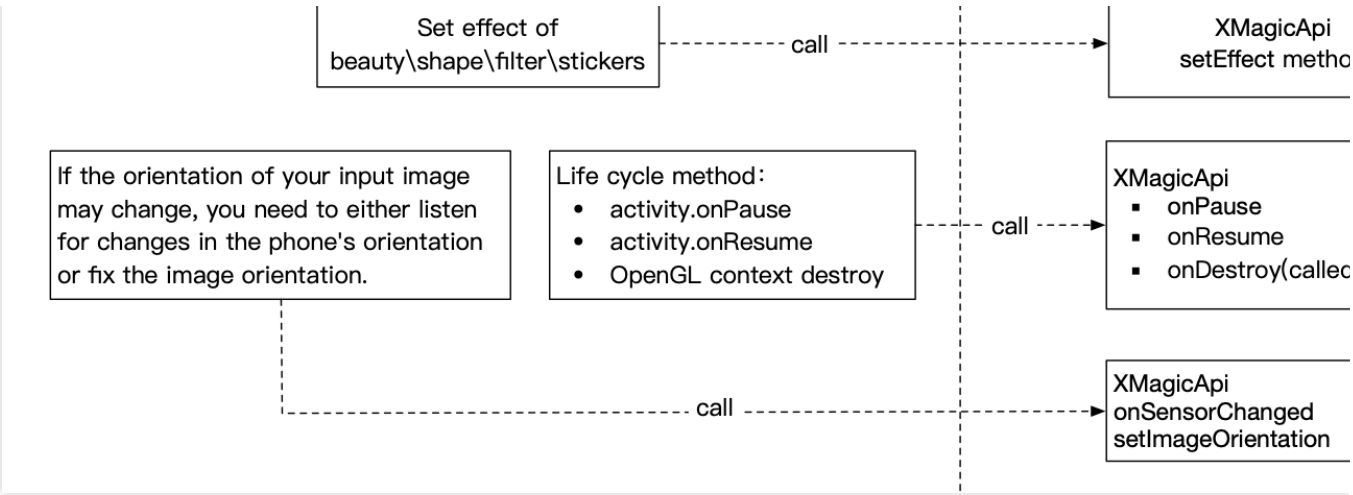
3. When overlaying, effects triggered by actions (such as triggering an effect by reaching out or smiling) are complex effects and need to be placed in front, with simple effects overlaid on top.
4. Usage example: The anchor uses effect A, and then the audience sends gift effect B, which needs to be overlaid on A. After a period of time, B disappears and returns to effect A. The settings are as follows:
4.1. Set effect A, and set mergeWithCurrentMotion to false.
4.2. Set effect B, and set mergeWithCurrentMotion to true.
4.3. After a short period of time, set A again, and set mergeWithCurrentMotion to false.

Android

Last updated：2024-10-31 14:46:45

Tencent Effect SDK Core Interface Class `XmagicApi.java` , utilized for initializing the SDK, updating beauty metric values, invoking animated effects, amongst other features.
Overall API calling process is as follows:





List of static properties and methods of XmagicApi

API	Description
VERSION	The SDK version number can be retrieved via XmagicApi.VERSION (added in V3.5.0).
setLibPathAndLoad	Set the path of the '.so' Library. If the '.so' Library is built into the apk package, this interface is not needed.
addAiModeFilesFromAssets	Transfer the content located within the directories Light3DPlugin, LightCore, LightHandPlugin, LightBodyPlugin, LightSegmentPlugin under application assets to your designated directory.
addAiModeFiles	Copy the AI model files downloaded by the client to the corresponding folders.
getDeviceLevel	Get the device level.

List of Public Methods of XmagicApi

API	Description
XmagicApi	constructor.
process	Methods for rendering data with the SDK, used for processing images or video streams.
setEffect	Set effects such as beauty, aesthetic shape, filters, makeup, stickers, and segmentation, and can be invoked in any thread.
setXmagicLogLevel	Set the SDK's log level, which defaults to `Log.WARN`. During development and debugging, it can be set to `Log.DEBUG` if necessary. For official release,

	make sure to set it to `Log.WARN` or `Log.ERROR` to avoid performance issues caused by excessive logging. **Invoke after new XmagicApi().**
enableHighPerformance	Invoke this method to enable the high-performance pattern. Upon the activation of the high-performance pattern, the system CPU/GPU resources occupied by beauty filters are minimized, thereby reducing heat generation and latency issues in the mobile device. It is particularly suitable for prolonged use on low-end devices. **Invoke after new XmagicApi().**
setFeatureEnableDisable	Enable or disable specific capability.
setXmagicStreamType	Set the input data type; there are two types: camera data and image data. The default is camera data stream.
setAIDataListener	Configure the callback for face, gesture, and body detection statuses.
setAudioMute	Toggle mute when using motion effect material. Parameter: true means mute, false means not mute.
onPause	Pause the sound playback in the special effects, can be bound to the Activity onPause lifecycle.
onResume	Resume the sound playback in the special effects, can be bound to the Activity onResume lifecycle.
onDestroy	Terminate `xmagic`, which necessitates its invocation within the `GL` thread.
setImageOrientation	Set the image orientation so that AI can recognize faces from different directions. If set, the direction provided by `sensorChanged` will be ignored.
sensorChanged	Use the system sensor to judge the current phone's rotation angle, so that the AI can recognize faces in different orientations.
isDeviceSupport	Pass the path of a dynamic effect material to the SDK, and detect whether the current device fully supports this dynamic effect.
isSupportBeauty	Determine whether the current device supports refinement (OpenGL3.0).
exportCurrentTexture	Retrieve the screen on the current texture
setTipsListener	Setting up callback functions for animated hint text, designated to display hints on the frontend page.
updateProperty updateProperties setYTDataListener	This interface is deprecated and not recommended for use. For details, please click here.

onPauseAudio getDeviceAbilities getPropertyRequiredAbilities isBeautyAuthorized enableEnhancedMode setDowngradePerformance	

Static method: setLibPathAndLoad

Set the path of the so Library and trigger loading.
If the so Library is built into the apk package, this interface is not needed. If the so Library is downloaded dynamically, it should be invoked before authentication and `new XmagicApi()` .
Passing in null indicates loading the so from the default path. Please ensure that the so is built into the APK package.
Pass in non-null: such as `data/data/package name/files/xmagic_libs` , the so will be loaded from this directory.

```
static boolean setLibPathAndLoad(String path)
```

Static method: addAiModeFilesFromAssets

SDK's Model Resource Files can be built into the assets directory of the apk package or downloaded dynamically. If they are built into the assets directory, the model resource files need to be copied to the specified directory before the SDK is used for the first time. This operation only needs to be done once. After upgrading the SDK version, it needs to be executed again.
context applicationcontext.
resDir is the root directory for storing SDK model resources. This directory is the same as the path parameter passed in when creating a new XmagicApi() object.
Returned values:
0: Copy successful
-1: denotes that the context is null
-2: denotes an IO error

```
static int addAiModeFilesFromAssets(Context context, String resDir)
```

Static method: addAiModeFiles

If you have not embedded the SDK's model resource file in the APK package but download it dynamically, after downloading and decompressing, you can call this method to copy the downloaded SDK model resource files to the corresponding folder.
inputResDir is the directory of the successfully downloaded model files.

resDir is the root directory for storing SDK model resources. This directory is the same as the path parameter passed in when creating a new XmagicApi() object.

Returned values:

0: denotes success

-1:inputResDir is not exists

-2: Signifies an IO error

```
static int addAiModeFiles(String inputResDir, String resDir)
```

Static method: getDeviceLevel

Obtain device level, added in V3.7.0. You can enable or disable the related SDK feature based on the device level, or set high-performance mode on lower-level phones.

```
static DeviceLevel getDevicLevel(Context context)

public enum DeviceLevel {
    DEVICE_LEVEL_VERY_LOW(1),
    DEVICE_LEVEL_LOW(2),
    DEVICE_LEVEL_MIDDLE(3),
    DEVICE_LEVEL_MIDDLE_HIGH(4),
    DEVICE_LEVEL_HIGH(5);

    private final int value;

    DeviceLevel(int value) {
        this.value = value;
    }

    public int getValue() {
        return value;
    }
}
```

XmagicApi Constructor Method

```
XmagicApi(Context context, String resDir)

XmagicApi(Context context, String resDir,OnXmagicPropertyErrorListener xmagicProper

public interface OnXmagicPropertyErrorListener {
    void onXmagicPropertyError(String errorMsg, int code);
}
```

Parameter	Type	Meaning
context	Context	context.
resDir	String	SDK model resource file directory. Before using the SDK, you need to designate directory to store the SDK's model resource fi recommended to use the app's private directc `resDir = context.getFilesDir().getAbsolute + "/xmagic"` Additionally, before instantiating XmagicApi, p first invoke the static method addAiModeFilesFromAssets or addAiModeFi XmagicApi to copy the model resource files to
xmagicPropertyErrorListener	OnXmagicPropertyErrorListener	Error Callback Interface. During the use of the this interface will callback some internal error information. Generally used during developme debugging.

OnXmagicPropertyErrorListener returns a table of error codes and their meanings:

Error code	Meaning
-1	Unknown error.
-100	3D engine resource initialization failed.
-200	GAN materials are not supported.
-300	This device does not support this material component.
-400	The template JSON content is empty.
-500	The SDK version is too low.
-600	Splitting is not supported.
-700	OpenGL is not supported.
-800	Scripting is not supported.
5,000	The resolution of the split background image exceeds 2160x3840.
5001	Insufficient memory required to segment the background image.

5002	Failed to parse the video segmentation of the background.
5003	Background video segment exceeds 200 seconds.
5004	Background video segment format unsupported.
5005	Background image segment possesses rotation angle

process

Methods for rendering data with the SDK, used for processing images or video streams. When processing video streams, it can be used within the camera data callback function.

```
// Render the texture
int process(int srcTextureId, int srcTextureWidth, int srcTextureHeight)
// Render the bitmap
Bitmap process(Bitmap bitmap, boolean needReset)
```

Parameter	Meaning
int srcTextureId	The texture that needs to be rendered. The type is: OpenGL 2D texture format, and the pixel format is RGBA.
int srcTextureWidth	Width of the texture that needs to be rendered.
int srcTextureHeight	Height of the texture that needs to be rendered.
Bitmap bitmap	The recommended maximum size is 2160 x 4096. Larger images have poor face recognition results or cannot get faces recognized and are likely to cause OOM problems. Shrink such images first before passing them in.
boolean needReset	In the following scenarios, set needReset to true: Processing an image for the first time, or switching images. First time using partition. Initial use of animated effects. First-time usage of makeup.

setEffect

Set effects such as beauty, aesthetic shape, filters, makeup, stickers, and segmentation, and can be invoked in any thread. For specific parameters, see Effect Parameters.

```
void setEffect(String effectName, int effectValue, String resourcePath, Map<String,
```

setXmagicLogLevel

Invoke after new XmagicApi(). Set the SDK's log level, which defaults to `Log.WARN`. During development and debugging, it can be set to `Log.DEBUG` if necessary. For official release, make sure to set it to `Log.WARN` or `Log.ERROR` to avoid performance issues caused by excessive logging.
If `ITELogger` is set, the SDK's internal logs will be redirected to the user.

```
public void setXmagicLogLevel(int level);
public void setXmagicLogLevel(int level, final ITELogger logger);

public interface ITELogger{
    void log(int severity, String tag, String msg);

    void log(int severity, String tag, String msg, Throwable throwable);

}
```

enableHighPerformance

Invoke this method to enable the High-Performance Mode, new in V3.7.0. After enabling the High-Performance Mode, the system CPU/GPU resources used for beauty filters are minimized, reducing heat generation and latency issues on the mobile device. This mode is more suitable for prolonged use on low-end devices. However, compared to the normal mode, some features may be limited.
It needs to be invoked immediately after new XmagicApi(). For detailed instructions, refer to the High-Performance Mode Usage Guide.

```
void enableHighPerformance()
```

setFeatureEnableDisable

Enable or disable a specific capability as needed.

```
void setFeatureEnableDisable(String featureName, boolean enable)
```

Parameter	Meaning
String featureName	Name of the atomic capability Valid values: `XmagicConstant.FeatureName.SEGMENTATION_SKIN` Skin Segmentation Capability, after enabling, can make the skin smoothing and whitening areas more accurate. `XmagicConstant.FeatureName.SEGMENTATION_FACE_BLOCK` Face Occlusion Detection Capability, after enabling, can avoid makeup being applied to occluded areas.

	`XmagicConstant.FeatureName.WHITEN_ONLY_SKIN_AREA` Whitening only works on skin `XmagicConstant.FeatureName.SMART_BEAUTY` Intelligent Beauty (reduces beauty and makeup effects for men and babies) `XmagicConstant.FeatureName.ANIMOJI_52_EXPRESSION` Face Expression Capability `XmagicConstant.FeatureName.BODY_3D_POINT` Body Point Capability `XmagicConstant.FeatureName.HAND_DETECT` Gesture Detection Capability
boolean enable	true indicates to enable this capability, false indicates to disable this capability

Among the mentioned capabilities, the commonly used ones are `SEGMENTATION_SKIN` and `SEGMENTATION_FACE_BLOCK` . The SDK will enable these two capabilities by default based on the value of getDeviceLevel. If you need to enable them manually, it is recommended to enable `SEGMENTATION_SKIN` when level >= 4, and enable `SEGMENTATION_FACE_BLOCK` when level >= 5.

setXmagicStreamType

Set the input data type; there are two types: camera data and image data. The default is camera data stream.

```
void setXmagicStreamType(int type)
```

Parameter	Meaning
int type	Data source type, there are two options: `XmagicApi.PROCESS_TYPE_CAMERA_STREAM` : camera data source. `XmagicApi.PROCESS_TYPE_PICTURE_DATA` : Image data source.

setAIDataListener

```
public void setAIDataListener(final XmagicAIDataListener aiDataListener);

public interface OnAIDataListener {
    void onFaceDataUpdated(List<TEFaceData> faceDataList);
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);
    void onAIDataUpdated(String jsonString);
    @Deprecated
    void onHandDataUpdated(List<TEHandData> handDataList);
}
```

onFaceDataUpdated

The SDK will callback after starting to process the image. When a face is recognized, it callbacks List<FaceData>, the size of the list is the number of faces. When there are no faces, it callbacks an empty List.
The definition of TEFaceData is as follows: points are coordinates for 83 face points, each with x and y coordinates, so the length of points is fixed at 166. The coordinate values are not based on the original input image but are face coordinates obtained after scaling the short edge of the original image to 256.
Therefore, it is recommended to use the callback data here only to determine whether there are faces in the current screen and how many faces there are. If you need higher precision face points, please use the data from `onAIDataUpdated` callback.

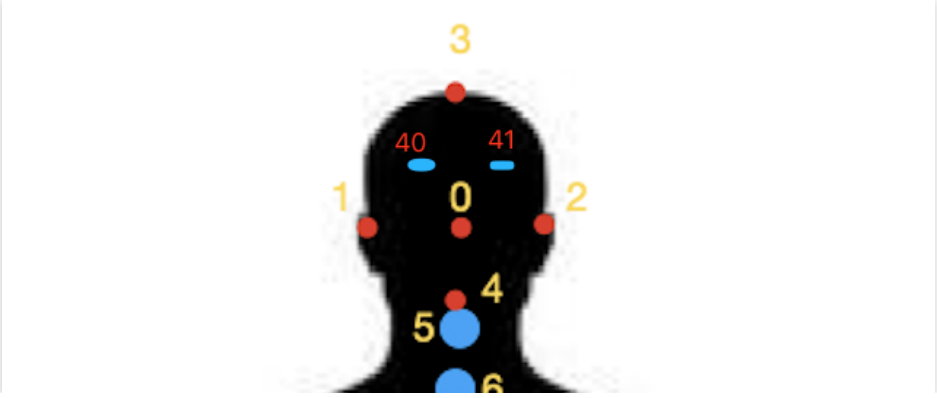
```
public class TEFaceData {
    public float[] points;

    public TEFaceData() {
    }

    public TEFaceData(float[] points) {
        this.points = points;
    }
}
```

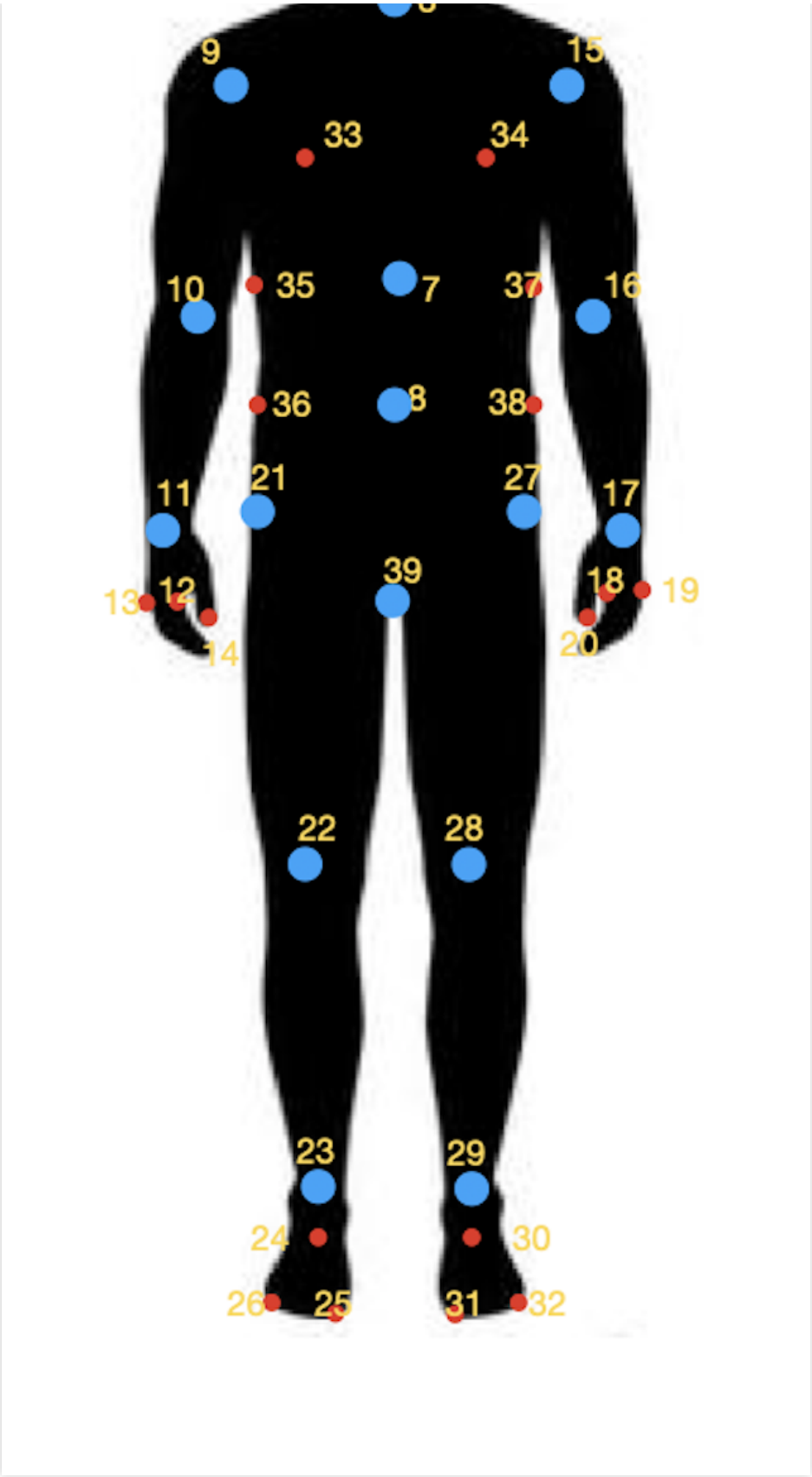
onHandDataUpdated

Deprecated interface, no data callback. The old version SDK callbacks when gesture effects are set and gestures are recognized.
If gesture data is needed, please use the data from `onAIDataUpdated` callback.

onBodyDataUpdated

Callbacks when body attributes are set and the body is recognized; no callbacks in other cases. The callback is the coordinates of the body's 42 points, the coordinate values are based on the width and height of the input image. If a point is not detected, the coordinate value is 0.





onAIDataUpdated

Callback detailed data of face, gesture, and body 3D usually requires a specific license to have data. An example data is as follows:

For detailed information about `face_info`, please check Facial Keypoint Detection.

For detailed information about `hand_info`, please check Gesture Recognition.

For detailed information about `body_3d_info`, please check Body Keypoints Android and Body Keypoints iOS.


```
{
    "face_info": [{
        "expression_weights": [0.001172, 0, 0.029249, ... , 0.060041, 0],
        "face_256_point": [211.844238, 673.247192, ... , 339.247925, 654.792603],
        "face_256_visible": [0.163925, 0.14921, ... , 0.99887, 0.99887],
        "face_3d_info": {
            "pitch": -3.860844850540161,
            "pitch_fixed": 2.1123428344726562,
            "roll": -12.797032356262207,
            "roll_fixed": 1.3187808990478516,
            "transform": [
                [0.8919625878334045, 0.2843534052371979, 0.3514907658100128, 0],
                [-0.17628398537635803, 0.9346542954444885, -0.3087802827358246, 0],
                [-0.41632509231567383, 0.21345829963684082, 0.88380366563797, 0],
                [-0.020958196371793747, -0.04502145200967789, -0.6078543663024902,
            ],
            "yaw": 24.824481964111328,
            "yaw_fixed": 25.02082061767578
        },
        "left_eye_high_vis_ratio": 0,
        "left_eyebrow_high_vis_ratio": 0,
        "mouth_high_vis_ratio": 1,
        "out_of_screen": false,
        "right_eye_high_vis_ratio": 1,
        "right_eyebrow_high_vis_ratio": 0.821429,
        "trace_id": 21
    }],
    "hand_info": {
        "gesture": "PAPER",
        "hand_point_2d": [180.71888732910156, 569.2958984375, ... , 353.87142944335
    },
    "body_3d_info": {
        "imageHeight": 652,
        "imageWidth": 320,
        "items": [{
            "index": 1,
            "pose": [0.049122653901576996, ... , 0],
            "position_x": [190.47494506835938, 235.23098754882812, ... , 4.94842433
            "position_y": [777.2109375, 836.488037109375, ... , 161.19752502441406,
            "position_z": [0, 0, ... , 0, 0],
```



```
            "rotation": [{
                "data": [0.9944382905960083, -0.09695644676685333, -0.0411277711391
            },
            ......
            {
                "data": [0.9907779693603516, 0.13549542427062988, 0, 0]
            }, {
                "data": [1, 0, 0, 0]
            }]
        }]
    }
}
```

setAudioMute

Whether to mute when using animation materials (Added in V2.5.0):

Parameter: true indicates mute, false denotes non-mute.

onPause

Pause the sound playback in the special effects, can be bound to the Activity onPause lifecycle.

```
void onPause()
```

onResume

Resume the sound playback in the special effects, can be bound to the Activity onResume lifecycle.

```
void onResume()
```

onDestroy

Clears GL thread resources and needs to be called within the GL thread. Sample code:

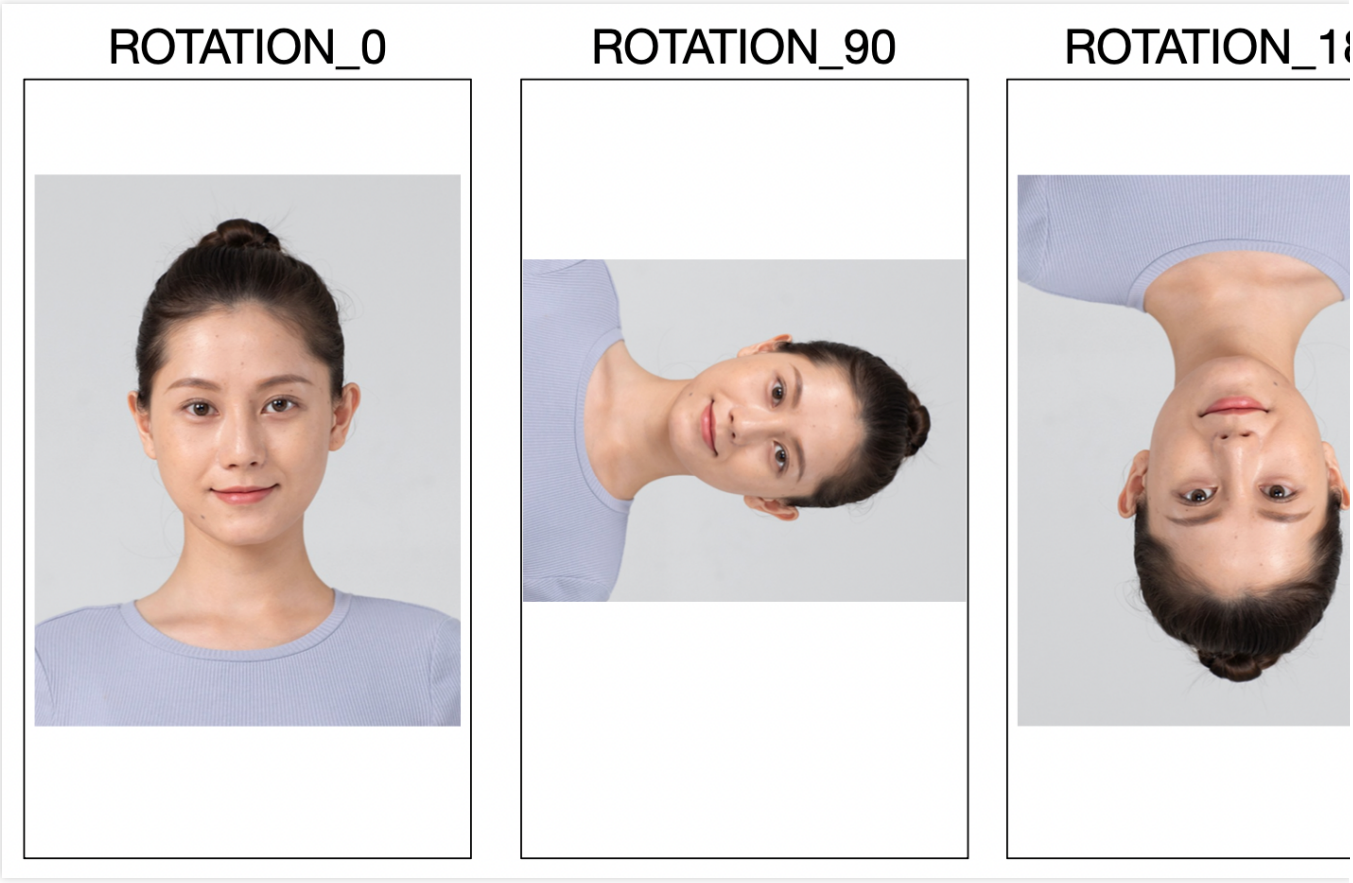
```
// Refer to the sample code in `TECameraBaseActivity.java`
public void onGLContextDestroy() {
    if (this.mXMagicApi != null) {
        this.mXMagicApi.onDestroy();
        this.mXMagicApi = null;
    }
}
```

setImageOrientation

```
void setImageOrientation(TEImageOrientation orientation)
```

```
public enum TEImageOrientation {
    ROTATION_0,
    ROTATION_90,
    ROTATION_180,
    ROTATION_270
}
```

Set the image orientation so that AI can recognize faces from different directions. If set, the direction provided by `sensorChanged` will be ignored. Example directions are as follows:



If the image you provide to the SDK is always upright (Rotation = 0), then you do not need to invoke this interface.

sensorChanged

```
void sensorChanged(android.hardware.SensorEvent event, android.hardware.Sensor acce
```

Use the system sensor to determine the current phone's rotation angle, so that the AI can recognize faces in different orientations. Note: If a fixed direction is set through `setImageOrientation`, the direction provided by `sensorChanged` will be ignored.

Usage example:

```
public class MyActivity implements SensorEventListener {
    private SensorManager mSensorManager;
    private Sensor mAccelerometer;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
    }

    @Override
    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_
    }

    @Override
    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (mXMagicApi != null) {
            mXMagicApi.sensorChanged(event, mAccelerometer);
        }
    }
}
```

isDeviceSupport

```
boolean isDeviceSupport(String motionResPath)
```

Pass the path of a dynamic effect material to the SDK, and detect whether the current device fully supports this dynamic effect.

For example, if a dynamic effect contains lipstick, facial stickers, and background segmentation, if the current device does not support background segmentation, this interface will return false. False does not mean that the dynamic effect cannot be used at all, but some effects cannot be presented.

isSupportBeauty

Determine whether the current model supports beauty (OpenGL3.0). Currently, most mobile phones on the market support OpenGL3.0, so usually there is no need to use this interface.

```
boolean isSupportBeauty()
```

exportCurrentTexture

Retrieve the screen on the current texture

```
void exportCurrentTexture(ExportTextureCallback callback)

public interface ExportTextureCallback {
    void onCallback(Bitmap bitmap);
}
```

setTipsListener

Establish the callback for animated effect cues, utilized for displaying prompts onto the frontend page. For instance, certain materials might instruct users to nod their heads, extend their palms, or make a heart shape.

Note: The callback method may not be on the main thread.

```
void setTipsListener(XmagicApi.XmagicTipsListener effectTipsListener)
```

XmagicTipsListener includes the following methods:

```
public interface XmagicTipsListener {

    /**
     * Display tips.
     * @param tips The returned prompt text information.
     * @param tipsIcon The file path of the tips icon. You can parse the correspond
     *                  Note: tipsIcon may be empty, return null if not configured
     * @param type Tips category, 0 means both tips and tipsIcon have values, 1 mea
     * @param duration Tips display duration, in milliseconds.
     */
    void tipsNeedShow(String tips, String tipsIcon, int type, int duration);

    /**
     * Hide tips.
     * @param tips The returned prompt text information.
     * @param tipsIcon The file path of the tips icon. You can parse the correspond
     *                 Note: tipsIcon may be empty, return null if not configured i
     * @param type Tips category, 0 means both tips and tipsIcon have values, 1 mea
     */
    void tipsNeedHide(String tips, String tipsIcon, int type);
}
```

The example code is as follows:

```
public void tipsNeedShow(final String tips, String tipsIcon, int type, int duration
    final int tipsType = type;
    final String tipsIconPath = tipsIcon;
    mMainThreadHandler.post(new Runnable() {
        @Override
        public void run() {
            if (tipsType == 0) {
                if (!TextUtils.isEmpty(tipsIconPath) && !mImageTipsIsShow) {
                    mTipsImageView.setVisibility(View.VISIBLE);
                    Bitmap bitmap = BitmapUtils.decodeSampleBitmap(AEModule.getCont
                    mTipsImageView.setImageBitmap(bitmap);
                    mImageTipsIsShow = true;
                } else {
                    mTipsImageView.setVisibility(View.GONE);
                }
                mTipsTextView.setText(tips);
            } else {
                pagFile = PAGFile.Load(tipsIconPath);
                tipsPAGView.post(new Runnable() {@
                    Override
                    public void run() {
                        if (pagFile != null) {
                            tipsPAGView.setRepeatCount(-1);
                            tipsPAGView.setComposition(pagFile);
                            tipsPAGView.setProgress(0);
                            tipsPAGView.play();
                            tipsPAGView.setVisibility(View.VISIBLE);
                        }
                    }
                });
            }
        }
    });
}
public void tipsNeedHide(String tips, String tipsIcon, int type) {
    final int tipsType = type;
    final String tipsIconPath = tipsIcon;
    mMainThreadHandler.post(new Runnable() {
        @Override
        public void run() {
            if (tipsType == 0) {
                mTipsContainer.setVisibility(View.GONE);
                mImageTipsIsShow = false;
            } else {
                tipsPAGView.post(new Runnable() {
                    @Override
```

```
                    public void run() {
                        tipsPAGView.stop();
                        tipsPAGView.setVisibility(View.GONE);
                    }
                });
            }
        }
    });
}
```

Summary of Deprecated Interfaces

updateProperty,updateProperties

Deprecated, please use the setEffect interface.

Modifies either a single beauty effect or multiple beauty effects, dynamic effects, filters in bulk, and can be invoked from any thread.

```
void updateProperty(XmagicProperty<?> p)
void updateProperties(List<XmagicProperty<?>> properties)
```

Parameter	Meaning
XmagicProperty<?> p	Tencent Effect data entity class. Taking "Skin Smoothing" as an example, an instance can be created as follows: `new XmagicProperty<>(Category.BEAUTY, null, null, BeautyConstant.BEAUTY_SMOOTH, new XmagicPropertyValues(0, 100, 50, 0, 1));` To take '2D Animated Effect Bunny Paste' as an example, you can instantiate a new instance in the following manner: `new XmagicProperty<>(Category.MOTION, "video_tutujiang", "path of the effect file", null, null);` If you want a certain animation/beauty/masking material to be overlaid on the current material, set the mergeWithCurrentMotion of that material's XmagicProperty object to true. For a detailed explanation of material overlay, see Material Overlay. For more examples, please refer to the configuration information in the Demo project's `assets/beauty_panel` folder.

XmagicProperty

Deprecated. Please use the setEffect interface to set beauty effects.

Beauty filter

Attribute	Description

Field	
category	Category.BEAUTY
ID	null Special case: The respective ID values for Natural, Goddess, and Handsome in the Face Slimming feature are: `BeautyConstant.BEAUTY_FACE_NATURE_ID`, `BeautyConstant.BEAUTY_FACE_FEMALE_GOD_ID`, `BeautyConstant.BEAUTY_FACE_MALE_GOD_ID` The ID value in the lipstick is: `XmagicConstant.BeautyConstant.BEAUTY_LIPS_LIPS_MASK` The ID value in the blush is: `XmagicConstant.BeautyConstant.BEAUTY_MAKEUP_MULTIPLY_MULTIPLY_MASK` The ID value in 3D is: `XmagicConstant.BeautyConstant.BEAUTY_SOFTLIGHT_SOFTLIGHT_MASK`
resPath	null Special Case: Lipstick, blush, and three-dimensional resPath represent the paths of resource images. For details, please refer to the `assets/beauty_panel/advanced_beauty.json` file in the Demo.
effkey	Required, refer to the Demo Example: Brightening BeautyConstant.BEAUTY_WHITEN
effValue	Required, refer to the Demo's assets/beauty_panel/advanced_beauty.json file. In the demo project, parse this file to construct an XmagicPropertyValues object. See Effect Parameters for the values of each attribute of XmagicPropertyValues

Body beautification

Attribute Field	Description
category	Category.BODY_BEAUTY
ID	null
resPath	null
effkey	Required, refer to the Demo Example: Long-legged BeautyConstant.BODY_LEG_STRETCH
effValue	Required, refer to the Demo's assets/beauty_panel/beauty_body.json file. In the demo project, parse this file to construct an XmagicPropertyValues object. See Effect Parameters for the values of each attribute of XmagicPropertyValues

Filters

Attribute Field	Description
category	Category.LUT
ID	Image name, required Sample: dongjing_lf.png The "none" ID corresponds to XmagicProperty.ID_NONE
resPath	Filter image path, mandatory, "none" setting as null
effkey	null
effValue	Required, set to null if "none". This is an XmagicPropertyValues object. See Effect Parameters for the values of each attribute of XmagicPropertyValues

Animated Effects

Attribute Field	Description
category	Category.MOTION
ID	Resource folder name, required Example: video_lianliancaomei The "none" ID corresponds to XmagicProperty.ID_NONE
resPath	Required, refer to the Demo
effkey	null
effValue	null

Makeup

Attribute Field	Description
category	Category.MAKEUP
ID	Resource folder name, required Example: video_xuejiezhuang The "none" ID corresponds to XmagicProperty.ID_NONE
resPath	Required, refer to the Demo
effkey	Mandatory. The value is: makeup.strength"None" Setting is null

effValue	Mandatory, "None" Setting is null. This is an XmagicPropertyValues object, for the values of various properties of XmagicPropertyValues, see Beauty Parameter Explanation

Divide

Attribute Field	Description
category	Category.SEGMENTATION
ID	Resource folder name, required Example: video_segmentation_blur_45 The "none" ID corresponds to XmagicProperty.ID_NONE From the Definition, the ID value must use: `XmagicConstant.SegmentationId.CUSTOM_SEG_ID`
resPath	Required, refer to the Demo
effkey	null (excluding custom definition background), the value of the custom definition background is the selected resource path
effValue	null

setYTDataListener

Deprecated interface, please use the setAIDataListener callback onAIDataUpdated.

onPauseAudio

Invoke this function when only the cessation of the audio is necessitated, without the need to release the GL thread.

getPropertyRequiredAbilities

Inputs an animated effect resources list, returns the SDK atomic abilities list used for each resource.
The usage scenario of this method:
If you have purchased or created a number of animated effect materials, by invoking this method, it will returning a list of atomic abilities required for each material. For instance, material 1 requires abilities A, B, C while material 2 need abilities B, C, D. Subsequently, you maintain such a list of abilities on your server. Later on, when a user wants to download the animated effect materials from the server, the user first accesses the list of atomic abilities that his mobile has through the getDeviceAbilities method (for example, this phone possess abilities A, B, C, but lacks D), transmits this abilities list to the server. The server, on determining the device does not have ability D, therefore does not issue material 2 to the user.

Parameter	Meaning
List<XmagicProperty<?>> assets	List of animated effect resources for which to check the atomic

	capabilities.

Returned value `Map<XmagicProperty<?>,ArrayList<String>>` :
key: animated effect resource material entity class.
value: list of used atomic capabilities.

getDeviceAbilities

It returns the atomic capability table that the current device supports. To be used in conjunction with the getPropertyRequiredAbilities method. For more details, please refer to the description of the getPropertyRequiredAbilities.

```
Map<String,Boolean> getDeviceAbilities()
```

Return value `Map<String,Boolean>` :
key: atomic capability name (corresponding to the material capability name).
value: whether it is supported by the current device.

isBeautyAuthorized

Determine which beauty or body modifier features the current License authorization supports. Only detection of BEAUTY and BODY_BEAUTY types are supported. The resultant evaluation will be allocated to each beauty object's `XmagicProperty.isAuth` field. If the isAuth field returns false, the entrances for these features can be concealed in the UI.

```
void isBeautyAuthorized(List<XmagicProperty<?>> properties)
```

enableEnhancedMode

Enable Beauty Enhancement Mode. Disabled by default. See Enhancement Mode Usage Guide for details.

setDowngradePerformance

Please use the enableHighPerformance interface to enable High-Performance Mode.

Flutter

Last updated：2024-08-30 17:36:51

`TencentEffectApi` is the core API class of the Tencent Effect Flutter SDK. It offers capabilities including setting the effect strength and applying animated effects.

Public Member APIs

API	Description
setResourcePath	Set the local storage path of beauty resources (V0.3.5.0 version)
initXmagic	Initializes data. You need to call this API before using the Tencent Effect SDK（**V0.3.1.1 and earlier**）.
setLicense	Configures the license.
setXmagicLogLevel	Sets the log level of the SDK. We recommend you set it to `Log.DEBUG` for debugging and `Log.WARN` for official release. If you set it to `Log.DEBUG` in a production environment, the output of a large amount of log data may affect your application's performance.
onResume	Resumes rendering. Call this API when the page is visible.
onPause	Pauses rendering. Call this API when the page is invisible.
enableEnhancedMode	enable enhanced mode,for specific instructions, please refer to the enhanced mode usage guide.
setDowngradePerformance	Invoke this method to enable the high-performance pattern. Upon the activation of the high-performance pattern, the system CPU/GPU resources occupied by beauty filters are minimized, thereby reducing heat generation and latency issues in the mobile device. It is particularly suitable for prolonged use on low-end devices.
setAudioMute	set mute（**Because some stickers have sound**）
setFeatureEnableDisable	enable or disable the feature
@Deprecated updateProperty	Updates an effect property. This API can be called in any thread.**(This interface has been deprecated)**
setEffect	Updates an effect property（**V0.3.5.0 Version**）

setOnCreateXmagicApiErrorListener	Configures the callback for creating an effect object. The callback will be triggered if an error occurs.
setTipsListener	Configures the callback for animated effect tips. The tips can be displayed on the UI.
setYTDataListener	Configure the callback of facial keypoints and other data, callback will only be available with the License authorization required to acquire facial keypoints (such as Atomic Capability X102).
setAIDataListener	Configures the callback of face, gesture, and body detection results.
@Deprecated isBeautyAuthorized	Checks whether the current license supports a particular type of effects. This API can only check the authorization of `BEAUTY` and `BODY_BEAUTY` effects. The result returned determines the value of `XmagicProperty.isAuth` .**(This interface has been deprecated)**
isSupportBeauty	Checks whether the current device supports effects (OpenGL 3.0).
getDeviceAbilities	Gets a list of Tencent Effect capabilities supported by the current device.
@Deprecated isDeviceSupport	Checks whether a list of animated effect resources are supported. The result is indicated by `XmagicProperty.isSupport` . For unsupported resources, you can either disable tapping on the UI or delete them from the resource list.**(This interface has been deprecated)**
isDeviceSupportMotion	Check if the current device supports this material.
@Deprecated getPropertyRequiredAbilities	Gets the Tencent Effect capabilities used by different animated effect resources.**(This interface has been deprecated)**

API Description

setResourcePath （V0.3.5.0）

To set the local path for storing beauty resources

```
/// Set the local path for storing beauty resources. This method must be called bef
/// Added in v0.3.5.0.
void setResourcePath(String xmagicResDir);
```

Parameters

Parameter	Description
String xmagicResDir	The resource directory.

initXmagic

Initialize beauty data. In versions prior to `V0.3.1.1`, this method must be called before using beauty effects. Starting from `V0.3.5.0`, this method only needs to be called once per version, and the setResourcePath method must be called before this method to set the resource path. In `V0.3.5.0`, the previous xmagicResDir parameter has been removed. Please refer to the latest demo for more information.

V0.3.5.0 :

```
void initXmagic(InitXmagicCallBack callBack);

typedef InitXmagicCallBack = void Function(bool reslut);
```

V0.3.1.1 and earlier :

This API is used to initialize the Tencent Effect SDK.

```
void initXmagic(String xmagicResDir,InitXmagicCallBack callBack);

typedef InitXmagicCallBack = void Function(bool reslut);
```

Parameters

Parameter	Description
String xmagicResDir	The resource directory.
InitXmagicCallBack callBack	The initialization callback.

setLicense

This API is used to set the license.

```
  ///Set the Tencent Effect license
void setLicense(String licenseKey, String licenseUrl, LicenseCheckListener checkLis
//The callback of the authorization result
typedef LicenseCheckListener = void Function(int errorCode, String msg);
```

Parameters

Parameter	Description

String licenseKey	The license key.
String licenseUrl	The license URL.
LicenseCheckListener checkListener	The callback of the authorization result.

setXmagicLogLevel

This API is used to set the log level of the SDK.

```
void setXmagicLogLevel(int logLevel);
```

Parameters

Parameter	Description
int logLevel	You can set the log level using a type defined for `LogLevel`.

onResume

This API is used to resume effect rendering.

```
void onResume();
```

onPause

This API is used to pause effect rendering.

```
void onPause();
```

enableEnhancedMode

enable enhanced mode,for specific instructions, please refer to the enhanced mode usage guide.

```
void enableEnhancedMode();
```

setDowngradePerformance（V0.3.1.1）

Invoke this method to enable the high-performance pattern

```
void setDowngradePerformance();
```

setAudioMute（V0.3.1.1）

To set the mute status, the parameter "true" represents mute, while "false" represents unmute.

```
/// Is the background music muted?
void setAudioMute(bool isMute);
```

setFeatureEnableDisable（V0.3.1.1）

enable or disable one feature

```
/// enable or disable one feature
void setFeatureEnableDisable(String featureName, bool enable);
```

Parameter

Parameter	Meaning
String featureName	feature Name Values： `"ai.3dmmV2.enable"` facial expressions feature. `"ai.body3dpoint.enable"` 3D body data feature. `"ai.hand.enable"` gesture detection. `"beauty.onlyWhitenSkin"` Brightening only applies to skin. `"ai.segmentation.skin.enable"` segmentation skin. `"auto_beauty_switch"` smart beauty(reducing the intensity of beauty and makeup effects for males and babies).
boolean enable	"true" indicates enabling a capability, while "false" indicates disabling a capability. **Note: If it is in downgrade mode, enabling skin segmentation is not allowed.**

updateProperty (This interface has been deprecated)

This API is used to set an effect value, an animated effect, or a filter. You can call it in any thread.

```
void updateProperty(XmagicProperty xmagicProperty);
```

Parameters

Parameter	Description
XmagicProperty xmagicProperty	The object of the effect property.

setEffect（V0.3.5.0）

You can set beautification, filters, makeup, stickers, and segmentation effects. This can be done from any thread.

Please refer to the specific parameters for more detailsBeautification Parameter Table。

```
///update beautification parameters
void setEffect(String effectName,int effectValue,String? resourcePath,Map<String,St
```

setOnCreateXmagicApiErrorListener

This API is used to configure the callback for errors for the creation of an effect object.

```
  void setOnCreateXmagicApiErrorListener(OnCreateXmagicApiErrorListener? errorListe
/// The callback for errors for the creation of an effect object
typedef OnCreateXmagicApiErrorListener = void Function(String errorMsg, int code);
```

Parameters

Parameter	Description
OnCreateXmagicApiErrorListener? errorListener	The callback for errors for the creation of an effect object.

Error codes:

Error Code	Description
-1	Unknown error.
-100	Failed to initialize the 3D engine.
-200	GAN materials are not supported.
-300	The device does not support this material component.
-400	The JSON template is empty.
-500	The SDK version is too old.
-600	Keying is not supported.
-700	OpenGL is not supported.
-800	The script is not supported.
5000	The resolution of the video to be keyed exceeds 2160 x 3840.
5001	Insufficient memory for keying.
5002	Failed to parse the video to be keyed.
5003	The video to be keyed is longer than 200 seconds.

5004	Unsupported video format for keying.

setTipsListener

This API is used to configure the callback for animated effect tips. The tips can be displayed on the UI, asking users to nod, show their palms, or make finger hearts.

```
void setTipsListener(XmagicTipsListener? xmagicTipsListener);

abstract class XmagicTipsListener {
  /// Show the tip
  /// @param tips: The content of the tip (string).
  /// @param tipsIcon: The icon for the tip.
  /// @param type: The display type. If it is set to `0`, both the tip string and i
  /// @param duration: How long (milliseconds) to show the tip.
  void tipsNeedShow(String tips, String tipsIcon, int type, int duration);

  /// *
  /// Hide the tip
  /// @param tips: The content of the tip (string).
  /// @param tipsIcon: The icon for the tip.
  /// @param type: The display type. If it is set to `0`, both the tip string and i
  void tipsNeedHide(String tips, String tipsIcon, int type);
}
```

Parameters

Parameter	Description
XmagicTipsListener xmagicTipsListener	The callback implementation class.

setYTDataListener

This API is used to configure the callback of facial keypoints and other data.

```
  /// Configure the callback of facial keypoints and other data (only available in
void setYTDataListener(XmagicYTDataListener? xmagicYTDataListener);
Configure the callback of facial keypoints and other data

abstract class XmagicYTDataListener {
  // YouTu AI data
  void onYTDataUpdate(String data);
}
```

`onYTDataUpdate` returns a JSON string structure that contains the information of up to 5 faces:

```
{
 "face_info":[{
  "trace_id":5,
  "face_256_point":[
    180.0,
    112.2,
    ...
  ],
  "face_256_visible":[
    0.85,
    ...
  ],
  "out_of_screen":true,
  "left_eye_high_vis_ratio:1.0,
  "right_eye_high_vis_ratio":1.0,
  "left_eyebrow_high_vis_ratio":1.0,
  "right_eyebrow_high_vis_ratio":1.0,
  "mouth_high_vis_ratio":1.0
 },
 ...
 ]
}
```

Fields

Field	Type	Range	Description
trace_id	int	[1,INF)	The face ID. If the faces obtained from a continuous video stream have the same face ID, they belong to the same person.
face_256_point	float	[0,screenWidth] or [0,screenHeight]	512 values in total for 256 facial keypoints. (0,0) is the top-left corner of the screen.
face_256_visible	float	[0,1]	The visibility of the 256 facial keypoints.
out_of_screen	bool	true/false	Whether only part of the face is captured.
left_eye_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the left eye.
right_eye_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the right eye.
left_eyebrow_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the left eyebrow.

right_eyebrow_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the right eyebrow.
mouth_high_vis_ratio	float	[0,1]	The percentage of keypoints with high visibility for the mouth.

Parameters

Parameter	Description
XmagicYTDataListener	The callback implementation class.

setAIDataListener

This API is used to configure the callback of face, gesture, and body detection results.

```
void setAIDataListener(XmagicAIDataListener? aiDataListener);

abstract class XmagicAIDataListener {
  void onFaceDataUpdated(String faceDataList);

  void onHandDataUpdated(String handDataList);

  void onBodyDataUpdated(String bodyDataList);
}
```

isBeautyAuthorized (This interface has been deprecated)

This API is used to check whether the current license supports a particular type of effects. It can only check the authorization of `BEAUTY` and `BODY_BEAUTY` effects. The result returned determines the value of `XmagicProperty.isAuth`. If `isAuth` is `false`, you can disable the corresponding effects on the UI.

```
Future<List<XmagicProperty>> isBeautyAuthorized(
      List<XmagicProperty> properties);
```

Parameters

Parameter	Description
List<XmagicProperty> properties	The type of effects to check.

isSupportBeauty

This API is used to check whether the current device supports effects (OpenGL 3.0).

```
Future&lt;bool> isSupportBeauty();
```

Response

A Boolean value indicating whether effects are supported.

getDeviceAbilities

This API is used to get a list of Tencent Effect capabilities supported by the current device. You can use it together with `getPropertyRequiredAbilities`.

```
Future<Map<String, bool>> getDeviceAbilities();
```

Response

`Map<String,Boolean>`:
key: The name of a capability (the material name).
value: Whether the current device supports the capability.

isDeviceSupport (This interface has been deprecated)

This API is used to check whether a list of animated effect resources are supported. The result is indicated by `XmagicProperty.isSupport`. For unsupported resources, you can either disable tapping on the UI or delete them from the resource list.

```
Future&lt;List<XmagicProperty>> isDeviceSupport(List<XmagicProperty> assetsList);
```

Parameters

Parameter	Description
List<XmagicProperty> assetsList	A list of animated effect resources to check.

isDeviceSupportMotion (V0.3.5.0)

To check if the current device supports a particular material

```
Future<bool> isDeviceSupportMotion(String motionResPath);
```

Parameters

Parameter	Description
motionResPath	the sticker local file path

getPropertyRequiredAbilities (This interface has been deprecated)

This API is used to get the Tencent Effect capabilities used by different animated effect resources.
Use case:
This API is useful if you have purchased animated effects or made your own animated effect materials. It returns the capabilities each material uses. For example, material 1 uses capabilities A, B, and C, and material 2 relies on capabilities B, C, and D. You can store such information in the server. When a user downloads the two materials from the server, call `getDeviceAbilities` first to get the capabilities supported by their device. The result is then passed to the server. For example, if a user's device supports capabilities A, B, and C, but not D, the server will not provide material 2 to the user.

```
Future<Map<XmagicProperty, List<String>?>> getPropertyRequiredAbilities(
    List<XmagicProperty> assetsList);
```

Parameters

Parameter	Description
List<XmagicProperty> assetsList	A list of the animated effects to check.

Response

Map<XmagicProperty, List<String>?>:
key: The entity class of the animated effect.
value: A list of capabilities used by the effect.

Web

Last updated：2023-05-06 15:45:32

This document describes the core parameters and methods of the Beauty AR Web SDK.

Note:

The Beauty AR Web SDK relies on hardware acceleration to achieve smooth rendering (this does not need to be checked in mini programs). The SDK allows you to check whether a browser supports hardware acceleration. You can block the browser if it does not support hardware acceleration.

```
import {ArSdk, isWebGLSupported} from 'tencentcloud-webar'

if(isWebGLSupported()) {
    const sdk = new ArSdk({
    ...
})
} else {
    // The browser blocking logic
}
```

Initialization Parameters

```
import { ArSdk } from 'tencentcloud-webar'
// Initialize the SDK
const sdk = new ArSdk({
...
})
```

`Config` of the SDK supports the following initialization parameters:

Parameters	Description	Type
module	The module configuration	`type bea seg }`
auth	The authentication parameter	`type lic app`

		aut s t }> }
input	Source	MediaStr
camera	Built-in Camera	type w h m f }
beautify	The beauty filter parameter	type w c l s e c }
background	The background parameter	type t s }
loading	The configuration of the built-in loading icon	type e s l s }
language	The language (supported since v1.0.6). Chinese (zh) and English (en) are supported.	String: zh

Callbacks

```
let effectList = [];
let filterList = [];
// Using the callbacks of the SDK
sdk.on('created', () => {
    // Pull and display the filter and effect list in the `created` callback
    sdk.getEffectList({
        Type: 'Preset',
        Label: 'Makeup',
    }).then(res => {
        effectList = res
    });
    sdk.getCommonFilter().then(res => {
        filterList = res
    })
})
sdk.on('cameraReady', async () => {
    // By getting the output stream in the `cameraReady` callback, you can display
    // You can choose this method if you want to display a video image as soon as p
    // You don't need to update the stream after the effects start to work.
    const arStream = await ar.getOutput();
    // Play the stream locally
    // localVideo.srcObject = arStream

})
sdk.on('ready', () => {
    // Get the output stream in the `ready` callback. The initialization parameters
    // You can get the output stream in `ready` if you want your video to show effe
    // Between the two methods, choose the one that fits your needs.
    const arStream = await ar.getOutput();
    // Play the stream locally
    // localVideo.srcObject = arStream

    // Call `setBeautify`, `setEffect`, or `setFilter` in the `ready` callback
    sdk.setBeautify({
        whiten: 0.3
    });
    sdk.setEffect({
        id: effectList[0].EffectId,
        intensity: 0.7
    });
    sdk.setEffect({
        id: effectList[0].EffectId,
        intensity: 0.7,
```

```
        filterIntensity: 0.5 // In v0.1.18 and later, you can use this parameter to
    });
    sdk.setFilter(filterList[0].EffectId, 0.5)
})
```

Events	Description	Callback Parameter
created	The SDK authentication was completed and the instance was created successfully.	-
cameraReady	The SDK generated a video output (the video is not yet processed).	-
ready	Detection has been initialized. Effects are now applied to the output video. You can change the effect settings.	-
error	This callback is triggered when the SDK encounters an error.	The `error` object

APIs

API	Parameters
async getOutput(fps)	fps (optional): The output frame rate.
setBeautify(options)	`type BeautifyOptions = { whiten?: number, // The brightening effect. Value dermabrasion?: number // The smooth skin effect. lift?: number // The slim face effect. Value rang shave?: number // The face width. Value range: 0- eye?: number // The big eyes effect. Value range: chin?: number // The chin effect. Value range: 0- }`
	effects: Effect ID | Effect object | Effect ID / An effect array

setEffect(effects, callback)	`effect:{ id: string, intensity: number, // The effect strength. Valu filterIntensity: number // The filter strength }` callback: The callback for successful configuration
setAvatar(params)	`{ mode: 'AR' | 'VR', effectId?: string, // Pass through `effectId` t url?: string, // Pass through `url` to use the backgroundUrl?: string, // Background image URI }`
setBackground(options)	`{ type: 'image|blur|transparent', src: string // This parameter is required only }`
setFilter(id, intensity, callback)	id: The filter ID intensity: The filter strength. Value range: 0–1. callback: The callback for successful configuration
getEffectList(params)	`{ PageNumber: number, PageSize: number, Name: '', Label: Array, Type: 'Custom|Preset' }`

getAvatarList(type)	`type = 'AR' | 'VR'`
getEffect(effectId)	effectId: The effect ID
getCommonFilter()	-
async updateInputStream(src:MediaStream) **(supported since v0.1.19)**	src: New input stream (`MediaStream`)
disable()	-
enable()	-

async startRecord()	-
async stopRecord()	`{ useOriginAudio: boolean, // Whether to record t musicPath: string, // The background music URL, }`
async takePhoto()	-
destroy()	-

Error Handling

The error object returned by the error callback includes the error code and error message, which facilitate troubleshooting.

```
sdk.on('error', (error) => {
    // Handle an error in the error callback
    const {code, message} = error
    ...
})
```

Error Code	Description	Remarks
10000001	The current browser environment is incompatible.	Recommend that the user use Chrome, Firefox, Safari, or the Weixin browser.
10000002	The render context is missing.	-
10000003	The rendering is slow.	Consider reducing the video resolution or disabling the feature.
10000005	An error occurred while parsing the input source.	-
10000006	Lag may occur due to insufficient browser support.	Recommend that the user use Chrome, Firefox, Safari, or the Weixin browser.
10001101	An error occurred while configuring the effect.	-
10001102	An error occurred while configuring the filter.	-
10001103	The effect strength parameter is incorrect.	-
10001201	Failed to turn on the camera.	-
10001202	The camera stopped.	-
10001203	Failed to get the camera permission.	The user needs to enable the camera permission by going to **Settings** > **Privacy** > **Camera**.
20002001	The authentication parameter is missing.	-
20001001	Authentication failed	Make sure you have created a license and the signature is correct.
20001002	The API request failed.	The error callback will return the data

		returned by the API. For details, see API Error Codes.
20001003	Failed to authenticate the effect configuration API.	The API is inaccessible. A Standard license cannot use the features of an Advanced license.
30000001	Failed to call `startRecord` in the mini program.	-
30000002	Failed to call `stopRecord` in the mini program.	-
40000000	An uncaught exception occurred.	-
40000001	As the current SDK version is too early, certain effects cannot be correctly displayed. Upgrade the SDK version.	-
50000002	The effect is lost due to the resolution change.	The effect needs to be reconfigured.

Handling the missing render context error

On some computers, if the SDK is in the background for a long time, the `contextlost` error may occur. In such cases, you can call `ArSdk.prototype.resetCore(input: MediaStream)` to resume rendering.

```
sdk.on('error', async (error) => {
    if (error.code === 10000002) {
        const newInput = await navigator.mediaDevices.getUserMedia({...})
        await sdk.resetCore(newInput)
    }
})
```