腾讯特效 SDK

API 文档

产品文档

腾讯云

文档目录

API 文档
- iOS
- Android
- Flutter
- Web

API 文档

iOS

最近更新时间：2024-03-19 15:50:12

腾讯特效 SDK 核心接口类 `XMagic.h`，用于初始化 SDK、更新美颜数值、调用动效等功能。

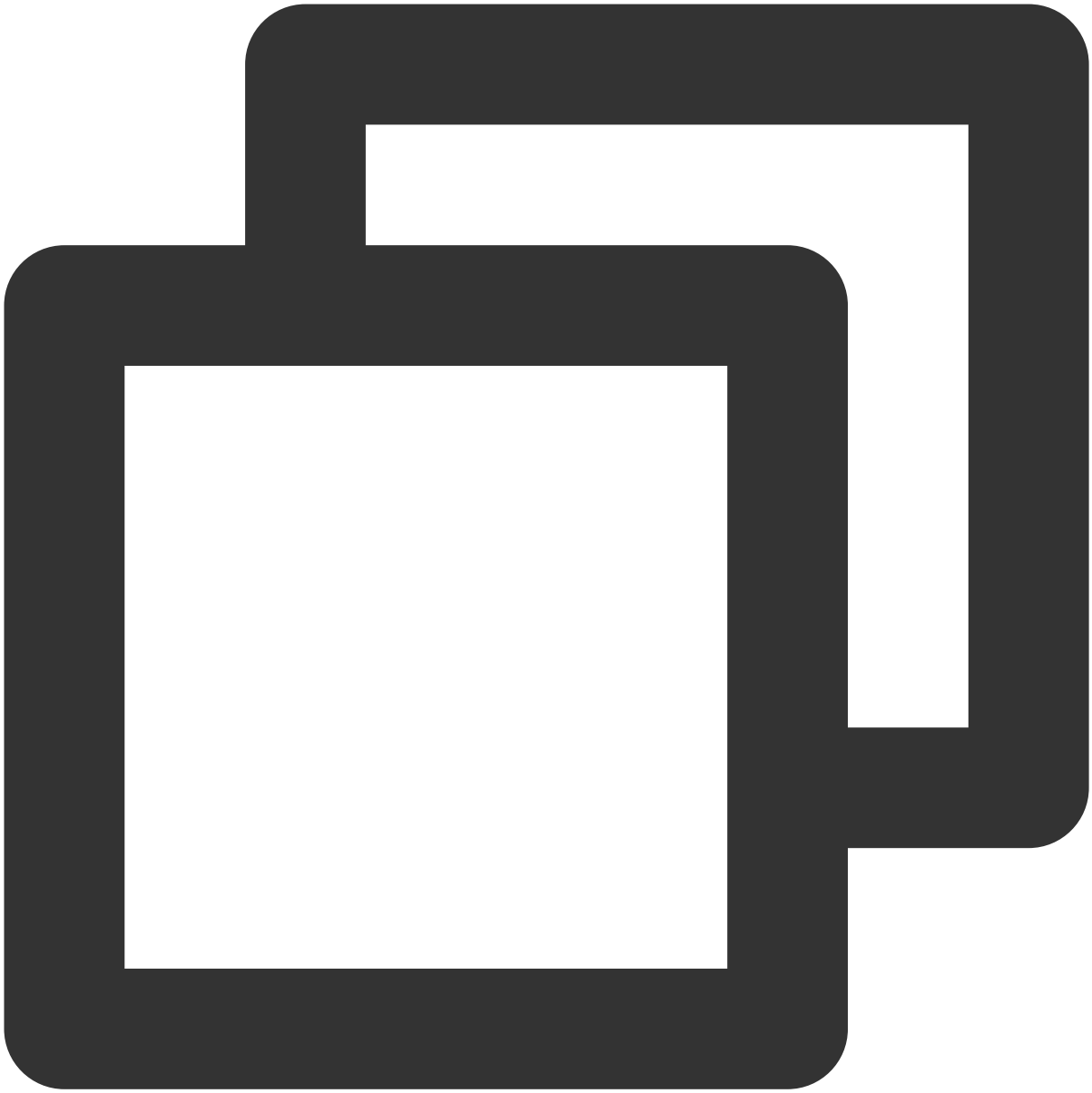
Public 成员函数

API	描述
initWithRenderSize	初始化接口
initWithGlTexture	初始化接口
configPropertyWithType	配置美颜各种效果
setEffect	配置美颜各种效果(3.5.0.2新增)
emitBlurStrengthEvent	设置后处理模糊强度（作用于所有模糊组件）
setRenderSize	设置 renderSize
deinit	资源释放接口
process:	图像数据处理接口，输入美颜前的图像，返回美颜后的图像。
process:withOrigin:withOrientation:	图像数据处理接口，输入美颜前的图像，返回美颜后的图像，比上接口多两个参数。
processUIImage	处理图片
getConfigPropertyWithName	获取美颜参数配置信息
registerLoggerListener	日志注册接口
registerSDKEventListener	SDK 事件监听接口
clearListeners	注册回调清理接口
getCurrentGlContext	获取当前 GL 上下文接口
onPause	SDK 暂停接口
onResume	SDK 恢复接口

setAudioMute	动效素材使用时是否开启静音（V2.5.0新增） 参数：YES表示静音，NO表示非静音
enableEnhancedMode	开启美颜增强模式（V2.5.1新增）。默认未开启。 未开启时，应用层可以设置的各美颜项的强度范围为 0到1 或 -1到1，如果超出此范围，SDK 会取边界值。例如应用层设置瘦脸为1.2，SDK判断其超出了最大值1.0，则在内部把瘦脸值修正为1.0。 开启增强模式后，应用层可以设置更大范围的数值。例如想要瘦脸程度更大，则可以把瘦脸值设置为1.2，SDK 会接受并使用1.2这个数值，不会将其修正为1.0。 **说明：** 开启增强模式后，需要应用层自己管理每个美颜项可以设置的最大值，让用户在此范围内调整数值。我们提供了一份 参考值，您可以根据产品需求自由调整，但不建议超出我们的推荐值，否则美颜效果可能变差。
素材叠加	如果想要某个动效/美妆/分割素材叠加在当前素材上，则设置该素材时，在 withExtraInfo 的字典中设置 mergeWithCurrentMotion 为 true
高性能模式	在 SDK 初始化时，字典中添加 `@"setDowngradePerformance":@(YES)` 。高性能模式开启后，美颜占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象，更适合低端机长时间用。

initWithRenderSize

初始化接口



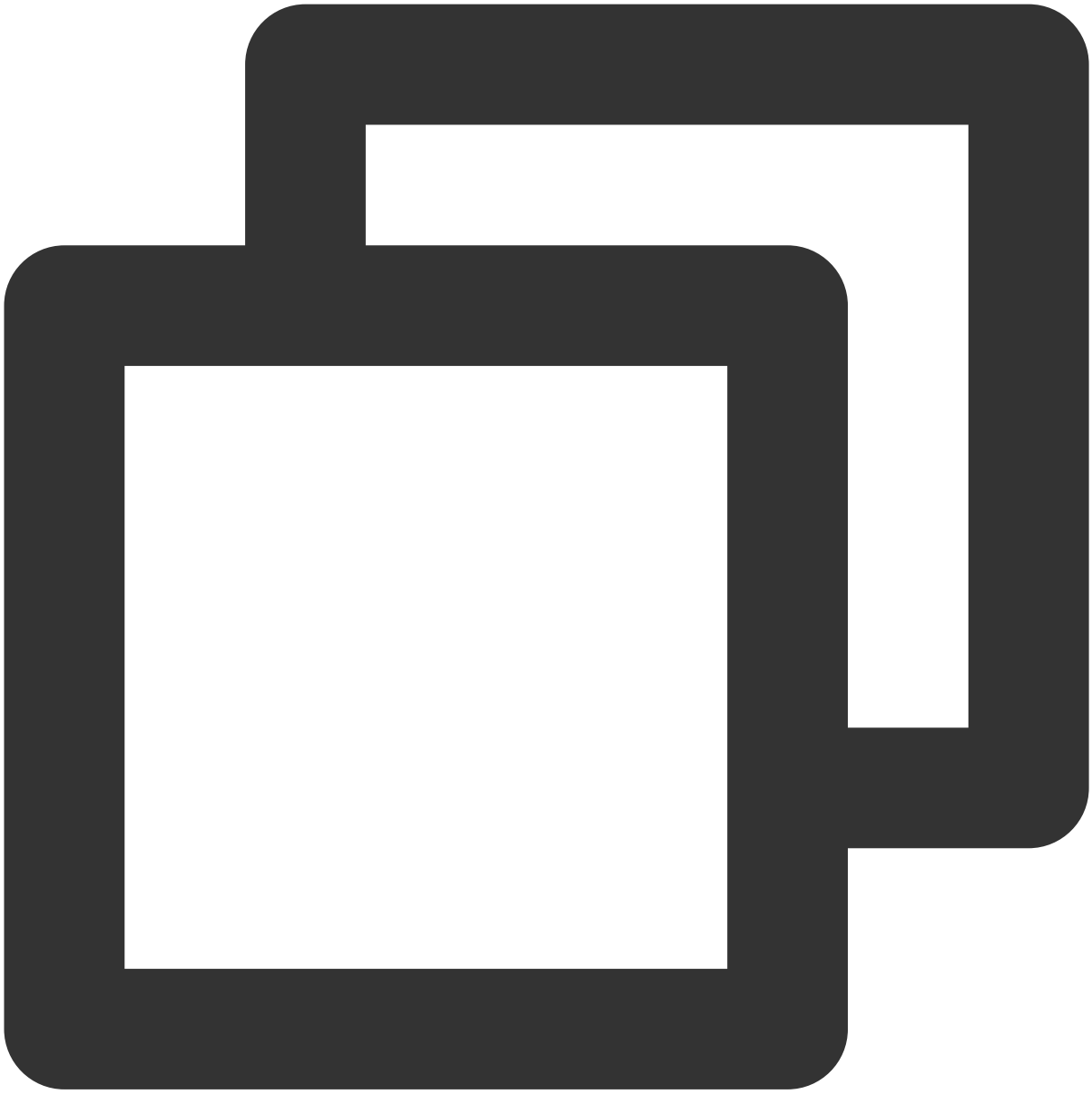
```
- (instancetype _Nonnull)initWithRenderSize:(CGSize)renderSize
                                assetsDict:(NSDictionary* _Nullable)assetsDict;
```

参数

参数	含义
renderSize	渲染尺寸
assetsDict	资源 Dict

initWithGlTexture

初始化接口



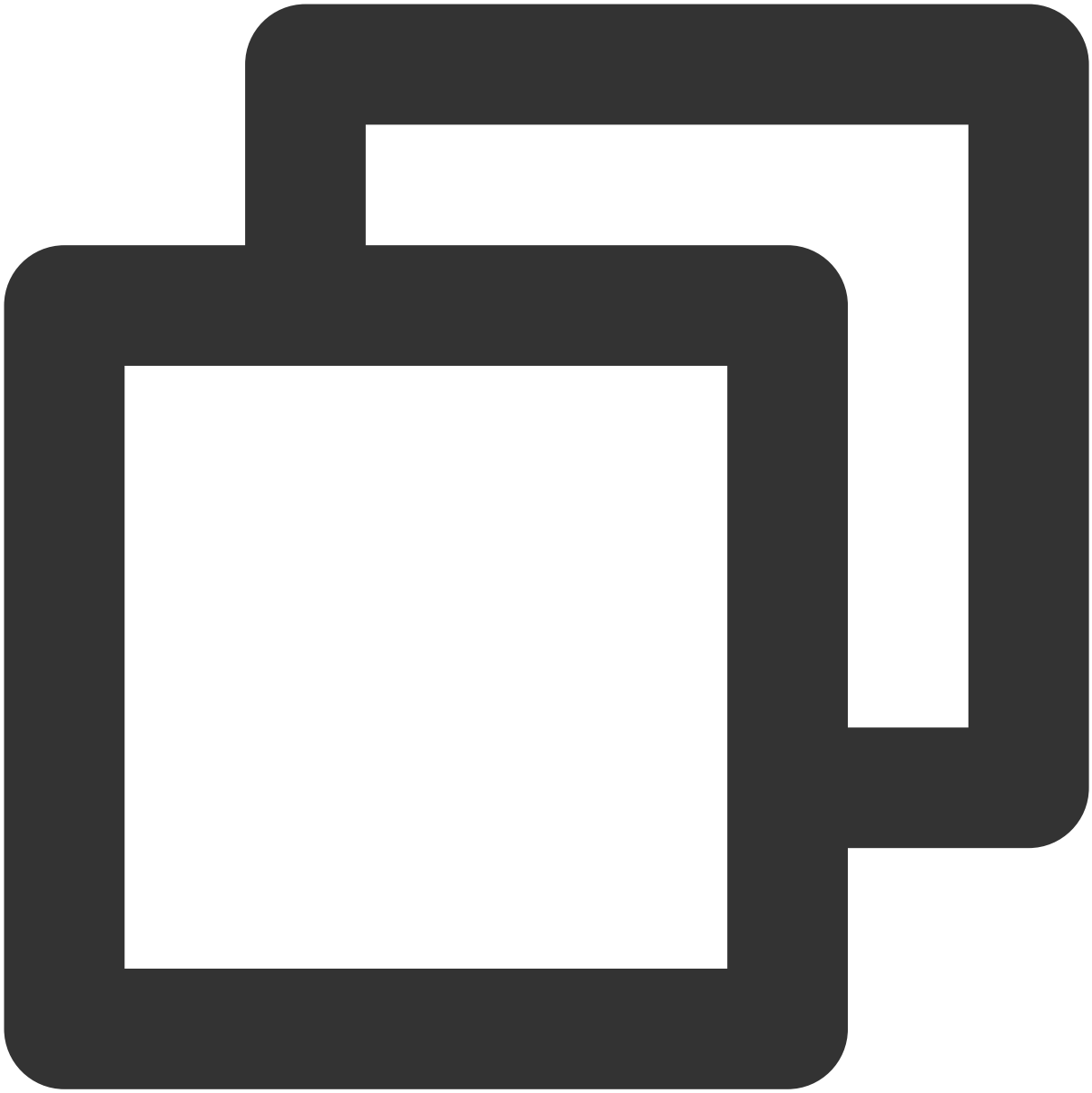
```
- (instancetype _Nonnull)initWithGlTexture:(unsigned)textureID
                        width:(int)width
                        height:(int)height
                        flipY:(bool)flipY
                        assetsDict:(NSDictionary* _Nullable)assetsDict;
```

参数

参数	含义
textureID	纹理 ID
width	渲染尺寸
height	渲染尺寸
flipY	是否翻转图片
assetsDict	资源 Dict

configPropertyWithType

配置美颜各种效果



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

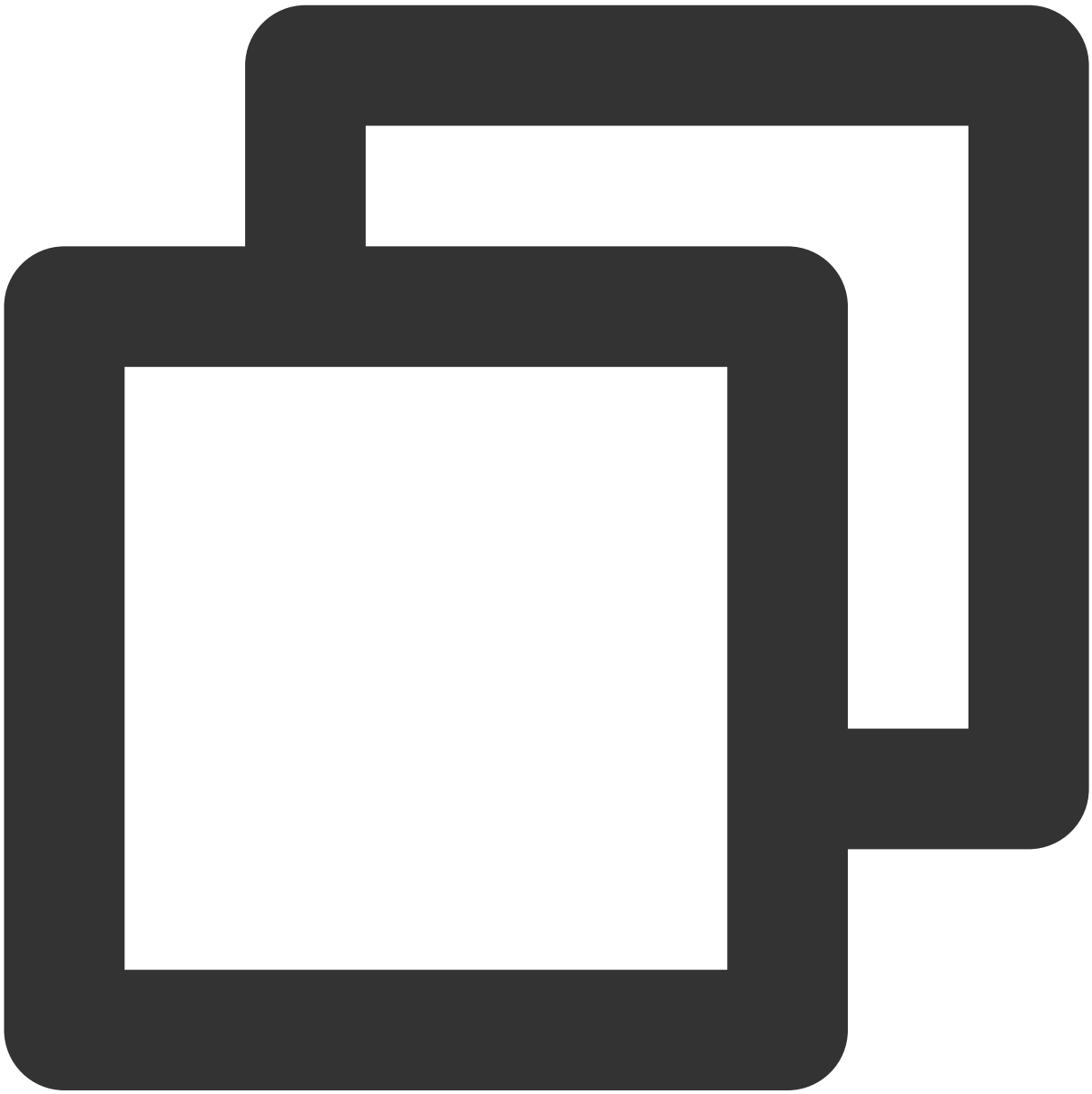
参数

参数	含义
propertyType	效果类型
propertyName	效果名称
propertyValue	效果数值

extraInfo	预留扩展, 附加额外配置 Dict

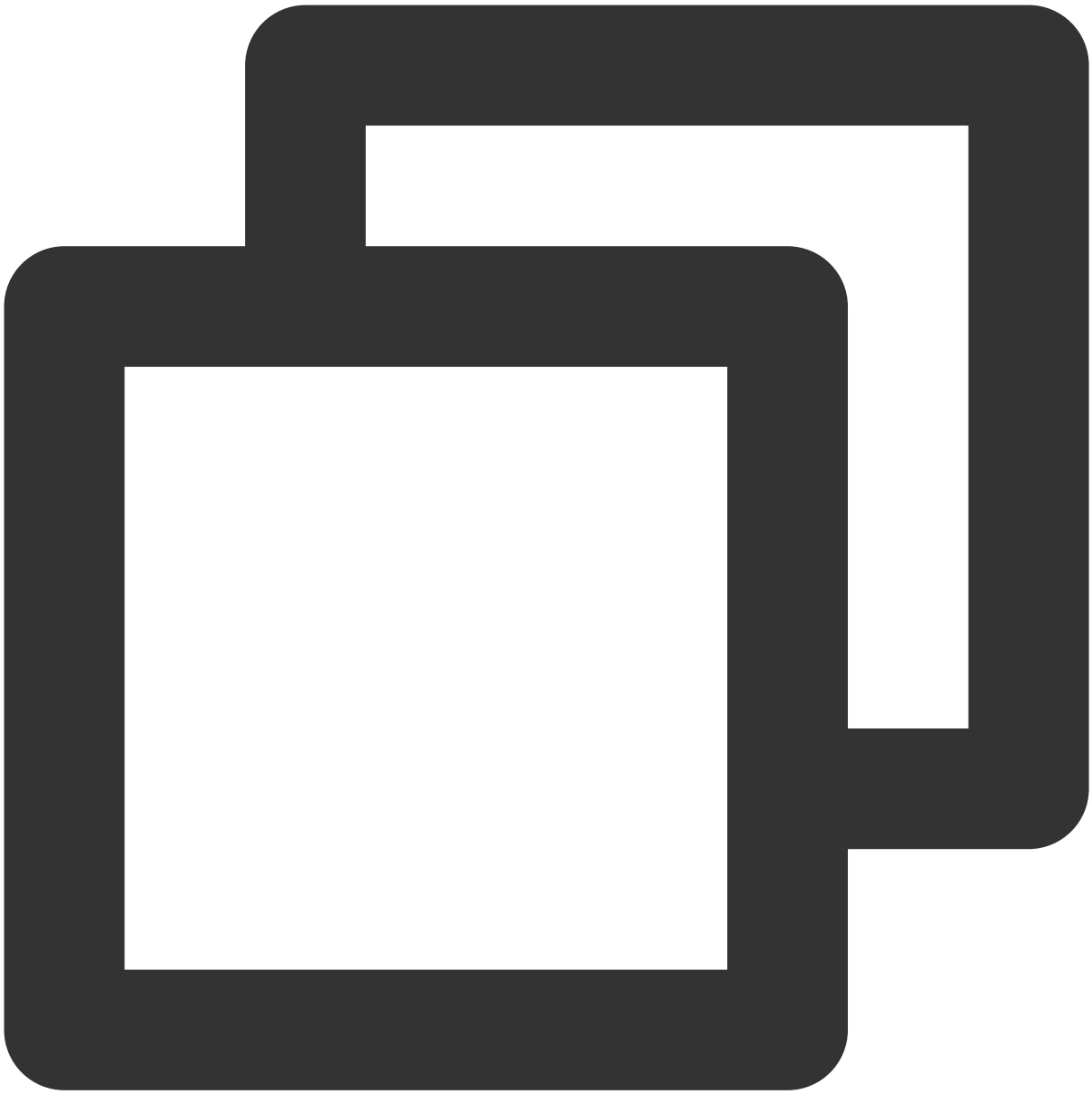
配置美颜效果示例

美颜：配置美白效果



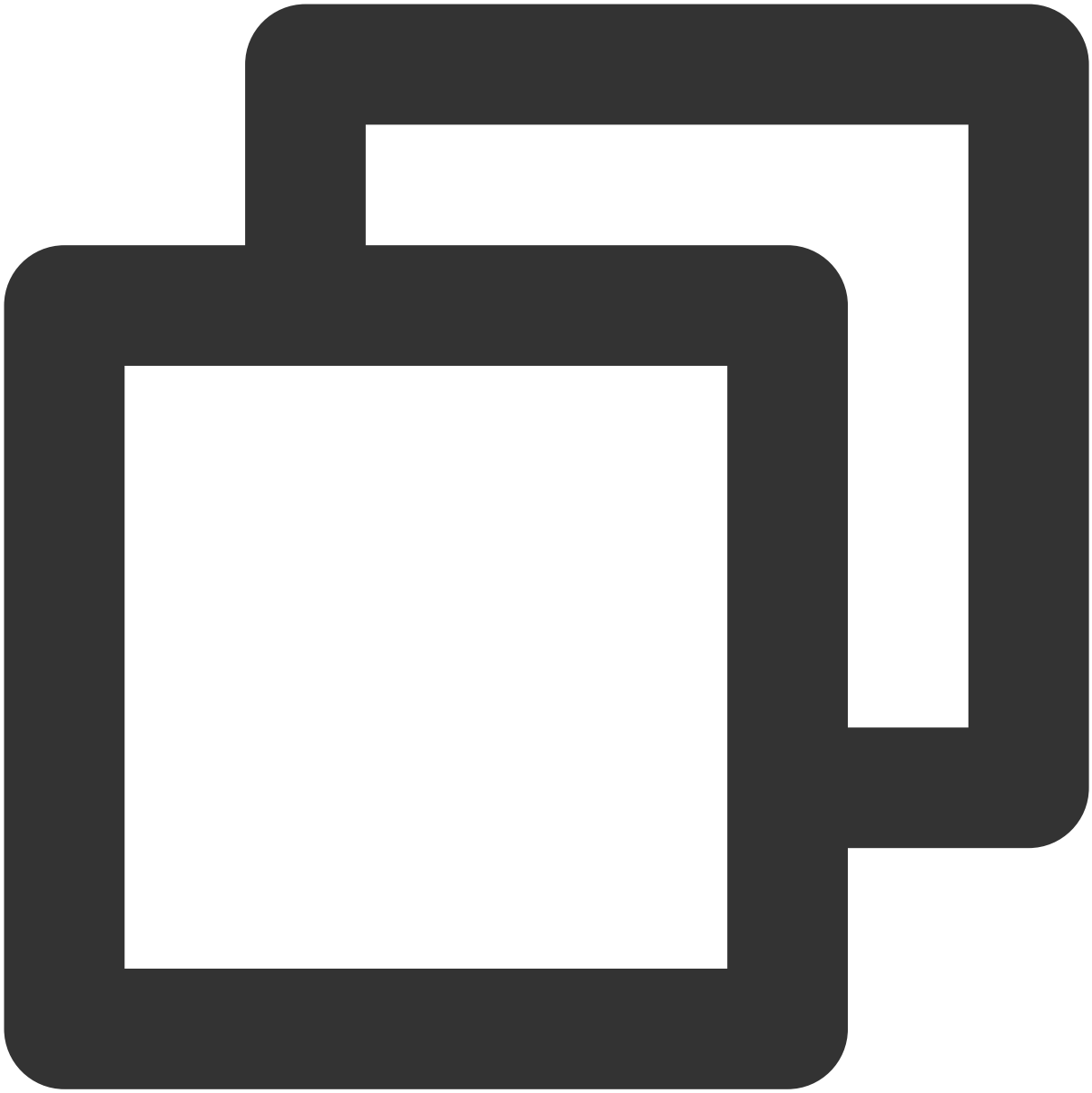
```
NSString *propertyType = @"beauty";          //配置美颜的效果类型，这里以美颜为例
NSString *propertyName = @"beauty.whiten"; //配置美颜的名称，这里以美白为例
NSString *propertyValue = @"60";             //配置美白的效果数值
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

滤镜：配置心动效果



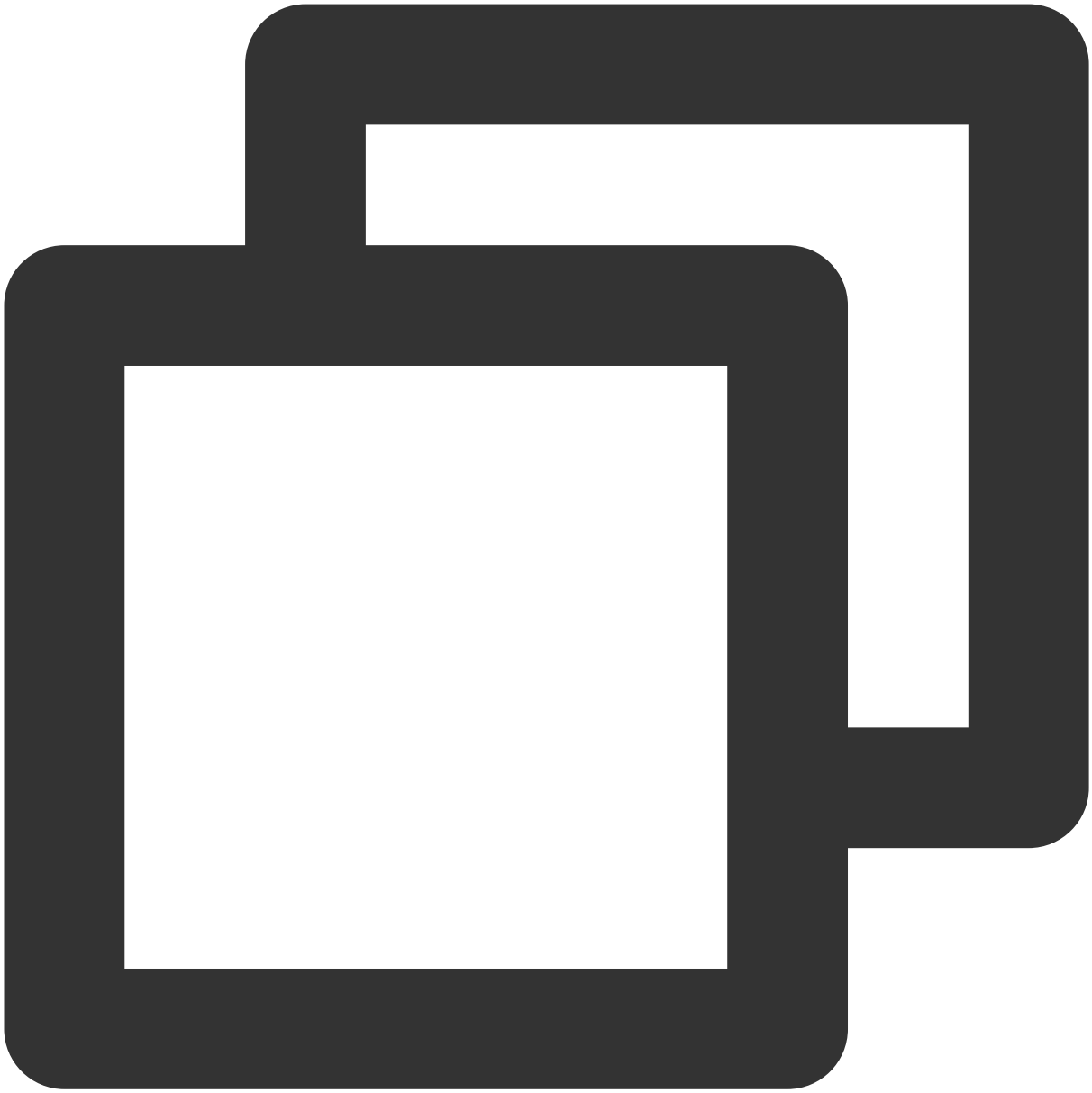
```
NSString *propertyType = @"lut";          //配置美颜的效果类型，这里以滤镜为例
NSString *propertyName = [@"lut.bundle/" stringByAppendingPathComponent:@"xindong_l
NSString *propertyValue = @"60";            //配置滤镜的效果数值
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

美体：配置长腿效果



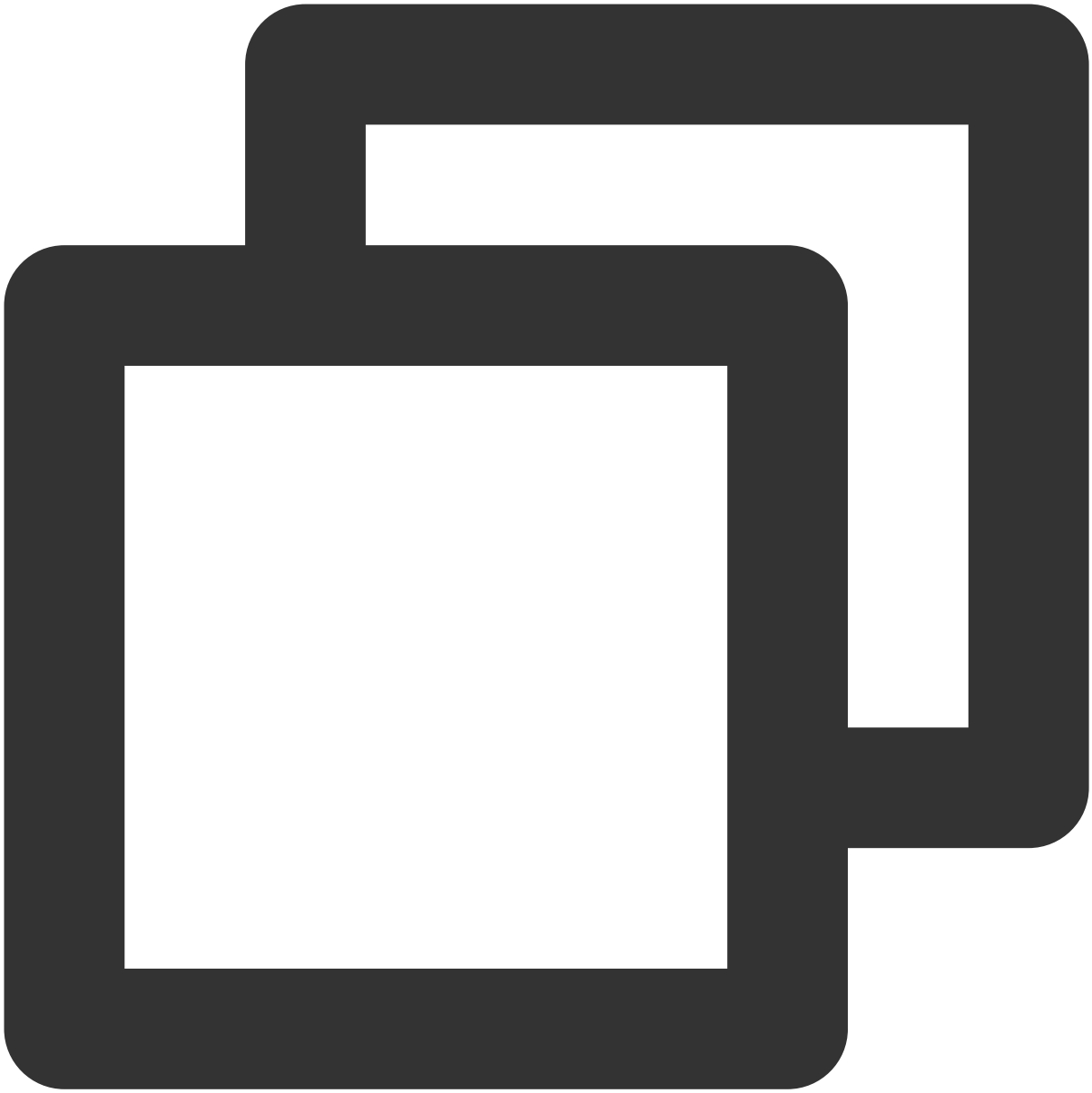
```
NSString *propertyType = @"body";          //配置美颜的效果类型，这里以美体为例
NSString *propertyName = @"body.legStretch"; //配置美颜的名称，这里以长腿为例
NSString *propertyValue = @"60";             //配置长腿的效果数值
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

动效：配置2D动效的可爱涂鸦效果



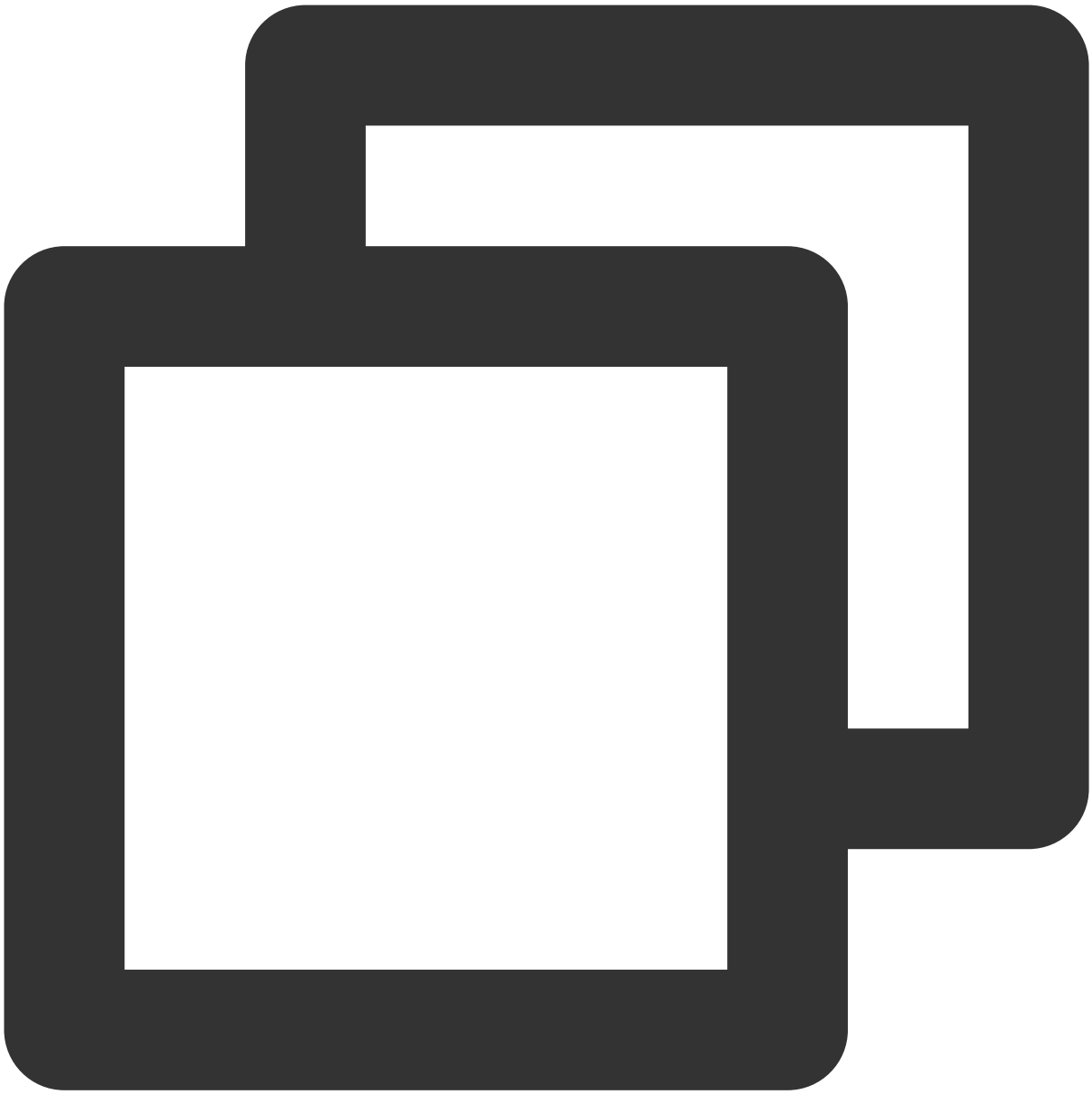
```
NSString *motion2dResPath = [[NSBundle mainBundle] pathForResource:@"2dMotionRes"
NSString *propertyType = @"motion";          //配置美颜的效果类型，这里以动效为例
NSString *propertyName = @"video_keaituya"; //配置美颜的名称，这里以2D动效的可爱涂鸦为例
NSString *propertyValue = motion2dResPath;   //配置动效的路径
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData
```

美妆：配置女团妆效果



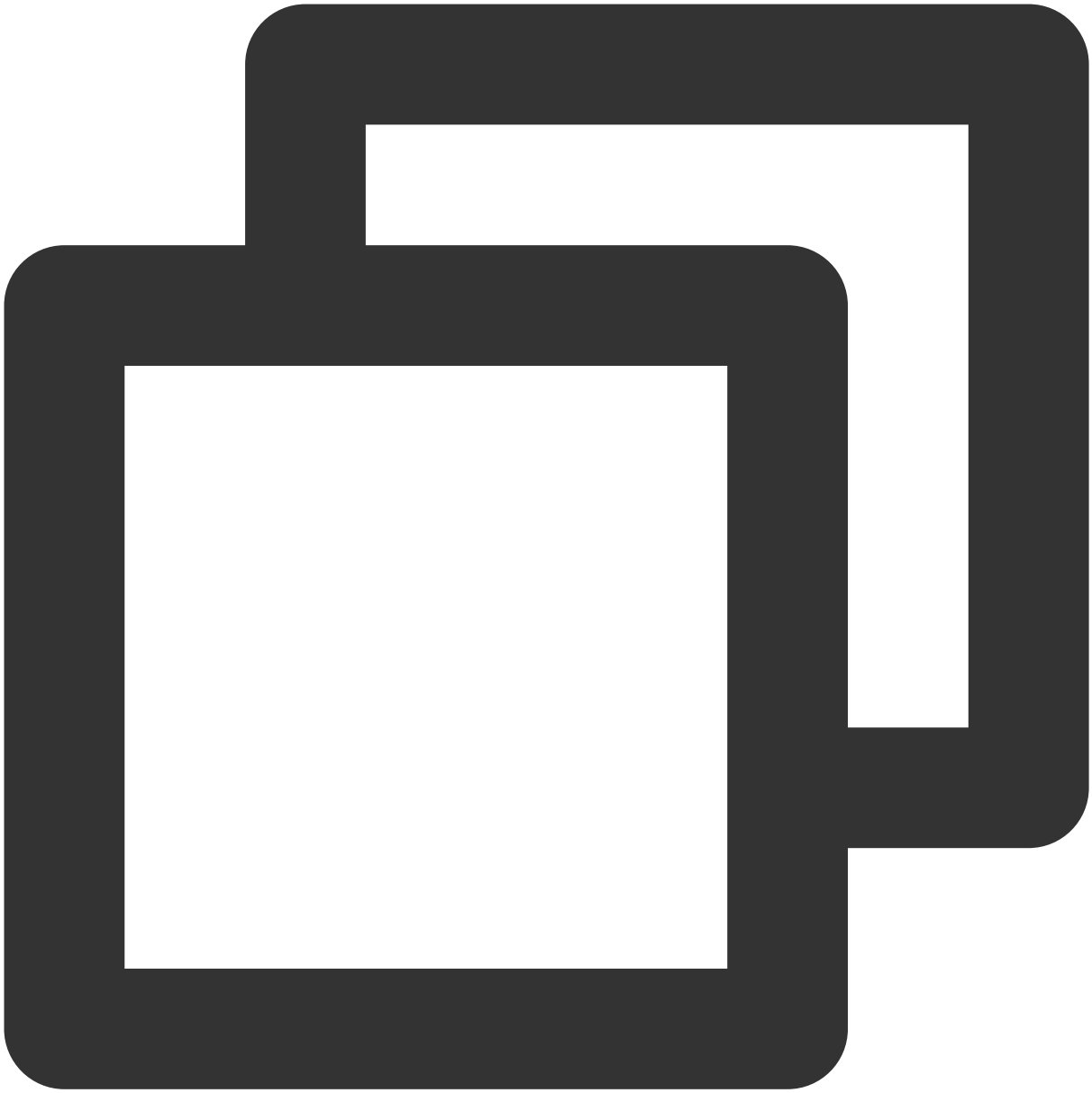
```
NSString *motionMakeupResPath = [[NSBundle mainBundle] pathForResource:@"makeupMoti
NSString *propertyType = @"motion";          //配置美颜的效果类型，这里以美妆为例
NSString *propertyName = @"video_nvtuanzhuang"; //配置美颜的名称，这里以女团妆为例
NSString *propertyValue = motionMakeupResPath;  //配置动效的路径
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
//下面是要配置美妆的数值（上面的动效只需要调用一次，下面的配置美妆数值可以多次调用）
 NSString *propertyTypeMakeup = @"custom";          //配置美颜的效果类型，这里以美妆为例
 NSString *propertyNameMakeup = @"makeup.strength"; //配置美颜的名称，这里以女团妆为例
 NSString *propertyValueMakeup = @"60";              //配置美妆的效果数值
 [self.xmagicApi configPropertyWithType:propertyTypeMakeup withName:propertyNameMak
```

分割：配置背景模糊（强效果）



```
NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotion
NSString *propertyType = @"motion";          //配置美颜的效果类型，这里以分割为例
NSString *propertyName = @"video_segmentation_blur_75"; //配置美颜的名称，这里以背景模糊
NSString *propertyValue = motionSegResPath;  //配置动效的路径
NSDictionary *dic = @{@"bgName":@"BgSegmentation.bg.png", @"bgType":@0, @"timeOffse
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

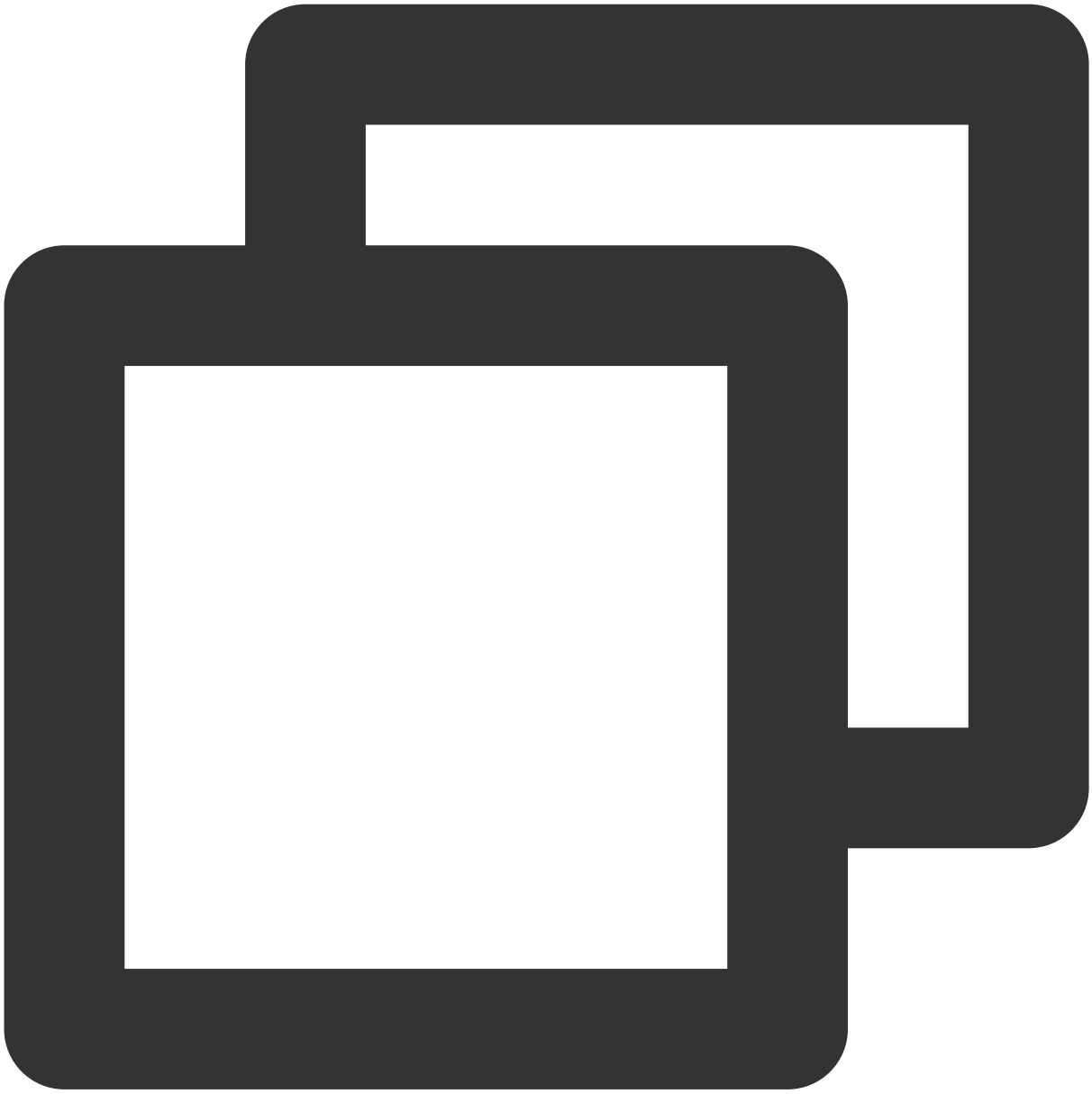
自定义背景：



```
NSString *motionSegResPath = [[NSBundle mainBundle] pathForResource:@"segmentMotion
NSString *propertyType = @"motion";          //配置美颜的效果类型，这里以分割为例
NSString *propertyName = @"video_empty_segmentation"; //配置美颜的名称，这里以自定义背景
NSString *propertyValue = motionSegResPath;  //配置动效的路径
NSString *imagePath = @"/var/mobile/Containers/Data/Application/06B00BBC-9060-450F-
int bgType = 0;//自定义背景的类型。 0表示图片，1表示视频
int timeOffset = 0；//时长。图片背景时，为0；视频背景时为视频的时长
NSDictionary *dic = @{@"bgName":imagePath, @"bgType":@(bgType), @"timeOffset": @(ti
[self.xmagicApi configPropertyWithType:propertyType withName:propertyName withData:
```

setEffect（3.5.0.2新增）

配置美颜各种效果，具体使用示例请参考 美颜参数说明。



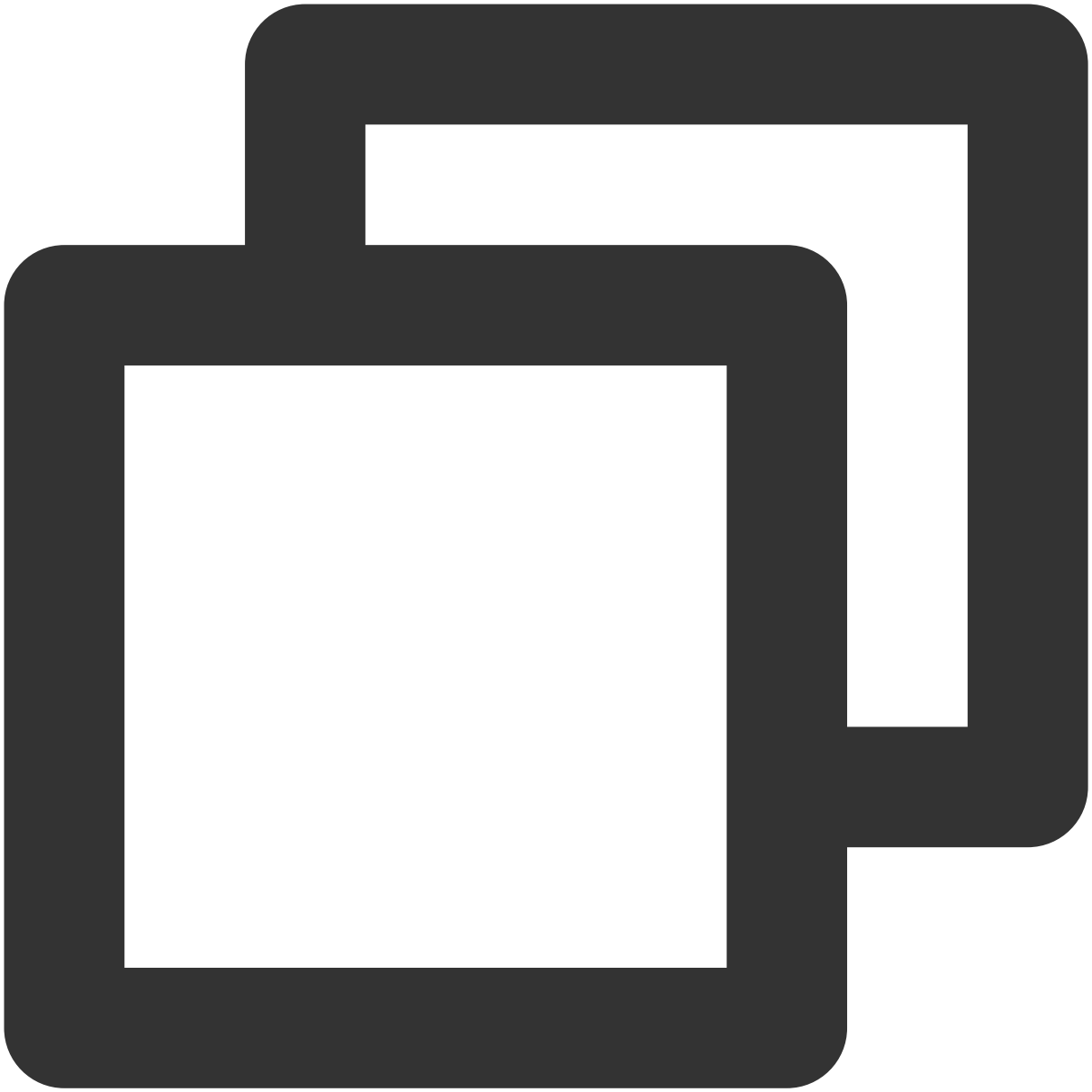
```
- (void)setEffect:(NSString * _Nullable)effectName
      effectValue:(int)effectValue
     resourcePath:(NSString * _Nullable)resourcePath
        extraInfo:(NSDictionary * _Nullable)extraInfo;
```

参数

参数	含义
effectName	效果类型
effectValue	效果数值
resourcePath	素材路径
extraInfo	预留扩展, 附加额外配置

emitBlurStrengthEvent

设置后处理模糊强度（作用于所有模糊组件）



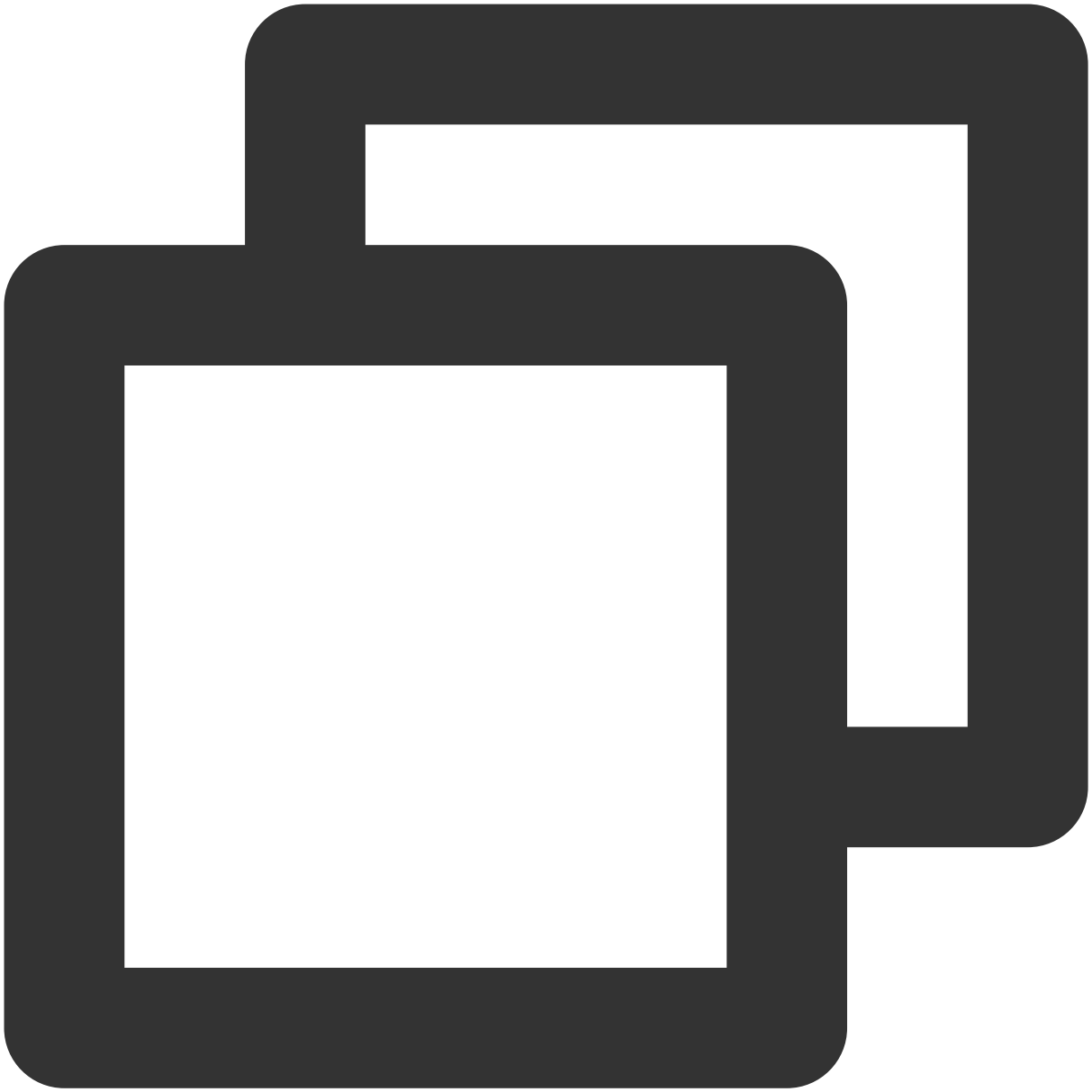
```
- (void)emitBlurStrengthEvent:(int)strength;
```

参数

参数	含义
strength	效果数值

setRenderSize

设置 renderSize



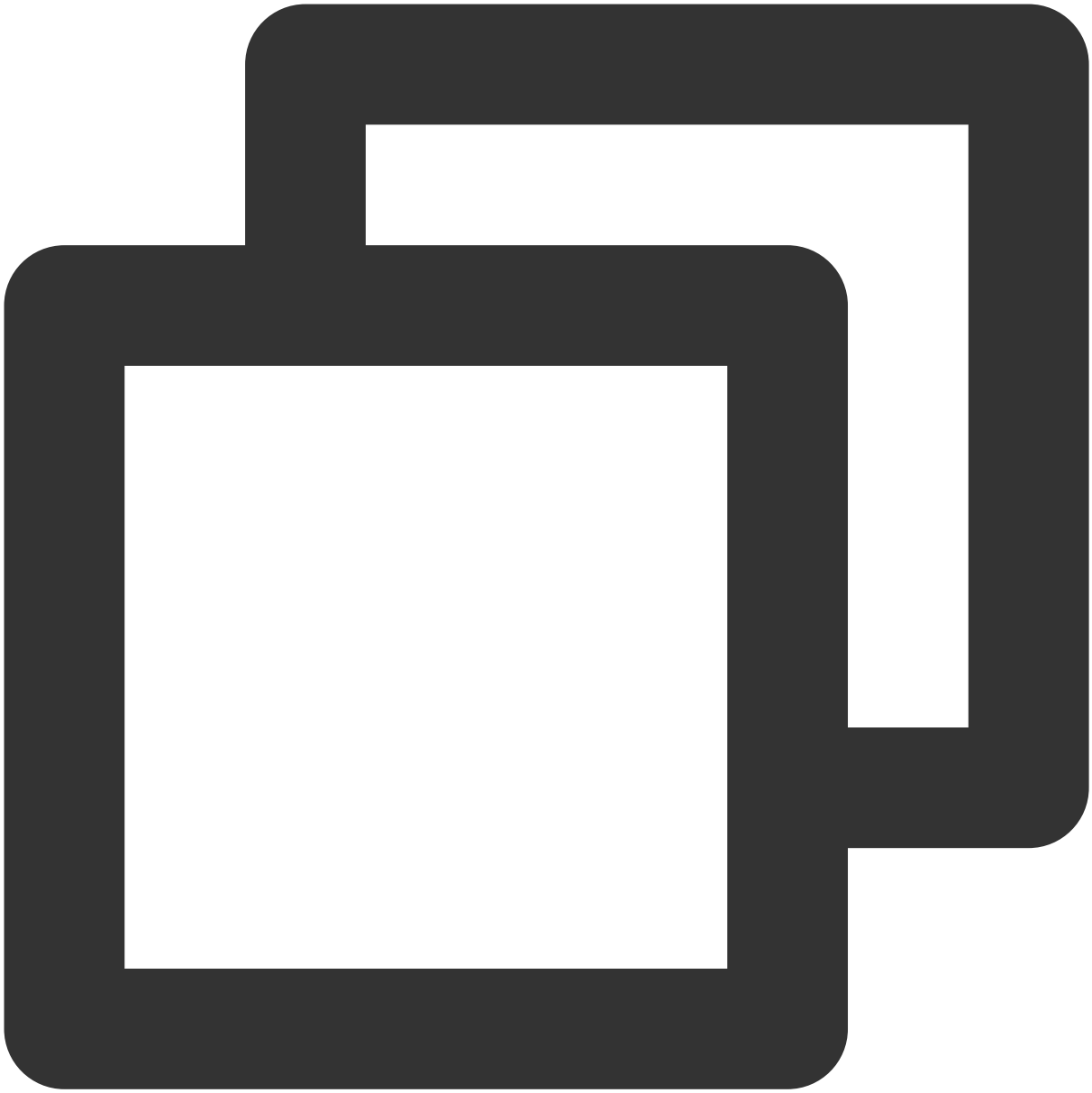
```
- (void)setRenderSize:(CGSize)size;
```

参数

参数	含义
size	渲染尺寸

deinit

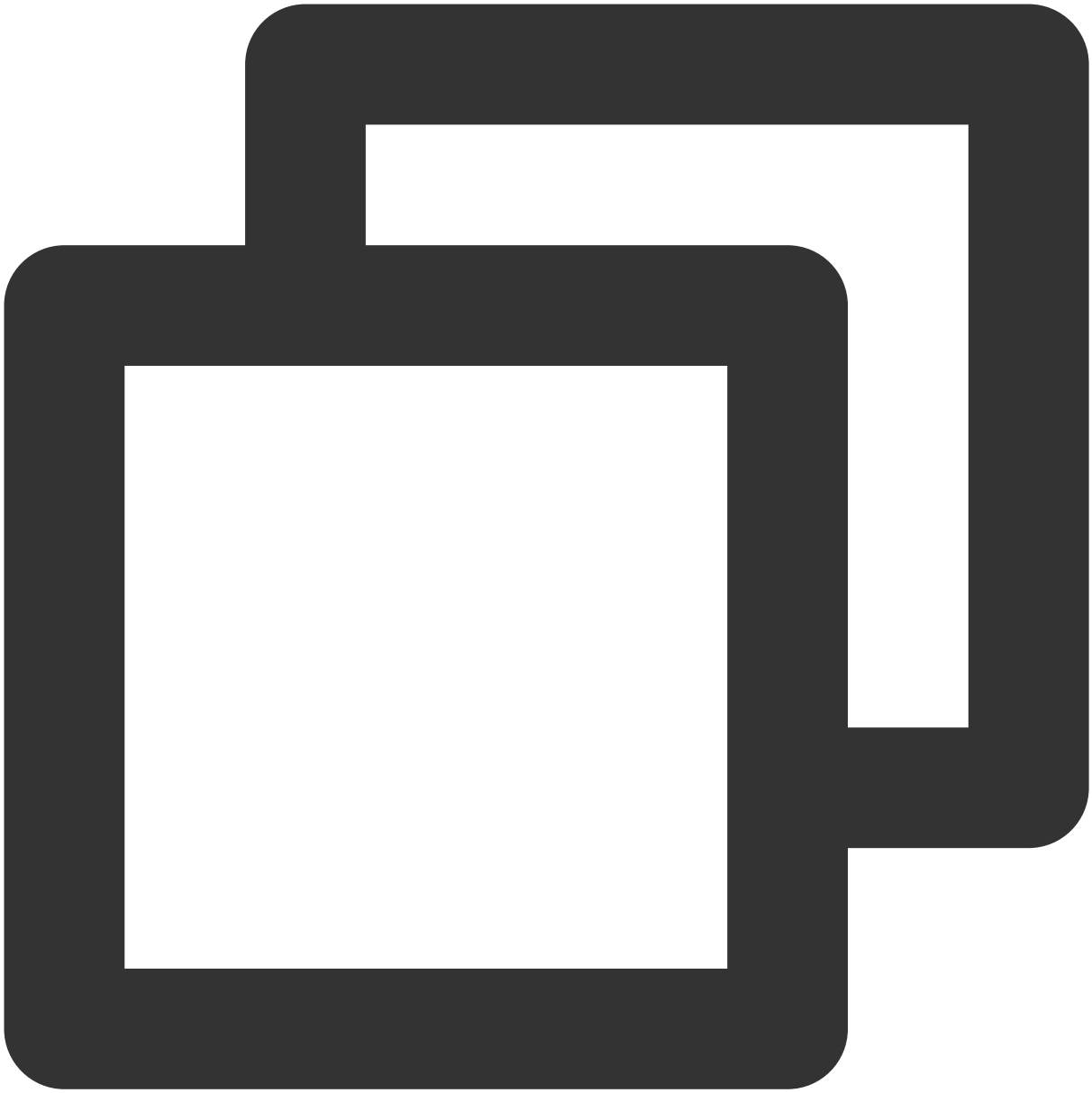
资源释放接口



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

process

处理数据接口，输入的数据格式有 `YTImagePixelData`、`YTTextureData`、`YTImageRawData`、`YTUIImageData` ，输出对应的数据格式。其中 `YTImagePixelData` 中的像素格式为 `RGBA`，`YTTextureData` 中的纹理格式为 `OpenGL 2D` 。



```
/// @brief 处理输入4选1      Process input 4 choose 1
@interface YTProcessInput : NSObject
/// 相机数据对象      camera data object
@property (nonatomic, strong) YTImagePixelData * _Nullable pixelData;
/// 纹理对象      texture object
@property (nonatomic, strong) YTTextureData * _Nullable textureData;
/// 原始数据对象      raw data object
@property (nonatomic, strong) YTImageRawData * _Nullable rawData;
/// UIImage对象      UIImage object
@property (nonatomic, strong) YTUIImageData * _Nullable UIImageData;
/// 输入数据类型      input data type
```

```
@property (nonatomic) enum YTProcessDataType dataType;
@end
/// @brief 处理输出     process output
@interface YTProcessOutput : NSObject
/// 纹理输出对象（一定会有）     Texture output object (always output)
@property (nonatomic, strong) YTTextureData * _Nullable textureData;
/// 相机输出对象（如果输入是相机采集数据）     Camera output object (if the input is camera
@property (nonatomic, strong) YTImagePixelData * _Nullable pixelData;
/// 原始输出对象（如果输入是原始数据）     raw output object (if input is raw data)
@property (nonatomic, strong) YTImageRawData * _Nullable rawData;
/// UIImage输出对象（如果输入是UIImage对象）     UIImage output object (if the input is
@property (nonatomic, strong) YTUIImageData * _Nullable UIImageData;
/// 输出数据类型     output data type
@property (nonatomic) enum YTProcessDataType dataType;
@end

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

参数

参数	含义
input	输入处理数据信息，输入的格式四选一（YTImagePixelData、YTTextureData、YTImageRawData、YTUIImageData）

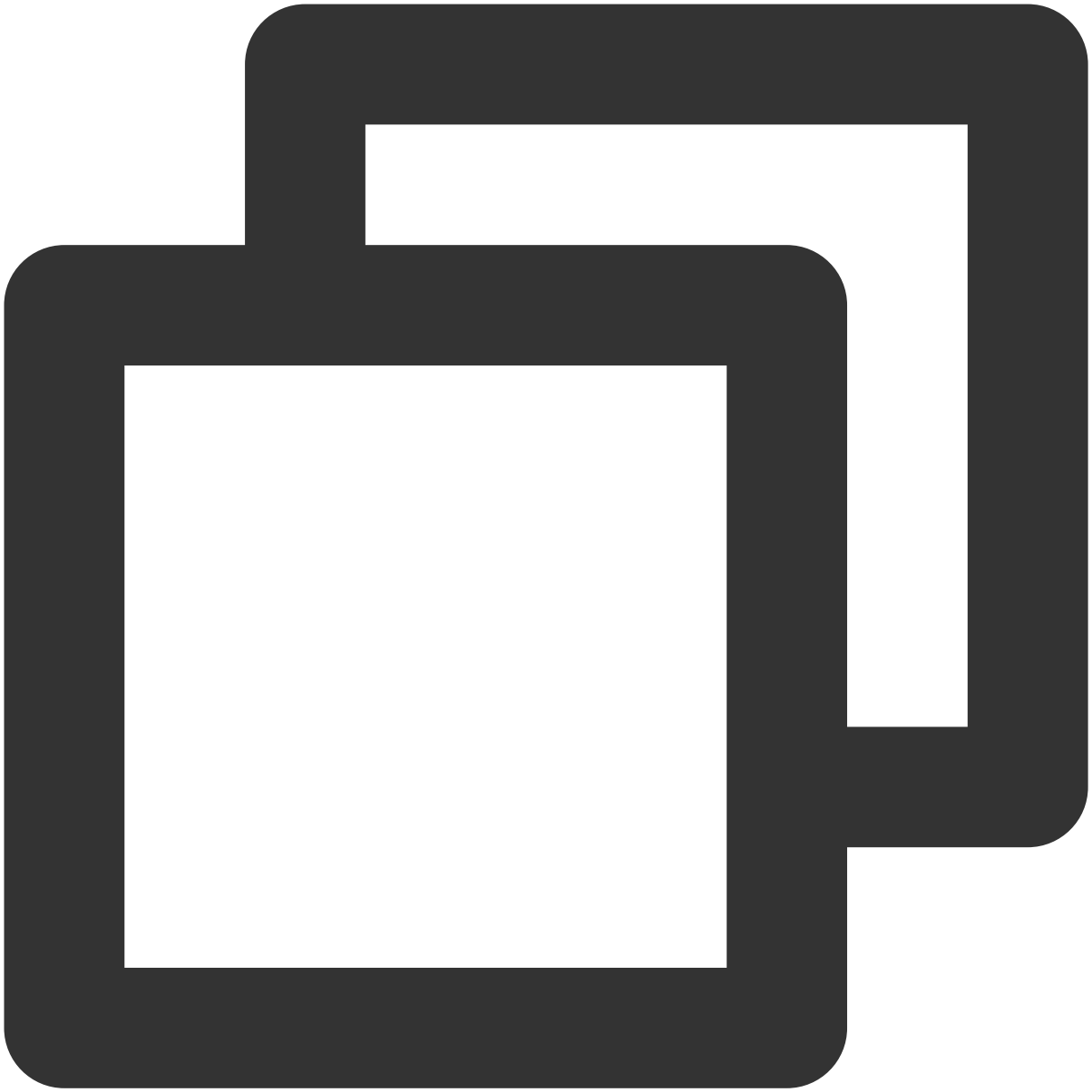
YTProcessInput 输入数据类型和说明

调用 process 接口时，输出数据类型跟输入数据类型一致，YTTextureData 类型总是会输出。

类型	含义
YTImagePixelData	相机数据对象，像素格式为 RGBA
YTTextureData	纹理对象，纹理格式为 OpenGL 2D
YTImageRawData	原始数据对象
YTUIImageData	UIImage 对象

process:withOrigin:withOrientation:

处理数据接口，输入和输出的数据格式跟process一致。withOrigin：设置图像是否上下镜像翻转，withOrientation：设置图像旋转方向。



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

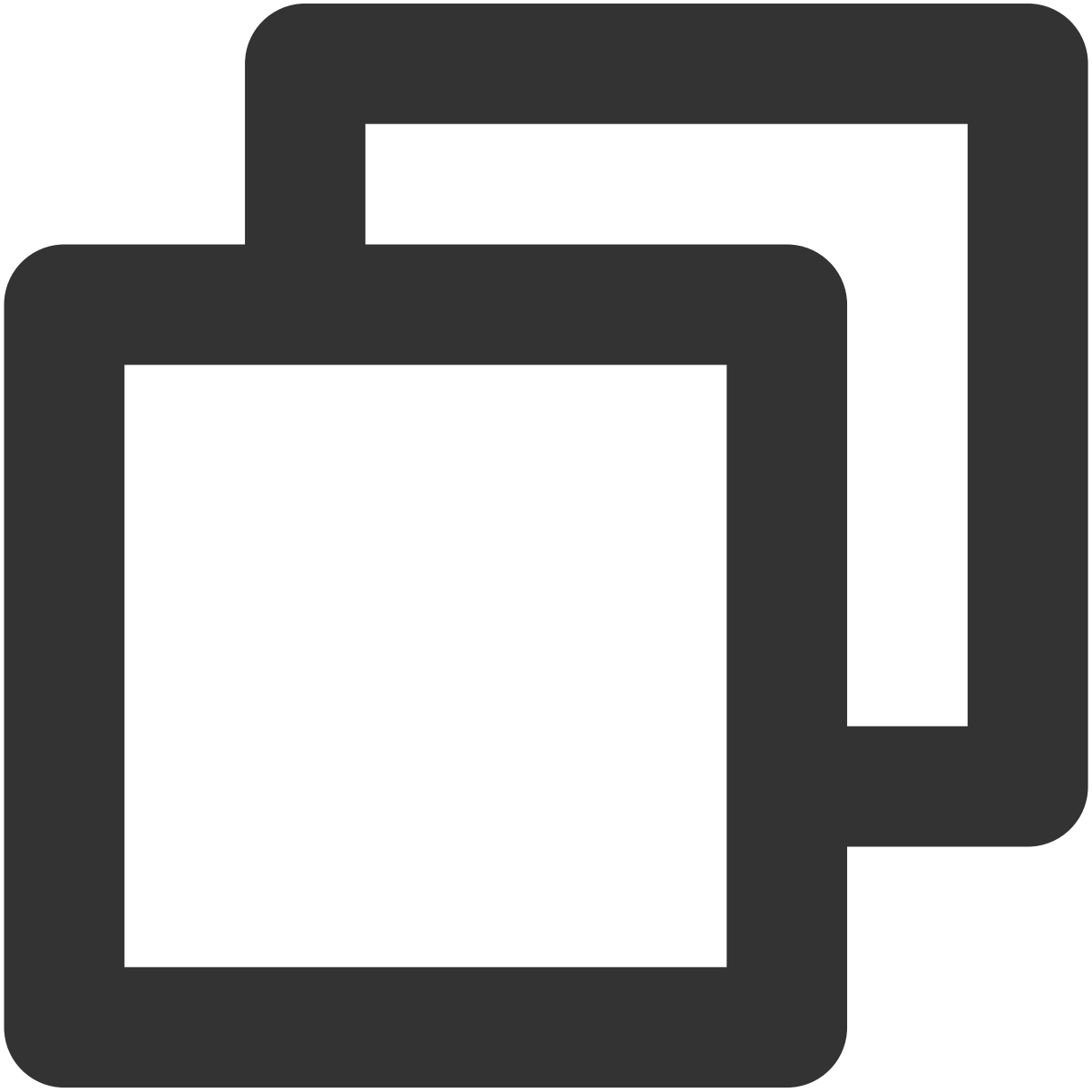
参数

参数	含义
input	输入处理数据信息
withOrigin	枚举值（YtLightImageOriginTopLeft、

	YtLightImageOriginBottomLeft），设置成 YtLightImageOriginBottomLeft 时，图像上下镜像翻转
withOrientation	枚举值：图像旋转角度，设置角度会修改输出的图像角度

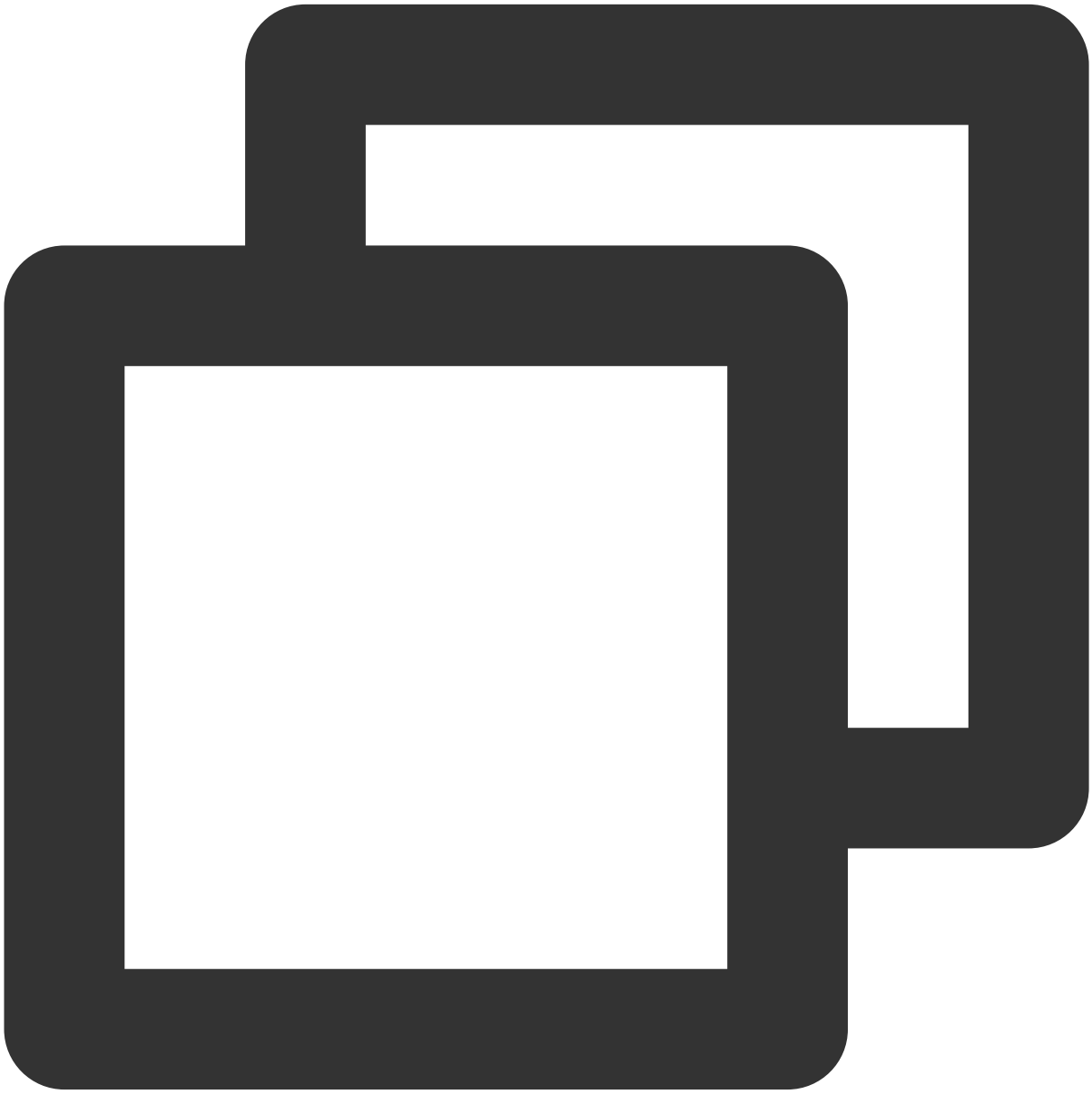
TEImageTransform 工具类

图像处理工具类，输入/输出的数据格式有 CVPixelBufferref、texture id。支持 CVPixelBufferref 数据的 bgra<-->yuv 格式的相互转换、旋转和上下/左右镜像。支持 texture id 格式输入的旋转和上下/左右镜像。



```
/// @param context 如果使用本类OpenGL接口,建议使用本方法初始化,可传[xMgiac getCurrentGlCor
- (instancetype)initWithEAGLContext:(EAGLContext *)context;
```

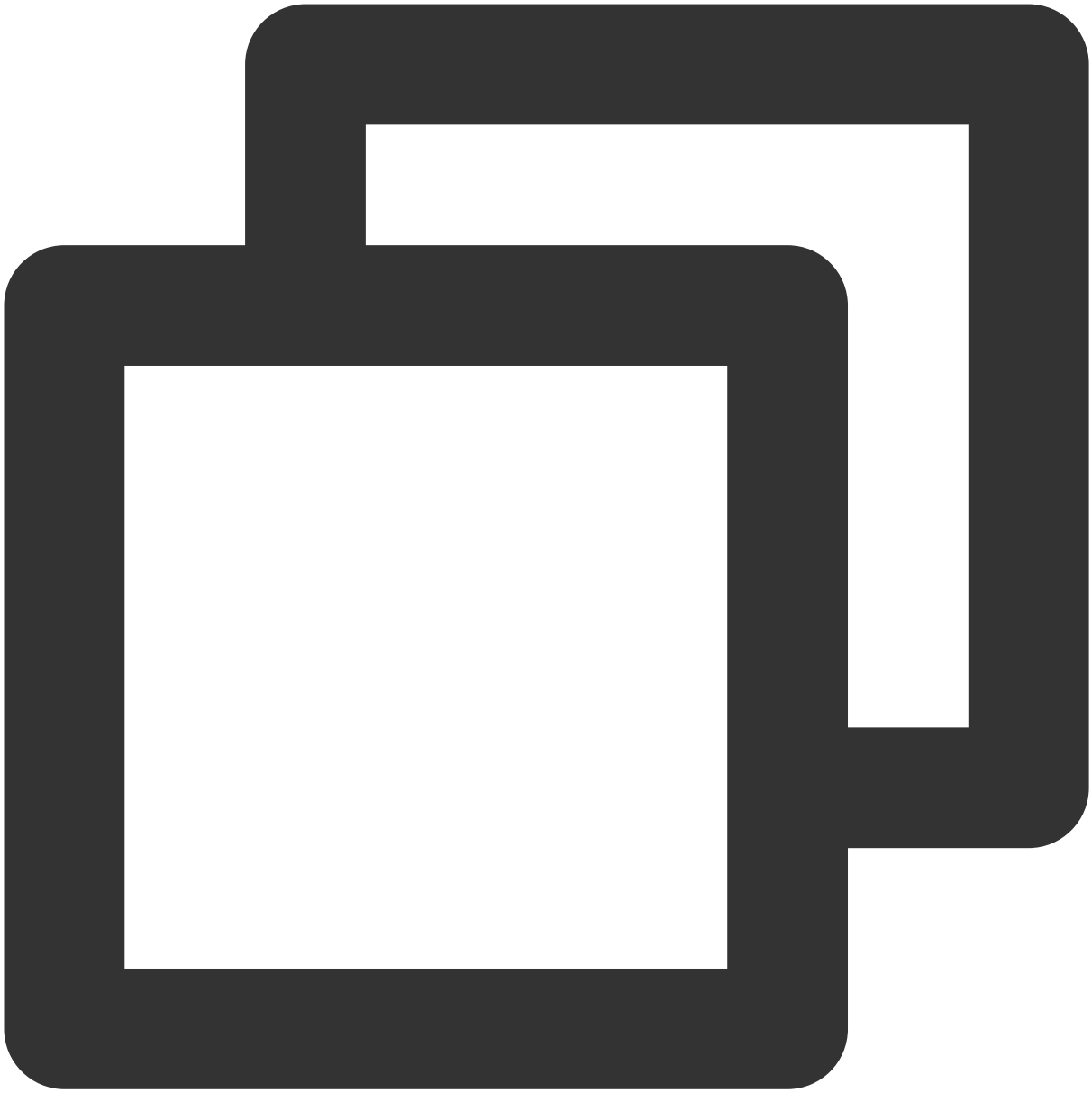
参数	含义
context	使用OpenGLES的上下文环境，可传[xMagic getCurrentGlContext]



```
/// @brief CVPixelBufferRef yuv/rgb相互转换接口，目前只支持TEPixelFormatType内的三种类型转
/// @param pixelBuffer 输入pixelBuffer     input pixelBuffer
```

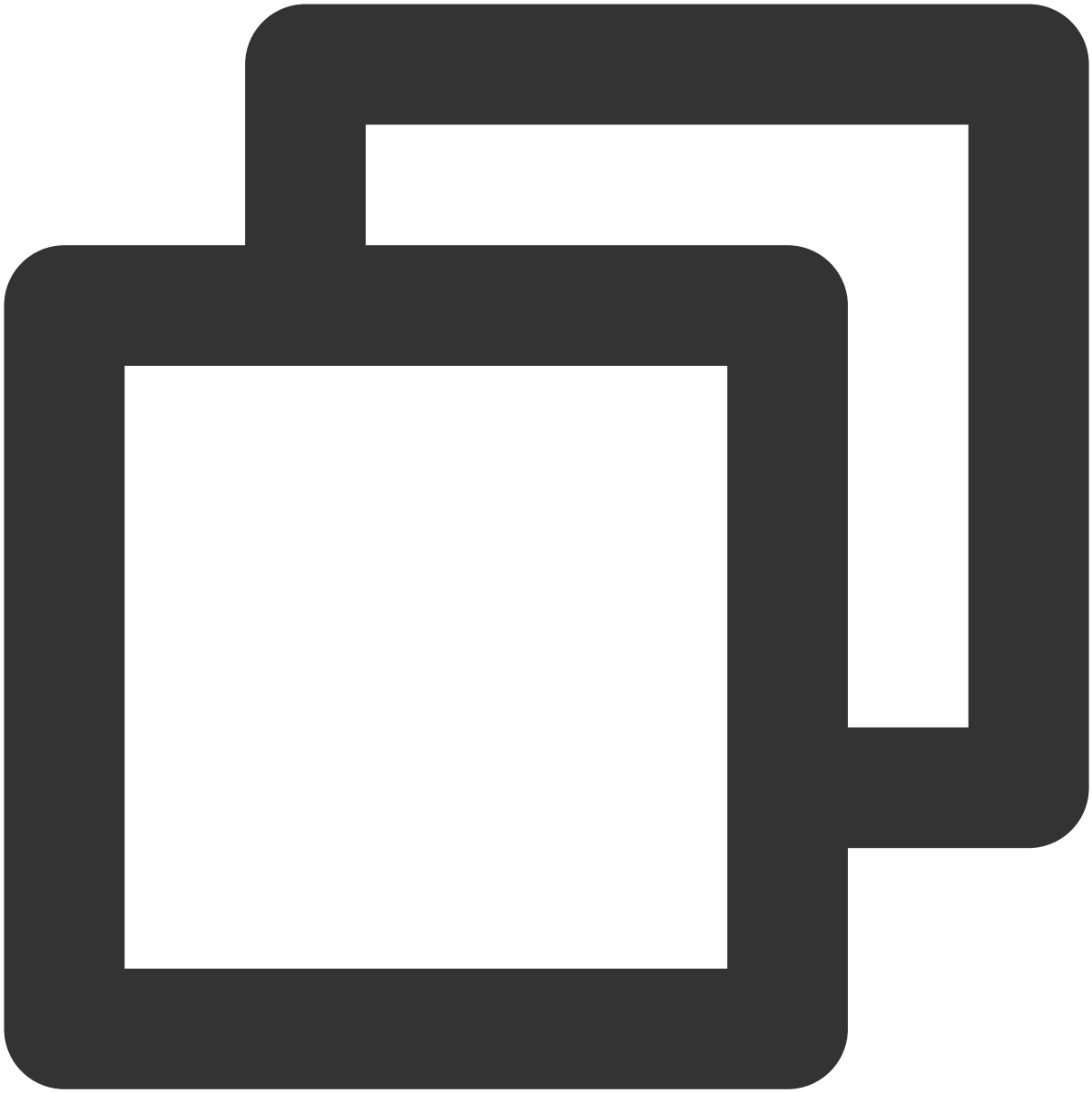
```
/// @param outputFormat 指定输出pixelBuffer的类型    out pixelBuffer format
- (CVPixelBufferRef)transformCVPixelBufferToBuffer:(CVPixelBufferRef)pixelBuffer ou
```

参数	含义
pixelBuffer	输入的 pixelBuffer 数据
outputFormat	输出的 pixelBuffer 格式，支持 BGRA、NV12F(kCVPixelFormatType_420YpCbCr8BiPlanarFullRange)、NV12V(kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange)



```
/// 将yuv/rgb pixelBuffer转换为bgra格式的纹理id
/// @param pixelBuffer 输入pixelBuffer
- (GLuint)transformPixelBufferToBGRATexture:(CVPixelBufferRef)pixelBuffer;
```

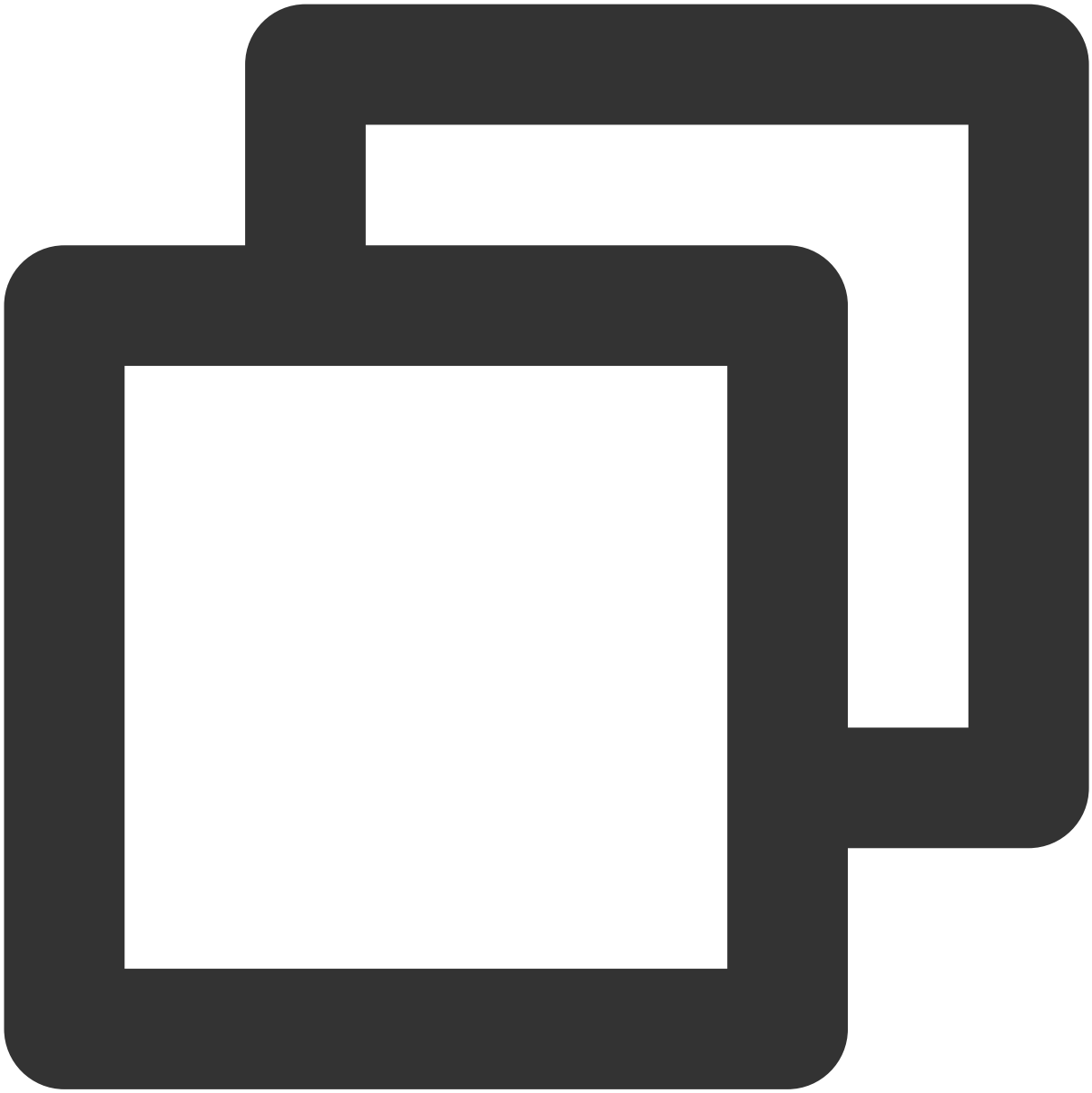
参数	含义
pixelBuffer	输入的 pixelBuffer 数据，支持 BGRA、NV12F(kCVPixelFormatType_420YpCbCr8BiPlanarFullRange)、NV12V(kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange)



```
/// 对CVPixelBufferRef进行旋转和镜像翻转，如果同时传旋转和镜像，处理逻辑为先镜像再旋转
- (CVPixelBufferRef)convertCVPixelBuffer:(CVPixelBufferRef)pixelBuffer rotaion:(YtL
```

参数	含义
pixelBuffer	输入的 pixelBuffer 数据
rotation	逆时针旋转角度，支持0度、90度、180度、270度。

flipType	镜像类型，水平镜像或者垂直镜像。如果同时传旋转和镜像，处理逻辑为先镜像再旋转



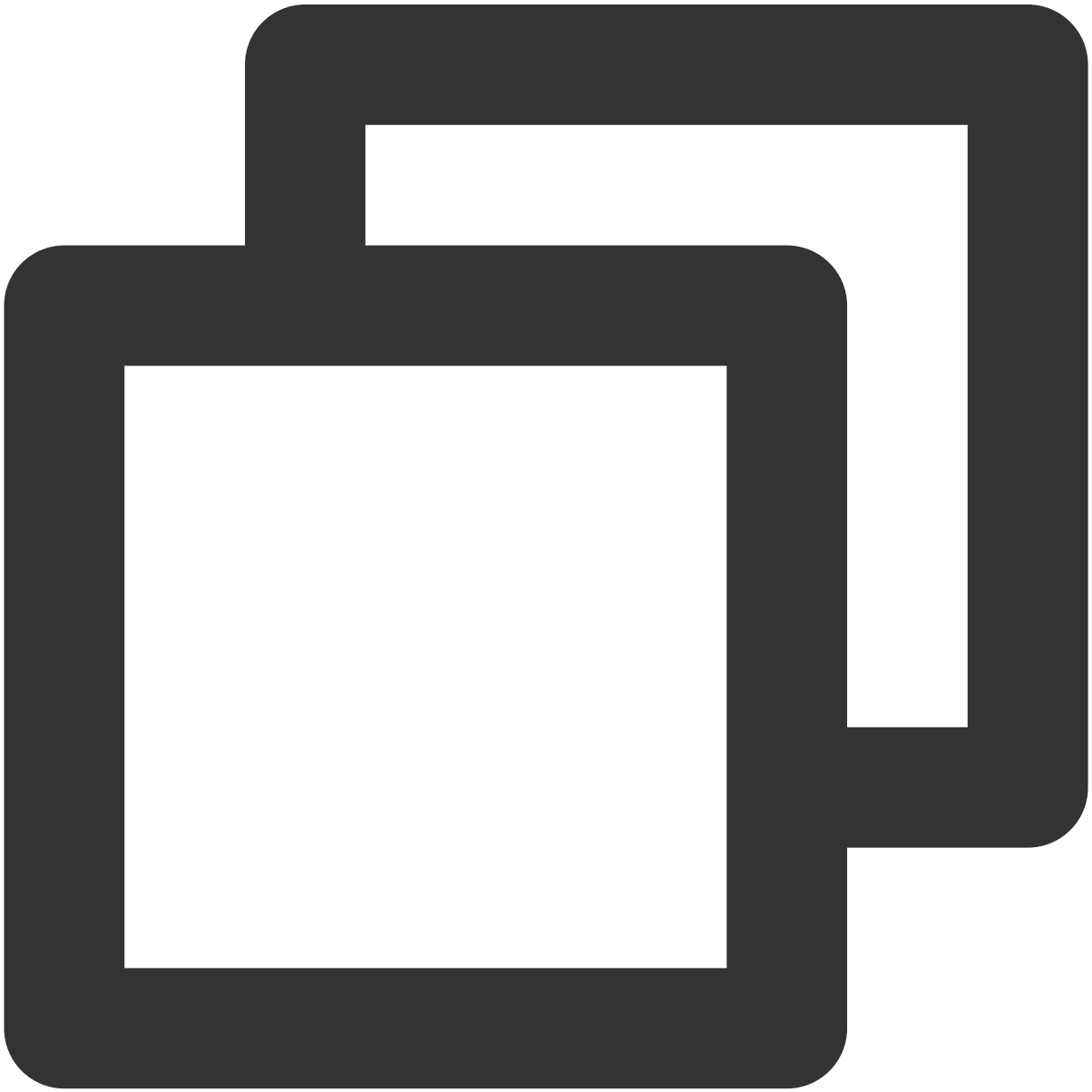
```
/// 对纹理Id进行旋转/镜像翻转,如果同时传旋转和镜像，处理逻辑为先镜像再旋转

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

参数	含义
srcId	输入的纹理 ID

width	纹理的宽度
height	纹理的高度
rotation	逆时针旋转角度，支持0度、90度、180度、270度。
flipType	镜像类型，水平镜像或者垂直镜像。如果同时传旋转和镜像，处理逻辑为先镜像再旋转

processUIImage

处理图片



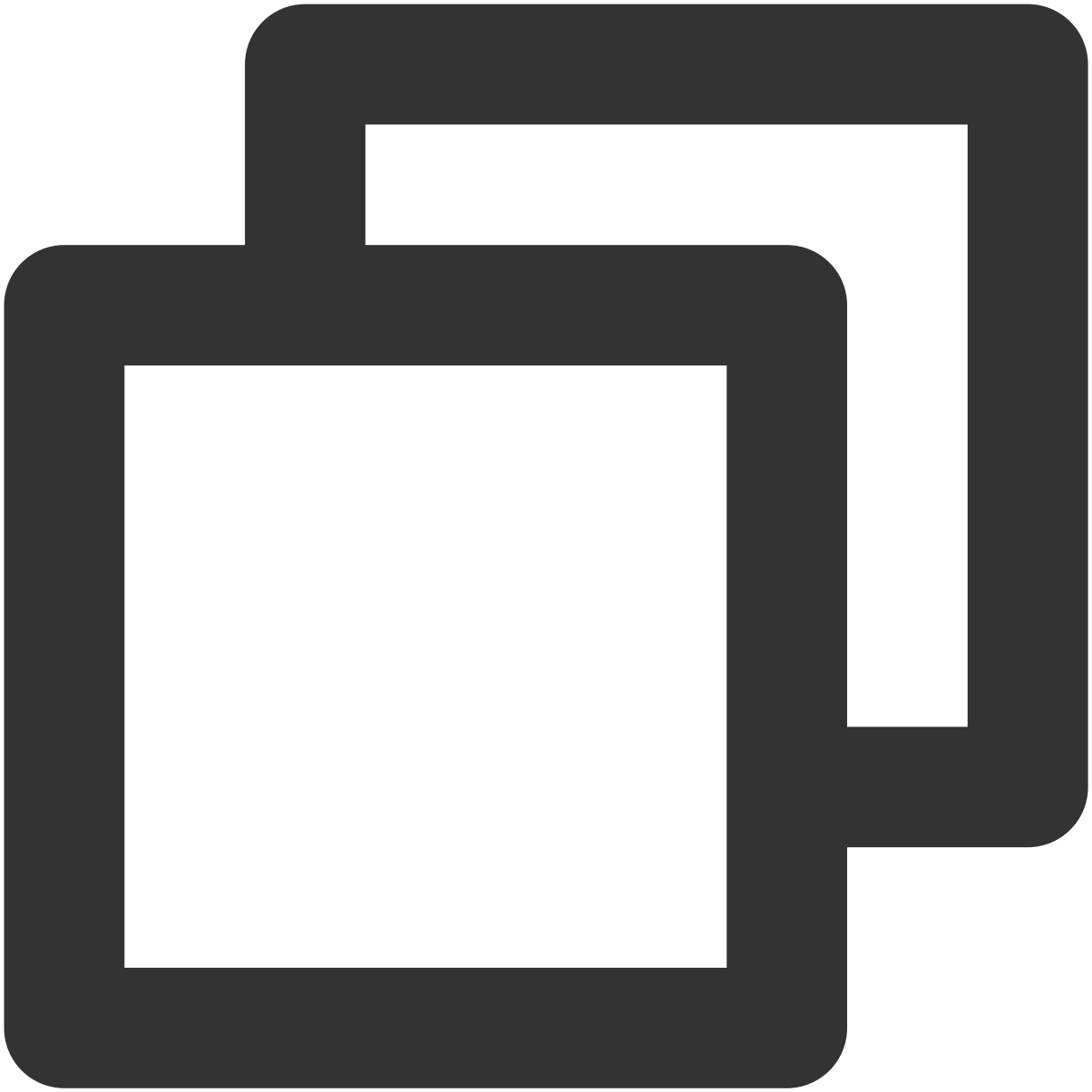
```
- (UIImage* _Nullable)processUIImage:(UIImage* _Nonnull)inputImage needReset:(bool)
```

参数

参数	含义
inputImage	输入图片建议最大尺寸 2160×4096。超过这个尺寸的图片人脸识别效果不佳或无法识别到人脸，同时容易引起 OOM 问题，建议把大图缩小后再传入
needReset	以下场景中 needReset 需设置 为 true： 切换图片 首次使用分割 首次使用动效 首次使用美妆

getConfigPropertyWithName

获取美颜参数配置信息



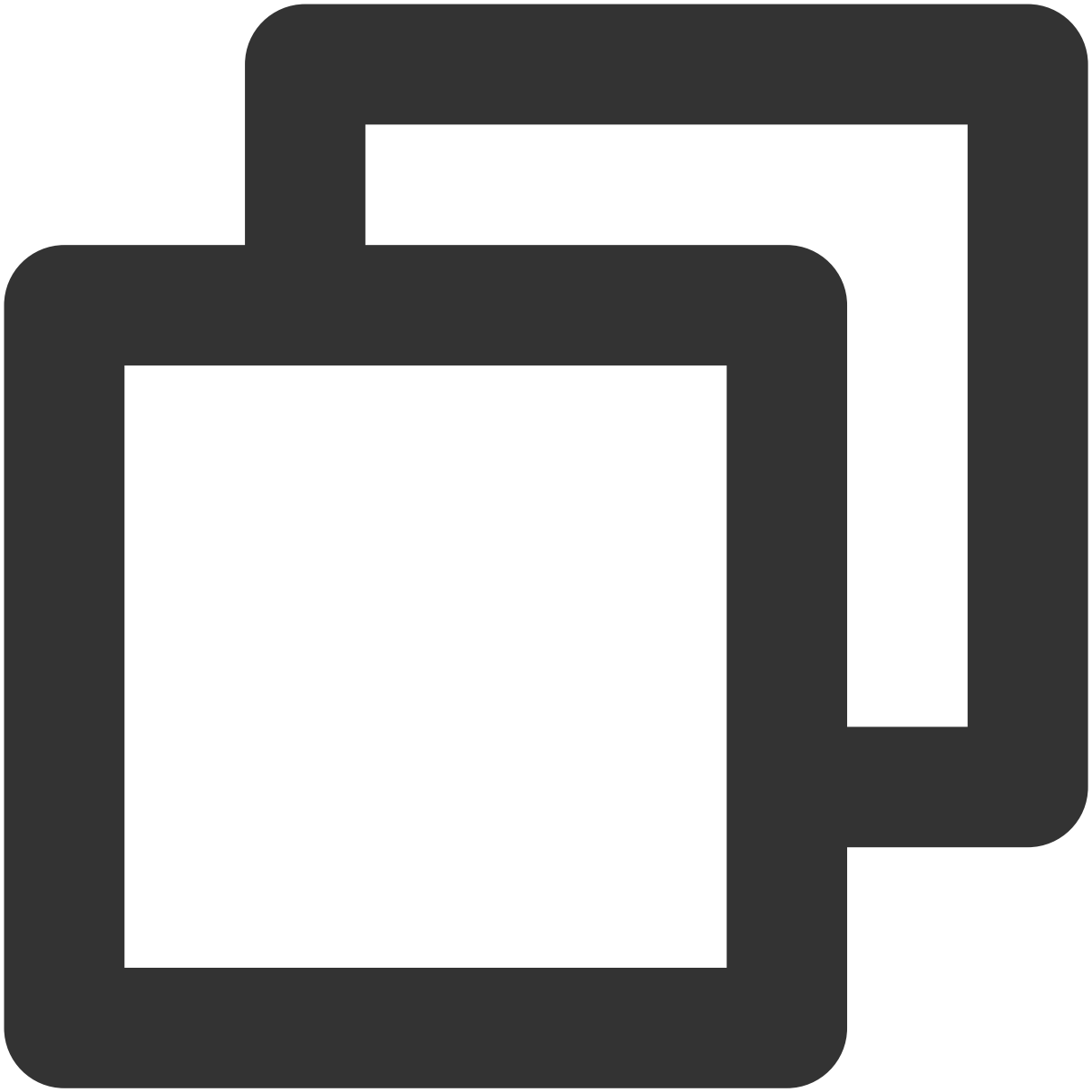
```
- (YTBeautyPropertyInfo * _Nullable)getConfigPropertyWithName:(NSString *_Nonnull)p
```

参数

参数	含义
propertyName	配置名称

registerLoggerListener

日志注册接口



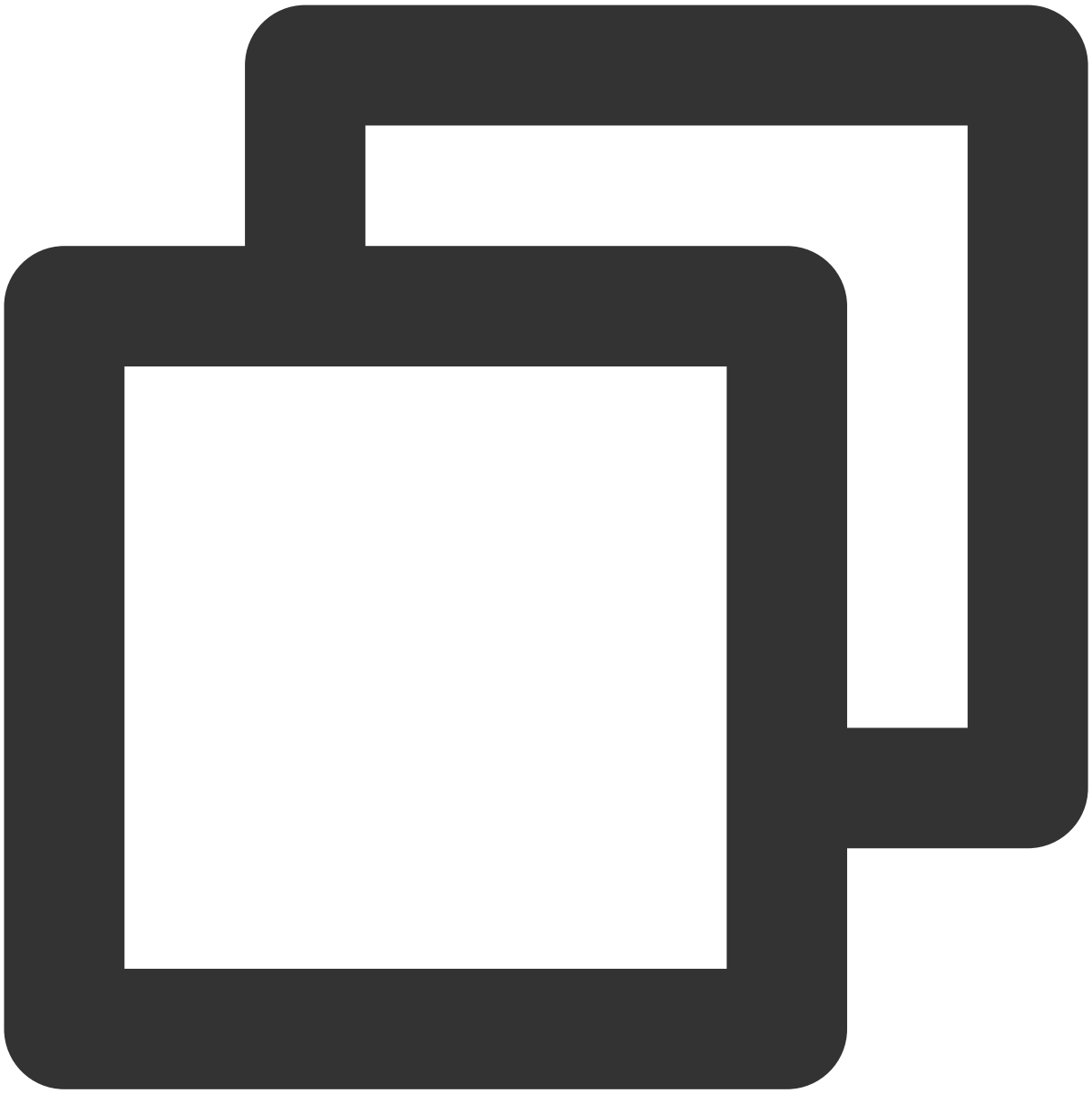
```
- (void)registerLoggerListener:(id<YTSDKLogListener> _Nullable)listener withDefault
```

参数

参数	含义
listener	日志回调接口
level	日志输出 level，默认 ERROR

registerSDKEventListener

SDK 事件监听接口



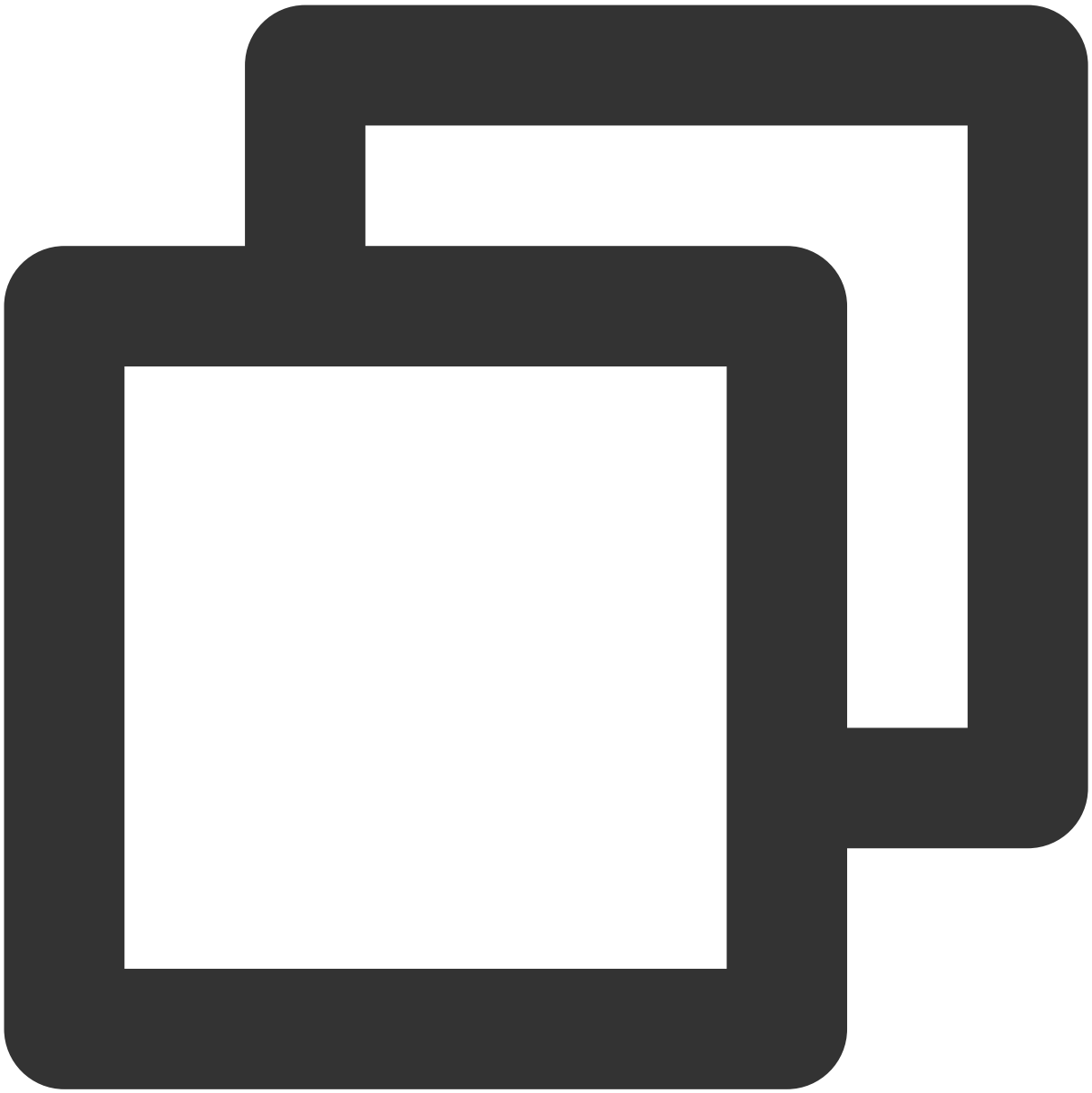
```
- (void)registerSDKEventListener:(id<YTSDKEventListener> _Nullable)listener;
```

参数

参数	含义
listener	事件监听器回调，主要分为 AI 事件，Tips 提示事件，Asset 事件

clearListeners

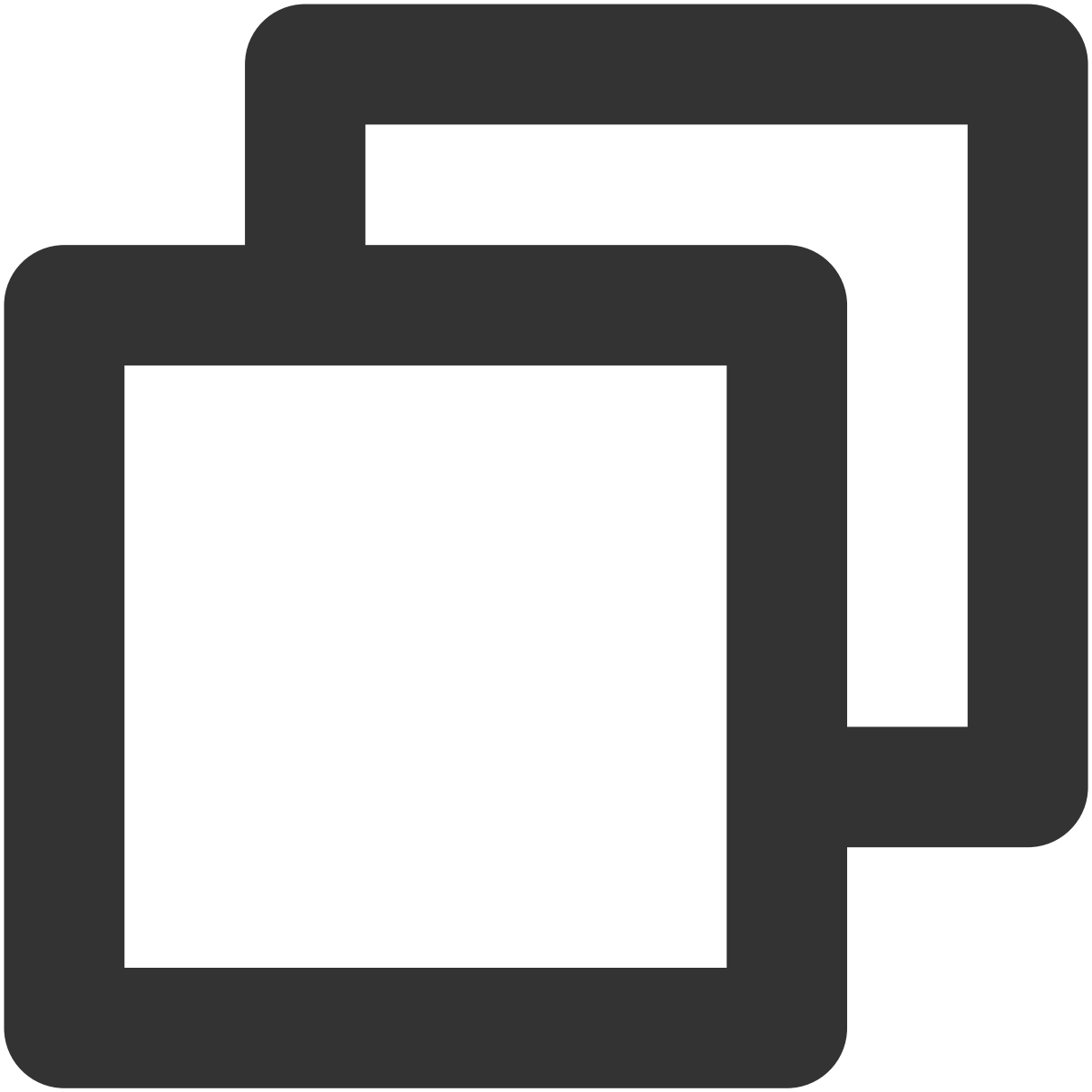
注册回调清理接口



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

getCurrentGlContext

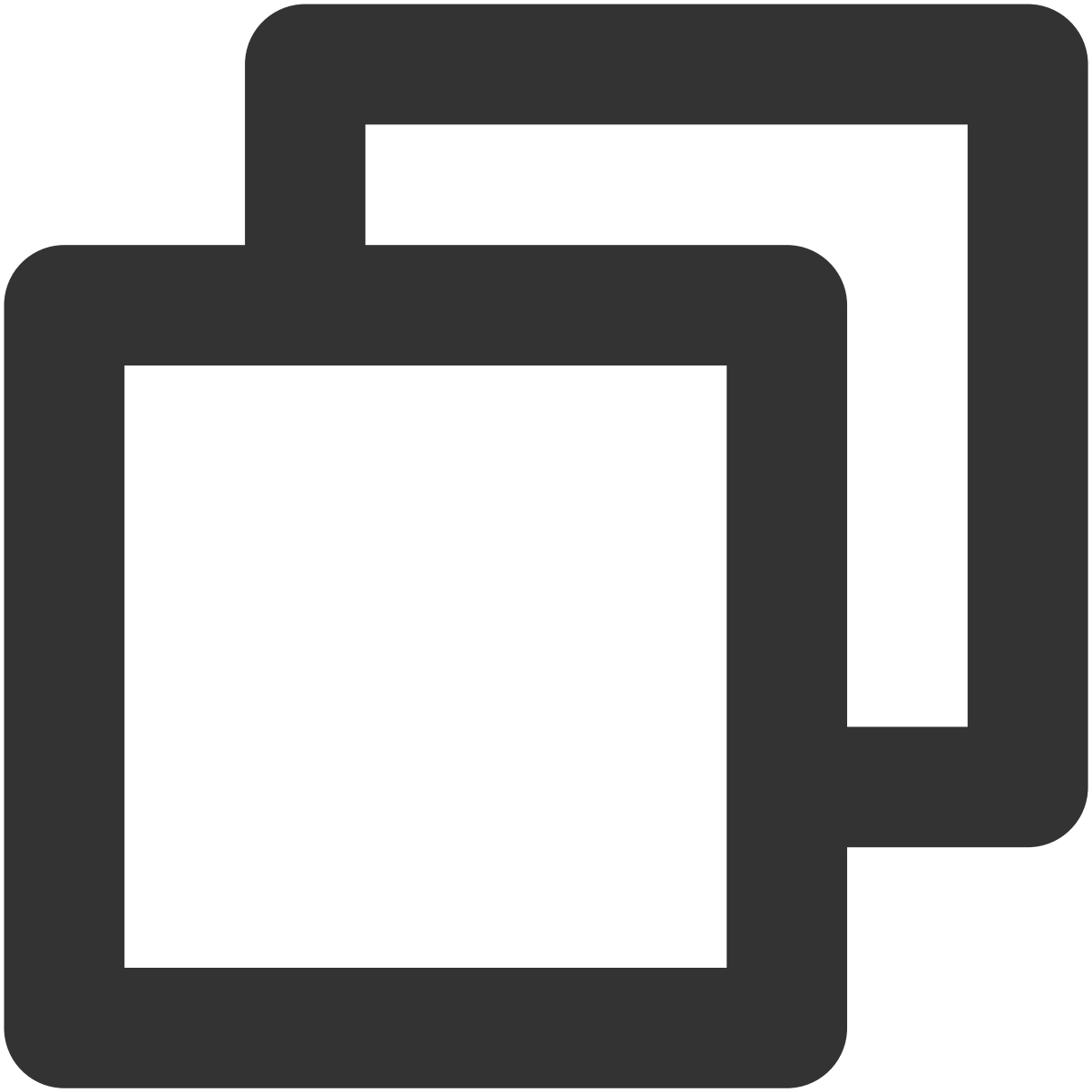
获取当前 GL 上下文接口



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

onPause

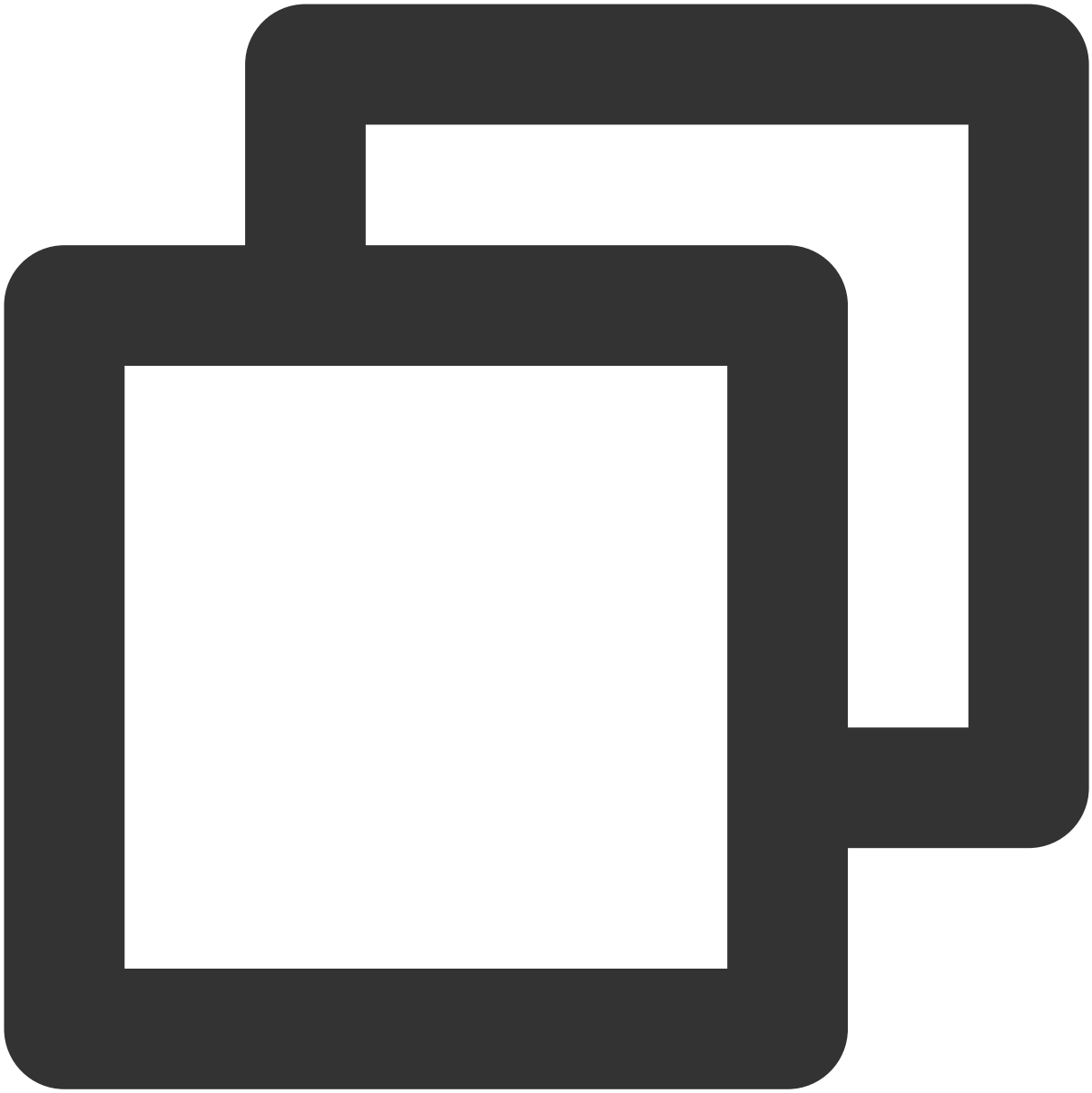
SDK 暂停接口



```
/// @brief APP暂停时候需要调用SDK暂停接口
- (void)onPause;
```

onResume

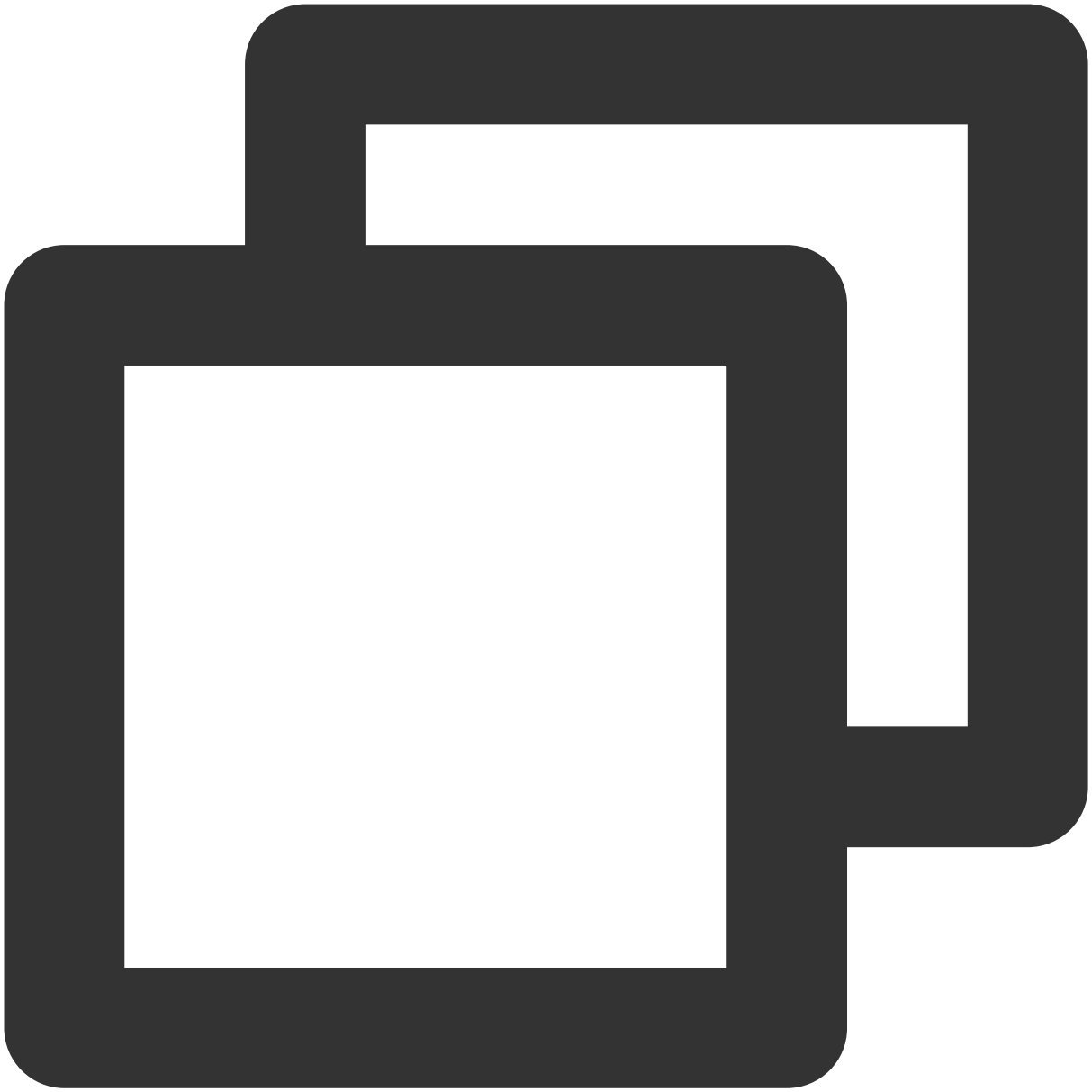
SDK 恢复接口



```
/// @brief APP恢复时候需要调用SDK恢复接口
- (void)onResume;
```

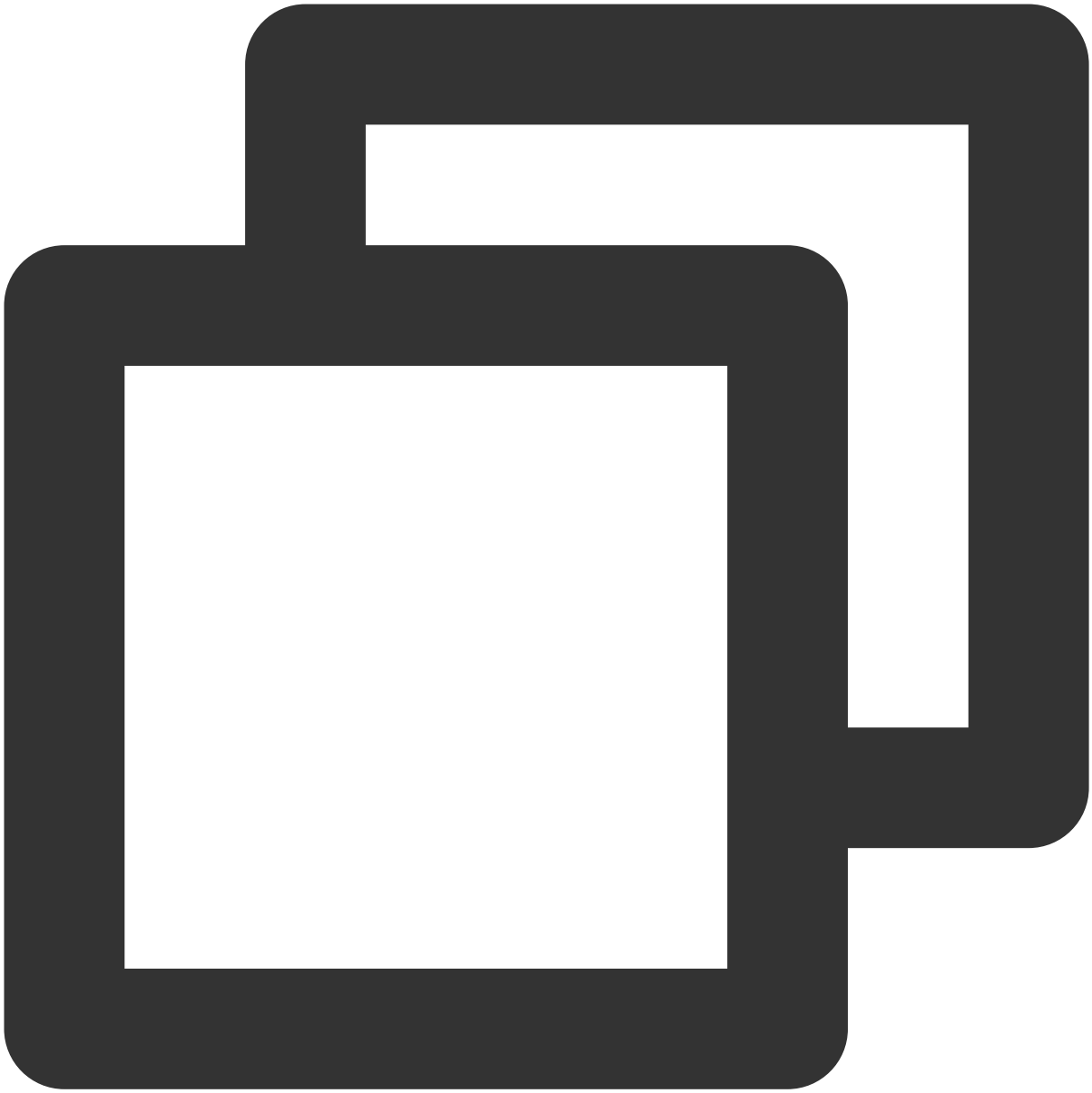
setAudioMute

动效素材使用时是否开启静音（V2.5.0新增）



```
/// @brief 设置静音
- (void)setAudioMute:(BOOL)isMute;
```

enableEnhancedMode



```
/// @brief 开启美颜增强模式
- (void)enableEnhancedMode;
```

开启美颜增强模式（V2.5.1新增）。默认未开启。
未开启时，应用层可以设置的各美颜项的强度范围为0到1或-1到1，如果超出此范围，SDK 会取边界值。例如应用层设置瘦脸为1.2，SDK 判断其超出了最大值1.0，则在内部把瘦脸值修正为1.0。
开启增强模式后，应用层可以设置更大范围的数值。例如想要瘦脸程度更大，则可以把瘦脸值设置为1.2，SDK 会接受并使用1.2这个数值，不会将其修正为1.0。

开启增强模式后，需要应用层自己管理每个美颜项可以设置的最大值，让用户在此范围内调整数值。我们提供了一份参考值，您可以根据产品需求自由调整，但不建议超出我们的推荐值，否则美颜效果可能变差。参考值见下：

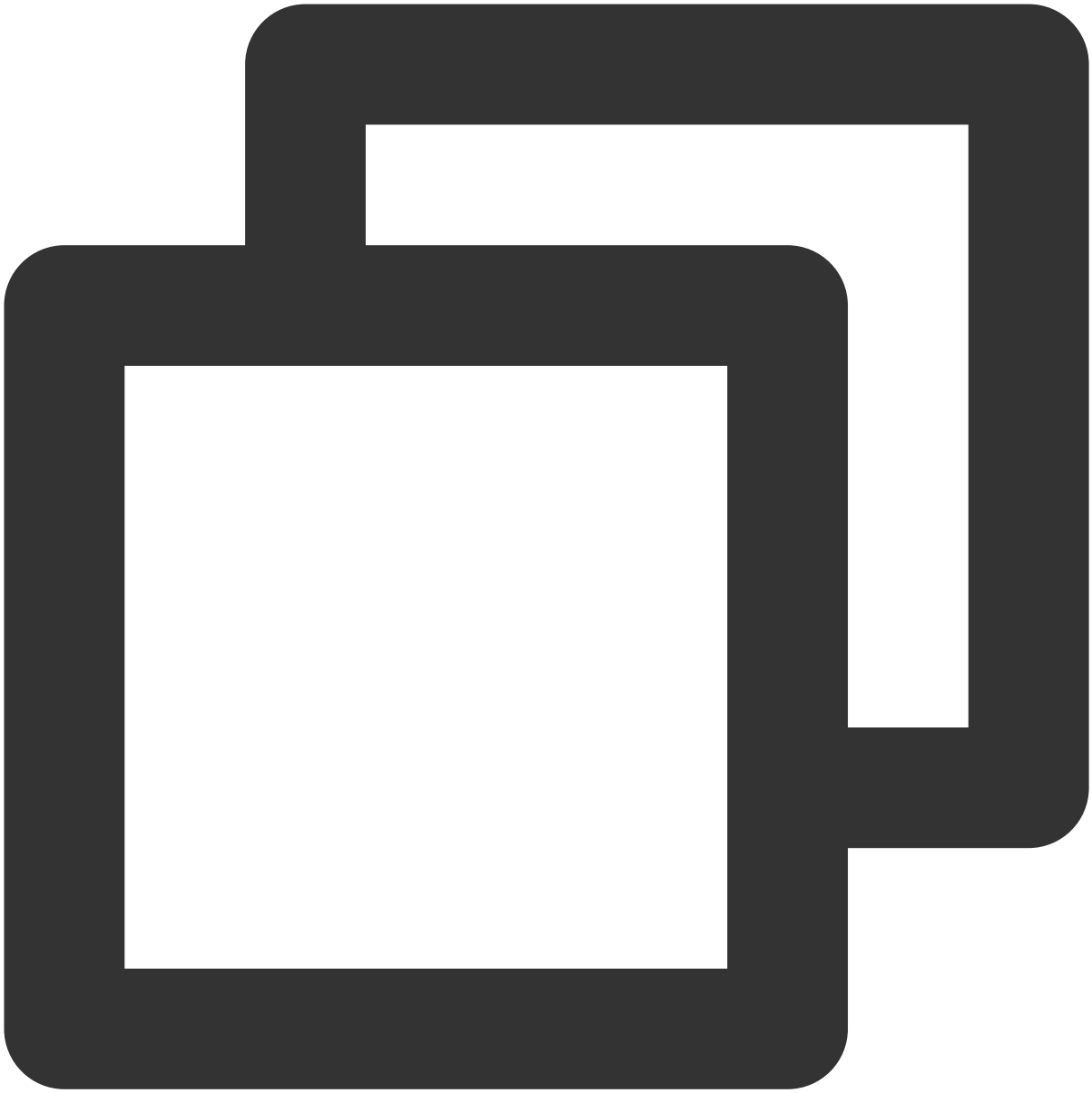
美颜项名称	增强模式下，建议的最大值（放大倍数）
美白，短脸，V脸，眼距，鼻子位置，祛法令纹，口红，立体	1.3
亮眼	1.5
腮红	1.8
其他	1.2

高性能模式（V3.1.0新增）

高性能模式开启后，美颜占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象，更适合低端机长时间用。

注意：开启高性能模式后，以下美颜项将不可用：

1. 眼部：眼宽、眼高、祛眼袋。
2. 眉毛：角度、距离、高度、长度、粗细、眉峰。
3. 嘴部：微笑唇。
4. 面部：瘦脸（自然，女神，英俊），收下颌，祛皱、祛法令纹。建议用“脸型”实现综合大眼瘦脸效果。



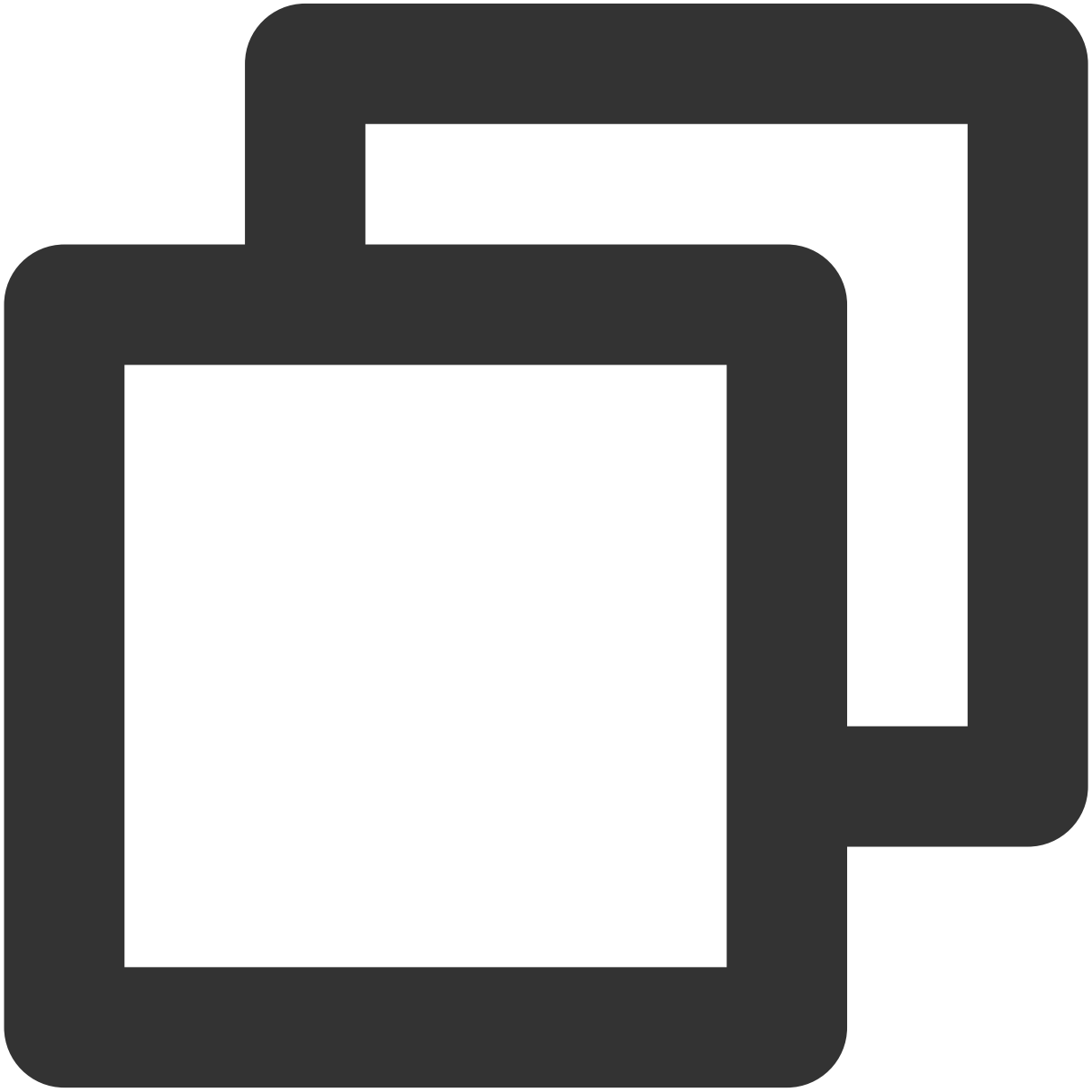
```
NSDictionary *assetsDict = @{@"core_name":@"LightCore.bundle",
                             @"root_path":[[NSBundle mainBundle] bundlePath],
                                @"setDowngradePerformance":@(YES) //开启高性能模式

    };
    self.beautyKit = [[XMagic alloc] initWithRenderSize:previewSize assetsDict:asse
```

静态函数

API	描述
isBeautyAuthorized	获取该美颜参数的授权信息

isBeautyAuthorized

获取该美颜参数的授权信息（仅支持美颜和美体）



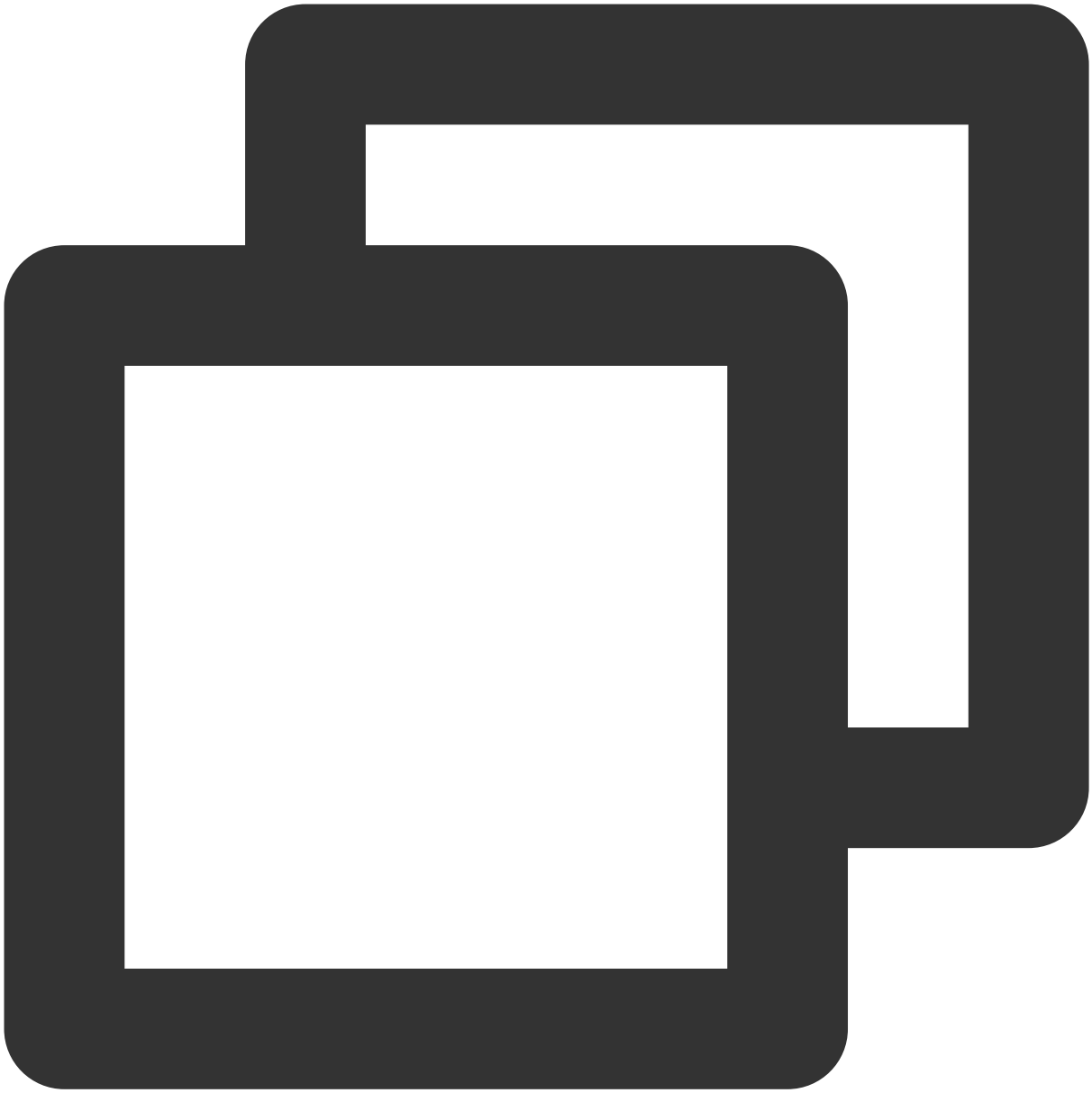
```
/// @param featureId 配置美颜参数
/// @return 返回对应美颜参数的授权结果
+ (BOOL)isBeautyAuthorized:(NSString * _Nullable)featureId;
```

回调

API	描述
YTSDKEventListener	SDK 内部事件回调接口
YTSDKLogListener	日志监听回调

YTSDKEventListener

SDK 内部事件回调接口



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

成员函数

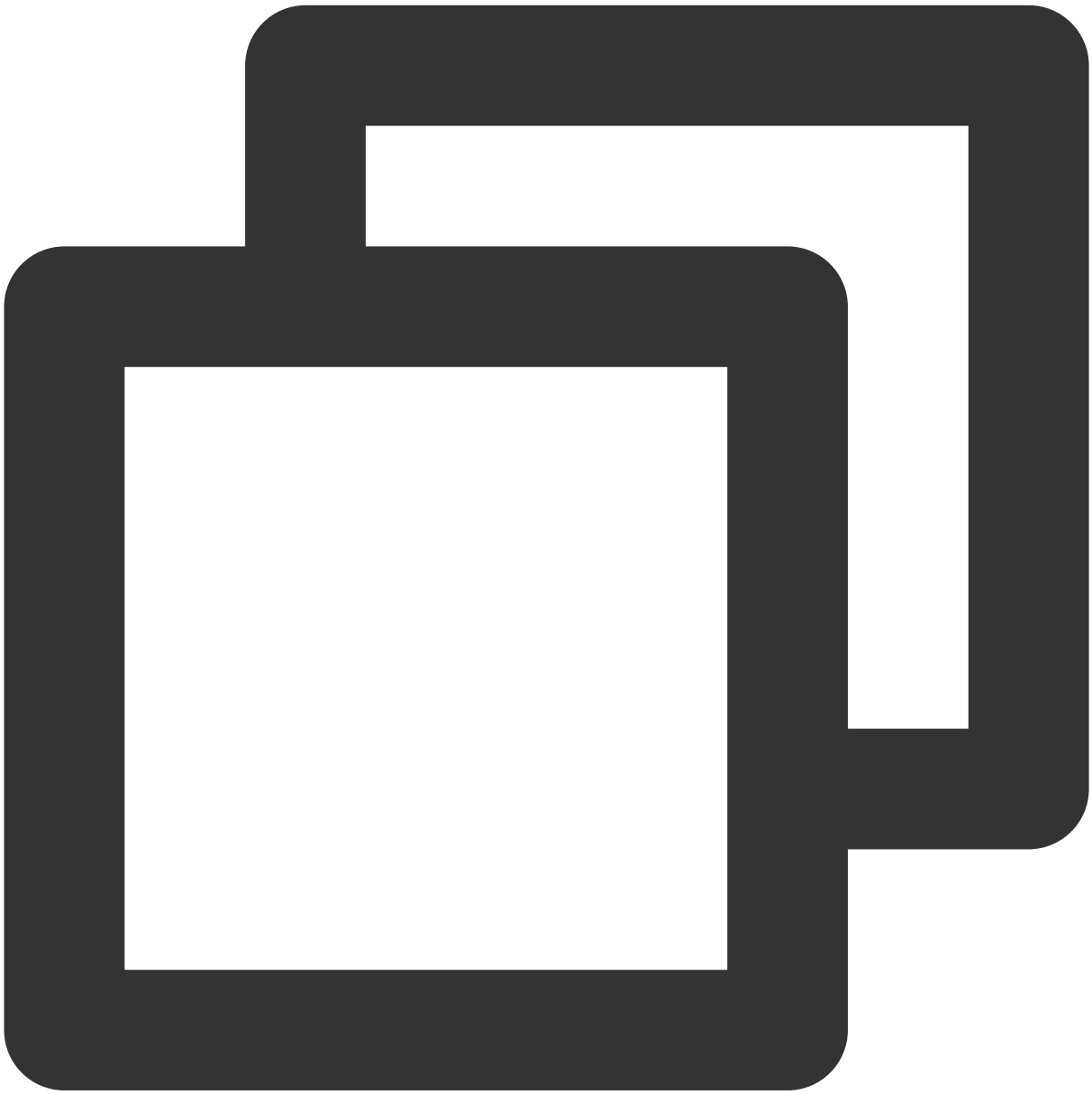
返回类型	名称
void	onAIEvent
void	onTipsEvent

void	onAssetEvent

函数说明

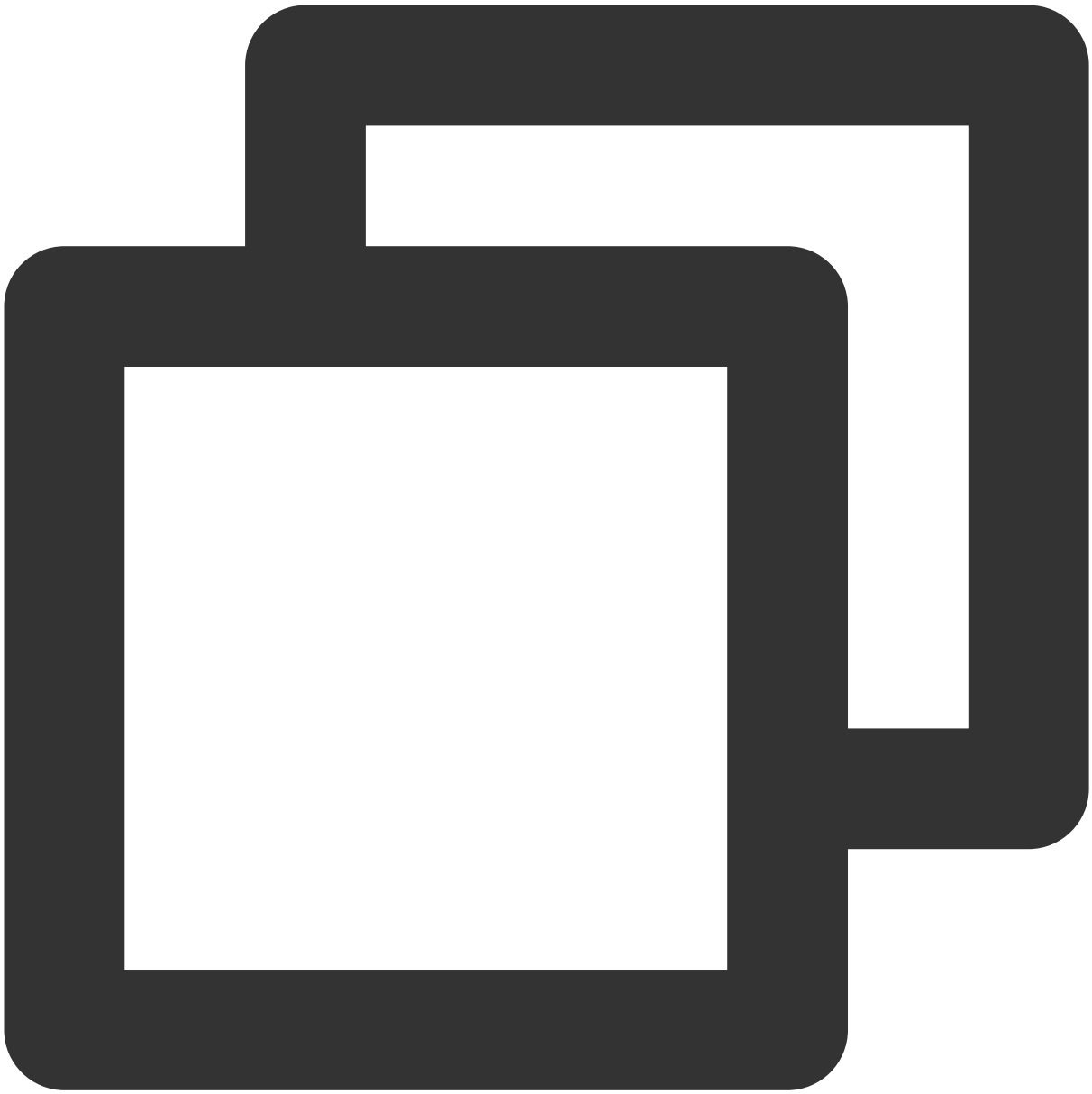
onAIEvent

AI 事件回调



```
/// @param event dict格式的回调
- (void)onAIEvent:(id _Nonnull)event;
```

最多返回5个人脸信息：



```
{
 "face_info":[{
  "trace_id":5,
  "face_256_point":[
    180.0,
    112.2,
    ...
  ],
  "face_256_visible":[
```

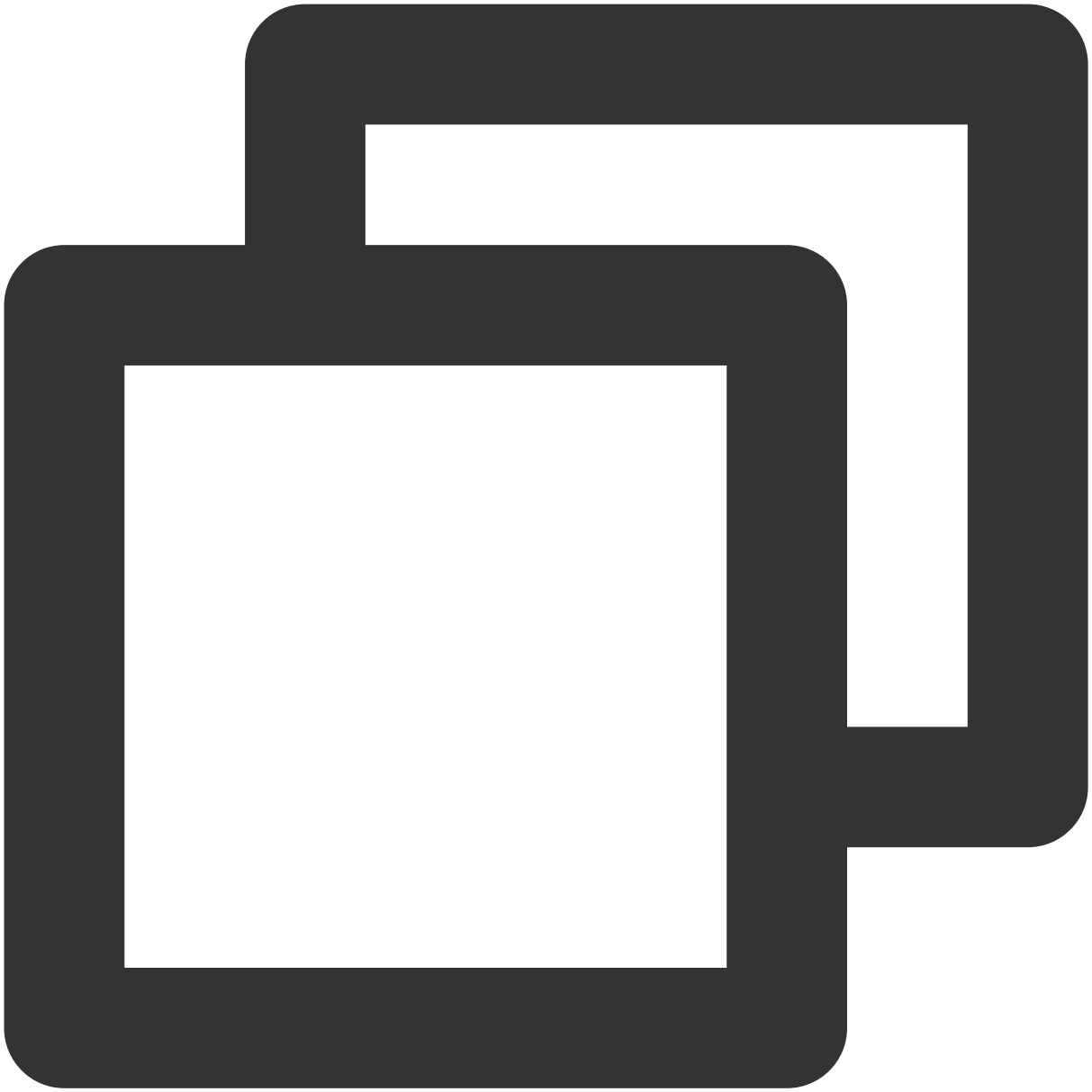
```
      0.85,
      ...
    ],
    "out_of_screen":true,
    "left_eye_high_vis_ratio":1.0,
    "right_eye_high_vis_ratio":1.0,
    "left_eyebrow_high_vis_ratio":1.0,
    "right_eyebrow_high_vis_ratio":1.0,
    "mouth_high_vis_ratio":1.0
  },
  ...
  ]
}
```

字段含义

字段	类型	值域	说明
trace_id	int	[1,INF)	人脸 ID，连续取流过程中，ID 相同的可以认为是同一张人脸
face_256_point	float	[0,screenWidth] 或 [0,screenHeight]	共512个数，人脸256个关键点，屏幕左上角为(0,0)
face_256_visible	float	[0,1]	人脸256关键点可见度
out_of_screen	bool	true/false	人脸是否出框
left_eye_high_vis_ratio	float	[0,1]	左眼高可见度点位占比
right_eye_high_vis_ratio	float	[0,1]	右眼高可见度点位占比
left_eyebrow_high_vis_ratio	float	[0,1]	左眉高可见度点位占比
right_eyebrow_high_vis_ratio	float	[0,1]	右眉高可见度点位占比
mouth_high_vis_ratio	float	[0,1]	嘴高可见度点位占比

onTipsEvent

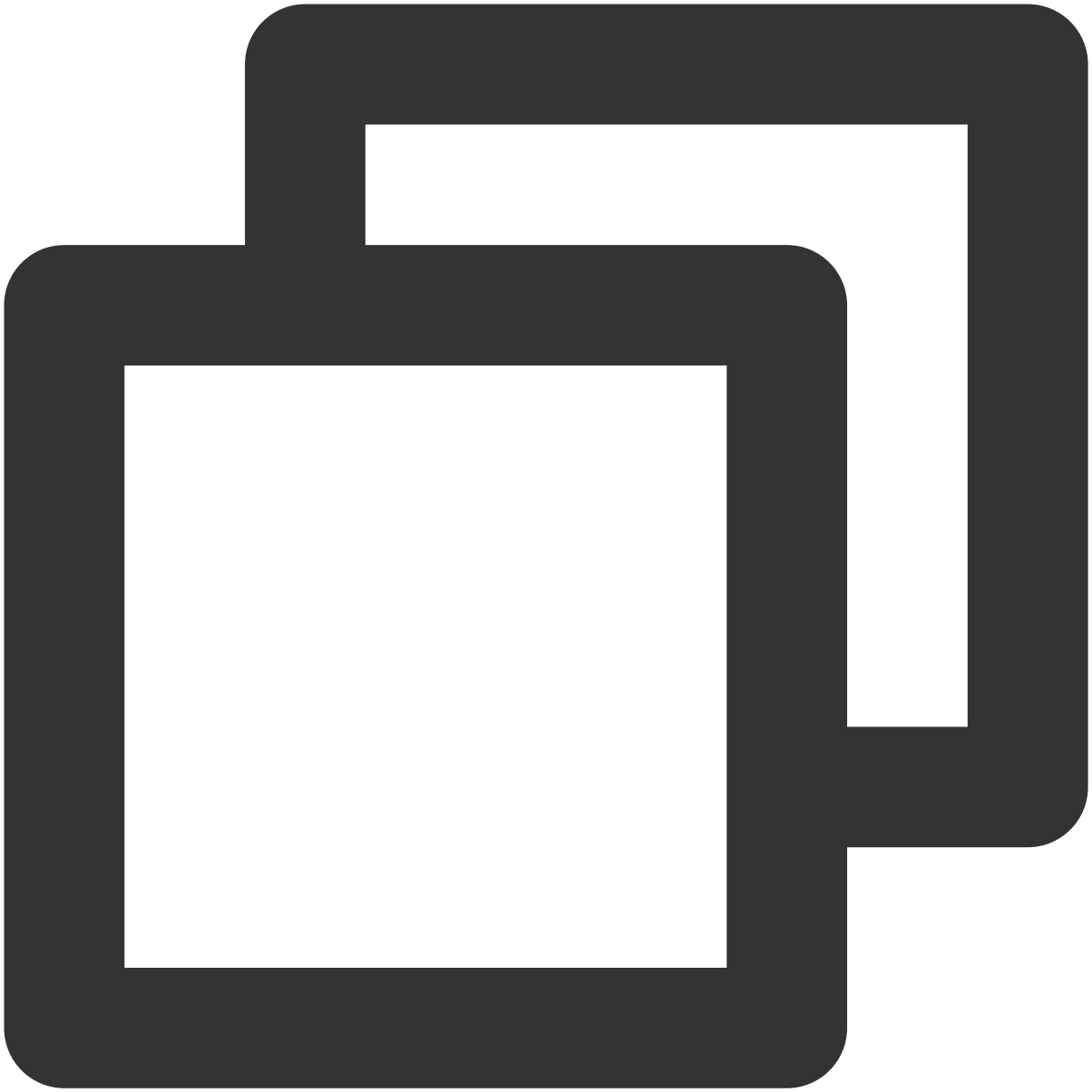
提示事件回调



```
/// @param event dict格式的回调
- (void)onTipsEvent:(id _Nonnull)event;
```

onAssetEvent

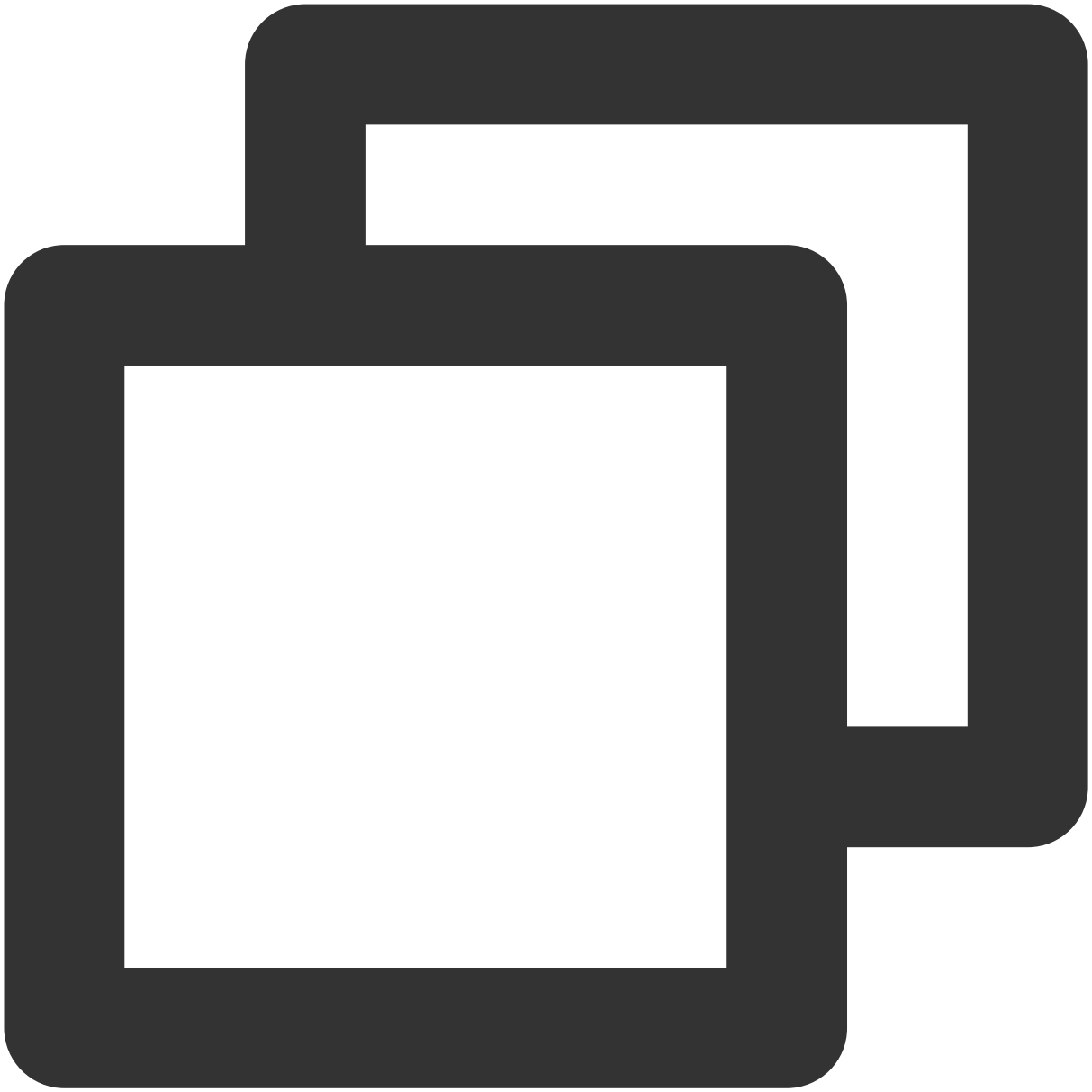
资源包事件回调



```
/// @param event string格式的回调
- (void)onAssetEvent:(id _Nonnull)event;
```

YTSDKLogListener

日志监听回调



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

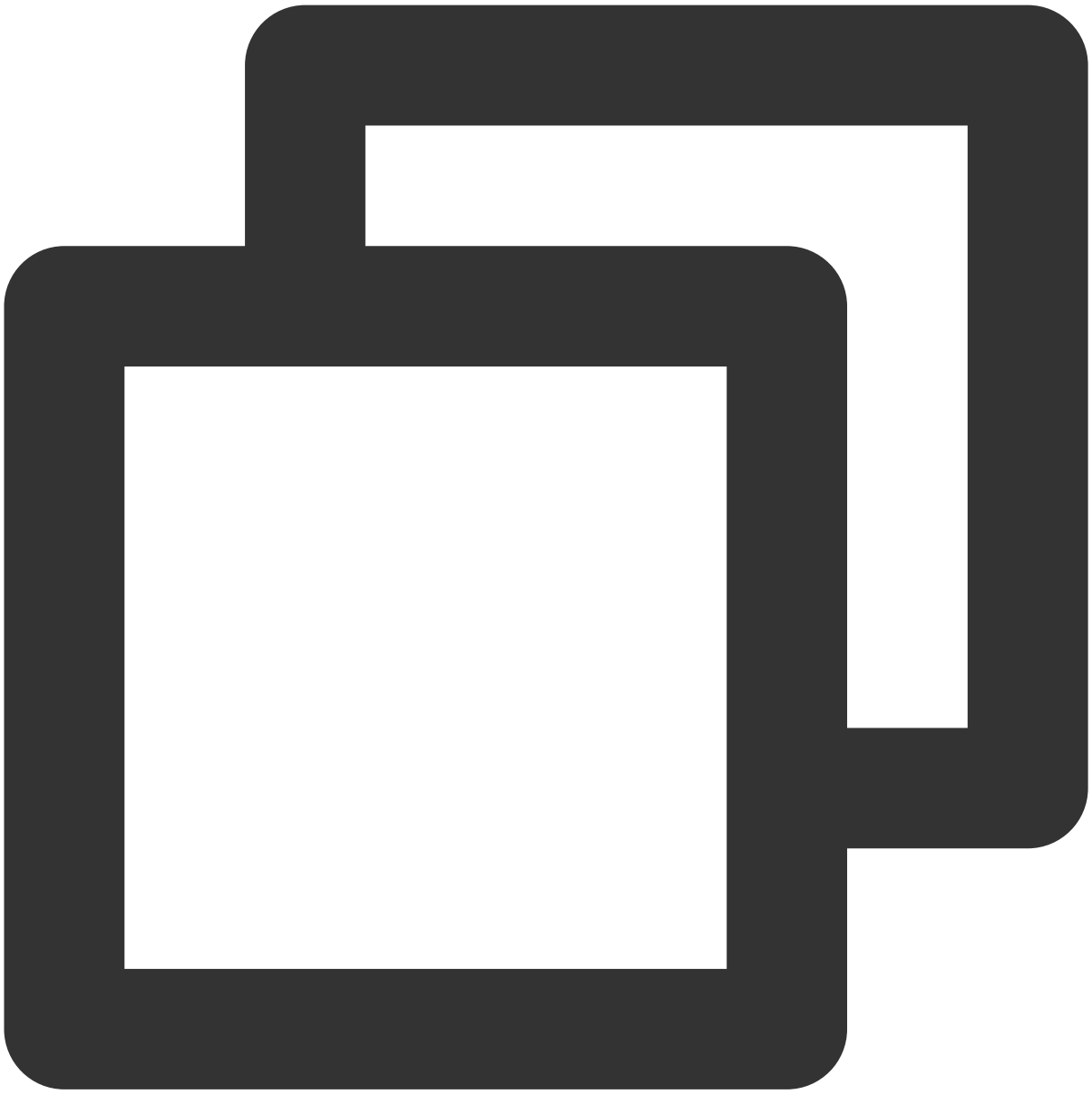
成员函数

返回类型	函数名称
void	onLog

函数说明

onLog

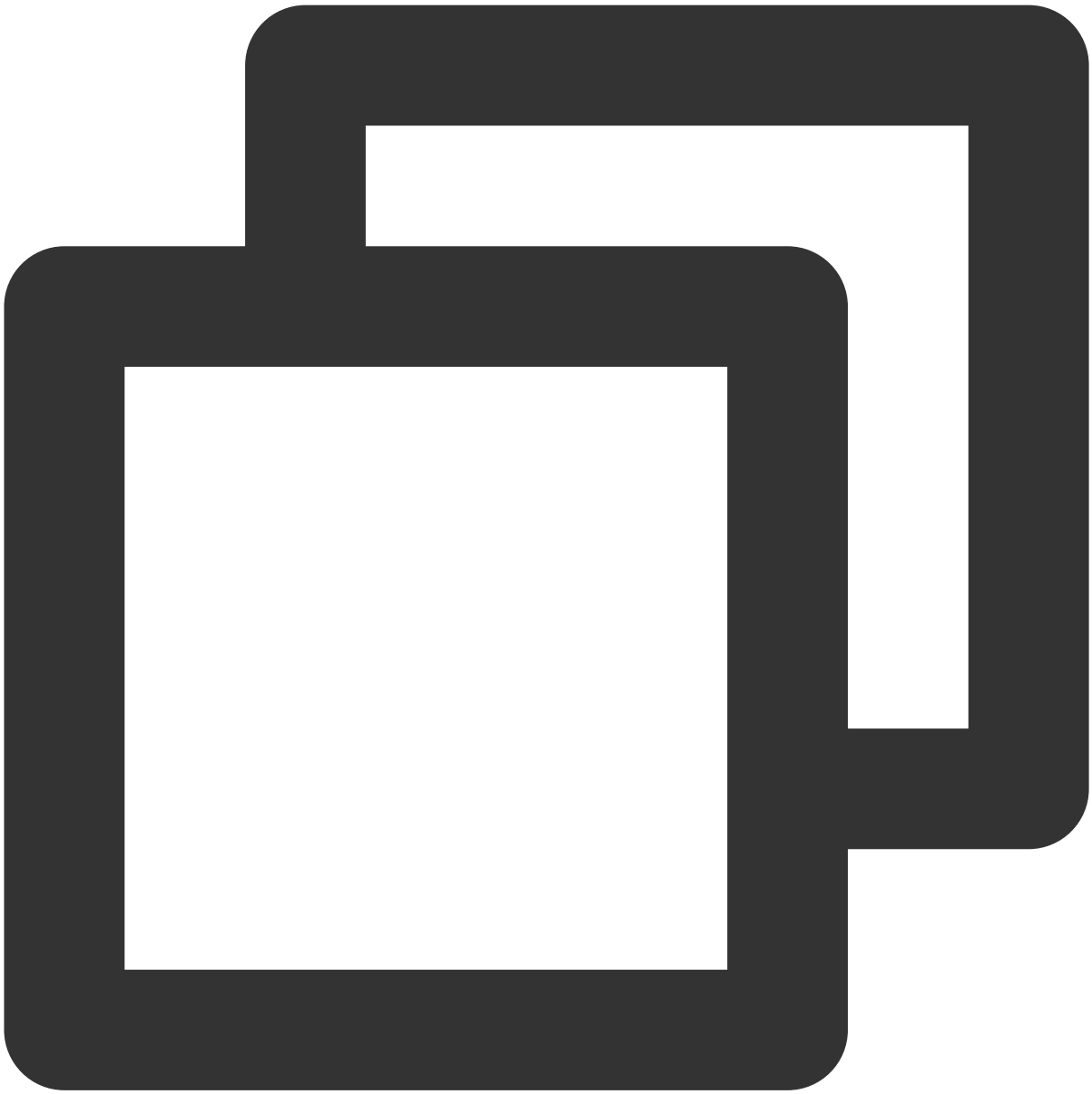
日志监听回调



```
/// @param loggerLevel 返回当前日志等级
/// @param logInfo 返回当前日志信息
- (void)onLog:(YtSDKLoggerLevel) loggerLevel withInfo:(NSString * _Nonnull) logInfo
```

素材叠加(3.0.1.5新增)

如果想要某个动效/美妆/分割素材叠加在当前素材上，则设置该素材时，在 withExtraInfo 的字典中设置 mergeWithCurrentMotion 为 true，示例如下：



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

素材叠加注意事项：

1. 客户需要自行管理素材之间是否适合叠加。举两个例子：

例1：特效 A 是变成贵妃脸，特效 B 是变成童话脸，这两个特效叠加后可能会导致画面非常别扭。

例2：特效 A 是个兔耳朵，特效 B 是猪耳朵，两个叠加后，就有两种耳朵。
例1和例2这两种情况不适合叠加。如果特效 A 是兔耳朵，特效 B 是送一个飞吻，这两个特效不会冲突，就适合叠加。
2. 只支持简单素材之间的叠加。简单素材是指只有单动效能力、或者单美妆效果、或者单抠背等，复杂素材是指包含了多种效果。简单素材和复杂素材没有明确的界定，建议客户充分测试后，自行管理哪些素材之间可以叠加，哪些不能叠加。
3. 叠加时，有动作触发的特效（例如伸出手触发某个特效、微笑触发某个特效等）属于复杂特效，需要放在前面，简单特效放在后面叠加在它之上。
4. 使用示例：主播使用了特效 A，然后观众送礼物特效 B，B 要叠加在 A 之上，一段时间后 B 消失，恢复成特效 A。那么设置步骤如下：
4.1 设置特效 A，mergeWithCurrentMotion 设置为 false。
4.2 设置特效 B，mergeWithCurrentMotion 设置为 true。
4.3 一小段时间后，再设置 A，mergeWithCurrentMotion 设置为 false。

Android

最近更新时间：2024-02-29 11:13:01

腾讯特效 SDK 核心接口类 `XmagicApi.java`，用于初始化 SDK、更新美颜数值、调用动效等功能。

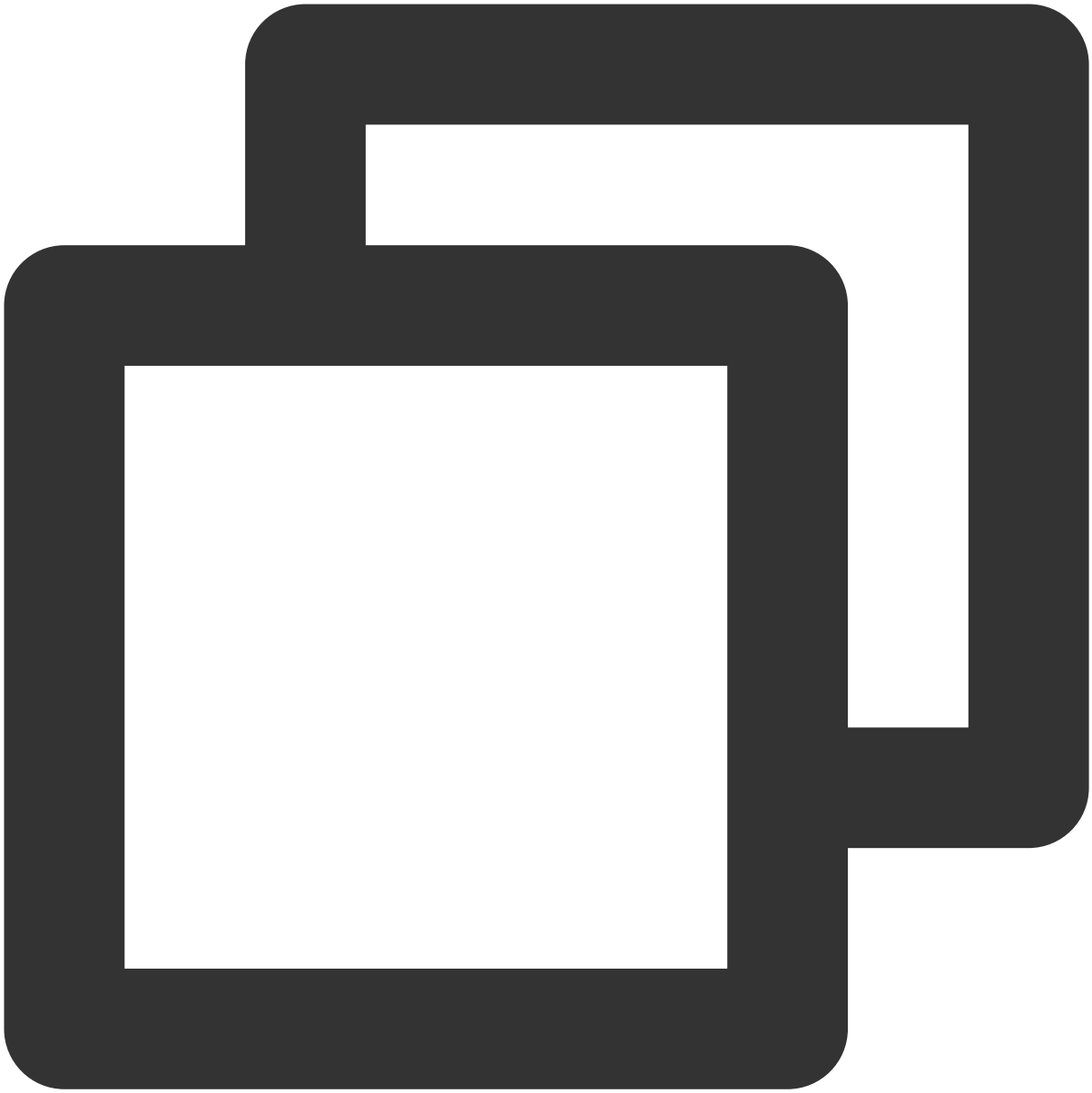
Public 成员函数

API	描述
XmagicApi	构造函数。
~~updateProperty~~	更新属性， 可在任意线程调用（**V3.3.0及以前**）。
~~updateProperties~~	批量更新属性，可在任意线程调用（**V3.3.0及以前**）。
setEffect	设置美颜、美型、滤镜、美妆、贴纸、分割等效果，可在任意线程调用（**V3.5.0版本新增**）
setTipsListener	设置动效提示语回调函数，用于将提示语展示到前端页面上。
setYTDataListener	设置人脸点位信息等数据回调，需要获得人脸点位的 Licence 授权（例如原子能力X102）才会有回调。
setAIDataListener	设置人脸、手势、身体检测状态回调。
onPause	暂停声音播放，可与 Activity onPause 生命周期绑定。
onResume	恢复渲染，可与 Activity onResume 生命周期绑定。
onDestroy	销毁 xmagic，需要在 GL 线程中调用。
process	SDK 渲染接受数据的方法，可在相机数据回调函数内使用。
onPauseAudio	当仅需要停止音频，但不需要释放 GL 线程时调用此函数。
sensorChanged	用于判断当前手机旋转的角度，从而调整 AI 识别人脸的判断角度依据。
isDeviceSupport	将动效资源列表传入 SDK 中做检测，执行后 `XmagicProperty.isSupport` 字段标识该原子能力是否可用。 根据 `XmagicProperty.isSupport` 可 UI 层控制单击限制，或者直接从资源列表删除。
getPropertyRequiredAbilities	传入一个动效资源列表，返回每一个资源所使用到的 SDK 原子能力列表。

getDeviceAbilities	返回当前设备支持的原子能力表。
isSupportBeauty	判断当前机型是否支持美颜（OpenGL3.0）。
isBeautyAuthorized	判断当前的 lic 授权支持哪些美颜。 仅支持 BEAUTY 和 BODY_BEAUTY 类型的美颜项检测。检测后的结果会赋值到各个美颜对象 XmagicProperty.isAuth 字段中。
setXmagicStreamType	设置输入数据类型，默认 Android camera 数据流。
setXmagicLogLevel	设置 SDK 的 log 等级，默认为 `WARN` 。开发调试阶段如有需要，可以将其设为 `Log.DEBUG`。 正式发布时务必设置为 `Log.WARN` 或 `Log.ERROR` ，否则大量的日志会影响性能。 **在 new XmagicApi() 之后调用。**
setAudioMute	动效素材使用时是否开启静音（V2.5.0新增）： 参数：true 表示静音，false 表示非静音。
enableEnhancedMode	开启美颜增强模式（V2.5.1新增）。默认未开启。 未开启时，应用层可以设置的各美颜项的强度范围为0到1或-1到1，如果超出此范围，SDK 会取边界值。例如应用层设置瘦脸为1.2，SDK 判断其超出了最大值1.0，则在内部把瘦脸值修正为1.0。 开启增强模式后，应用层可以设置更大范围的数值。例如想要瘦脸程度更大，则可以把瘦脸值设置为1.2，SDK会接受并使用1.2这个数值，不会将其修正为1.0。 **说明：** 开启增强模式后，需要应用层自己管理每个美颜项可以设置的最大值，让用户在此范围内调整数值。我们提供了一份 参考值，您可以根据产品需求自由调整，但不建议超出我们的推荐值，否则美颜效果可能变差。
setDowngradePerformance	调用此方法开启高性能模式。高性能模式开启后，美颜占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象，更适合低端机长时间使用。
exportCurrentTexture	获取当前纹理上的画面
setFeatureEnableDisable	开启或关闭某个能力

XmagicApi

构造函数。



```
XmagicApi(Context context, String resDir)
XmagicApi(Context context, String resDir,OnXmagicPropertyErrorListener xmagicProper
```

参数

参数	类型	含义
context	Context	上下文。
resDir	String	资源文件目录。

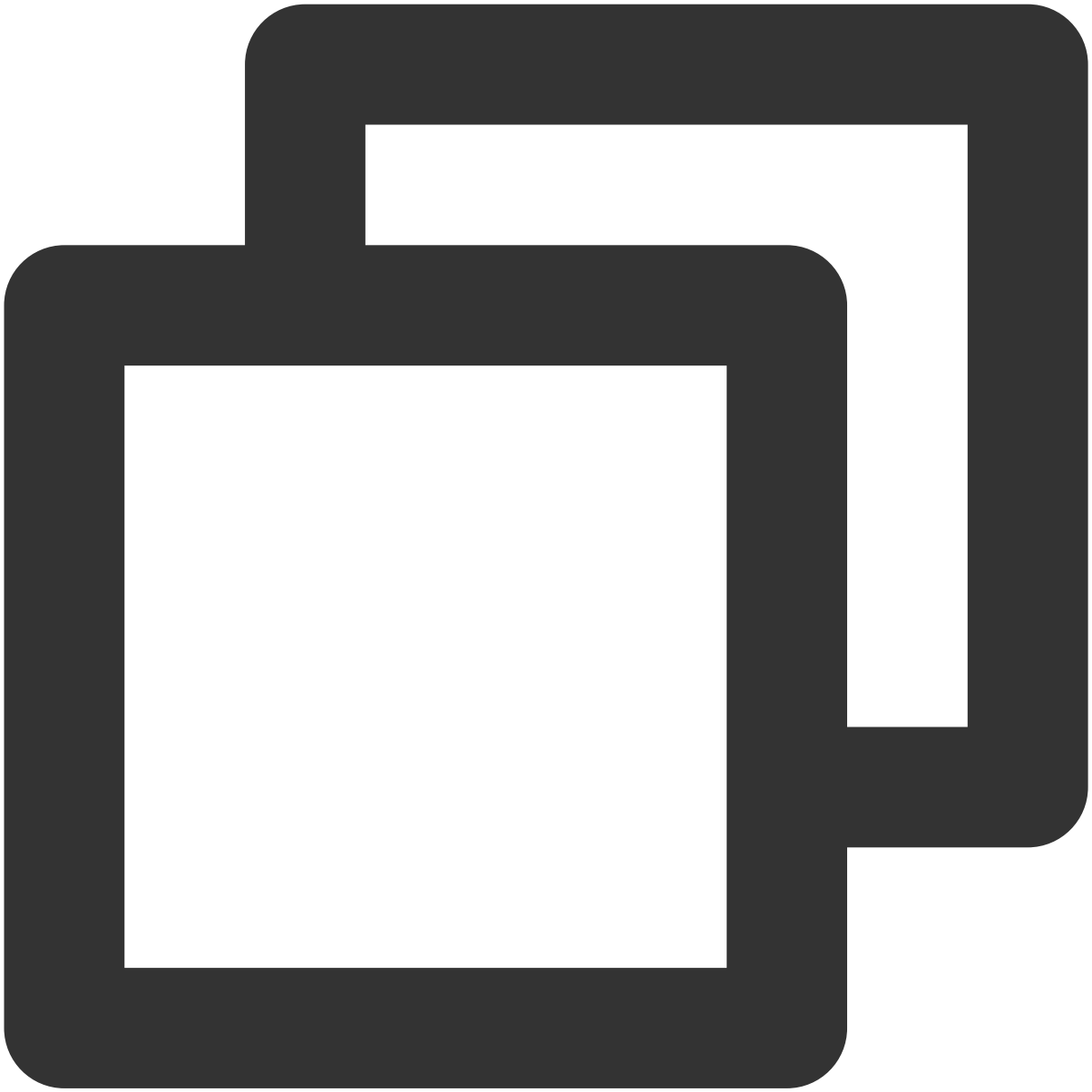
		如果 SDK 资源文件是内置在 assets 的，那么需要把资源 copy 到 App 的私有目录：先通过 `XmagicResParser.setResPath(new File(getFilesDir(), "xmagic").get` 设置资源路径，再通过 `XmagicResParser.copyRes(getAppli` 完成资源拷贝，详见 Demo 的 `TEMenuAct` 如果 SDK 资源文件是联网下载的，下载成功 `XmagicResParser.setResPath(valic` 设置资源路径。 通过 `XmagicResParser.getResPath()` 径。
xmagicPropertyErrorListener	OnXmagicPropertyErrorListener	错误回调接口。

返回错误码含义对照表：

错误码	含义
-1	未知错误。
-100	3D 引擎资源初始化失败。
-200	不支持 GAN 素材。
-300	设备不支持此素材组件。
-400	模板 JSON 内容为空。
-500	SDK 版本过低。
-600	不支持分割。
-700	不支持 OpenGL。
-800	不支持脚本。
5000	分割背景图片分辨率超过 2160×3840。
5001	分割背景图片所需内存不足。
5002	分割背景视频解析失败。
5003	分割背景视频超过200秒。
5004	分割背景视频格式不支持。
5005	分割背景图片存在旋转角度

~~updateProperty，updateProperties~~

更改某一项或批量更改多项美颜、动效、滤镜，可在任意线程调用。



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

参数

参数	含义

XmagicProperty<?> p	腾讯特效数据实体类。 以"磨皮"为例，可以按如下方式 new 一个实例：`new XmagicProperty<>(Category.BEAUTY, null, null, BeautyConstant.BEAUTY_SMOOTH, new XmagicPropertyValues(0, 100, 50, 0, 1));` 以“2D 动效兔兔酱”为例，可以按如下方式 new 一个实例：`new XmagicProperty<>(Category.MOTION, "video_tutujiang" , "动效的文件路径", null, null);` 如果想要某个动效/美妆/分割素材叠加在当前素材上，则将该素材 XmagicProperty 对象的 mergeWithCurrentMotion 设置为 true。关于素材叠加的详细说明见 素材叠加。 更多的例子，请参见 Demo 工程 assets/beauty_panel 文件夹下的配置信息。

~~XmagicProperty~~ （美颜参数说明）

美颜

属性字段	说明
category	Category.BEAUTY
ID	null 特殊情况： 瘦脸中的（自然、女神、英俊） ID 值分别为：`BeautyConstant.BEAUTY_FACE_NATURE_ID`、`BeautyConstant.BEAUTY_FACE_FEMALE_GOD_ID`、`BeautyConstant.BEAUTY_FACE_MALE_GOD_ID` 口红中的 ID 值为：`XmagicConstant.BeautyConstant.BEAUTY_LIPS_LIPS_MASK` 腮红中的 ID 值为：`XmagicConstant.BeautyConstant.BEAUTY_MAKEUP_MULTIPLY_MULTIPLY_MASK` 立体中的 ID 值为：`XmagicConstant.BeautyConstant.BEAUTY_SOFTLIGHT_SOFTLIGHT_MASK`
resPath	null 特殊情况：口红、腮红、立体的 resPath 是资源图片的路径，具体请参考 Demo 中 assets/beauty_panel/advanced_beauty.json 文件。
effkey	必填，参考 Demo 示例：美白 BeautyConstant.BEAUTY_WHITEN
effValue	必填，参考 Demo 中 assets/beauty_panel/advanced_beauty.json 文件，demo工程中解析该文件构造一个XmagicPropertyValues对象，XmagicPropertyValues各个属性取值见美颜参数说明

美体

属性字段	说明
category	Category.BODY_BEAUTY
ID	null
resPath	null
effkey	必填，参考 Demo 示例：长腿 BeautyConstant.BODY_LEG_STRETCH
effValue	必填，参考 Demo 中 assets/beauty_panel/beauty_body.json 文件。demo工程中解析该文件构造一个XmagicPropertyValues对象，XmagicPropertyValues各个属性取值见美颜参数说明

滤镜

属性字段	说明
category	Category.LUT
ID	图片名称，必填 示例：dongjing_lf.png “无” ID 为 XmagicProperty.ID_NONE
resPath	滤镜图片路径，必填，“无”设置为 null
effkey	null
effValue	必填，“无”设置为 null。是一个XmagicPropertyValues对象，XmagicPropertyValues各个属性取值见 美颜参数说明

动效

属性字段	说明
category	Category.MOTION
ID	资源文件夹名称，必填 示例：video_lianliancaomei “无” ID 为 XmagicProperty.ID_NONE
resPath	必填，参考 Demo
effkey	null
effValue	null

美妆

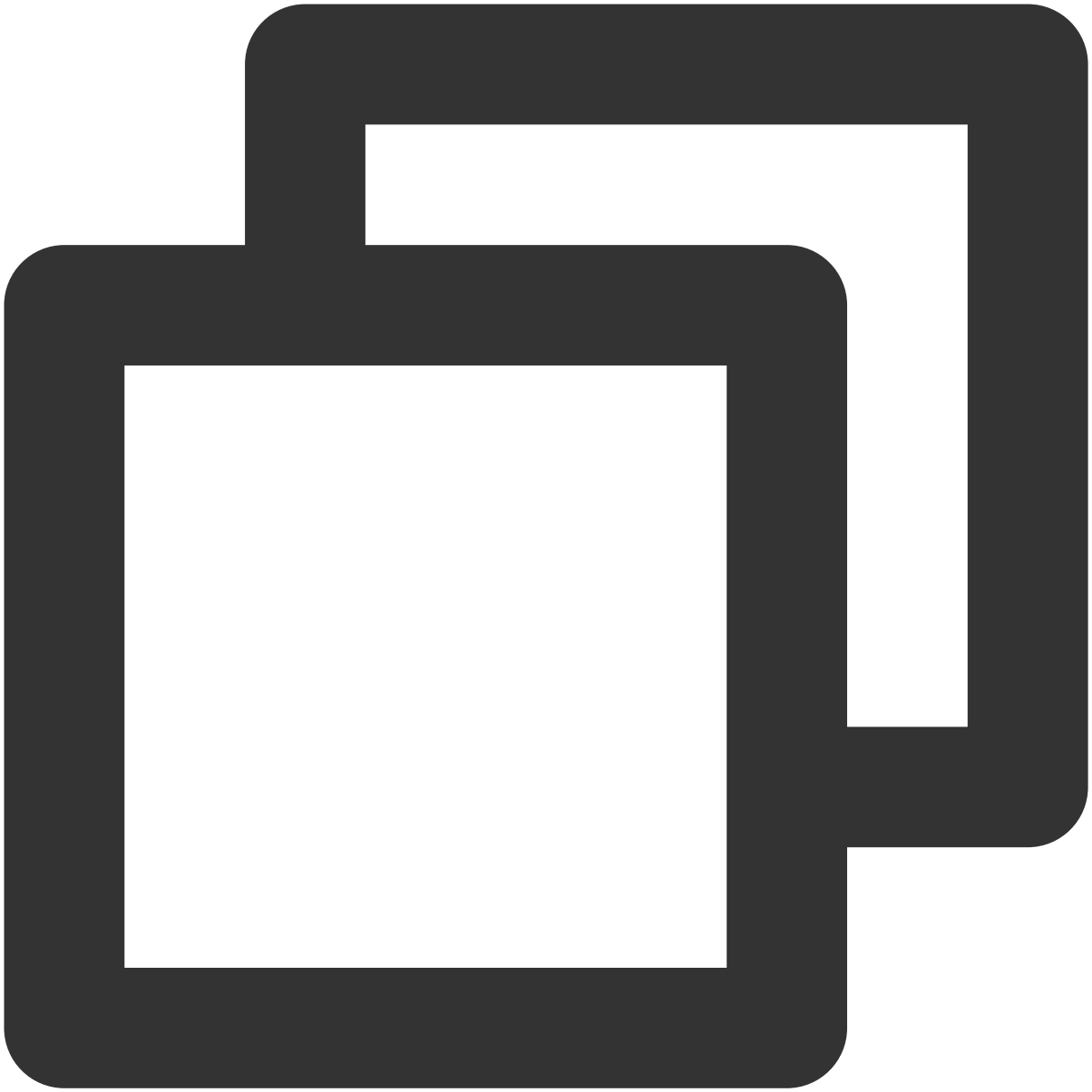
属性字段	说明
category	Category.MAKEUP
ID	资源文件夹名称，必填 示例：video_xuejiezhuang “无” ID 为 XmagicProperty.ID_NONE
resPath	必填，参考 Demo
effkey	必填，取值为：makeup.strength“无”设置为 null
effValue	必填，“无”设置为 null。是一个 XmagicPropertyValues 对象，XmagicPropertyValues 各个属性取值见 美颜参数说明

分割

属性字段	说明
category	Category.SEGMENTATION
ID	资源文件夹名称，必填 示例：video_segmentation_blur_45 “无” ID 为 XmagicProperty.ID_NONE 自定义分割 ID 值必须使用：`XmagicConstant.SegmentationId.CUSTOM_SEG_ID`
resPath	必填，参考 Demo
effkey	null（自定义背景除外），自定义背景的值为选择的资源路径
effValue	null

setEffect（V3.5.0新增）

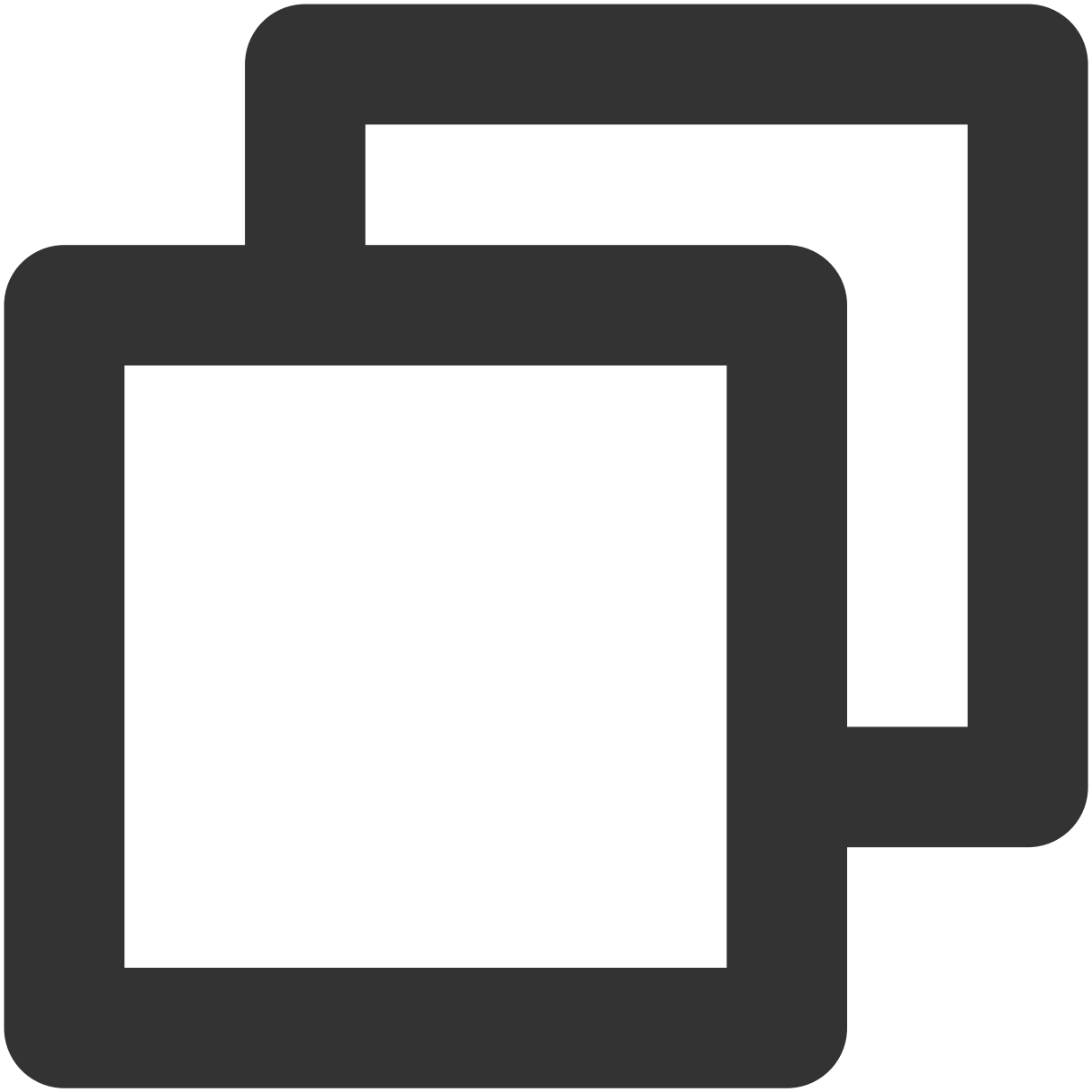
设置美颜、美型、滤镜、美妆、贴纸、分割等效果，可在任意线程调用。具体参数请参考 美颜参数说明。



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

setTipsListener

设置动效提示语回调函数，用于将提示语展示到前端页面上。例如某些素材会提示用户点点头、伸出手掌、比心等。



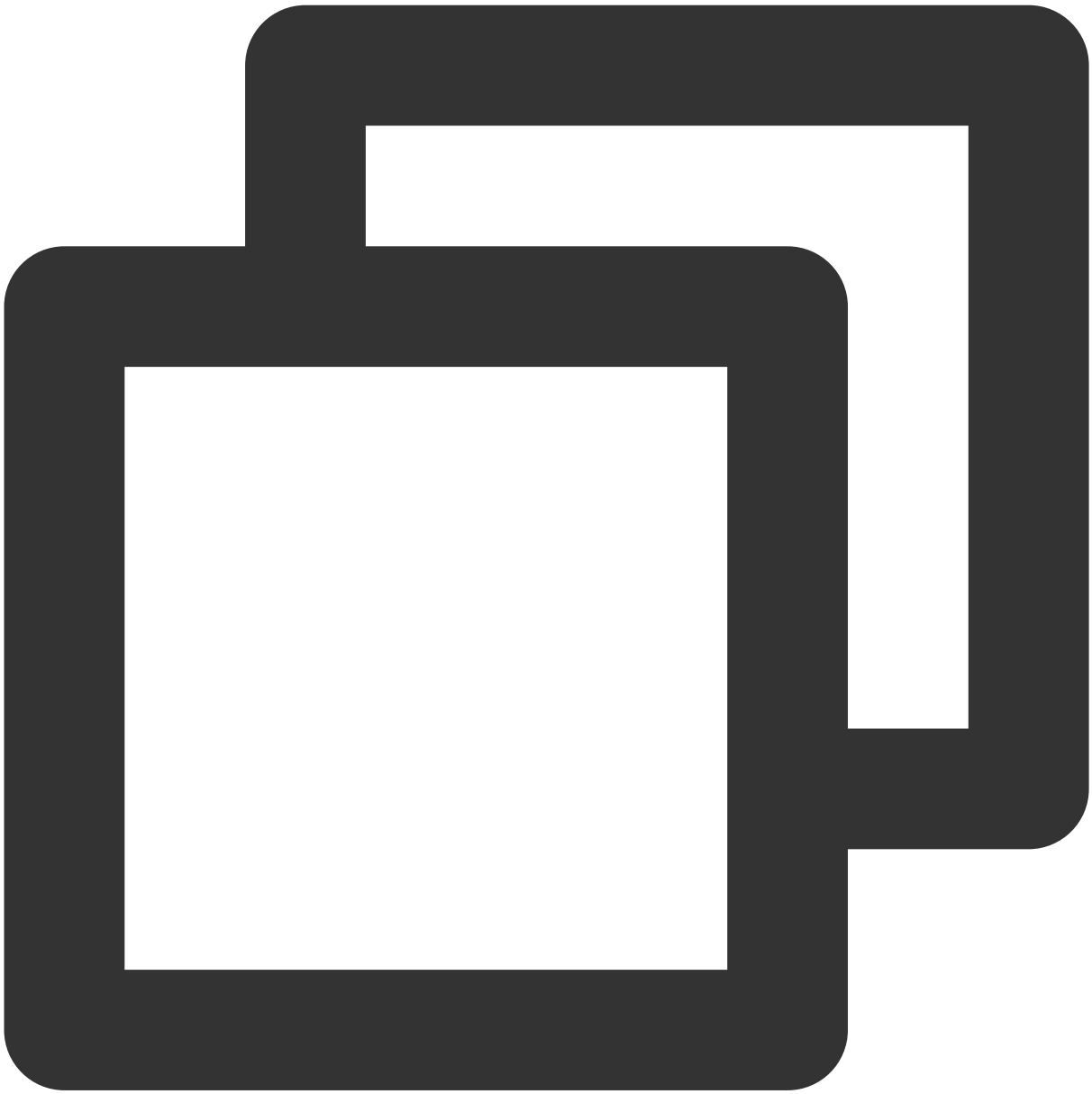
```
void setTipsListener(XmagicApi.XmagicTipsListener effectTipsListener)
```

参数

参数	含义
XmagicApi.XmagicTipsListener effectTipsListener	回调函数实现类，回调不一定在主线程。

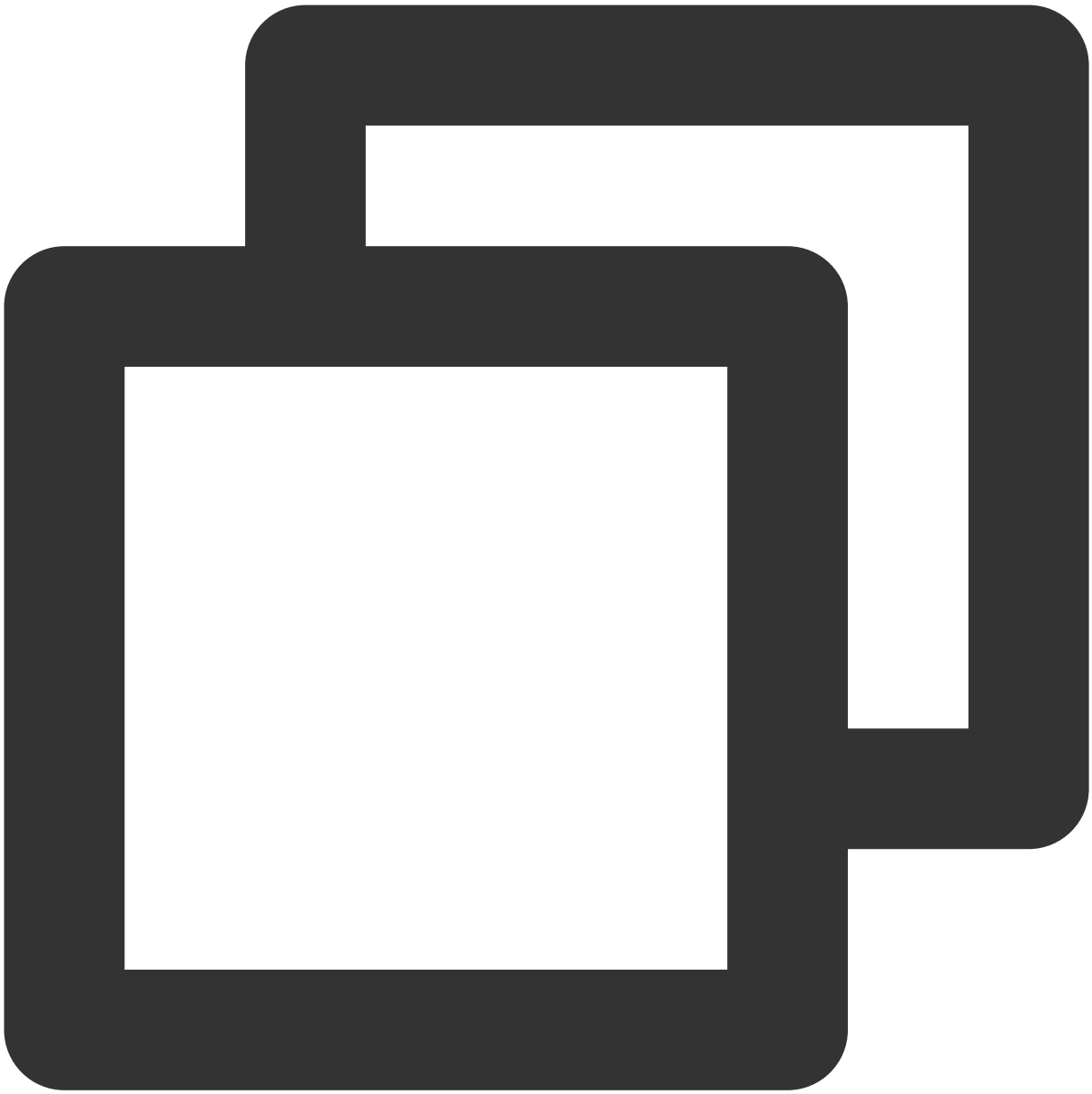
~~setYTDataListener~~（此接口在3.0.0版本已经删除，功能移植到了XmagicAIDataListener的onAIDataUpdated方法）

设置人脸点位信息等数据回调。



```
void setYTDataListener(XmagicApi.XmagicYTDataListener ytDataListener)
设置人脸信息等数据回调

public interface XmagicYTDataListener {
    void onYTDataUpdate(String data)
}
```

onYTDataUpdate 返回 JSON string 结构，最多返回5个人脸信息：



```
{
 "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
  },
  ...
  ]
}
```

字段含义

字段	类型	值域	说明
trace_id	int	[1,INF)	人脸 ID，连续取流过程中，ID 相同的可以认为是同一张人脸。
face_256_point	float	[0,screenWidth] 或 [0,screenHeight]	共512个数，人脸256个关键点，屏幕左上角为(0,0)。
face_256_visible	float	[0,1]	人脸256关键点可见度。
out_of_screen	bool	true/false	人脸是否出框。
left_eye_high_vis_ratio	float	[0,1]	左眼高可见度点位占比。
right_eye_high_vis_ratio	float	[0,1]	右眼高可见度点位占比。
left_eyebrow_high_vis_ratio	float	[0,1]	左眉高可见度点位占比。
right_eyebrow_high_vis_ratio	float	[0,1]	右眉高可见度点位占比。
mouth_high_vis_ratio	float	[0,1]	嘴高可见度点位占比。

参数

参数	含义
XmagicApi.XmagicYTDataListener ytDataListener	回调函数实现类。

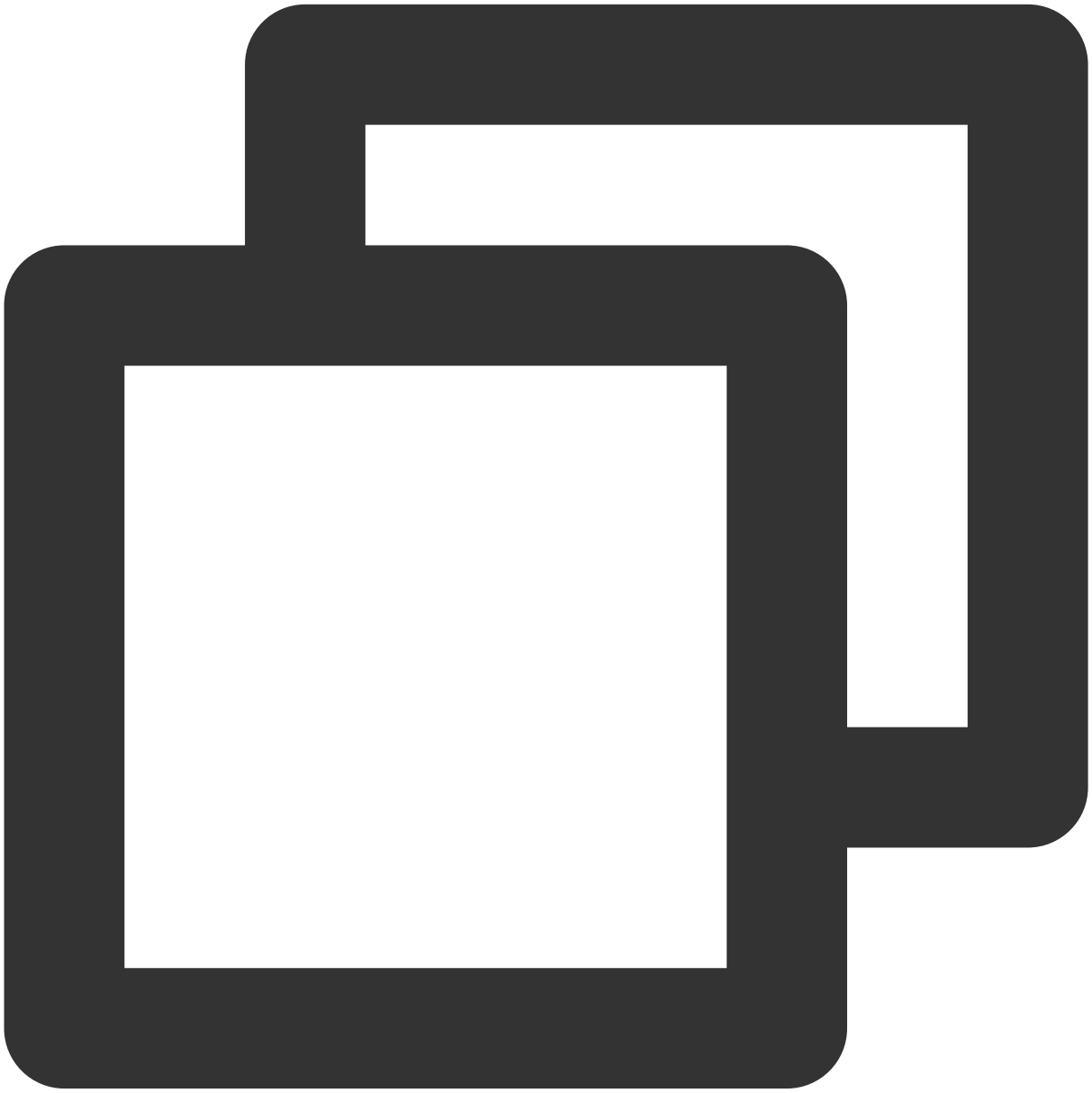
setAIDataListener

检测到人脸、身体、手势时，会回调这些部位的点位信息：

onFaceDataUpdated：开启美颜后就会有回调，识别到人脸时，回调 List<FaceData>，没有人脸时，回调一个空的 List。

onHandDataUpdated：设置了手势动效，且识别到手势时回调，其他情况不回调。

onBodyDataUpdated：设置了美体的属性并且识别到身体的时候回调，其他情况不回调。

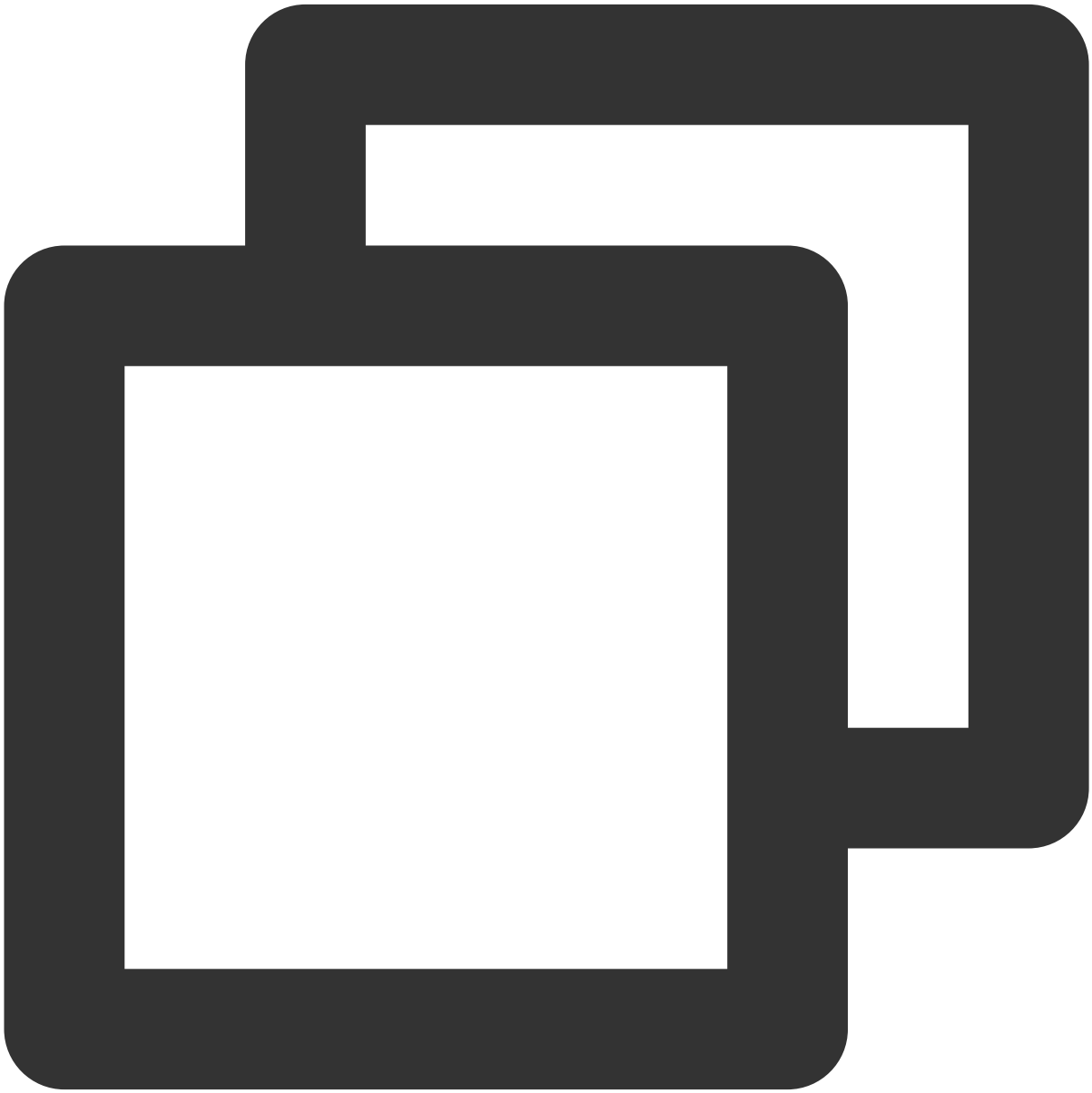


```
public interface OnAIDataListener {

    void onFaceDataUpdated(List<TEFaceData> faceDataList);
```

```
    void onHandDataUpdated(List<TEHandData> handDataList);
    void onBodyDataUpdated(List<TEBodyData> bodyDataList);

    void onAIDataUpdated(String jsonString);    //此方法是3.0.0版本新增方法，数据结构和之前
}
```

onPause

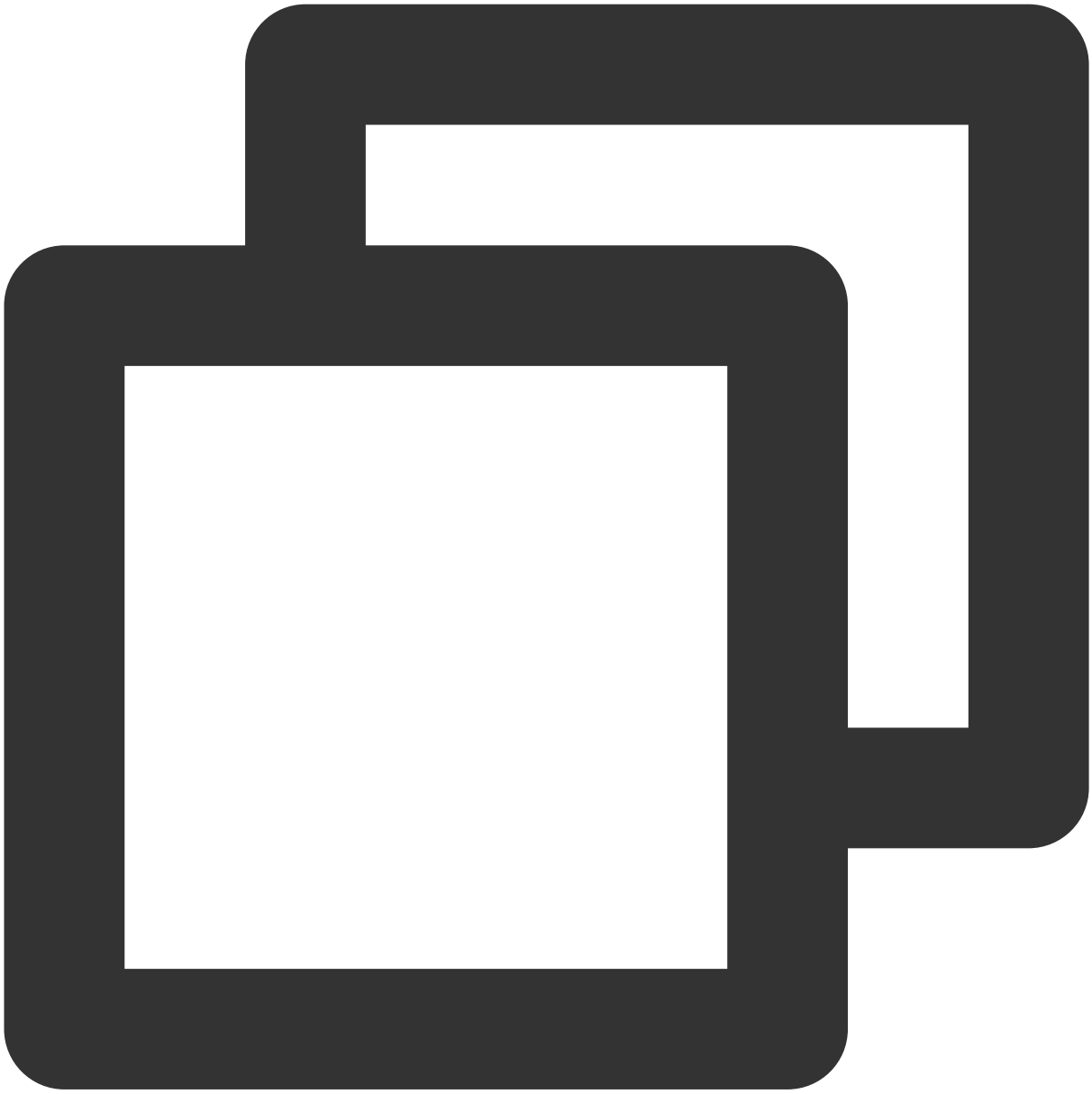
暂停渲染，可与 Activity onPause 生命周期绑定，目前内部仅调用 `onPauseAudio` 。



```
void onPause()
```

onResume

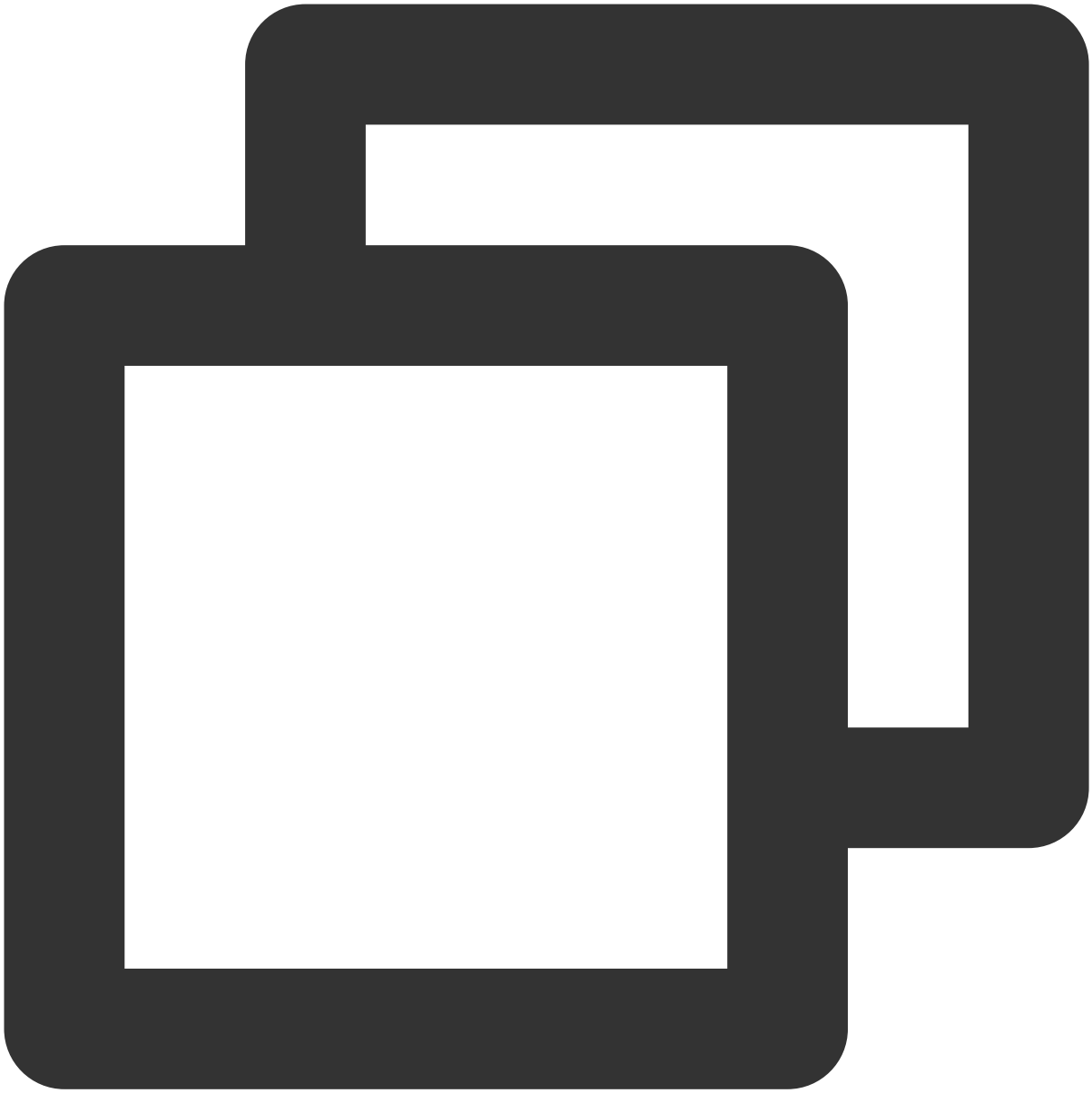
恢复渲染，可与 Activity onResume 生命周期绑定。



```
void onResume()
```

onDestroy

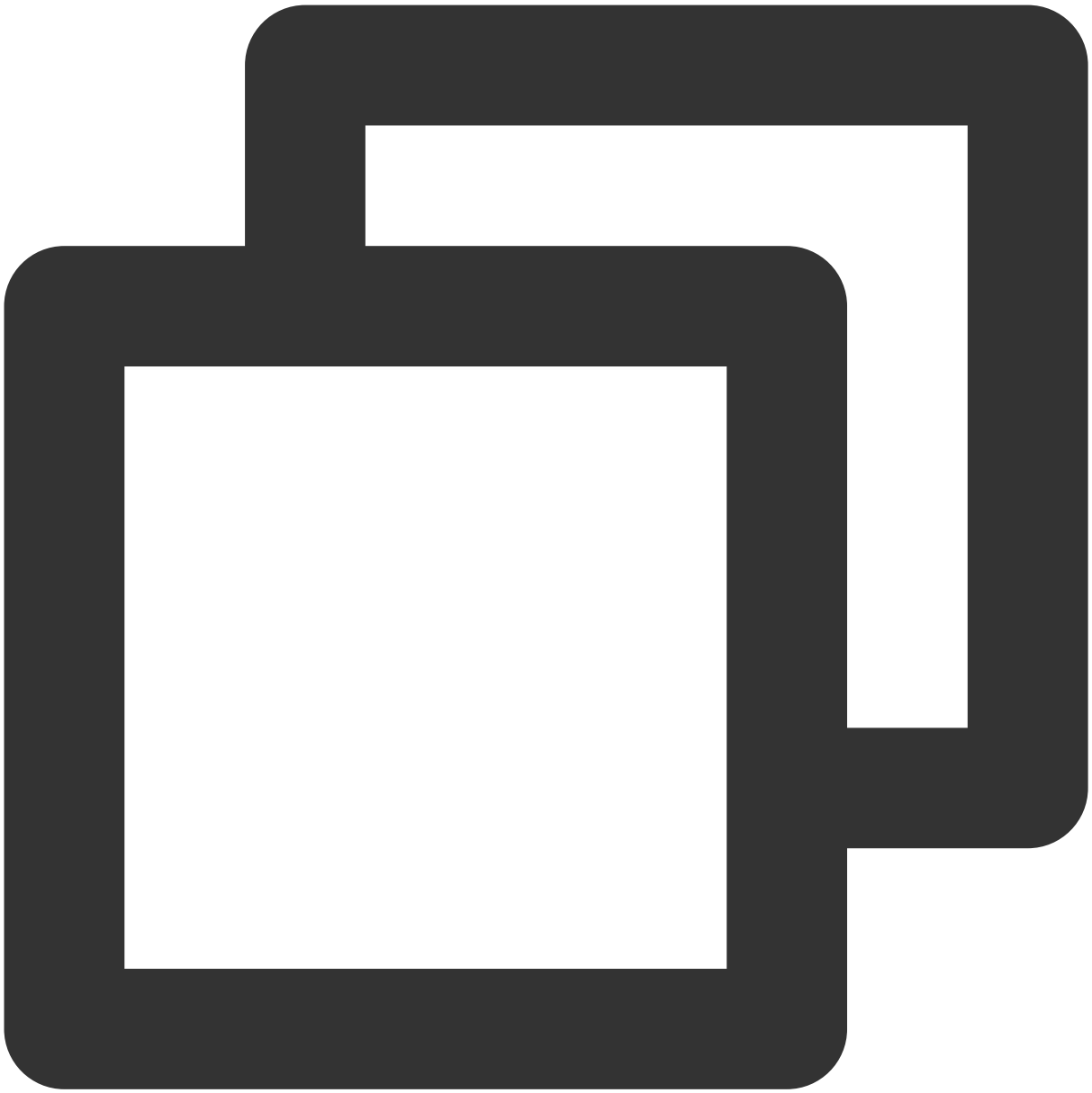
清理GL线程资源，需要在 GL 线程内调用。示例代码：



```
//示例代码见 TECameraBaseActivity.java
public void onGLContextDestroy() {
    if (this.mXMagicApi != null) {
        this.mXMagicApi.onDestroy();
        this.mXMagicApi = null;
    }
}
```

process

SDK 渲染接受数据的方法，可在相机数据回调函数内使用。



```
//渲染纹理
int process(int srcTextureId, int srcTextureWidth, int srcTextureHeight)
//渲染bitmap
Bitmap process(Bitmap bitmap, boolean needReset)
```

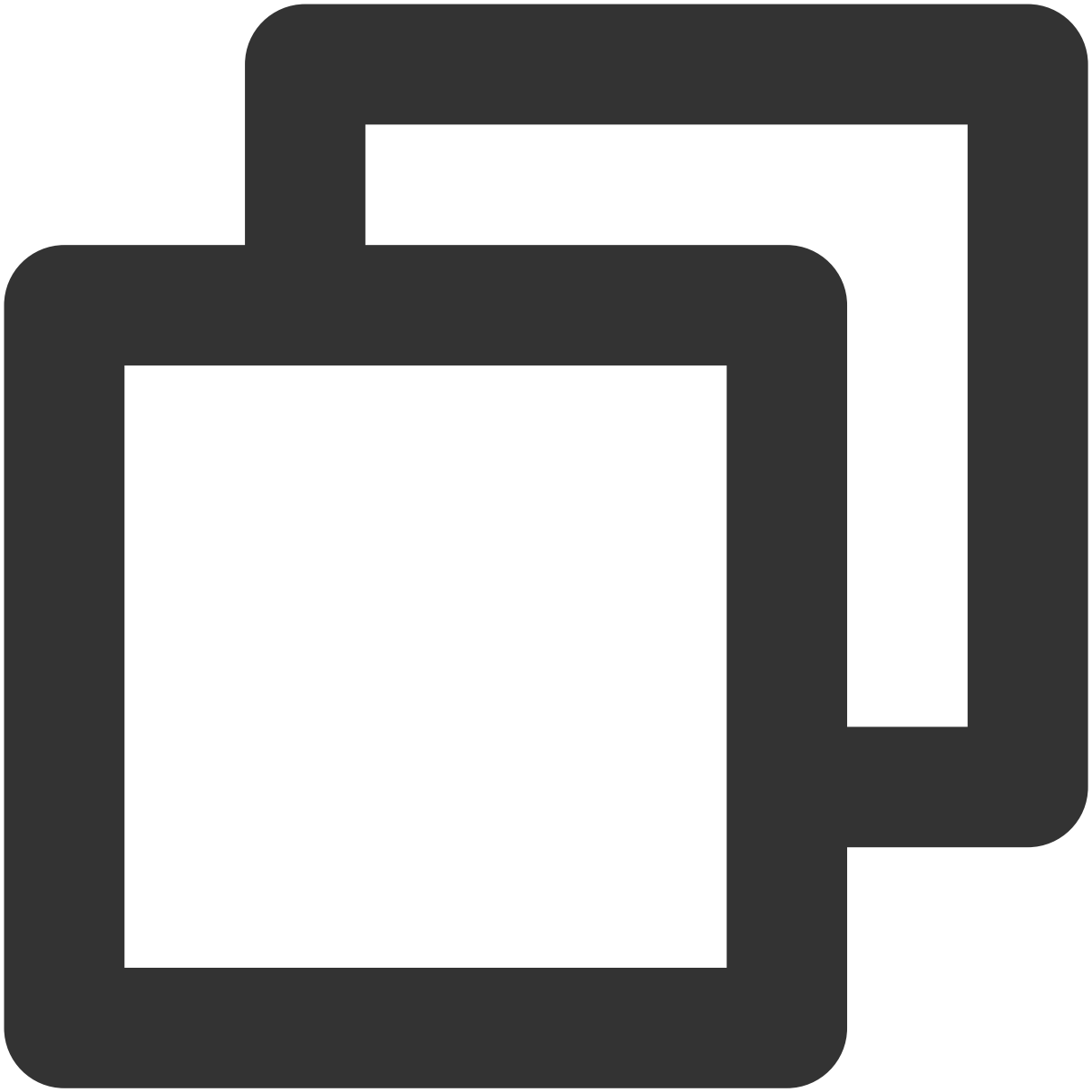
参数

参数	含义

int srcTextureId	需要被渲染的纹理。类型为：OpenGL 2D 纹理格式,像素格式为 RGBA。
id int srcTextureWidth	需要被渲染的纹理宽。
int srcTextureHeight	需要被渲染的纹理高。
Bitmap bitmap	建议最大尺寸 2160×4096。超过这个尺寸的图片人脸识别效果不佳或无法识别到人脸，同时容易引起 OOM 问题，建议把大图缩小后再传入。
boolean needReset	切换图片。 首次使用分割。 首次使用动效。 首次使用美妆。 这几种场景 needReset 设置为 true。

onPauseAudio

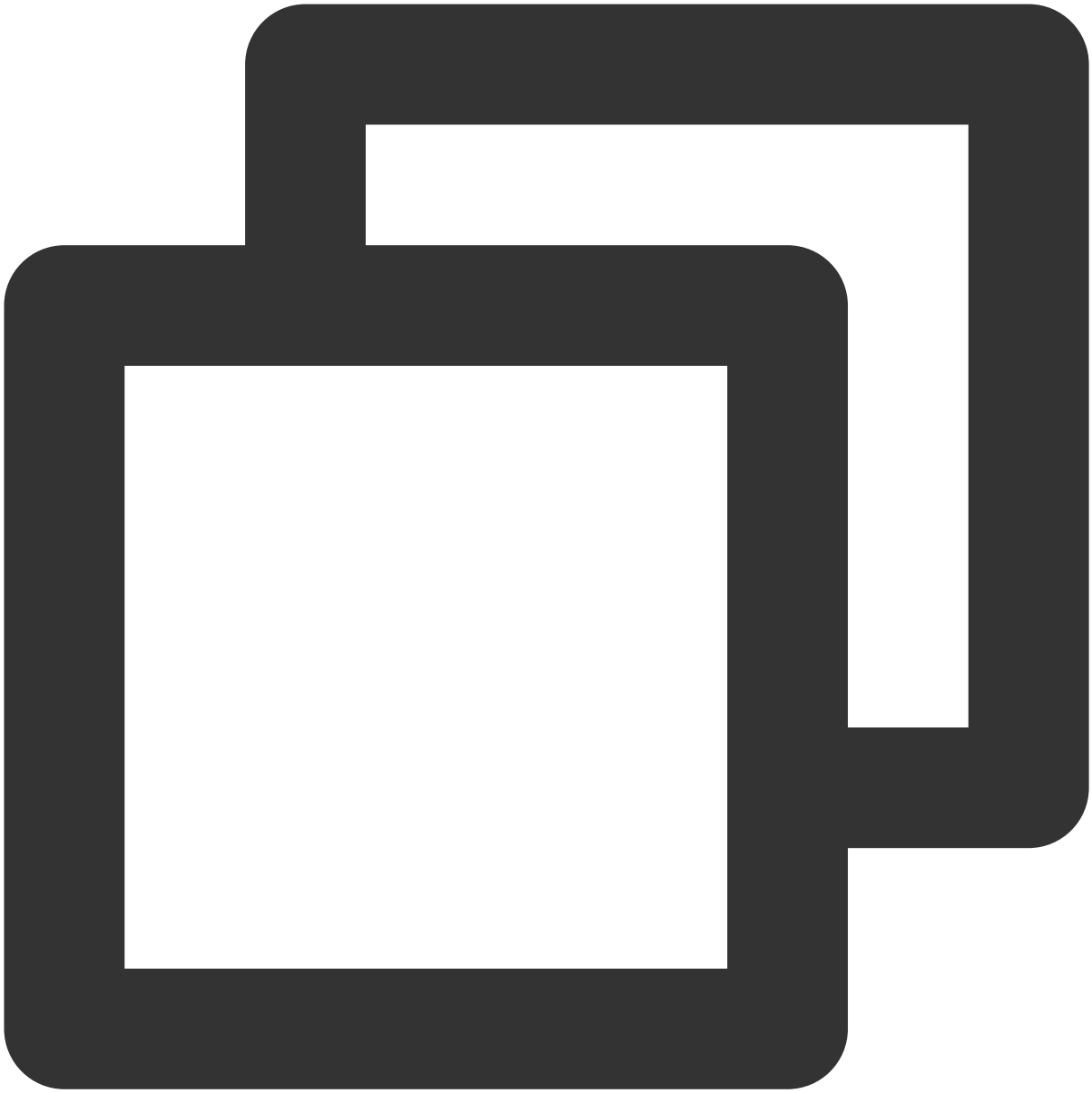
当仅需要停止音频，但不需要释放 GL 线程时调用此函数。



```
void onPauseAudio()
```

sensorChanged

用于判断当前手机旋转的角度，从而调整 AI 识别人脸的判断角度依据，需在陀螺仪传感器回调函数内调用。



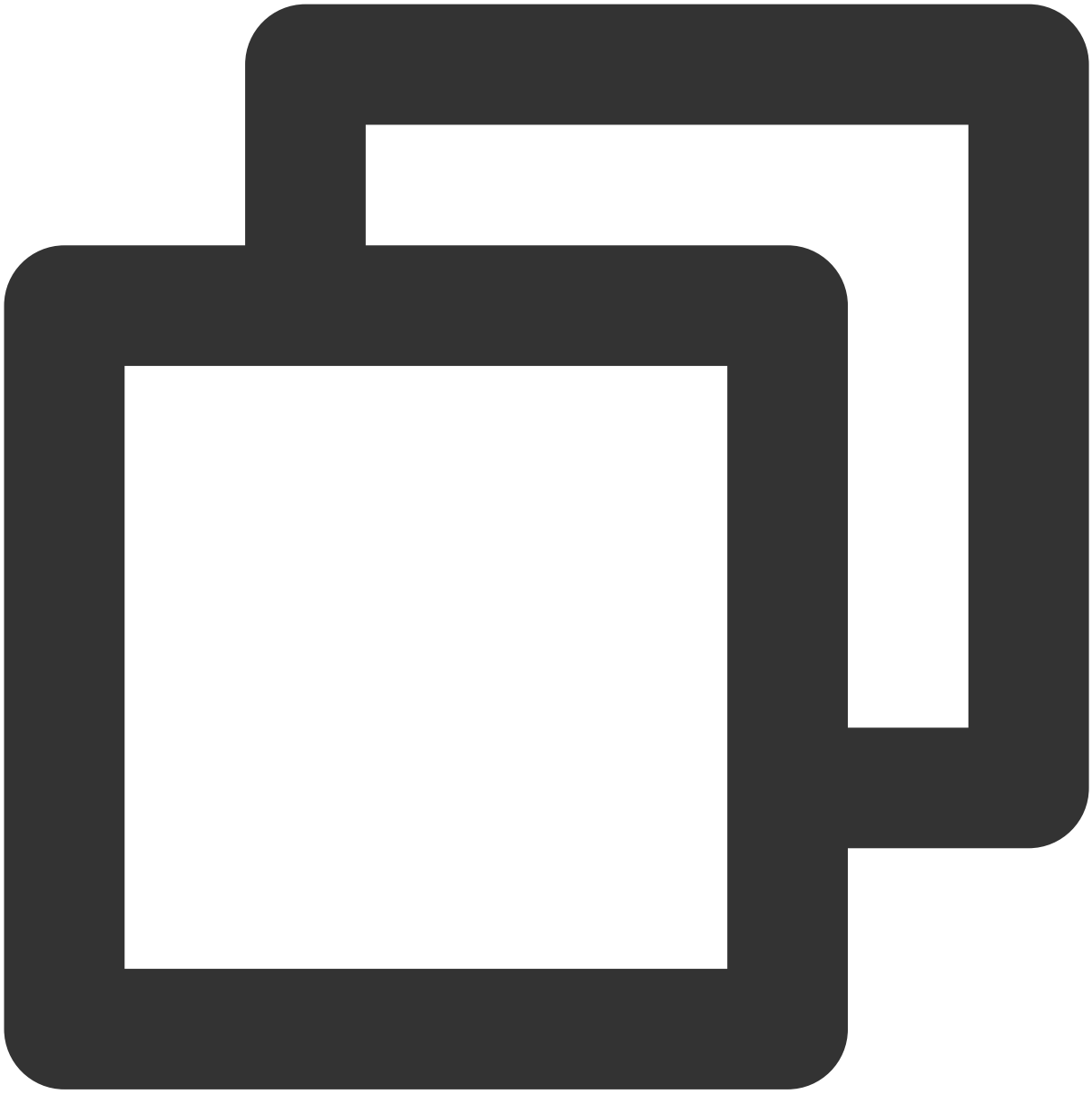
```
void sensorChanged(SensorEvent event, Sensor accelerometer)
```

参数

参数	含义
SensorEvent event	陀螺仪传感器回调函数 `onSensorChanged` 返回的事件实体类。
Sensor accelerometer	陀螺仪传感器示例。

isDeviceSupport

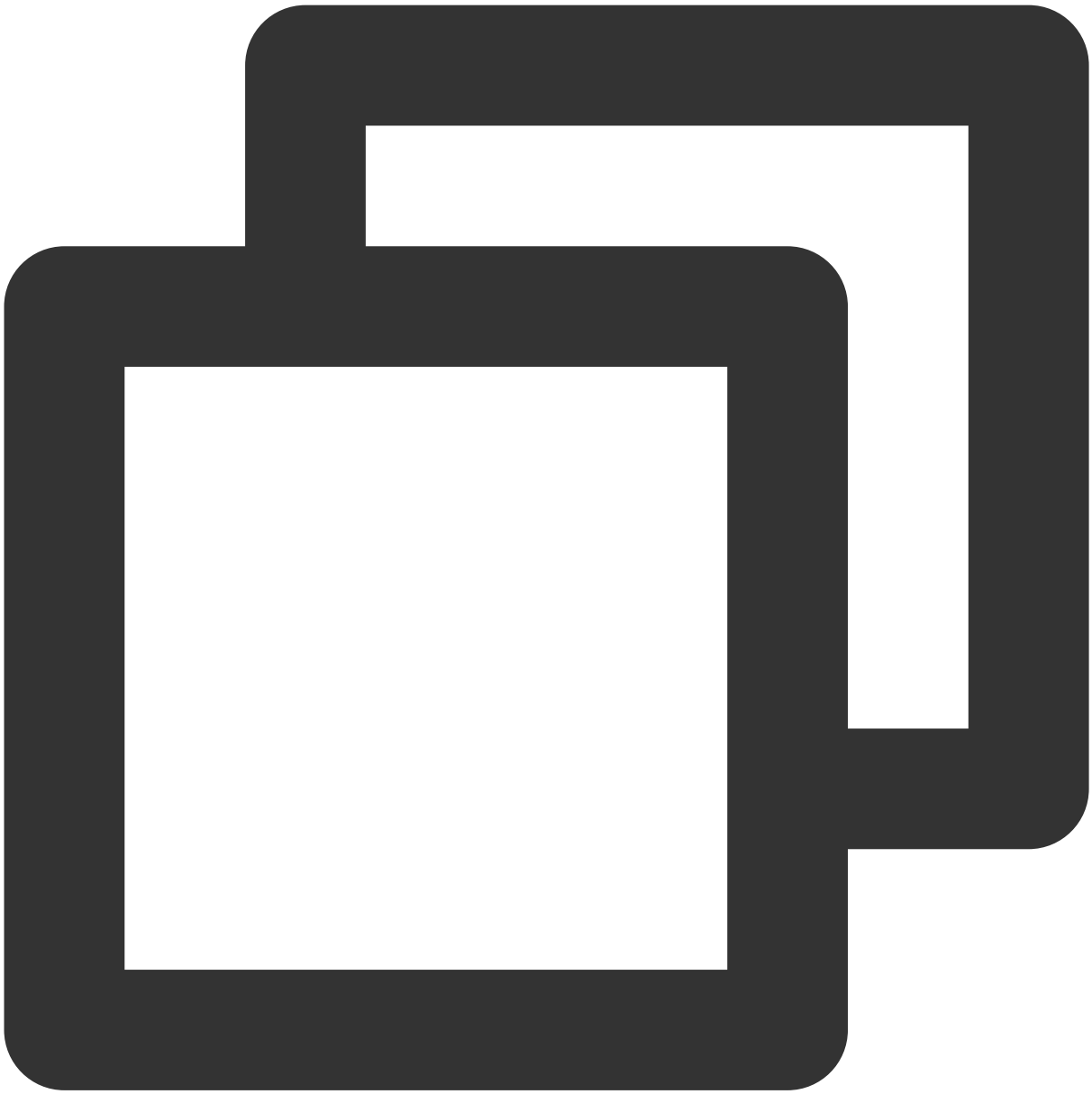
3.5.0版本及以后。



```
/**
 * 检测当前设备是否支持此素材
 * @param motionResPath 素材文件的路径
 * @return true 表示支持, false 表示不支持
 */
boolean isDeviceSupport(String motionResPath)
```

3.3.0版本及之前。

将动效资源列表传入 SDK 中做检测，执行后 `XmagicProperty.isSupport` 字段标识该素材是否可用。根据 `XmagicProperty.isSupport` 可 UI 层控制单击限制，或者直接从资源列表删除。



```
void isDeviceSupport(List<XmagicProperty<?>> assetsList)
```

参数

参数	含义
List<XmagicProperty<?>> assetsList	需要检测的动效素材列表。

getPropertyRequiredAbilities

传入一个动效资源列表，返回每一个资源所使用到的 SDK 原子能力列表。
方法的使用场景为：
您购买或制作了若干款动效素材，调用这个方法，会返回每一个素材需要使用的原子能力列表。例如素材1需要使用能力 A、B、C，素材2需要使用能力 B、C、D，然后您把这样的能力列表保持在服务器上。之后，当用户要从服务器下载动效素材时，用户先通过 getDeviceAbilities 方法获取他手机具备的原子能力列表（例如这台手机具备能力 A、B、C，但不具备能力 D），把他的能力列表传给服务器，服务器判断该设备不具备能力 D，因此不给该用户下发素材2。

参数

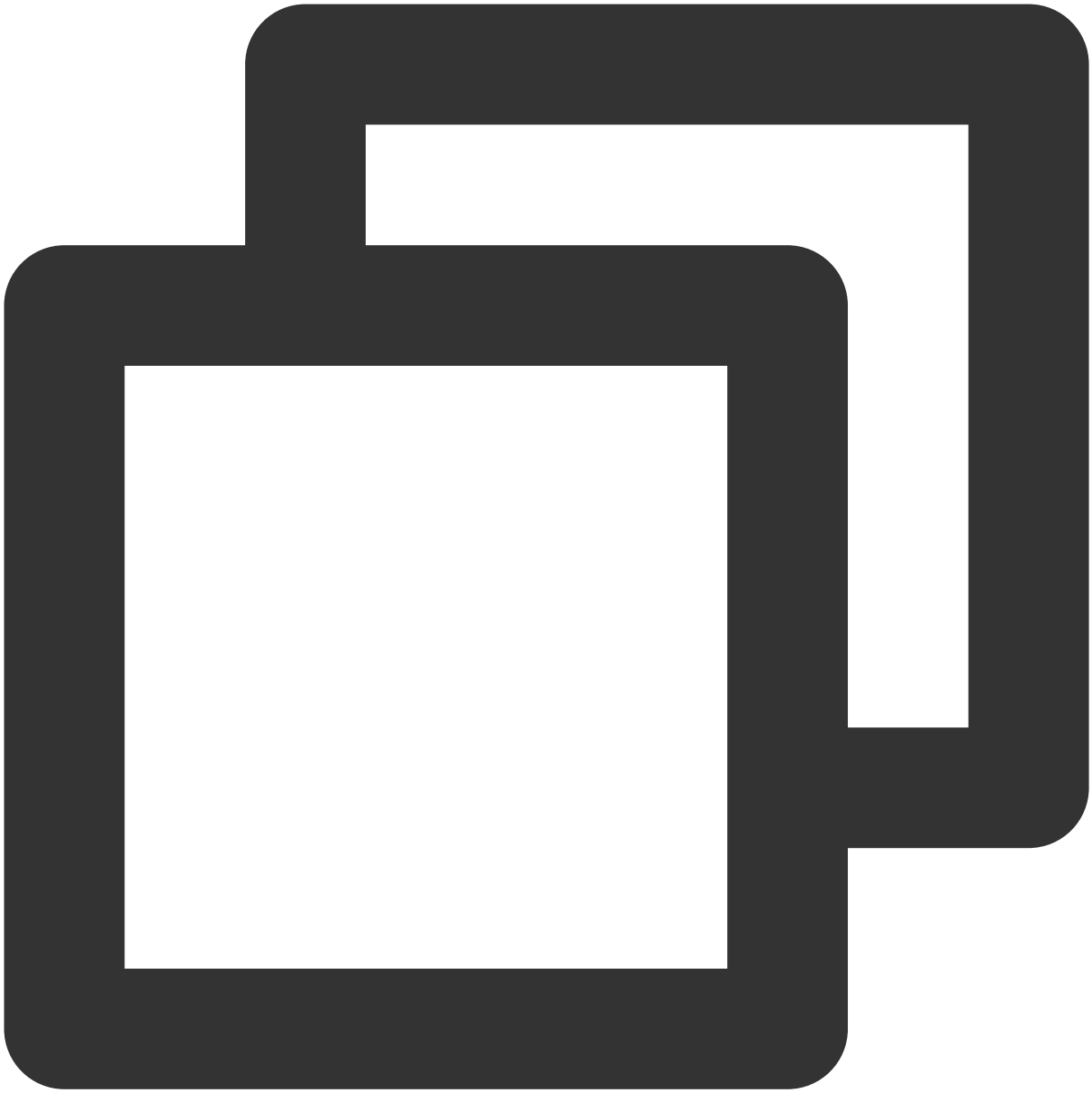
参数	含义
List<XmagicProperty<?>> assets	需要检测原子能力的动效资源列表。

返回

返回值 `Map<XmagicProperty<?>,ArrayList<String>>` ：
key：动效资源素材实体类。
value：所使用到的原子能力列表。

getDeviceAbilities

返回当前设备支持的原子能力表。与 getPropertyRequiredAbilities 方法搭配使用，详见 getPropertyRequiredAbilities 的说明。



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

返回

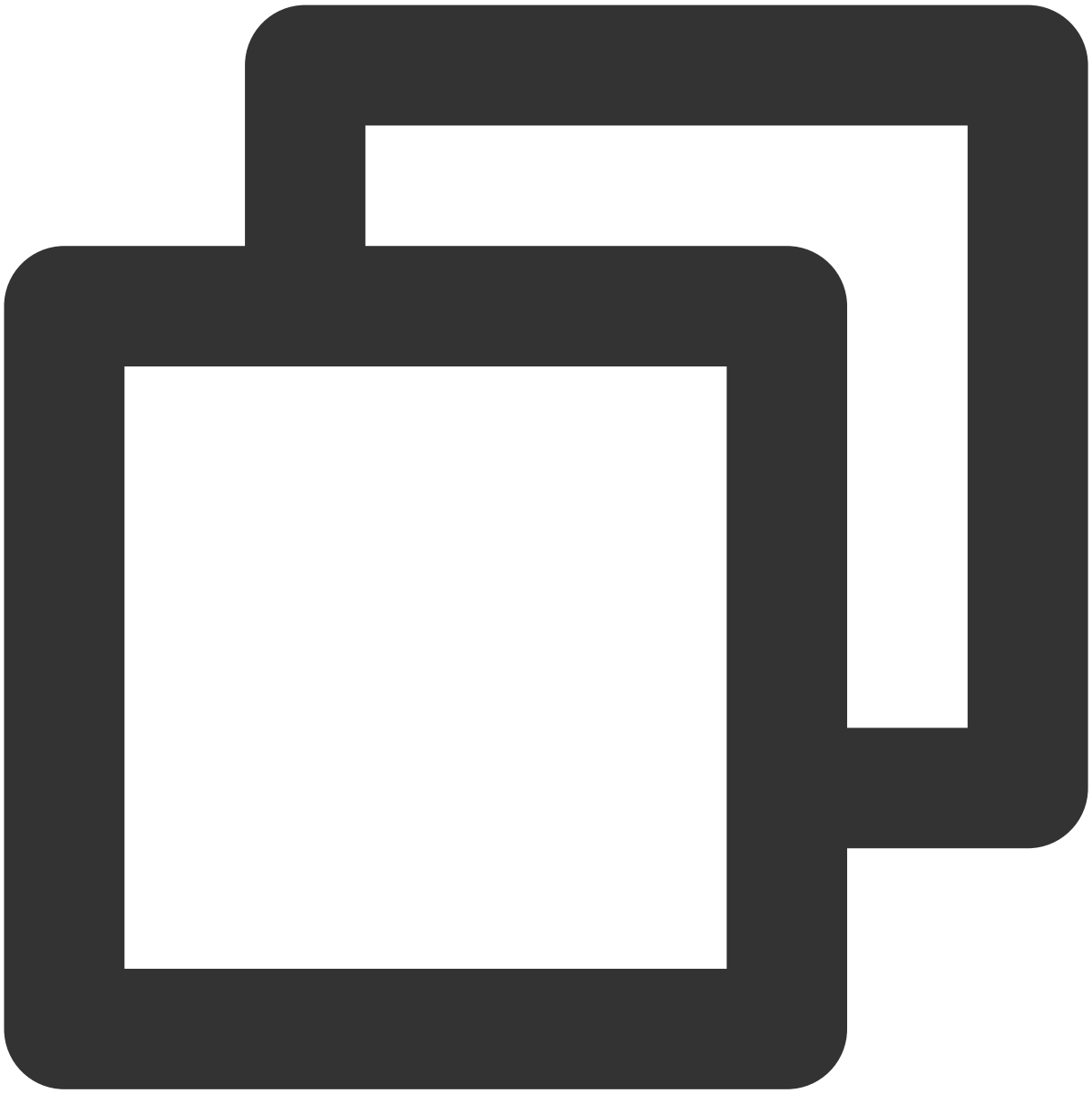
返回值 `Map<String,Boolean>` ：

key：原子能力名（与素材能力名字对应）。

value：当前设备是否支持。

isSupportBeauty

判断当前机型是否支持美颜（OpenGL3.0）。



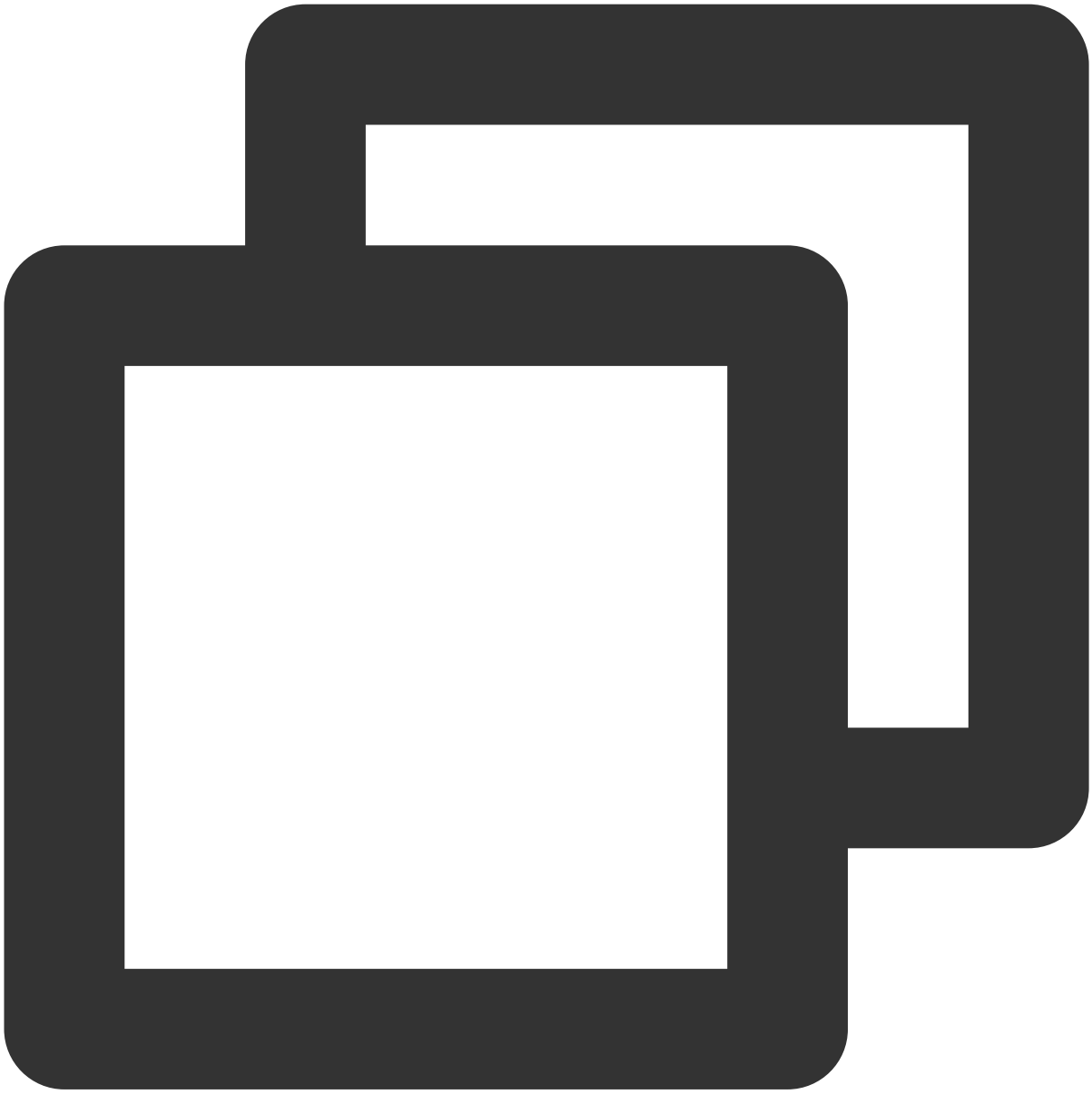
```
boolean isSupportBeauty()
```

返回

返回值 boolean：是否支持美颜。

isBeautyAuthorized

判断当前的 License 授权支持哪些美颜或美体项。 仅支持 BEAUTY 和 BODY_BEAUTY 类型的美颜项检测。检测后的结果会赋值到各个美颜对象 `XmagicProperty.isAuth` 字段中。如果 isAuth 字段为 false，可以在 UI 上屏蔽这些项的入口。



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

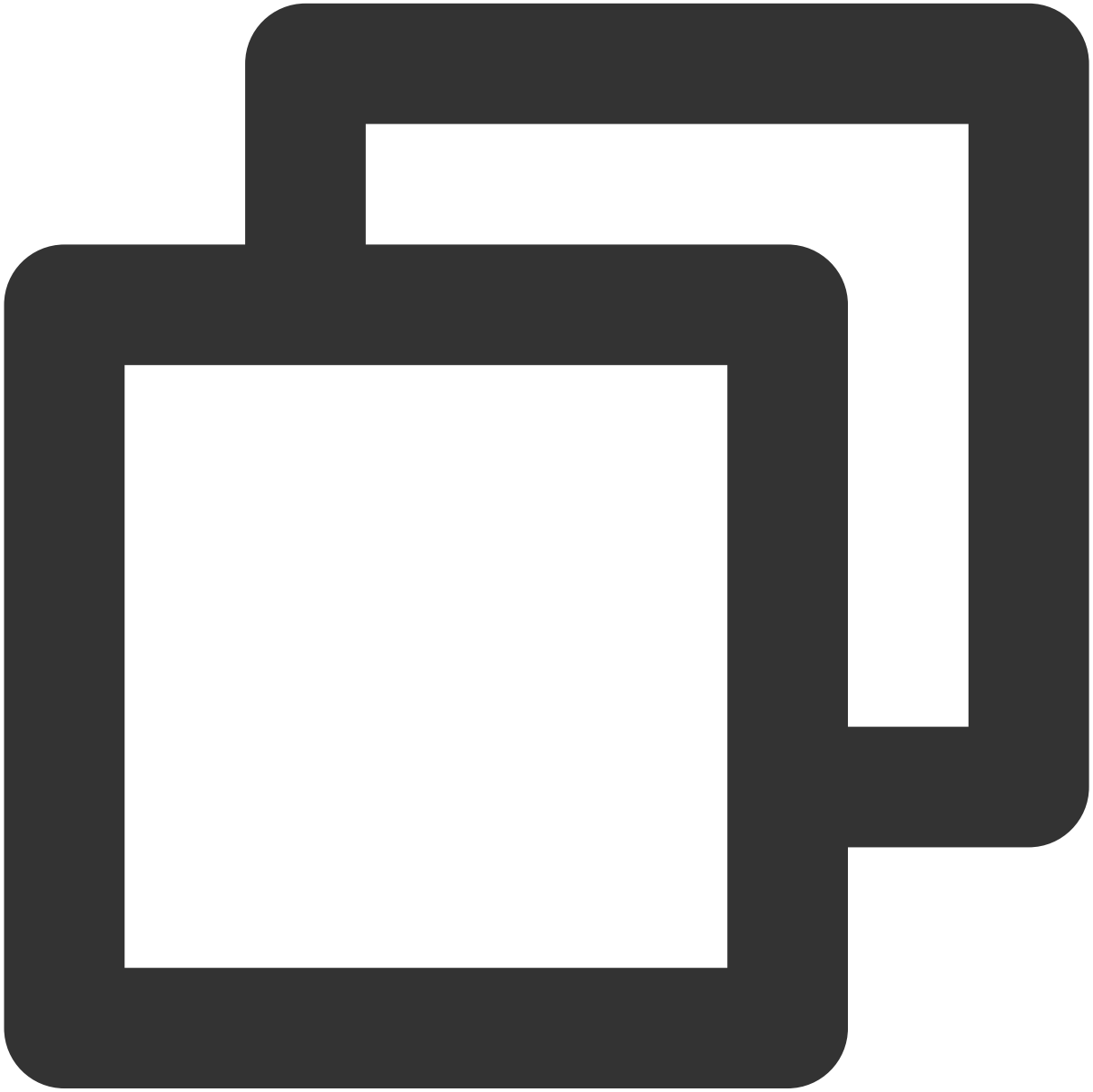
参数

参数	含义

List<XmagicProperty<?>> properties	需要检测的美颜项。

setXmagicStreamType

设置输入数据类型，默认 Android camera 数据流（XmagicApi.PROCESS_TYPE_CAMERA_STREAM）。



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

参数

参数	含义
int type	数据源类型，有以下两种选择： `XmagicApi.PROCESS_TYPE_CAMERA_STREAM` ：相机数据源。 `XmagicApi.PROCESS_TYPE_PICTURE_DATA` ：图片数据源。

setXmagicLogLevel

设置 SDK 的 log 等级，默认为WARN。开发调试阶段如有需要，可以将其设为 `Log.DEBUG`。正式发布时务必设置为 `Log.WARN` 或 `Log.ERROR` ，否则大量的日志会影响性能。

在 new XmagicApi() 之后调用。

setAudioMute

动效素材使用时是否开启静音（V2.5.0新增）：

参数：true 表示静音，false 表示非静音。

enableEnhancedMode

开启美颜增强模式（V2.5.1新增）。默认未开启。

未开启时，应用层可以设置的各美颜项的强度范围为 0到1 或 -1到1，如果超出此范围，SDK会取边界值。例如应用层设置瘦脸为1.2，SDK判断其超出了最大值1.0，则在内部把瘦脸值修正为1.0。

开启增强模式后，应用层可以设置更大范围的数值。例如想要瘦脸程度更大，则可以把瘦脸值设置为1.2，SDK会接受并使用1.2这个数值，不会将其修正为1.0。

开启增强模式后，需要应用层自己管理每个美颜项可以设置的最大值，让用户在此范围内调整数值。我们提供了一份参考值，您可以根据产品需求自由调整，但不建议超出我们的推荐值，否则美颜效果可能变差。参考值如下：

美颜项名称	增强模式下，建议的最大值（放大倍数）
美白，短脸，V脸，眼距，鼻子位置，祛法令纹，口红，立体	1.3
亮眼	1.5
腮红	1.8
其他	1.2

素材叠加（3.0.1.2新增）

如果想要某个动效/美妆/分割素材叠加在当前素材上，则将该素材 **XmagicProperty** 对象的 mergeWithCurrentMotion 设置为 true。XMagicProperty 对象的其他属性设置见 美颜参数设置。

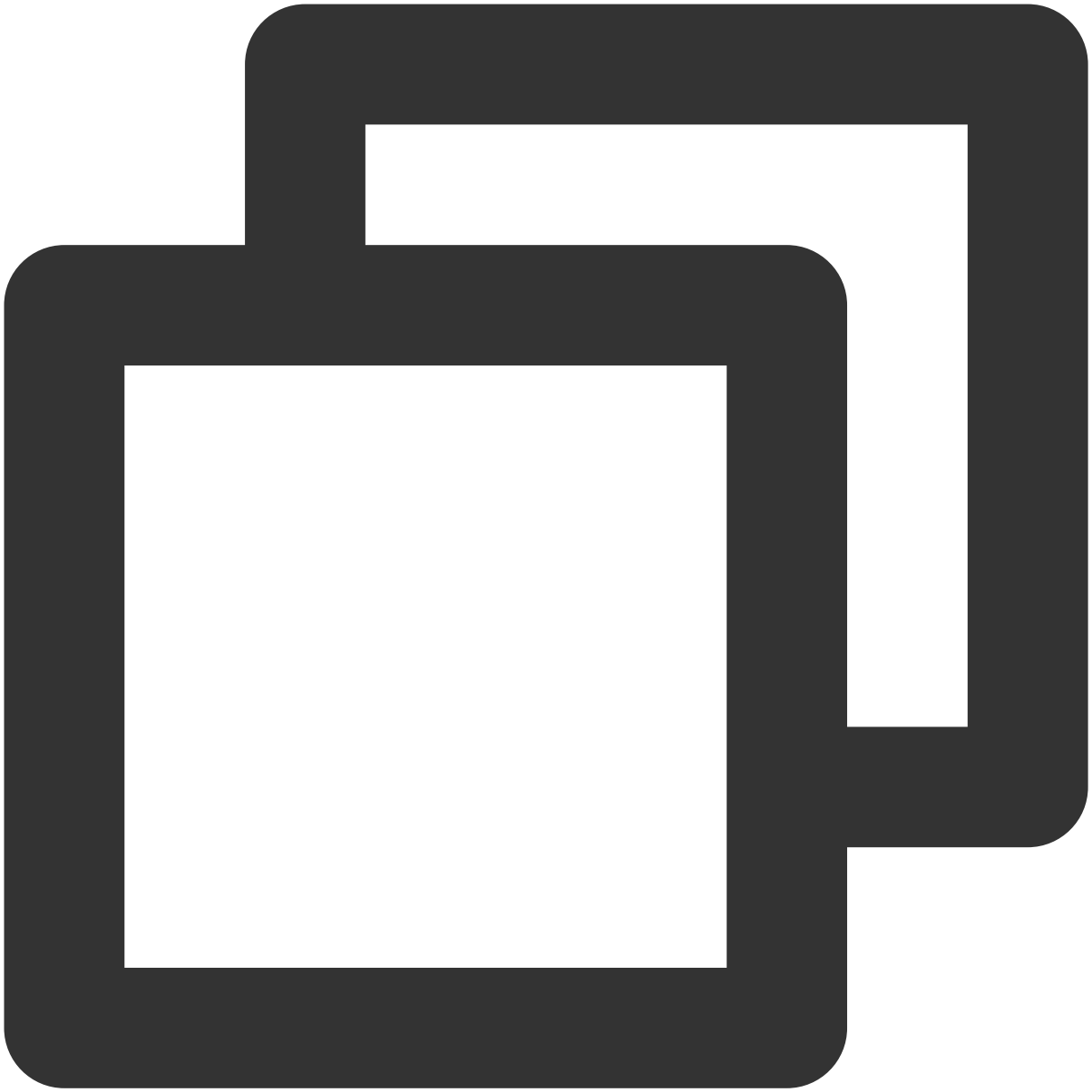
素材叠加注意事项：

1. 客户需要自行管理素材之间是否适合叠加。举两个例子：

例1：特效 A 是变成贵妃脸，特效 B 是变成童话脸，这两个特效叠加后可能会导致画面非常别扭。
例2：特效 A 是个兔耳朵，特效 B 是猪耳朵，两个叠加后，就有两种耳朵。
例1和例2这两种情况不适合叠加。如果特效 A 是兔耳朵，特效 B 是送一个飞吻，这两个特效不会冲突，就适合叠加。
2. 只支持简单素材之间的叠加。简单素材是指只有单动效能力、或者单美妆效果、或者单抠背等，复杂素材是指包含了多种效果。简单素材和复杂素材没有明确的界定，建议客户充分测试后，自行管理哪些素材之间可以叠加，哪些不能叠加。
3. 叠加时，有动作触发的特效（例如伸出手触发某个特效、微笑触发某个特效等）属于复杂特效，需要放在前面，简单特效放在后面叠加在它之上。
4. 使用示例：主播使用了特效 A，然后观众送礼物特效 B，B 要叠加在 A 之上，一段时间后 B 消失，恢复成特效 A。那么设置步骤如下：
4.1 设置特效 A，mergeWithCurrentMotion 设置为 false。
4.2 设置特效 B，mergeWithCurrentMotion 设置为 true。
4.3 一小段时间后，再设置 A，mergeWithCurrentMotion 设置为 false。

setDowngradePerformance（V3.1.0新增）

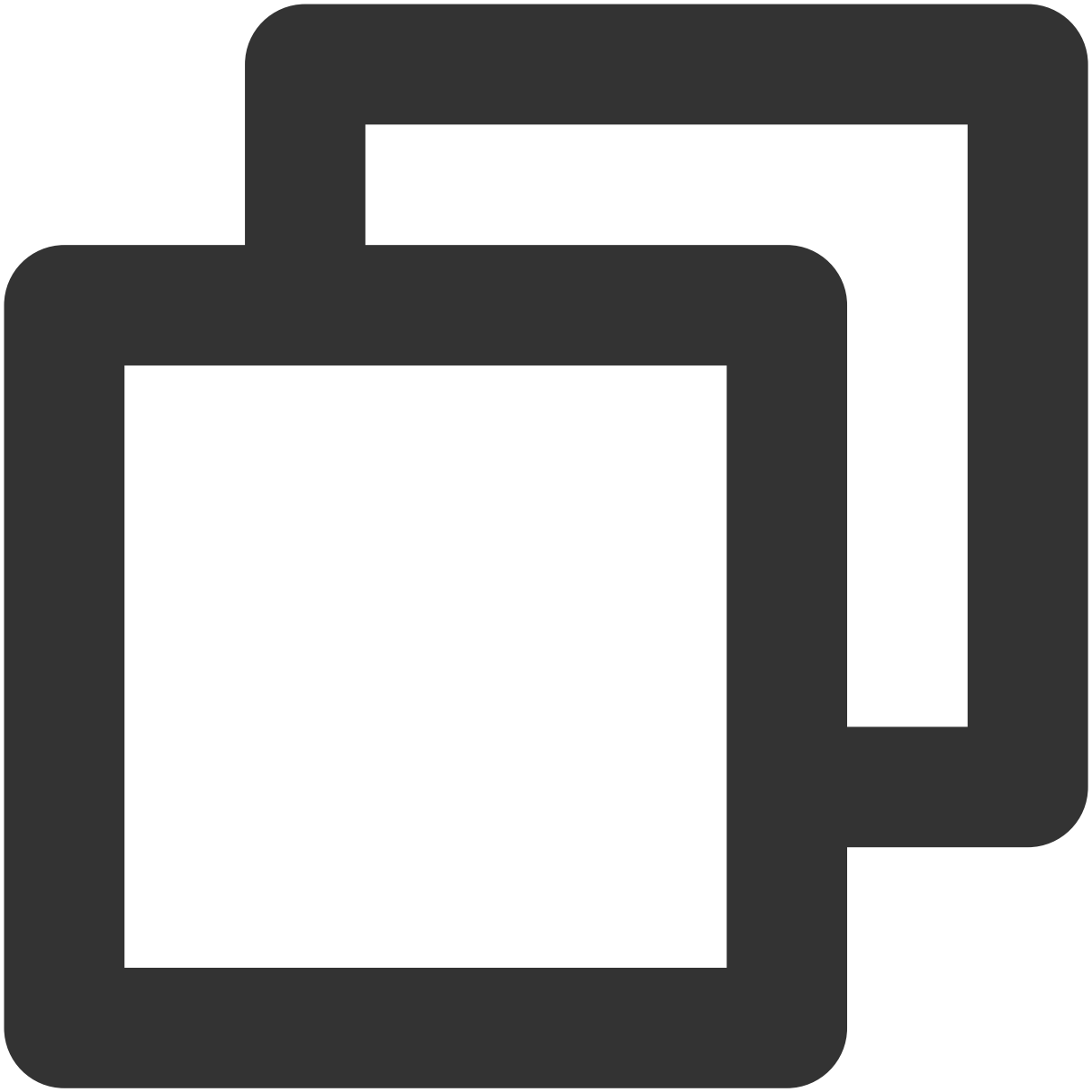
开启高性能模式后，美颜占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象，更适合低端机长时间使用。
注意：开启高性能模式后，以下美颜项将不可用：
1. 眼部：眼宽、眼高、祛眼袋 。
2. 眉毛：角度、距离、高度、长度、粗细、眉峰。
3. 嘴部：微笑唇。
4. 面部：瘦脸（自然，女神，英俊），收下颌，祛皱、祛法令纹。建议用“脸型”实现综合大眼瘦脸效果。



```
void boolean setDowngradePerformance()
```

exportCurrentTexture

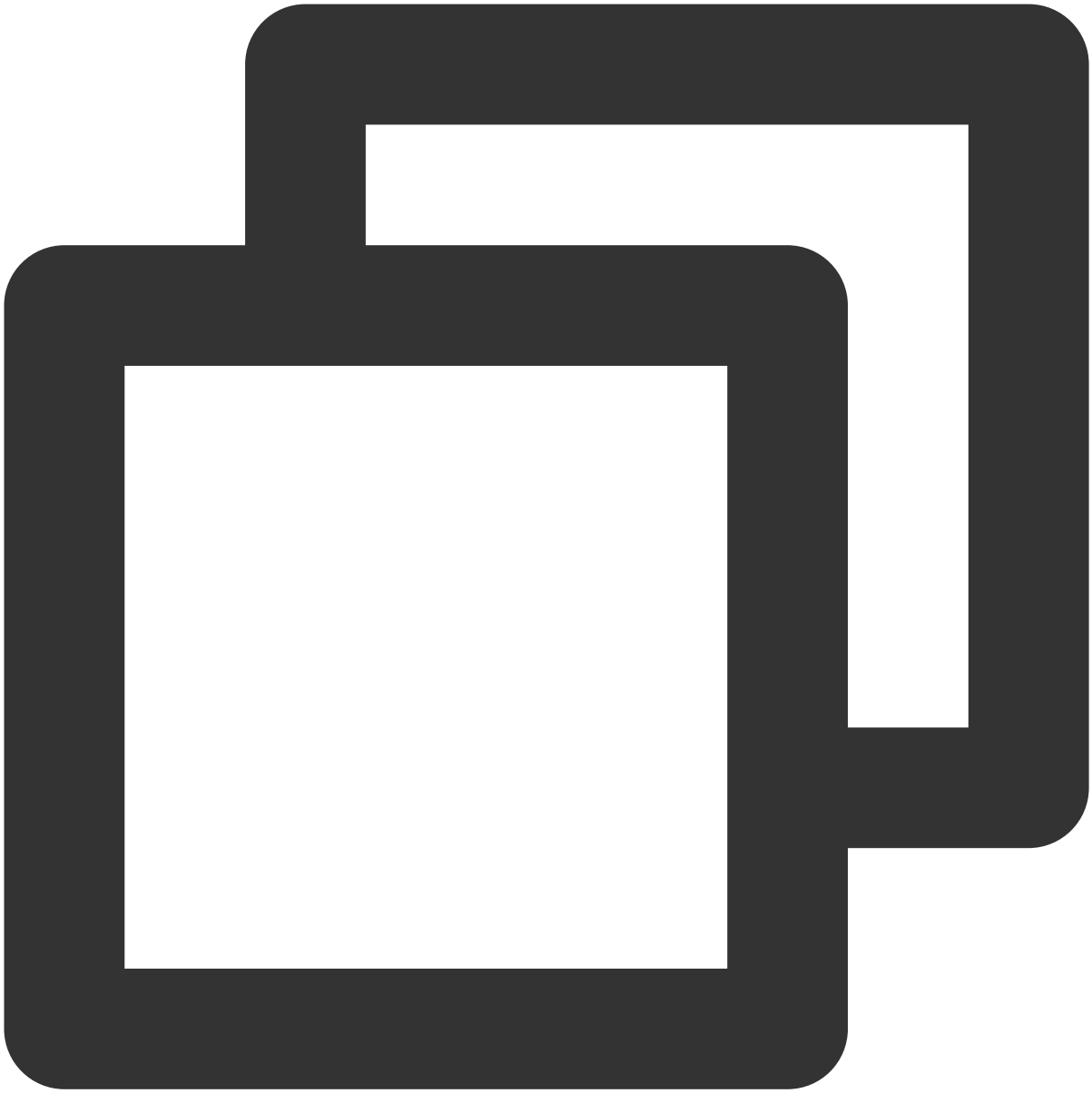
获取当前纹理上的画面



```
void exportCurrentTexture(ExportTextureCallback callback)
```

setFeatureEnableDisable

开启或关闭某个能力



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

参数

参数	含义
String featureName	原子能力名称 取值如下： `XmagicConstant.FeatureName.ANIMOJI_52_EXPRESSION` 人脸表情能力 `XmagicConstant.FeatureName.BODY_3D_POINT` 身体点位能力

	`XmagicConstant.FeatureName.HAND_DETECT` 手势检测能力 `XmagicConstant.FeatureName.WHITEN_ONLY_SKIN_AREA` 美白仅对皮肤生效 `XmagicConstant.FeatureName.SEGMENTATION_SKIN` 皮肤分割能力 `XmagicConstant.FeatureName.SMART_BEAUTY` 智能美颜(为男性、宝宝减淡美颜美妆效果)
boolean enable	true 表示开启此能力，false 表示关闭此能力 **注：如果是降级模式，则不允许开启皮肤分割**

静态属性和方法

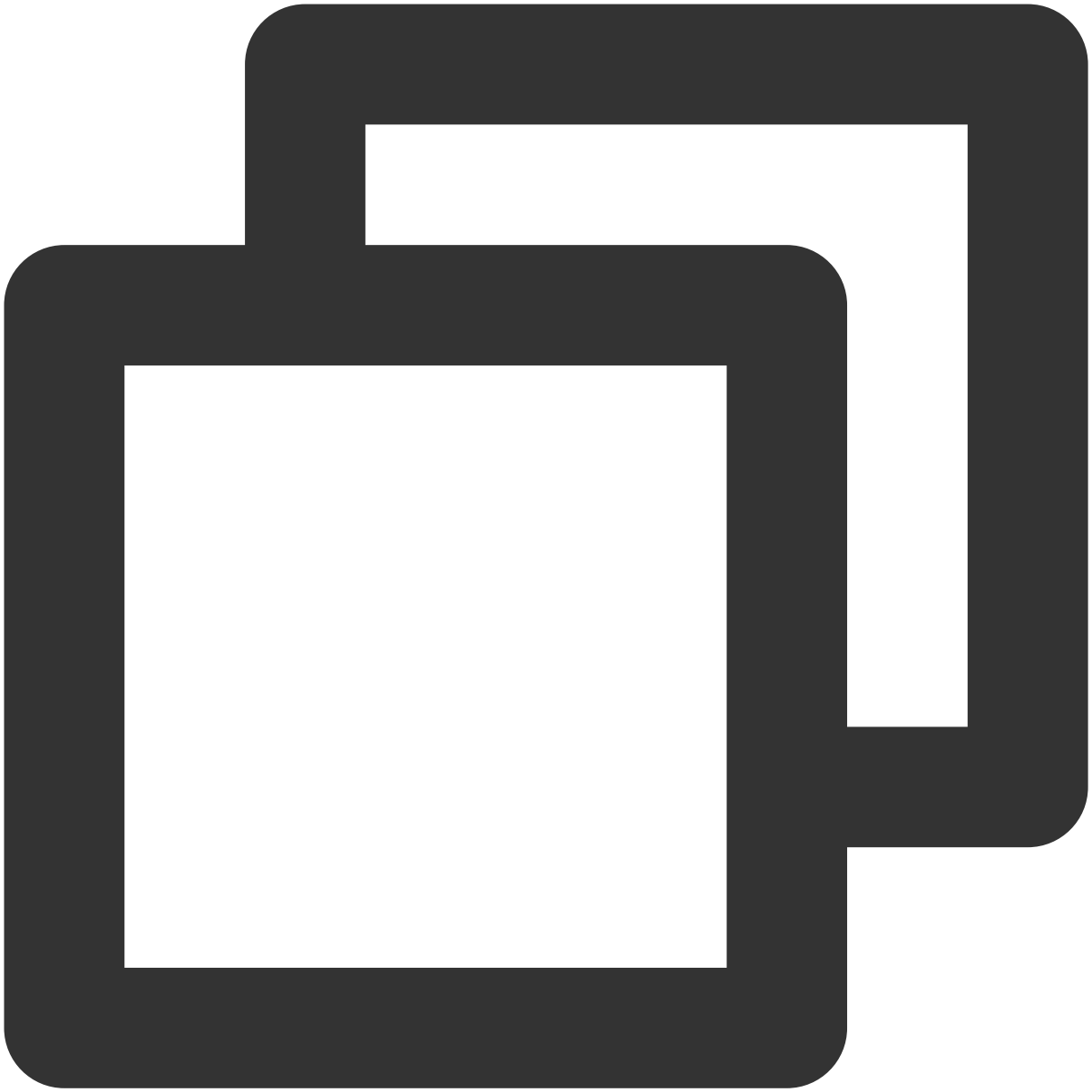
API	描述
VERSION	通过XmagicApi.VERSION可获取SDK的版本号（V**3.5.0新增**）
setLibPathAndLoad	设置 libPath。
addAiModeFilesFromAssets	将应用程序 assets 下 Light3DPlugin、LightCore、LightHandPlugin、LightBodyPlugin、LightSegmentPlugin 文件夹中的内容复制到指定目录中。
addAiModeFiles	将客户下载好的 AI 模型文件复制到对应的文件夹下。

setLibPathAndLoad

设置 so 的路径，并触发加载。如果 so 是内置在 assets 里的，则无需调用此方法。如果 so 是动态下载的，则需要在鉴权和 `new XmagicApi` 之前调用。

传入 null ：表示从默认路径加载 so，请确保 so 是内置在 APK 包里的。

传入非 null：如 `data/data/包名/files/xmagic_libs` ，将从这个目录去加载 so。



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

参数

参数	含义
String path	so 库存放路径。

addAiModeFilesFromAssets

将应用程序 assets 下 Light3DPlugin、LightCore、LightHandPlugin、LightBodyPlugin、LightSegmentPlugin 文件夹中的内容复制到指定目录中。

context 应用上下文。

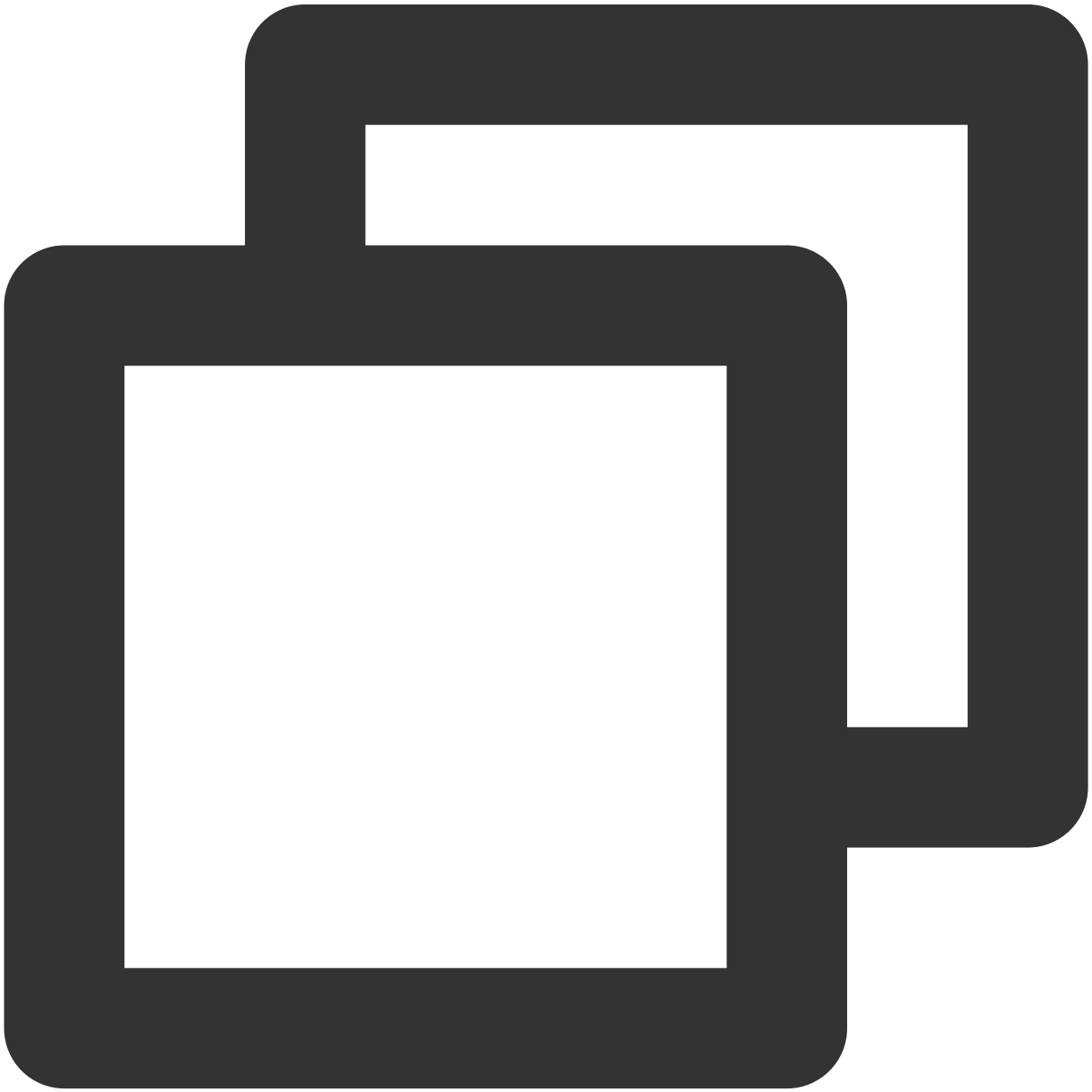
resDir 用于存放美颜资源的根目录，此目录和创建 xmagicApi 对象时传入的路径一致。

返回值：

0：复制成功

-1：表示 context 为 null

-2：表示 IO 错误



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

参数

参数	含义
Context context	应用上下文。
String resDir	用于存放美颜资源的根目录，此目录和创建 xmagicApi 对象时传入的路径一致。

addAiModeFiles

将客户下载好的 AI 模型文件复制到对应的文件夹下。

inputResDir 下载成功的模型文件的文件夹。

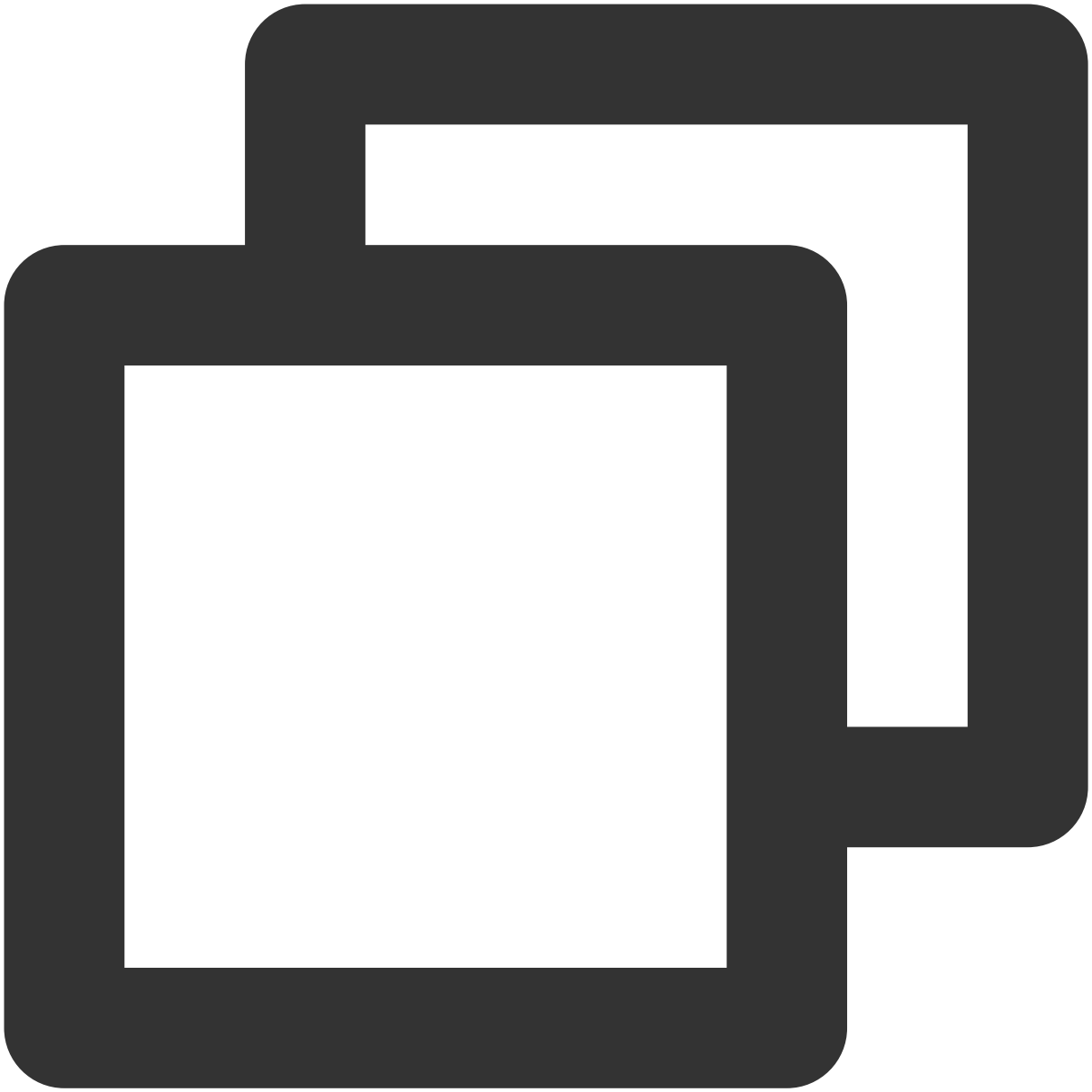
resDir 用于存放美颜资源的根目录，此目录和创建 xmagicApi 对象时传入的路径一致。

返回值：

0：表示成功

-1：inputResDir is not exists

-2:表示 IO 错误



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

参数

参数	含义
String inputResDir	下载成功的模型文件的文件夹。
String resDir	用于存放美颜资源的根目录，此目录和创建 xmagicApi 对象时传入的路径一致。

Flutter

最近更新时间：2024-02-29 17:38:46

腾讯特效 SDK Flutter版本核心接口类 `TencentEffectApi` ，更新美颜数值、调用动效等功能。

Public 成员函数

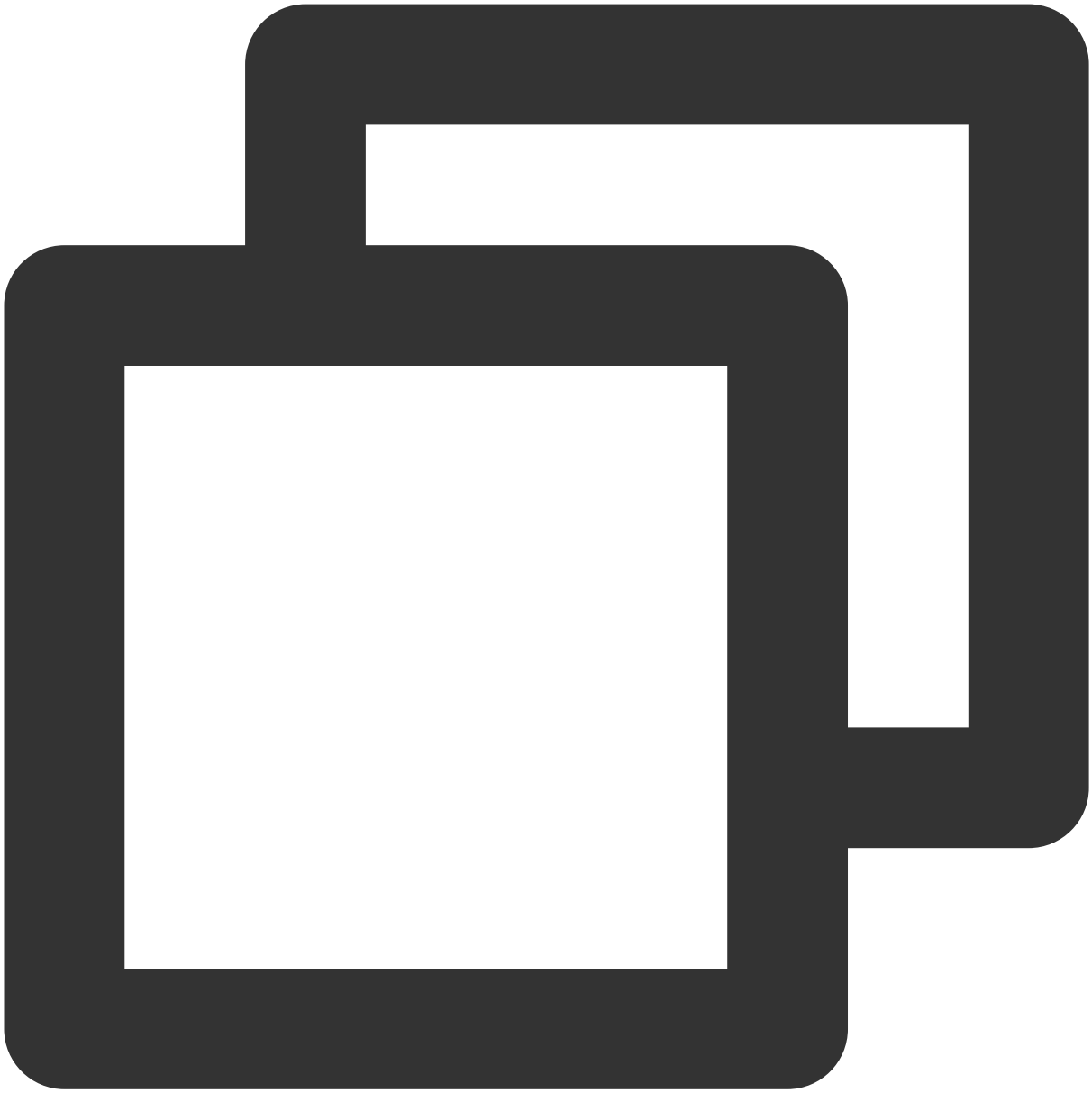
API	描述
setResourcePath	设置美颜资源的本地存储路径（**V0.3.5.0**版本新增）
initXmagic	初始化美颜数据，使用美颜前必须先调用此方法**(在V0.3.1.1版本及之前)**
setLicense	进行美颜授权
setXmagicLogLevel	设置 SDK 的 log 等级，建议开发调试时设为 Log.DEBUG，正式发布时设置为 Log.WARN，如果正式发布设置为 Log.DEBUG，大量的日志会影响性能
onResume	恢复渲染，页面可见时调用
onPause	暂停渲染，页面不可见时调用
~~enableEnhancedMode~~	开启美颜增强模式
setDowngradePerformance	调用此方法开启高性能模式。高性能模式开启后，美颜占用的系统 CPU/GPU 资源更少，可减少手机的发热和卡顿现象，更适合低端机长时间使用。**注意：需要在调用其他方法之前调用**
setAudioMute	设置静音（因为有些贴纸中有声音）
setFeatureEnableDisable	开启或关闭某个特性
~~updateProperty~~	更新美颜属性， 可在任意线程调用
setEffect	更新美颜属性（**V0.3.5.0**版本新增）
setOnCreateXmagicApiErrorListener	设置创建美颜对象时的回调接口（如果出错会回调此接口）
setTipsListener	设置动效提示语回调函数，用于将提示语展示到前端页面上
setYTDataListener	设置人脸点位信息等数据回调，需要获得人脸点位的 Licence 授权（例如原子能力X102）才会有回调。

setAIDataListener	设置人脸、手势、身体检测状态回调
~~isBeautyAuthorized~~	判断当前的 lic 授权支持哪些美颜。 仅支持 BEAUTY 和 BODY_BEAUTY 类型的美颜项检测。检测后的结果会赋值到各个美颜对象 XmagicProperty.isAuth 字段中
isSupportBeauty	判断当前机型是否支持美颜（OpenGL3.0）
getDeviceAbilities	返回当前设备支持的原子能力表
~~isDeviceSupport~~	将动效资源列表传入 SDK 中做检测，执行后 XmagicProperty.isSupport 字段标识该原子能力是否可用。 根据 XmagicProperty.isSupport 可 UI 层控制单击限制，或者直接从资源列表删除
isDeviceSupportMotion	检测当前设备是否支持此素材
~~getPropertyRequiredAbilities~~	传入一个动效资源列表，返回每一个资源所使用到的 SDK 原子能力列表

成员函数说明

setResourcePath （V0.3.5.0版本新增）

设置美颜资源存放的本地路径



```
///设置美颜资源存放的本地路径，使用美颜前必须先调用此方法。
///v0.3.5.0新增
void setResourcePath(String xmagicResDir);
```

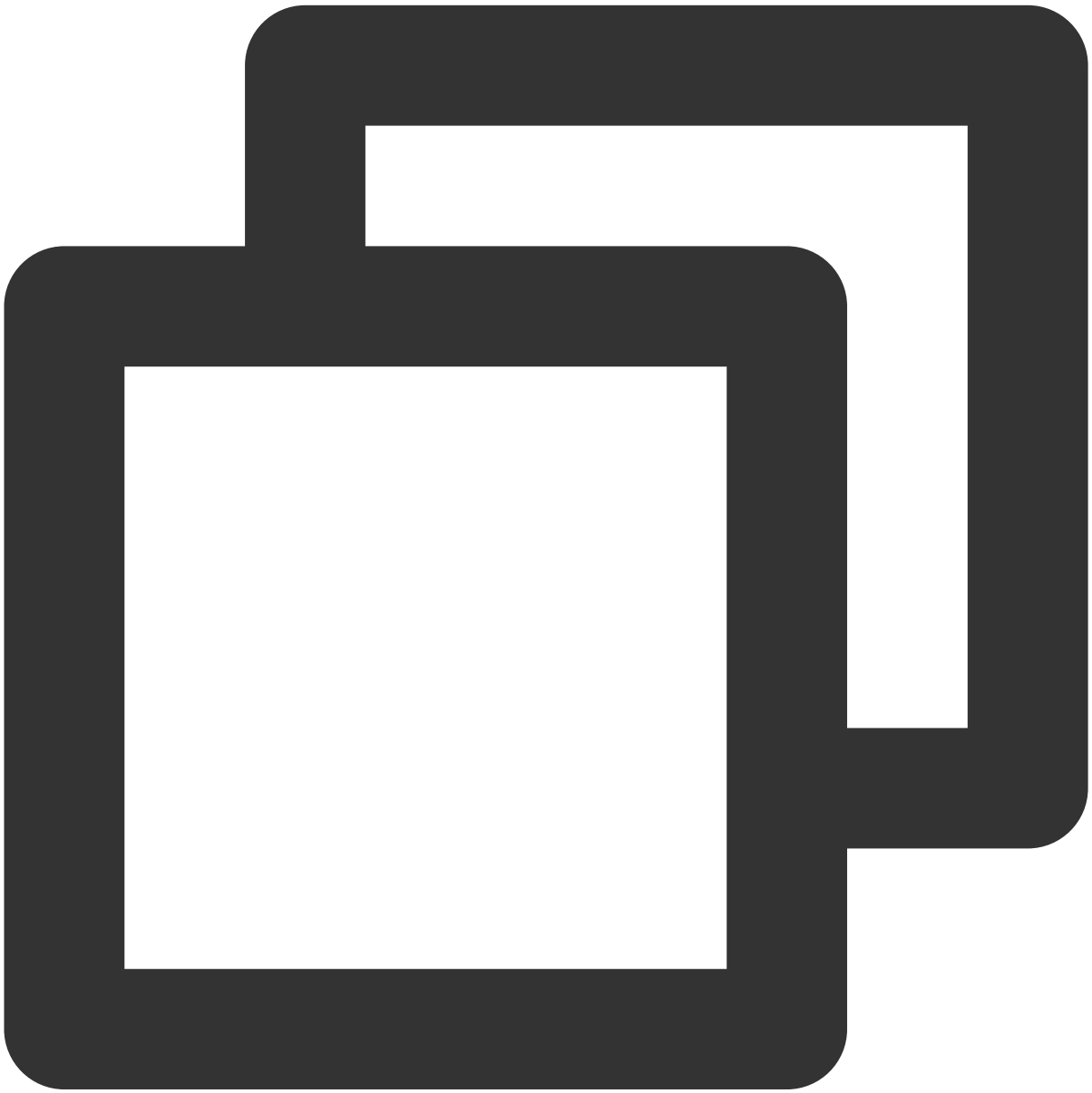
参数

参数	含义
String xmagicResDir	资源文件放置的目录

initXmagic

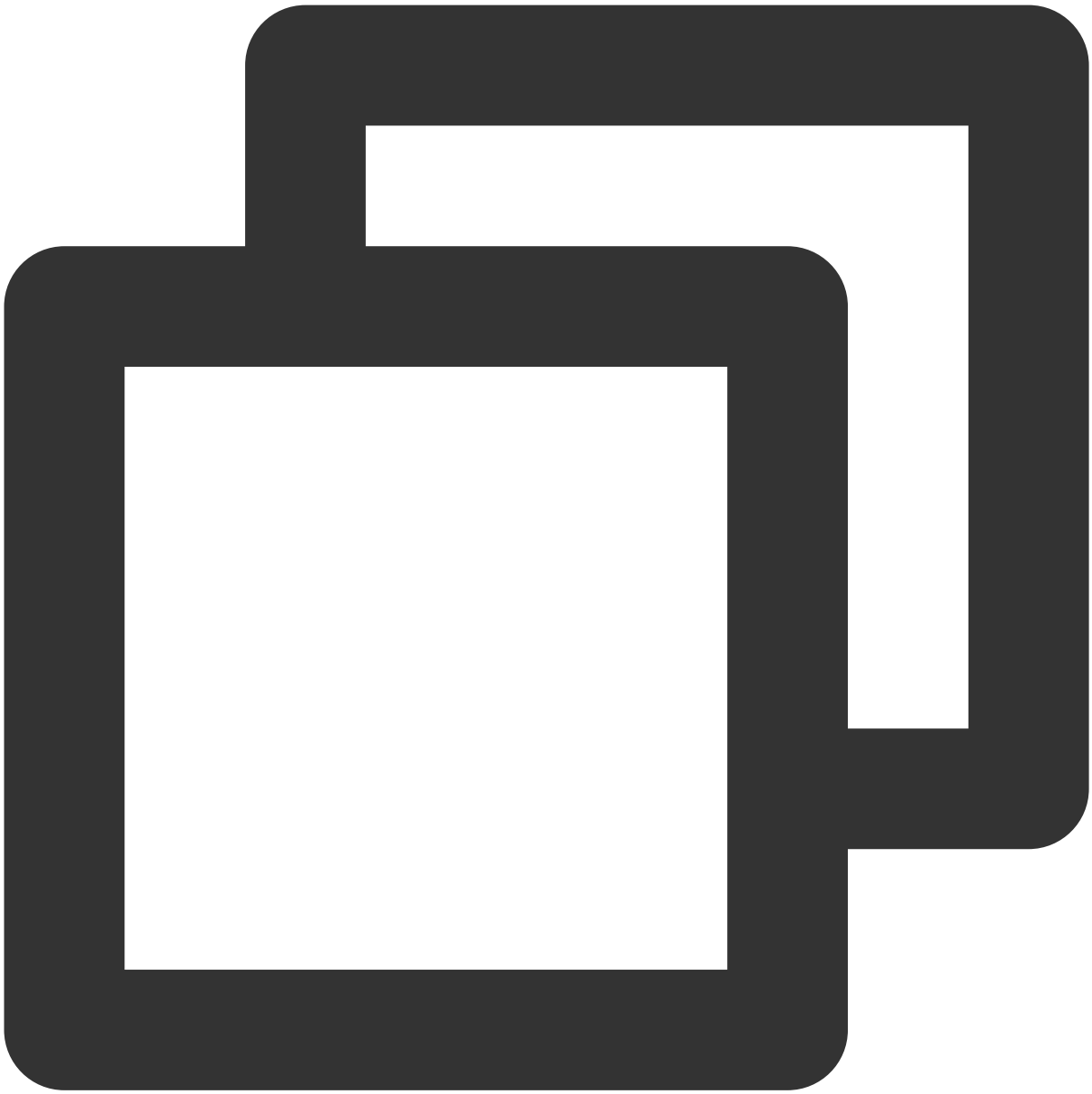
初始化美颜数据。在**V0.3.1.1**版本及之前，使用美颜前必须先调用此方法。在**V0.3.5.0**版本，此方法每个版本只需要调用一次，并且在调用了此方法之前必须先调用 `setResourcePath` 方法设置了资源路径，在**V0.3.5.0**版本中删除了之前的 xmagicResDir 参数，可参考最新 demo。

V0.3.5.0版本



```
void initXmagic(InitXmagicCallBack callBack);

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

V0.3.1.1版本及之前



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

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

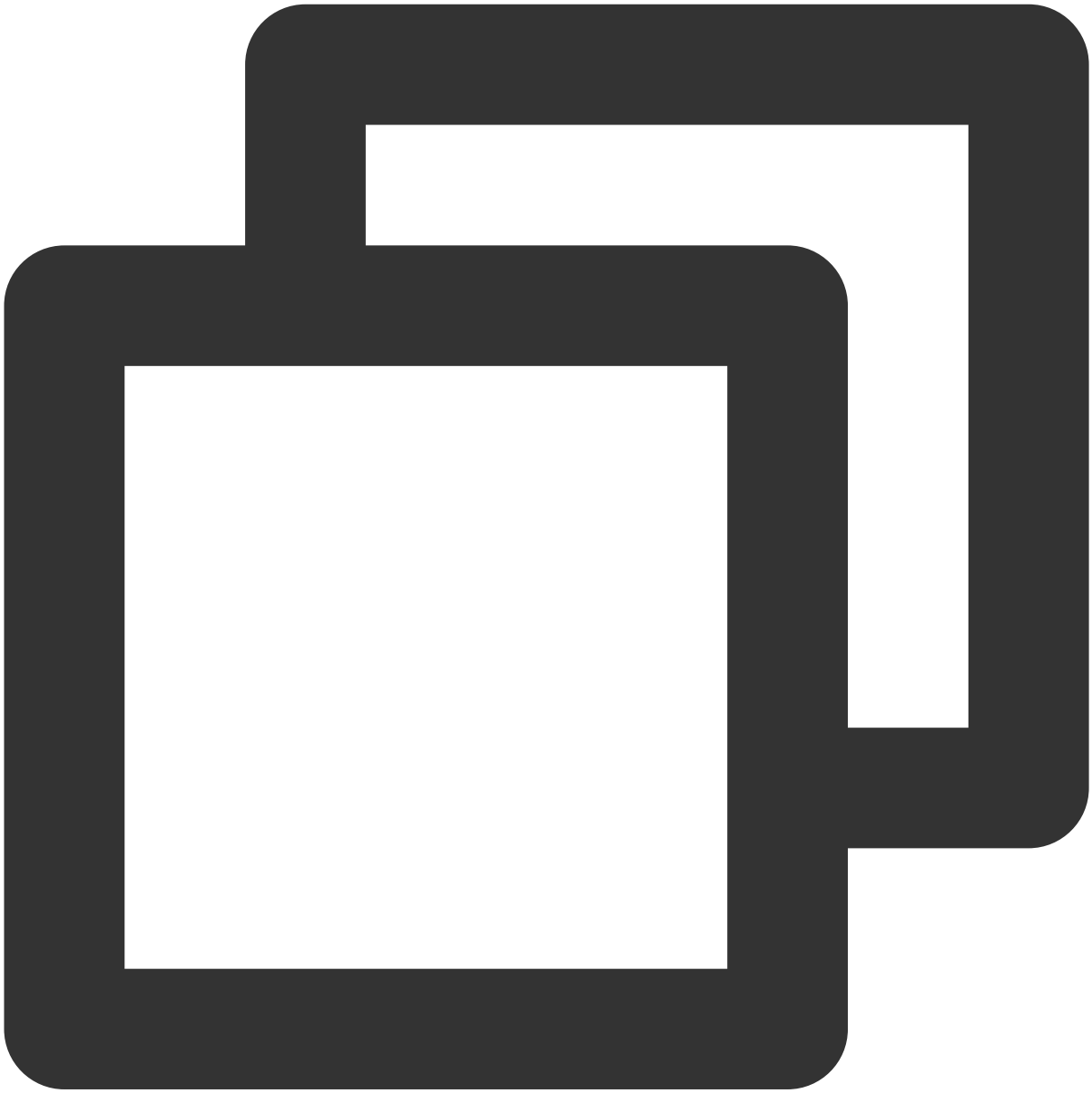
参数

参数	含义
String xmagicResDir	资源文件放置的目录

InitXmagicCallBack callBack	初始化回调接口

setLicense

设置鉴权数据，进行美颜授权。



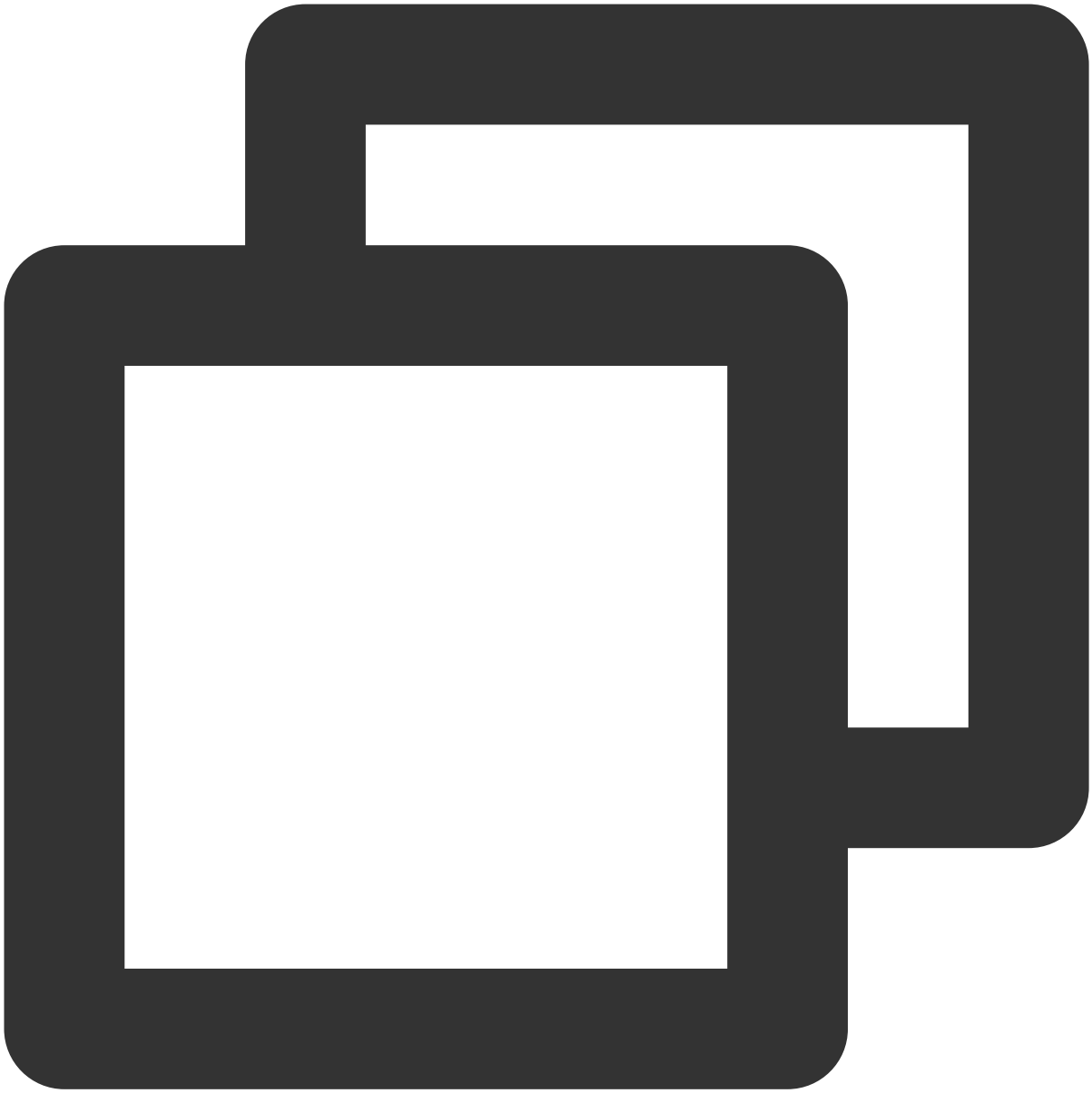
```
  ///美颜进行鉴权处理
void setLicense(String licenseKey, String licenseUrl, LicenseCheckListener checkLis
///授权校验的结果回调方法
typedef LicenseCheckListener = void Function(int errorCode, String msg);
```

参数

参数	含义
String licenseKey	鉴权的 LicenseKey
String licenseUrl	鉴权的 LicenseUrl
LicenseCheckListener checkListener	授权结果回调接口

setXmagicLogLevel

设置 SDK 的 log 等级



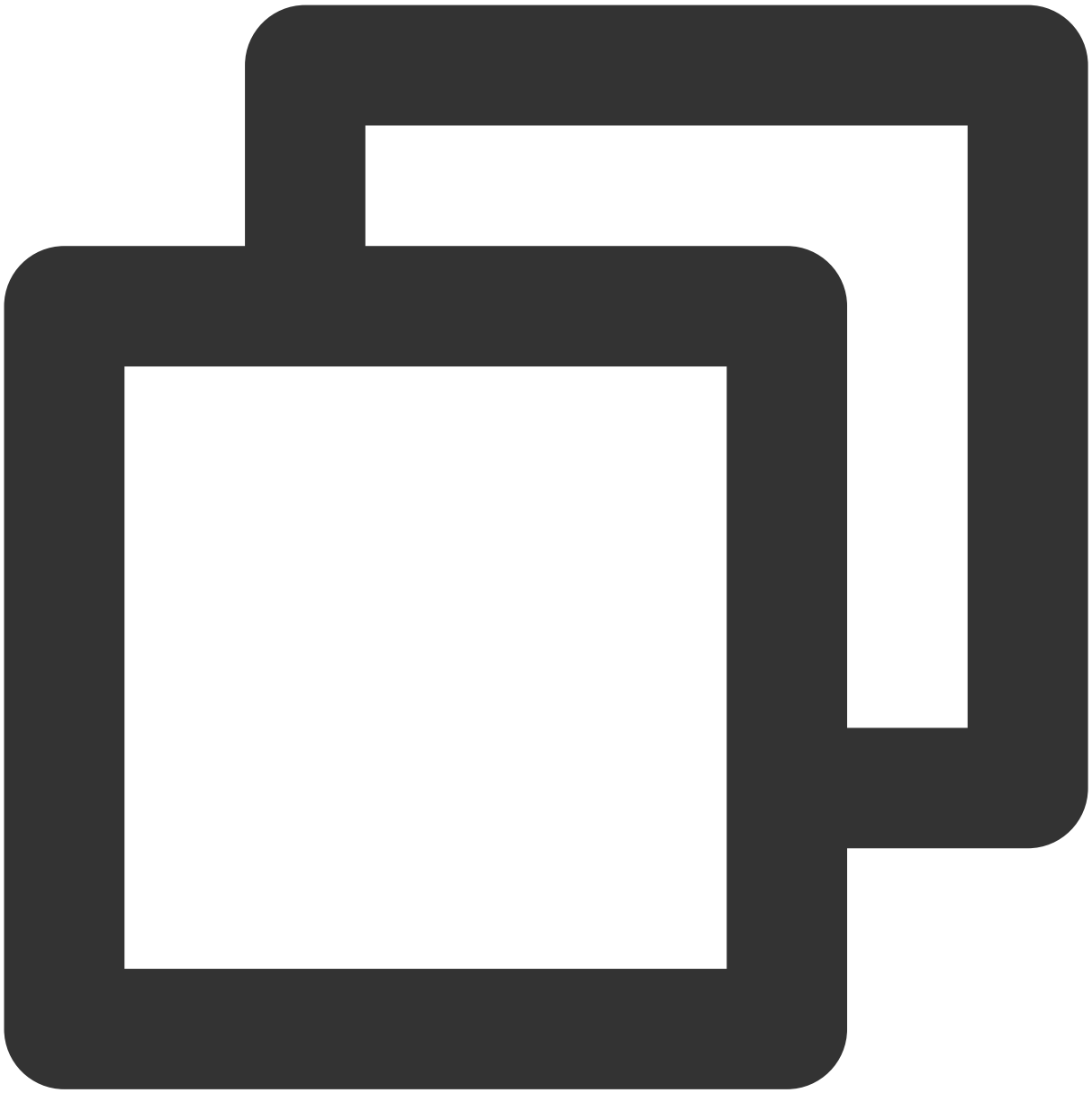
```
void setXmagicLogLevel(int logLevel);
```

参数

参数	含义
int logLevel	可使用 LogLevel 定义好的类型进行设置

onResume

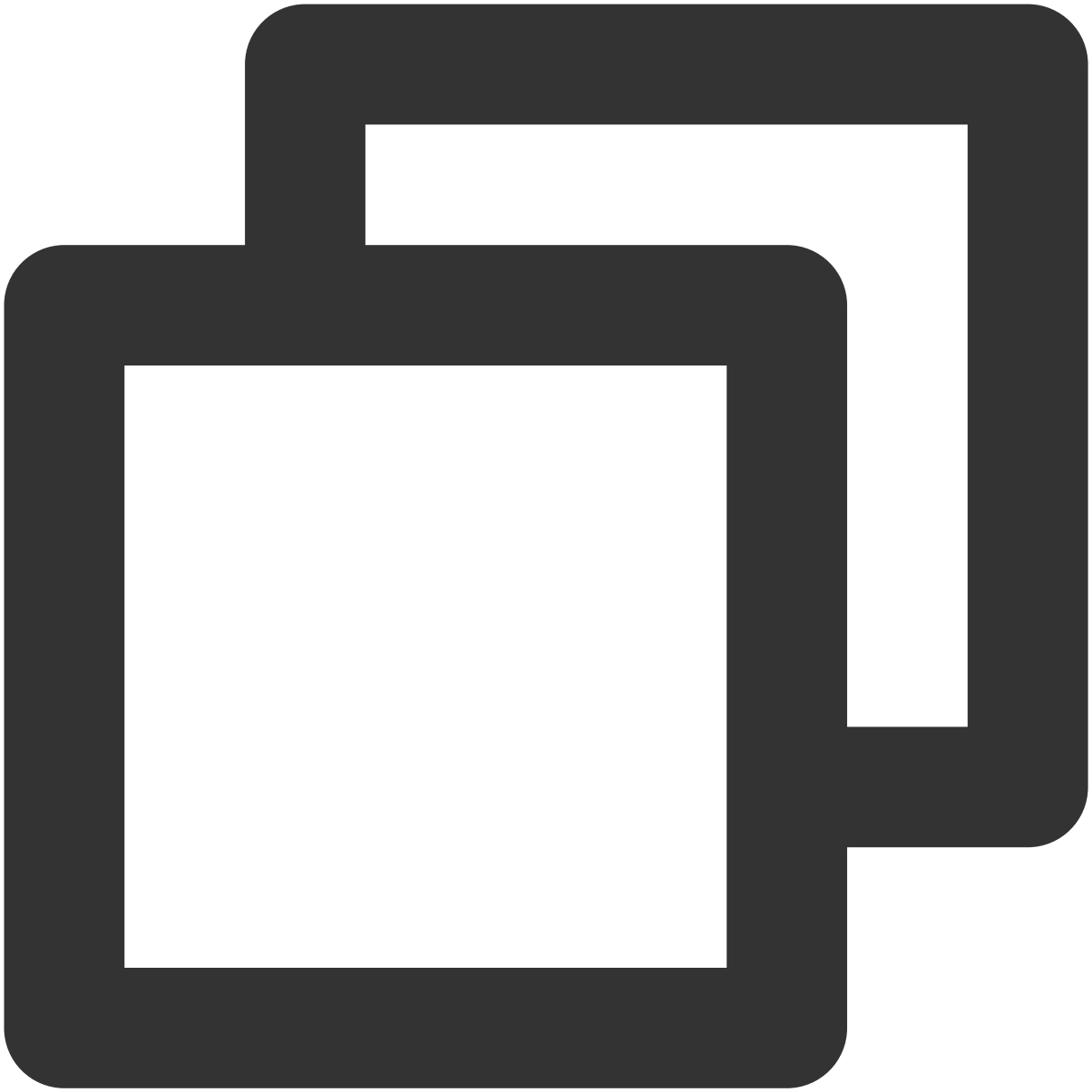
恢复美颜处理



```
void onResume();
```

onPause

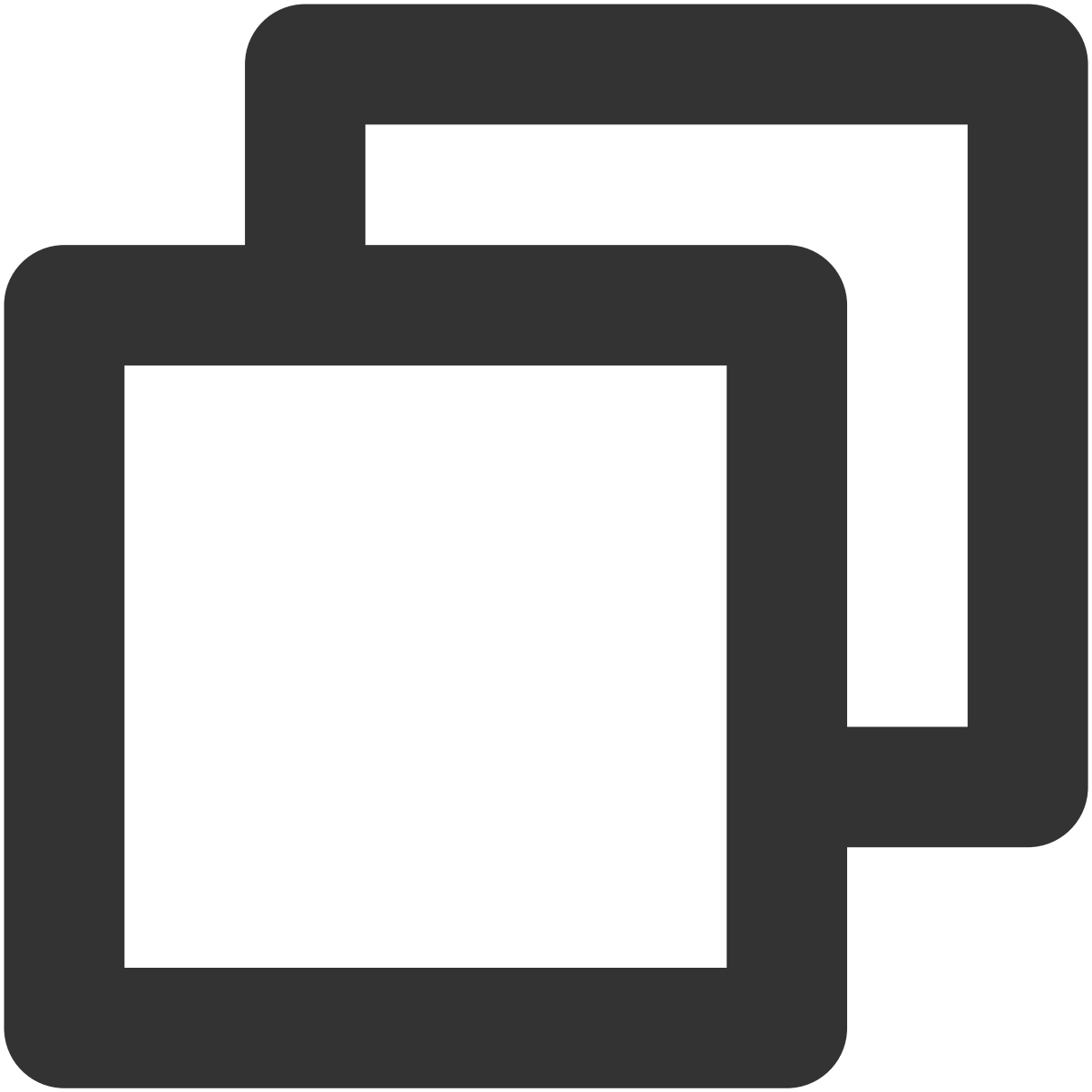
暂停美颜处理



```
void onPause();
```

~~enableEnhancedMode~~

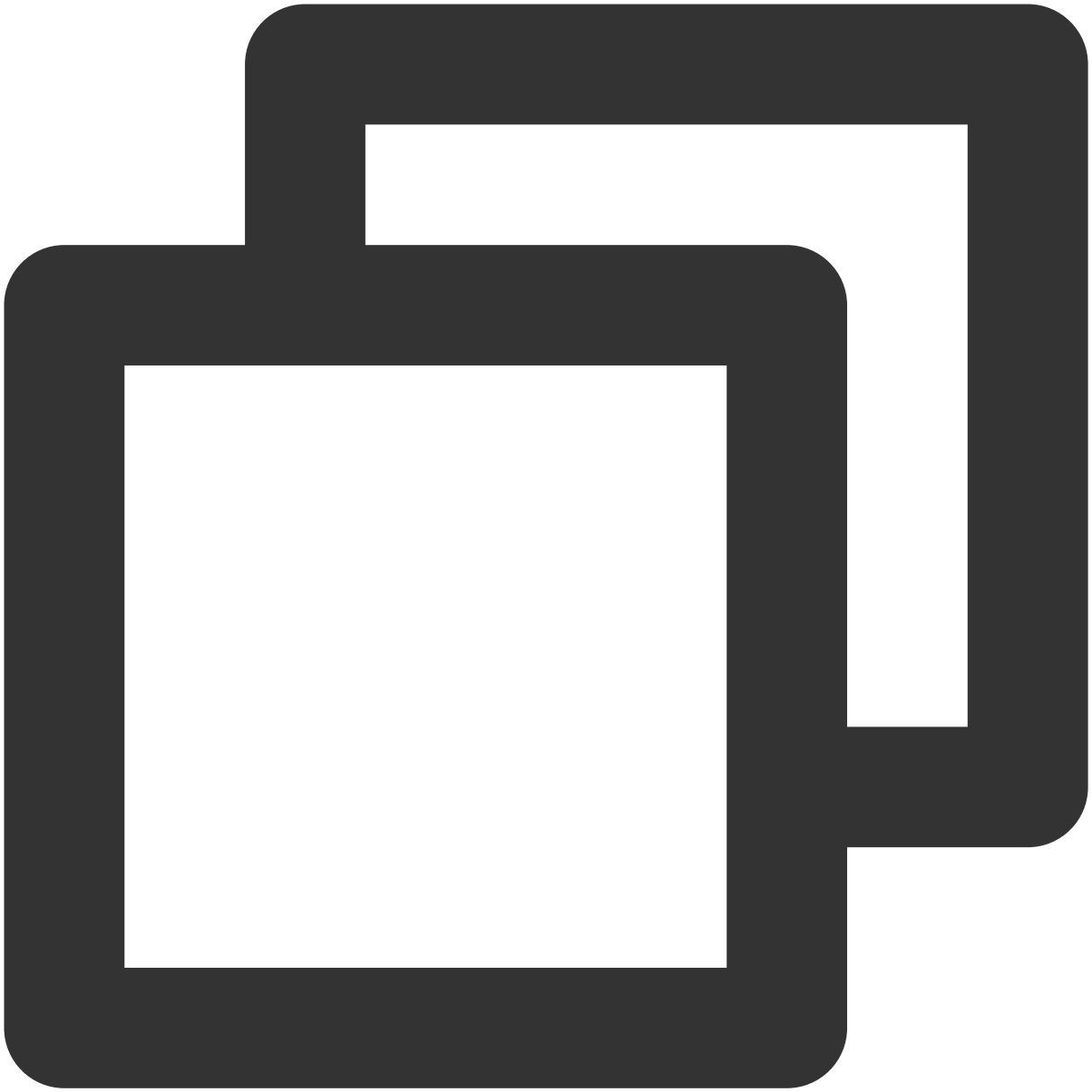
开启增强模式



```
void enableEnhancedMode();
```

setDowngradePerformance（V0.3.1.1新增）

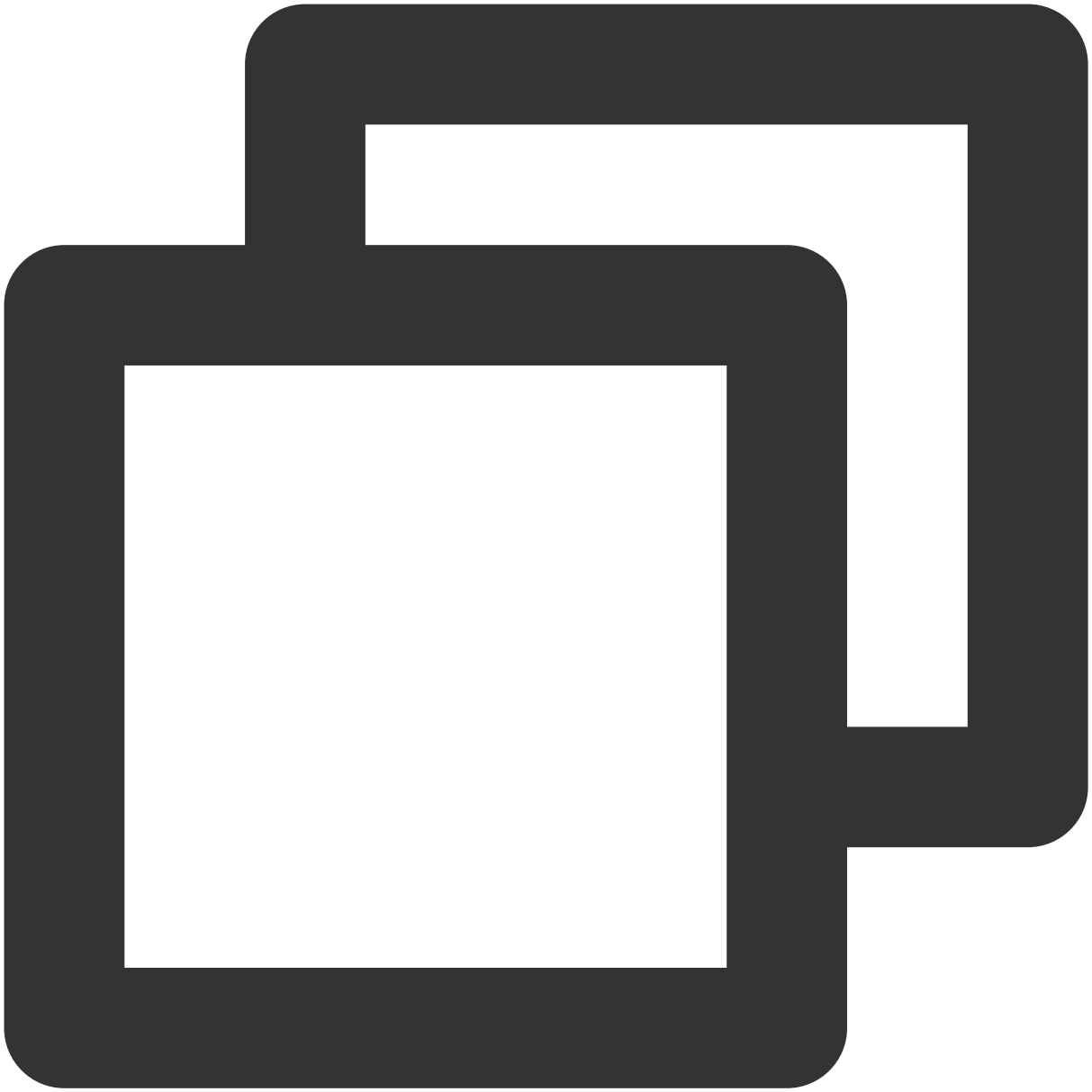
开启性能模式



```
void setDowngradePerformance();
```

setAudioMute（V0.3.1.1新增）

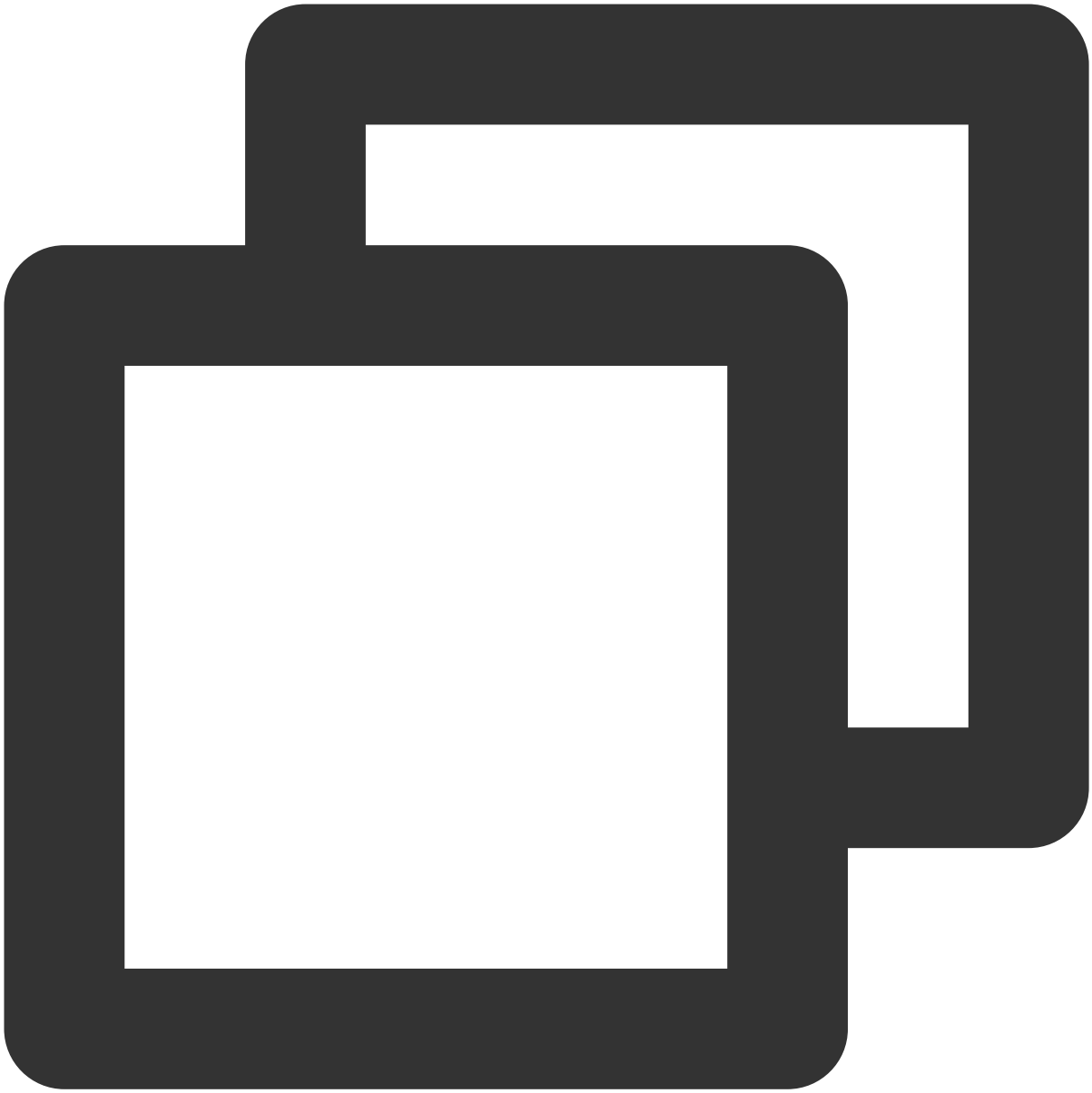
设置是否静音。参数：true 表示静音，false 表示非静音。



```
///背景音乐是否静音
void setAudioMute(bool isMute);
```

setFeatureEnableDisable（V0.3.1.1新增）

开启或关闭某个能力



```
/// 开启或关闭某个特性
void setFeatureEnableDisable(String featureName, bool enable);
```

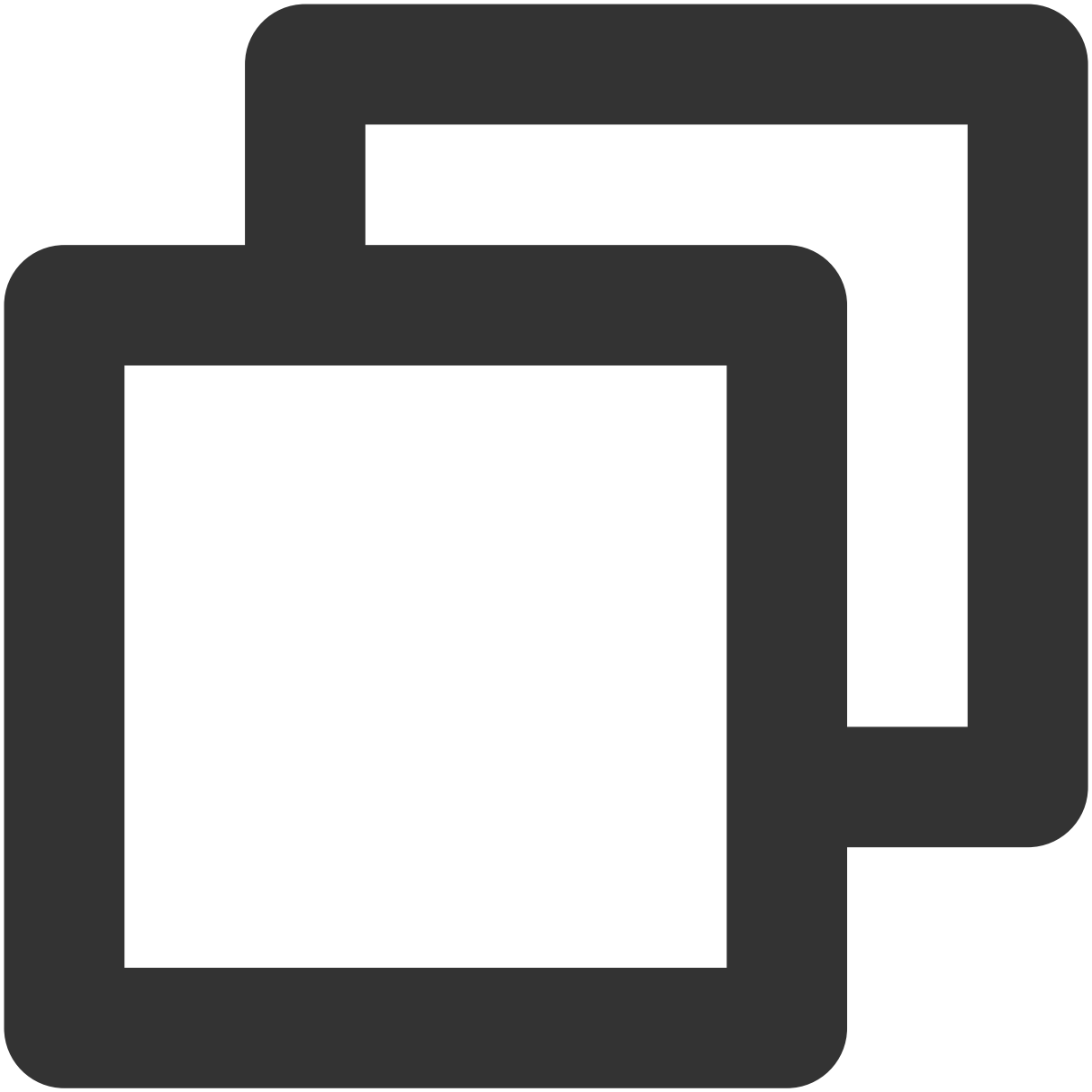
参数

参数	含义
String featureName	原子能力名称 取值如下： `"ai.3dmmV2.enable"` 人脸表情能力

	`"ai.body3dpoint.enable"` 身体点位能力 `"ai.hand.enable"` 手势检测能力 `"beauty.onlyWhitenSkin"` 美白仅对皮肤生效 `"ai.segmentation.skin.enable"` 皮肤分割能力 `"auto_beauty_switch"` 智能美颜(为男性、宝宝减淡美颜美妆效果)
boolean enable	true 表示开启此能力，false 表示关闭此能力 **注：如果是降级模式，则不允许开启皮肤分割**

~~updateProperty~~

设置某一项美颜数值或者动效、滤镜，可在任意线程调用。



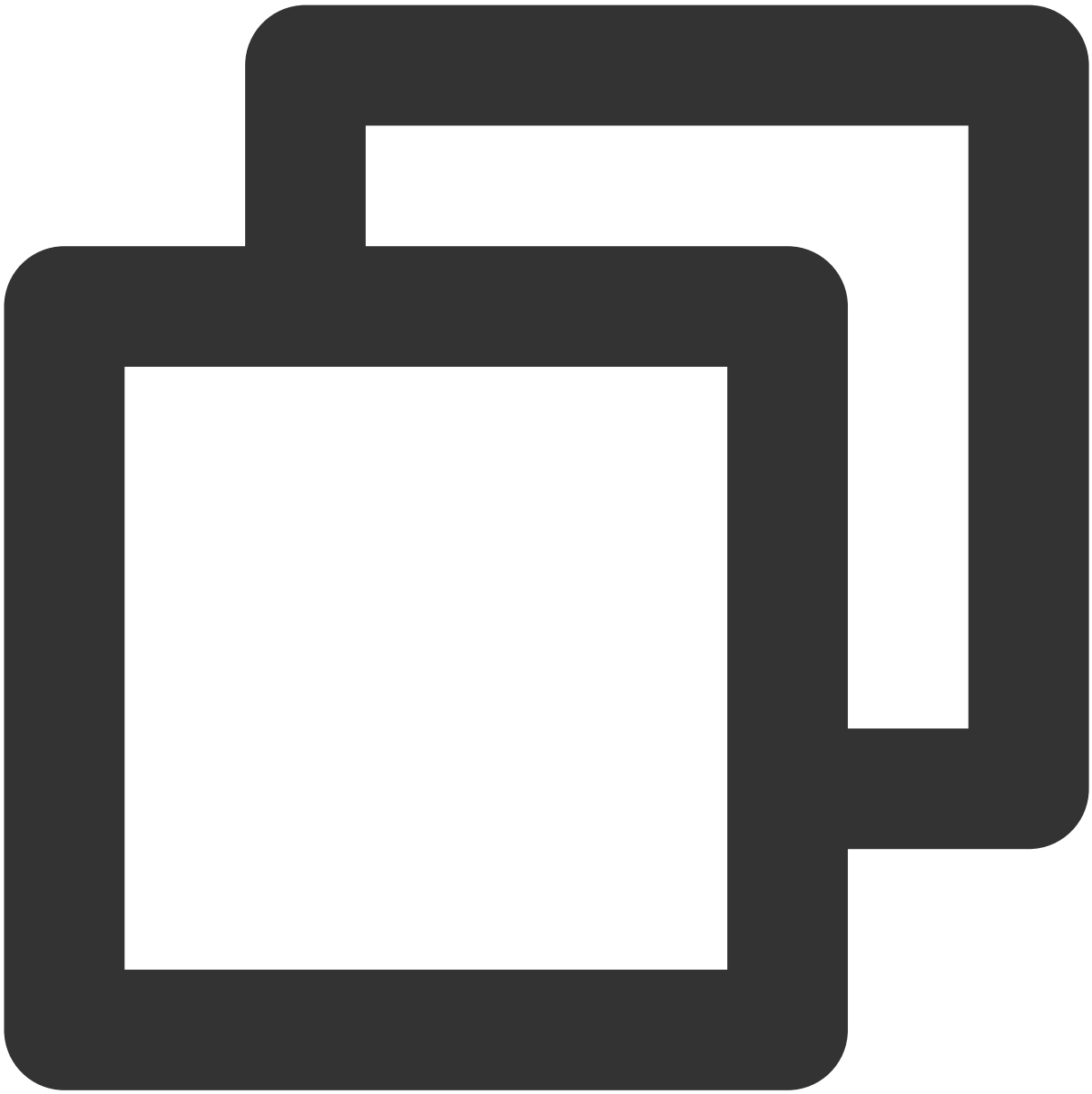
```
void updateProperty(XmagicProperty xmagicProperty);
```

参数

参数	含义
XmagicProperty xmagicProperty	美颜属性封装对象

setEffect（V0.3.5.0新增）

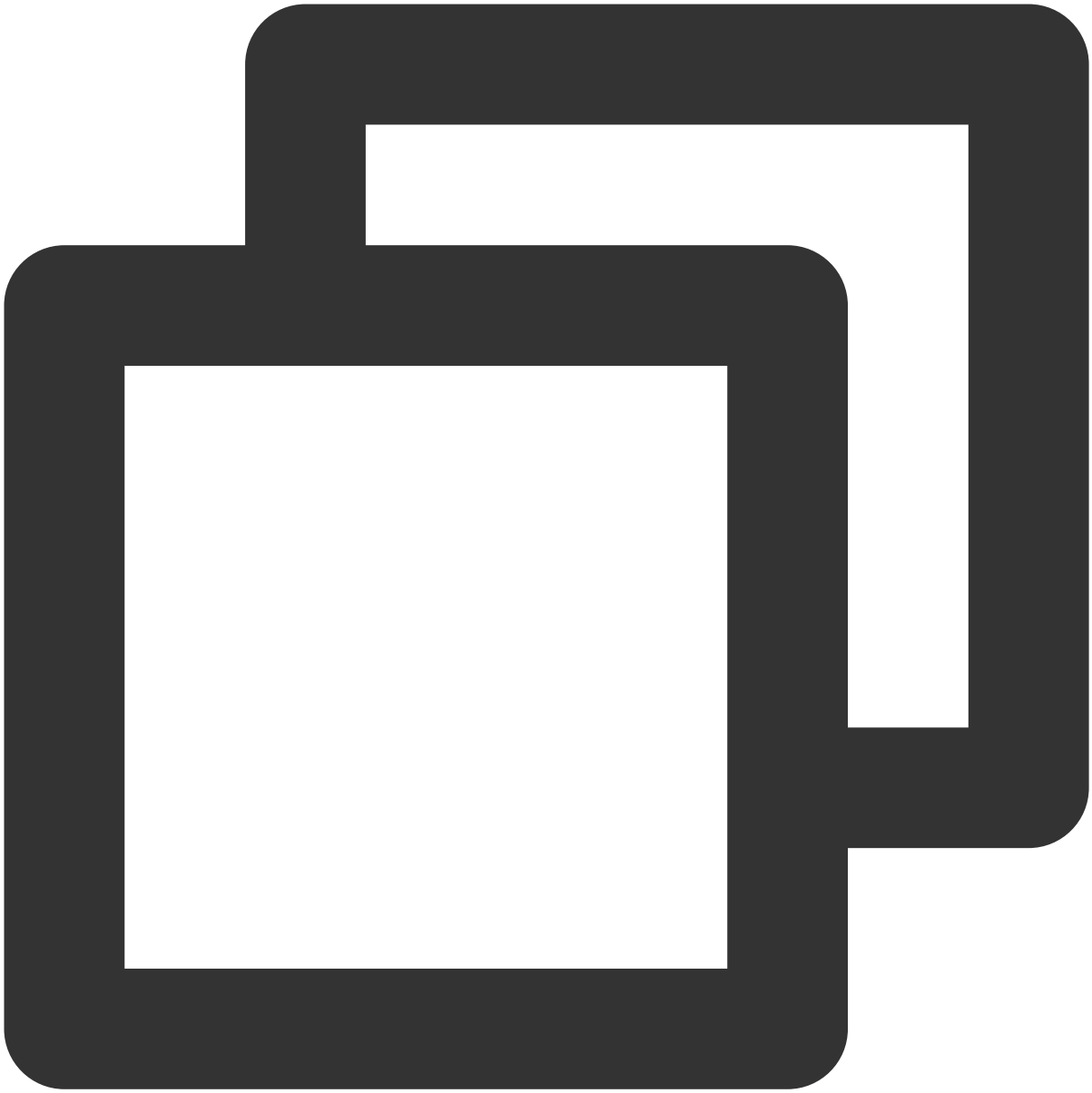
设置美颜、美型、滤镜、美妆、贴纸、分割等效果，可在任意线程调用。具体参数请参考 美颜参数说明。



```
///更新美颜属性。
void setEffect(String effectName,int effectValue,String? resourcePath,Map<String,St
```

setOnCreateXmagicApiErrorListener

设置美颜对象创建时的错误回调接口



```
  void setOnCreateXmagicApiErrorListener(OnCreateXmagicApiErrorListener? errorListe
///创建美颜实例时的错误回调方法
typedef OnCreateXmagicApiErrorListener = void Function(String errorMsg, int code);
```

参数

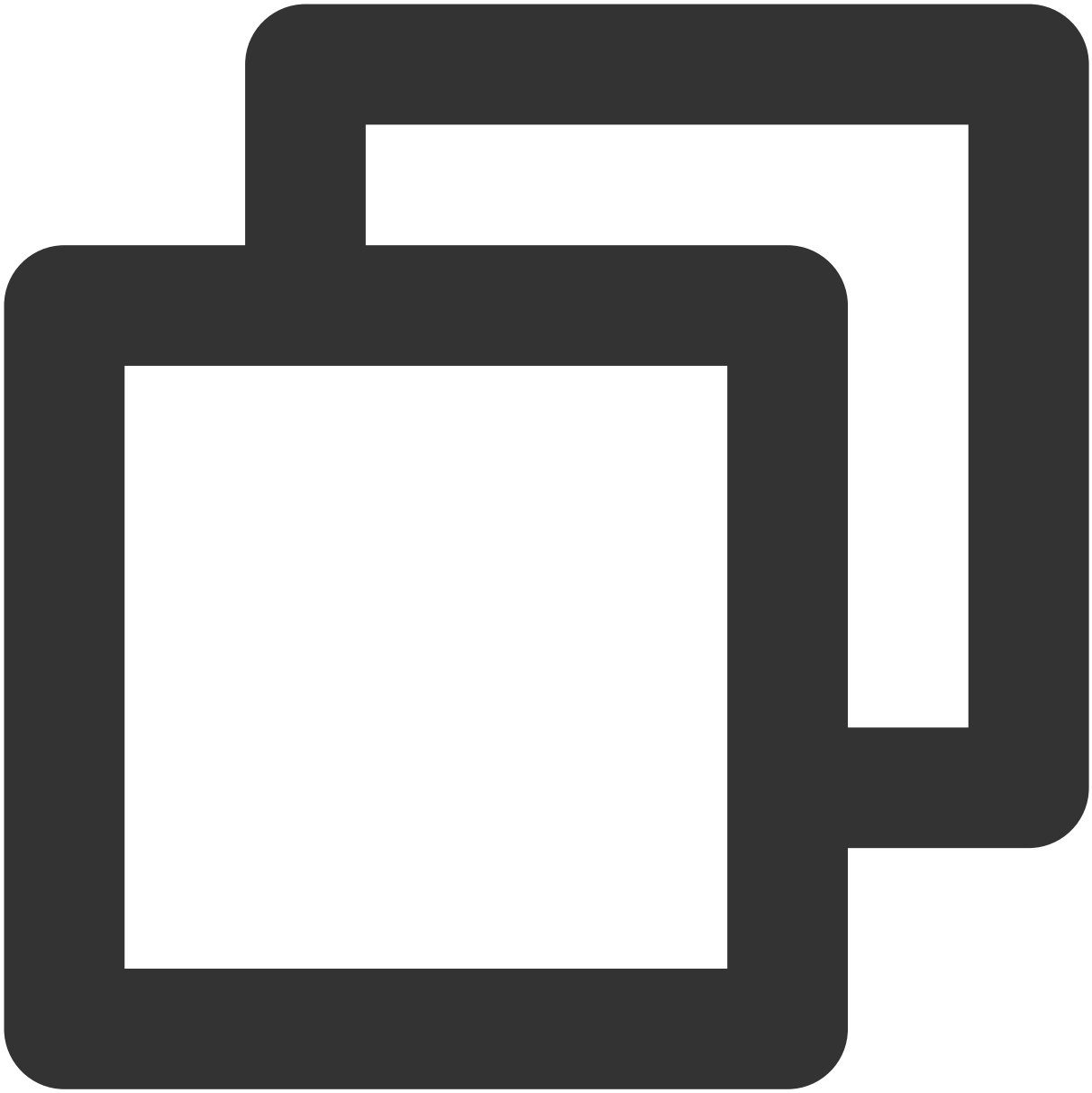
参数	含义
OnCreateXmagicApiErrorListener? errorListener	创建美颜对象时错误信息回调接口

返回错误码含义对照表：

错误码	含义
-1	未知错误
-100	3D 引擎资源初始化失败
-200	不支持 GAN 素材
-300	设备不支持此素材组件
-400	模板 JSON 内容为空
-500	SDK 版本过低
-600	不支持分割
-700	不支持 OpenGL
-800	不支持脚本
5000	分割背景图片分辨率超过 2160×3840
5001	分割背景图片所需内存不足
5002	分割背景视频解析失败。
5003	分割背景视频超过200秒
5004	分割背景视频格式不支持

setTipsListener

设置动效提示语回调函数，用于将提示语展示到前端页面上。比如某些素材会提示用户点点头、伸出手掌、比心等。



```
void setTipsListener(XmagicTipsListener? xmagicTipsListener);

abstract class XmagicTipsListener {
  /// 显示tips。Show the tip.
  /// @param tips tips字符串。Tip's content
  /// @param tipsIcon tips的icon。Tip's icon
  /// @param type tips类别，0表示字符串和icon都展示，1表示是pag素材只展示icon。tips catego
  /// @param duration tips显示时长，毫秒。Tips display duration, milliseconds
  void tipsNeedShow(String tips, String tipsIcon, int type, int duration);

  /// *
```

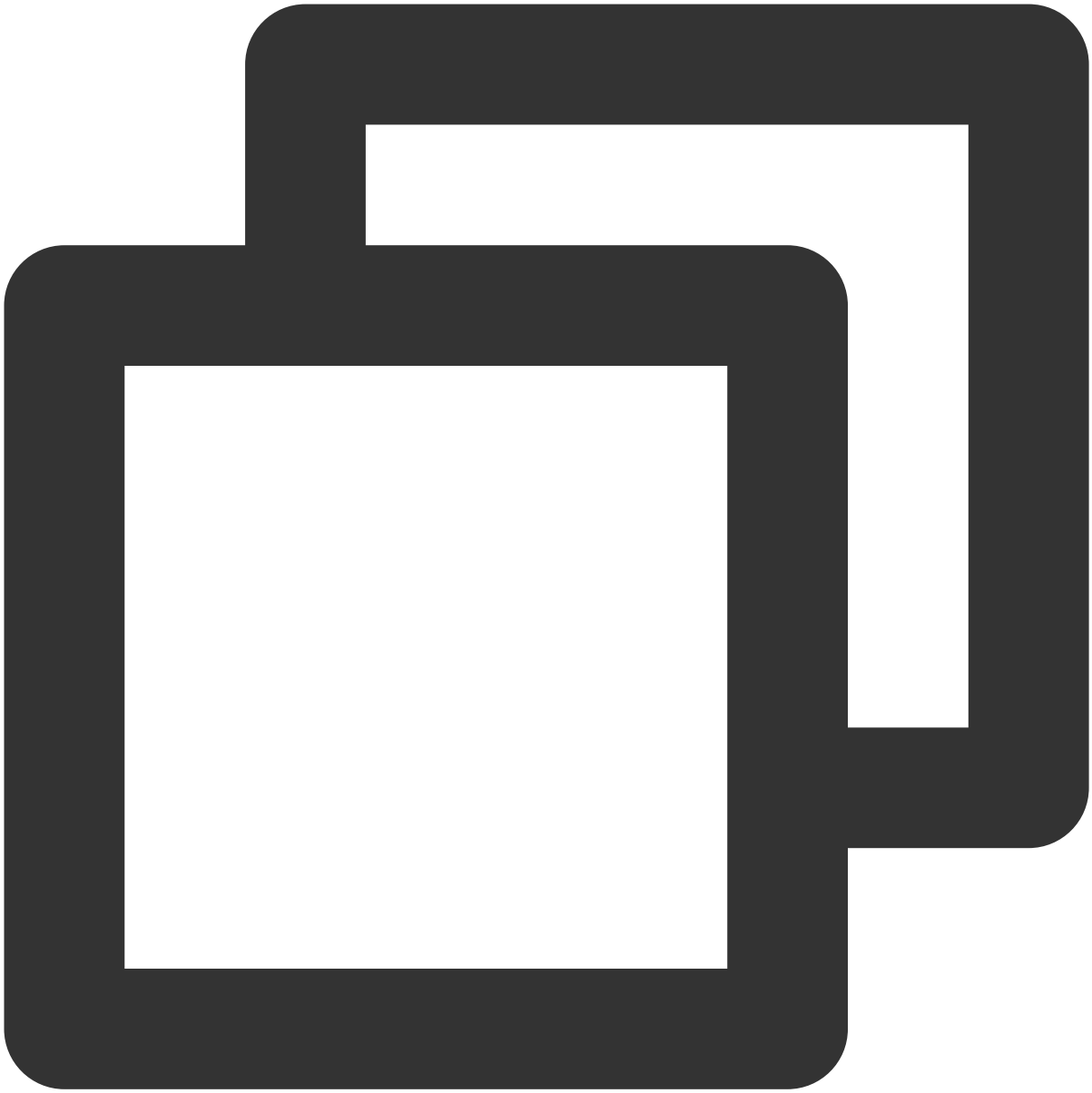
```
    /// 隐藏tips。Hide the tip.
    /// @param tips tips字符串。Tip's content
    /// @param tipsIcon tips的icon。Tip's icon
    /// @param type tips类别，0表示字符串和icon都展示，1表示是pag素材只展示icon。tips catego
    void tipsNeedHide(String tips, String tipsIcon, int type);
}
```

参数

参数	含义
XmagicTipsListener xmagicTipsListener	回调函数实现类

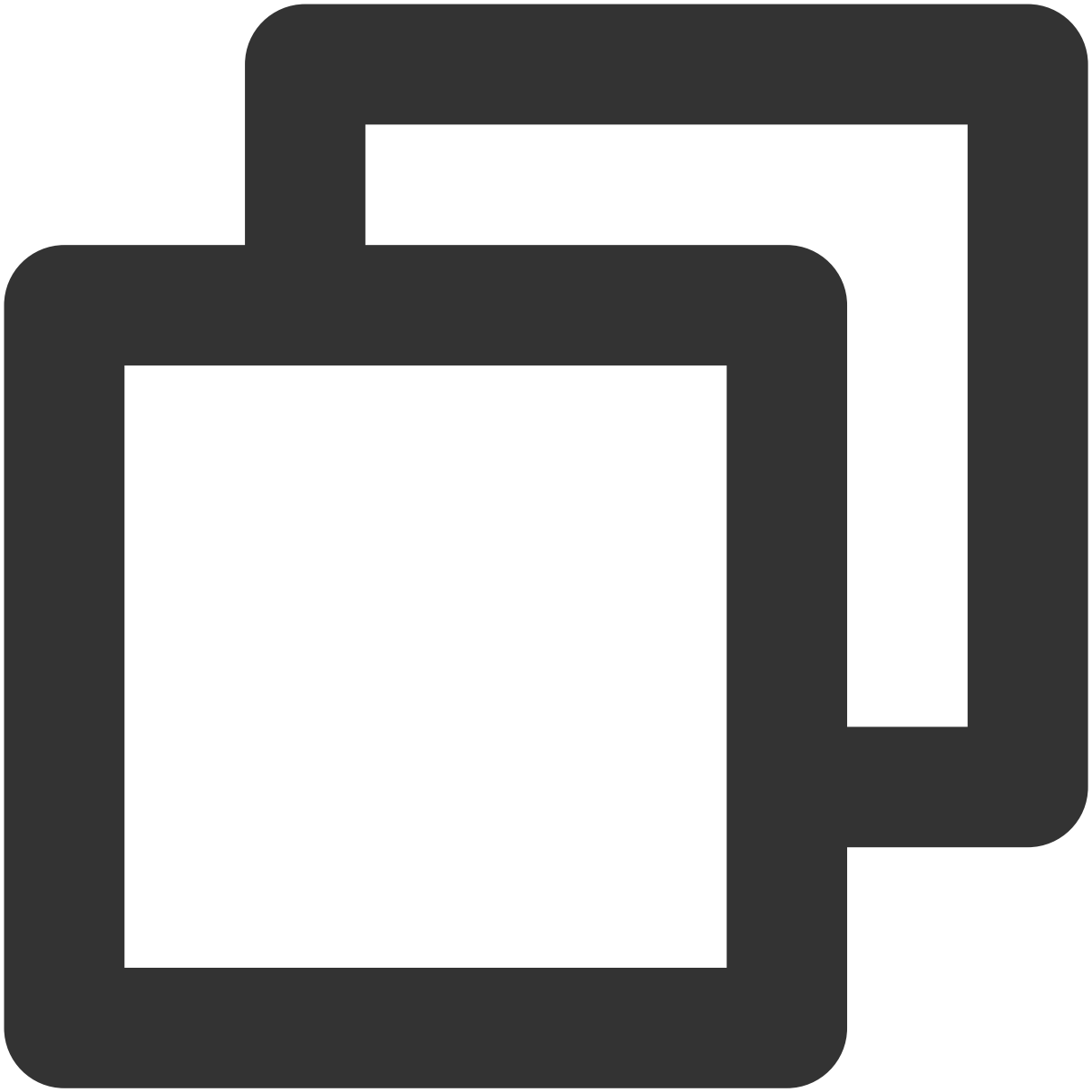
setYTDataListener

设置人脸点位信息等数据回调。



```
  ///设置人脸点位信息等数据回调（S1-05 和 S1-06 套餐才会有回调）
void setYTDataListener(XmagicYTDataListener? xmagicYTDataListener);
设置人脸信息等数据回调

abstract class XmagicYTDataListener {
  //优图AI数据回调。
  void onYTDataUpdate(String data);
}
```

onYTDataUpdate 返回 JSON string 结构，最多返回5个人脸信息：



```
{
 "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
 },
 ...
 ]
}
```

字段含义

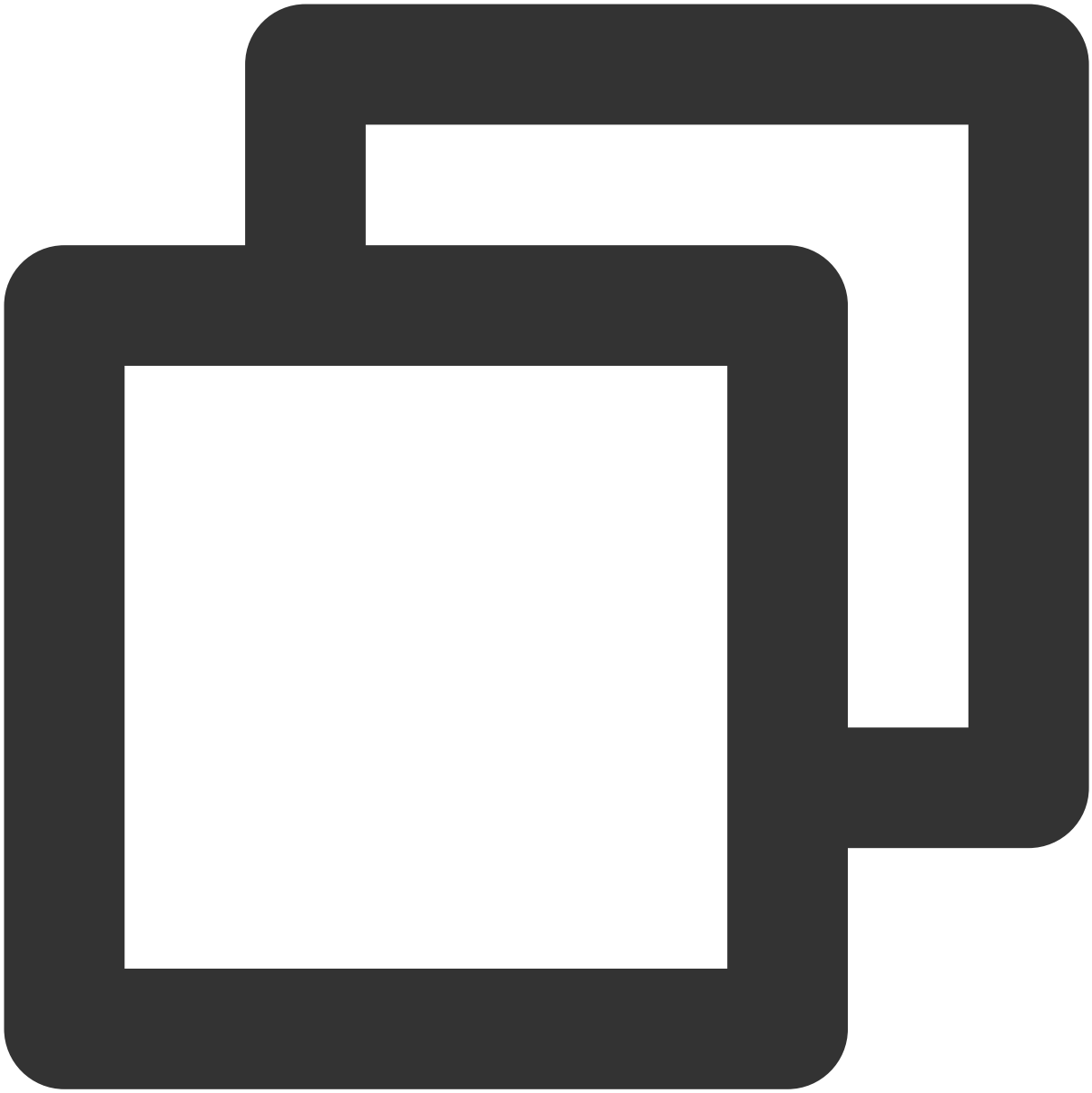
字段	类型	值域	说明
trace_id	int	[1,INF)	人脸 ID，连续取流过程中，ID 相同的可以认为是同一张人脸
face_256_point	float	[0,screenWidth] 或 [0,screenHeight]	共512个数，人脸256个关键点，屏幕左上角为(0,0)
face_256_visible	float	[0,1]	人脸256关键点可见度
out_of_screen	bool	true/false	人脸是否出框
left_eye_high_vis_ratio	float	[0,1]	左眼高可见度点位占比
right_eye_high_vis_ratio	float	[0,1]	右眼高可见度点位占比
left_eyebrow_high_vis_ratio	float	[0,1]	左眉高可见度点位占比
right_eyebrow_high_vis_ratio	float	[0,1]	右眉高可见度点位占比
mouth_high_vis_ratio	float	[0,1]	嘴高可见度点位占比

参数

参数	含义
XmagicYTDataListener xmagicYTDataListener	回调函数实现类

setAIDataListener

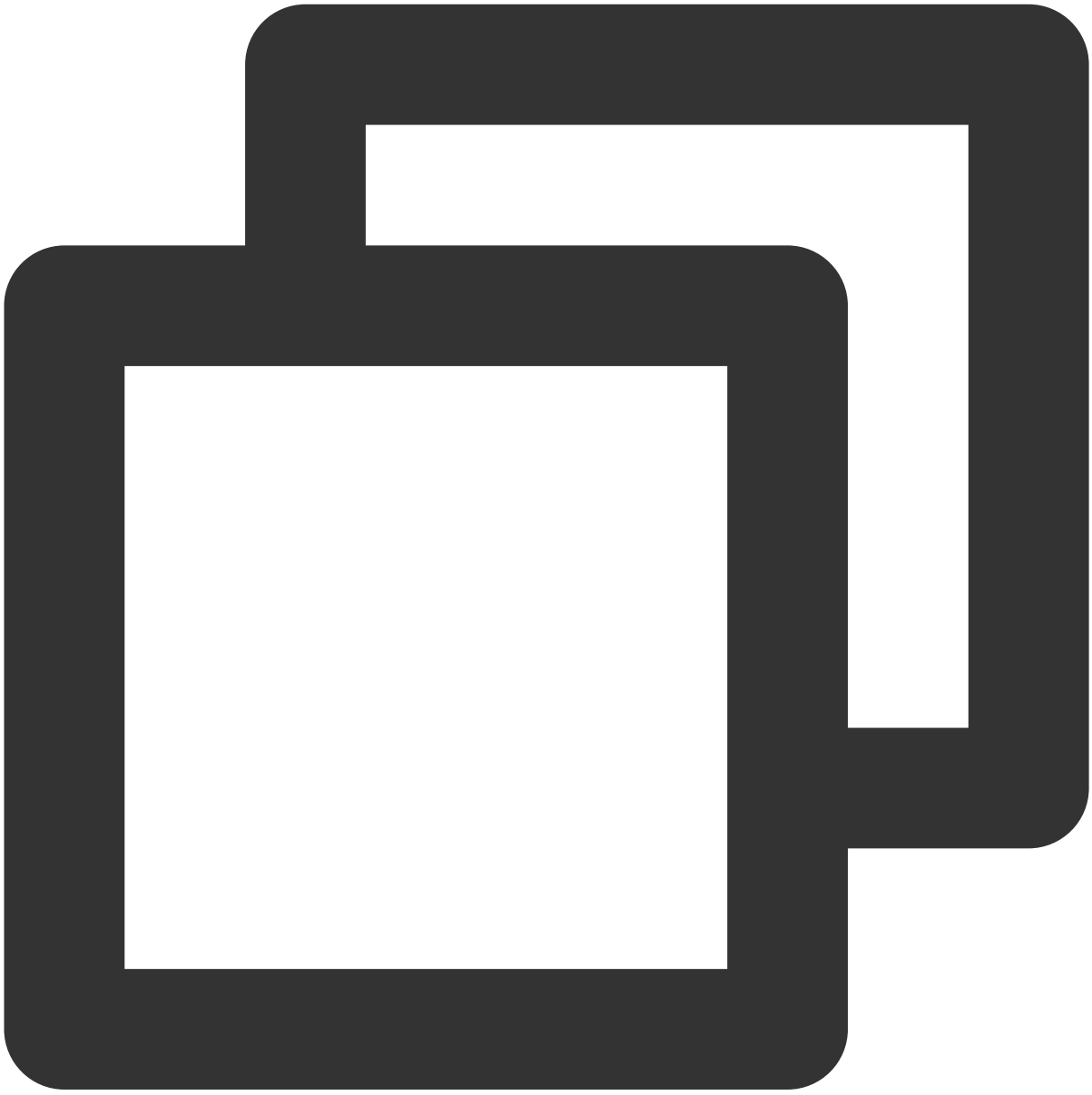
检测到人脸、身体、手势时，会回调这些部位的点位信息



```
void setAIDataListener(XmagicAIDataListener? aiDataListener);

abstract class XmagicAIDataListener {
  void onFaceDataUpdated(String faceDataList);

  void onHandDataUpdated(String handDataList);

  void onBodyDataUpdated(String bodyDataList);
}
```

~~isBeautyAuthorized~~

判断当前的 License 授权支持哪些美颜或美体项。 仅支持 BEAUTY 和 BODY_BEAUTY 类型的美颜项检测。检测后的结果会赋值到各个美颜对象 `XmagicProperty.isAuth` 字段中。如果 isAuth 字段为 false，可以在 UI 上屏蔽这些项的入口。



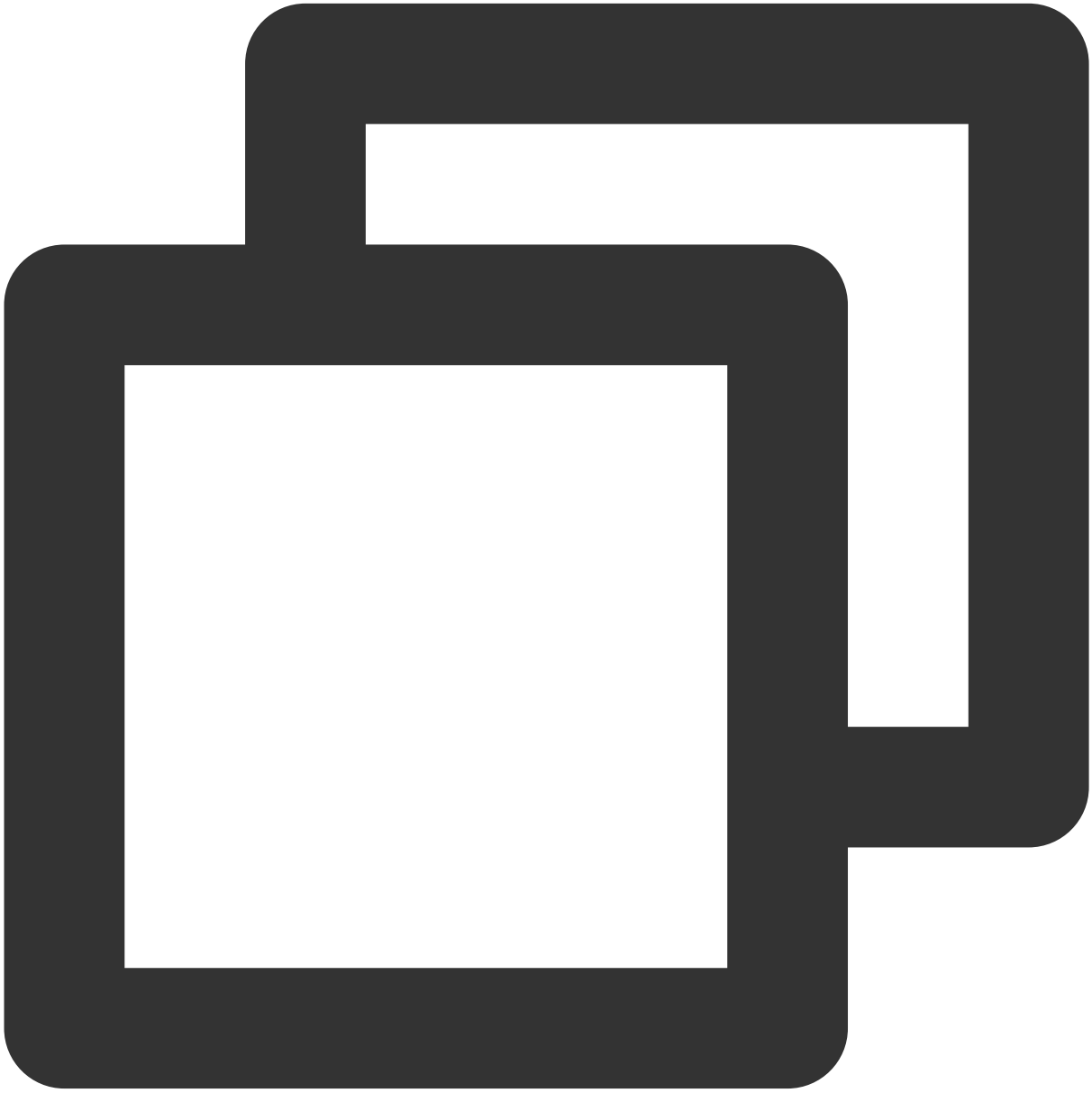
```
Future<List<XmagicProperty>> isBeautyAuthorized(
      List<XmagicProperty> properties);
```

参数

参数	含义
List<XmagicProperty> properties	需要检测的美颜项

isSupportBeauty

判断当前机型是否支持美颜（OpenGL3.0）。



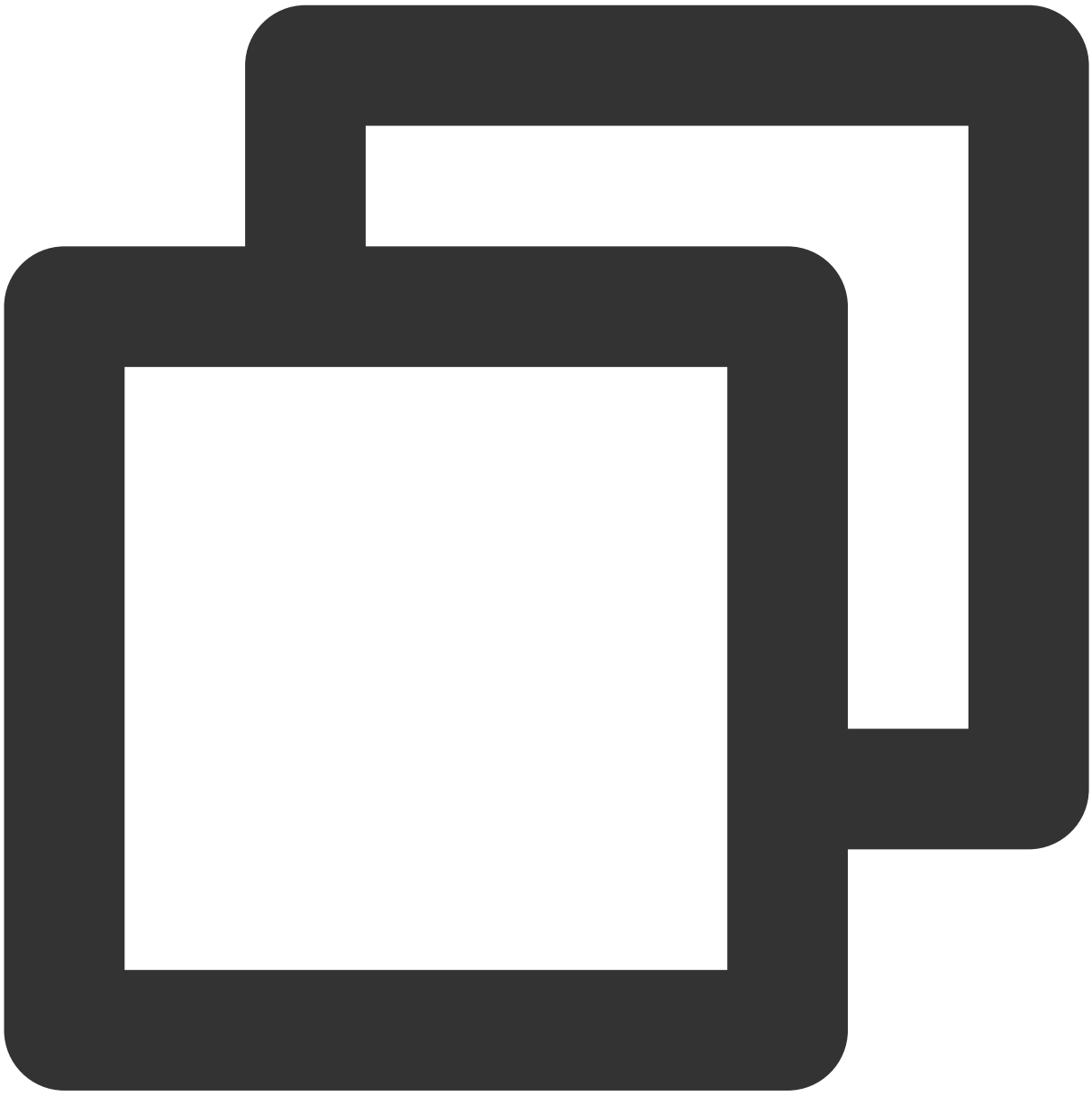
```
Future&lt;bool> isSupportBeauty();
```

返回

返回值 bool：是否支持支持美颜。

getDeviceAbilities

返回当前设备支持的原子能力表。与 getPropertyRequiredAbilities 方法搭配使用.



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

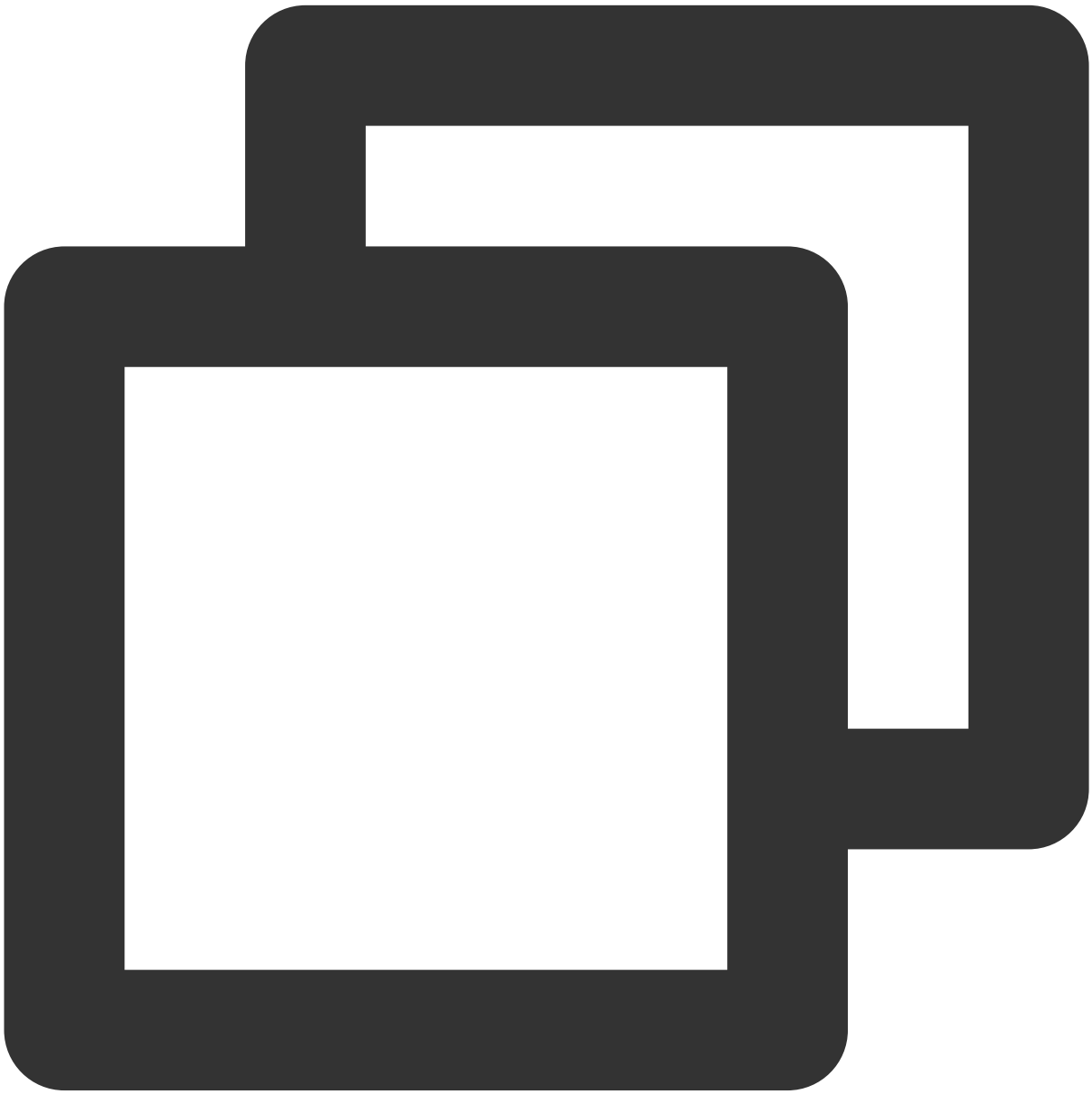
返回

返回值 `Map<String,bool>` ：

key：原子能力名（与素材能力名字对应）。
value：当前设备是否支持。

~~isDeviceSupport~~

将动效资源列表传入 SDK 中做检测，执行后 `XmagicProperty.isSupport` 字段标识该素材是否可用。根据 `XmagicProperty.isSupport` 可 UI 层控制单击限制，或者直接从资源列表删除。



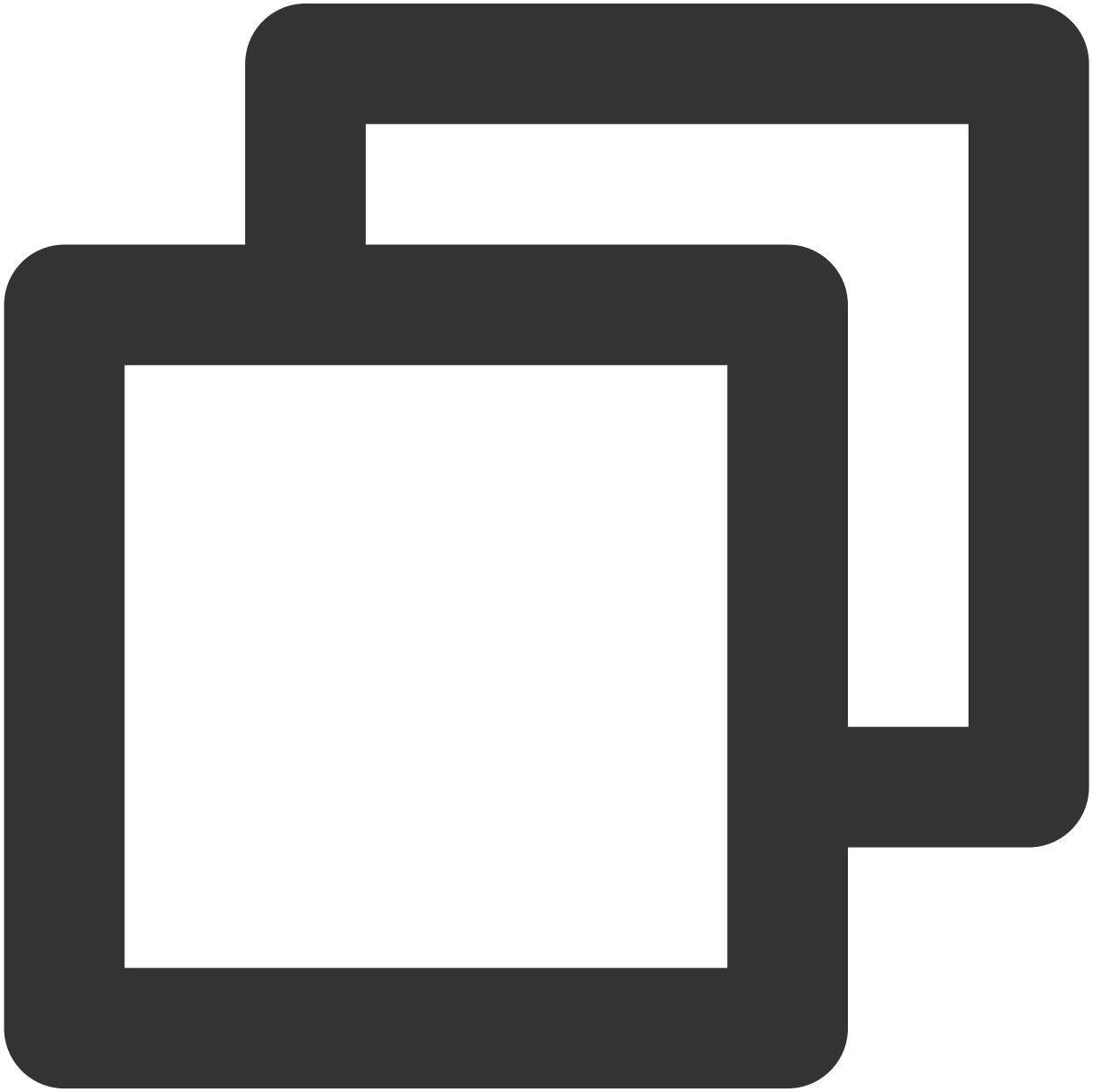
```
Future&lt;List<XmagicProperty>> isDeviceSupport(List<XmagicProperty> assetsList);
```

参数

参数	含义
List<XmagicProperty> assetsList	需要检测的动效素材列表

isDeviceSupportMotion (V0.3.5.0版本新增)

检测当前设备是否支持此素材



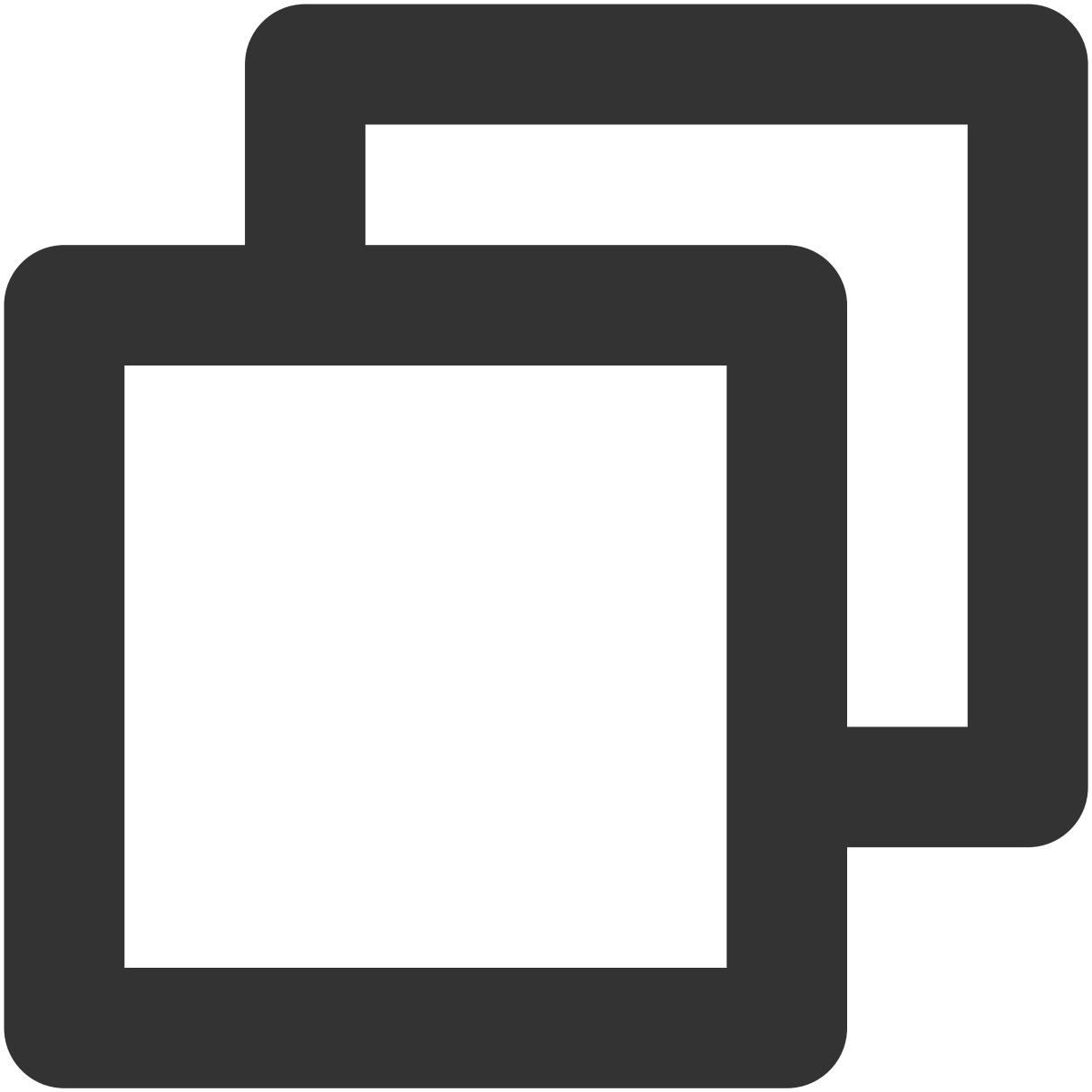
```
Future<bool> isDeviceSupportMotion(String motionResPath);
```

参数

参数	含义
motionResPath	需要检测的动效素材的本地文件地址

getPropertyRequiredAbilities

传入一个动效资源列表，返回每一个资源所使用到的 SDK 原子能力列表。
方法的使用场景为：
您购买或制作了若干款动效素材，调用这个方法，会返回每一个素材需要使用的原子能力列表。例如素材1需要使用能力 A、B、C，素材2需要使用能力 B、C、D，然后您把这样的能力列表保持在服务器上。之后，当用户要从服务器下载动效素材时，用户先通过 getDeviceAbilities 方法获取他手机具备的原子能力列表（比如这台手机具备能力 A、B、C，但不具备能力 D），把他的能力列表传给服务器，服务器判断该设备不具备能力 D，因此不给该用户下发素材2。



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

参数

参数	含义
List<XmagicProperty> assetsList	需要检测原子能力的动效资源列表

返回

返回值 Map<XmagicProperty, List<String>?> ：

key：动效资源素材实体类。

value：所使用到的原子能力列表。

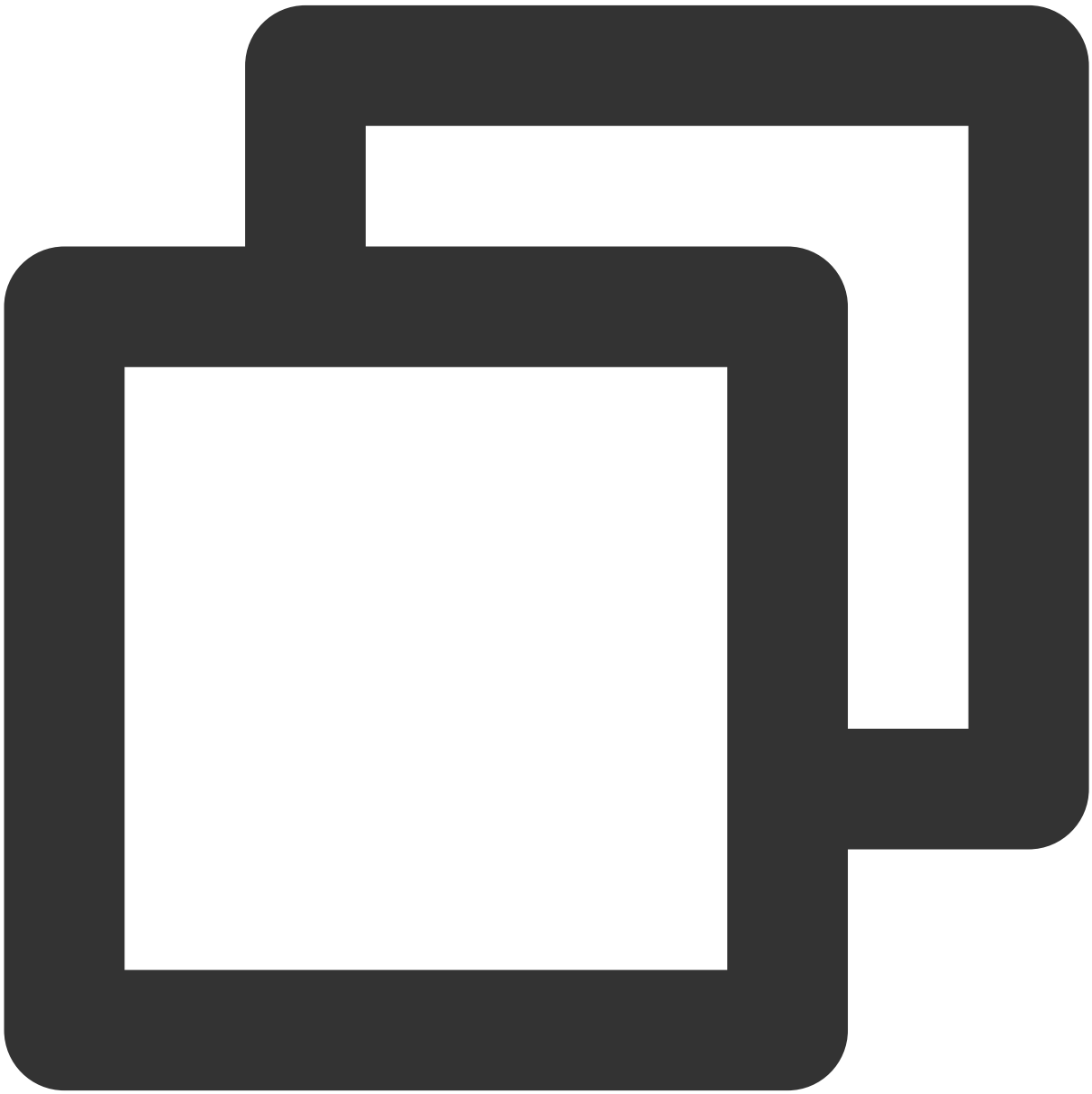
Web

最近更新时间：2023-05-06 15:45:49

本文档将介绍Web美颜特效SDK 核心参数及方法。

说明：

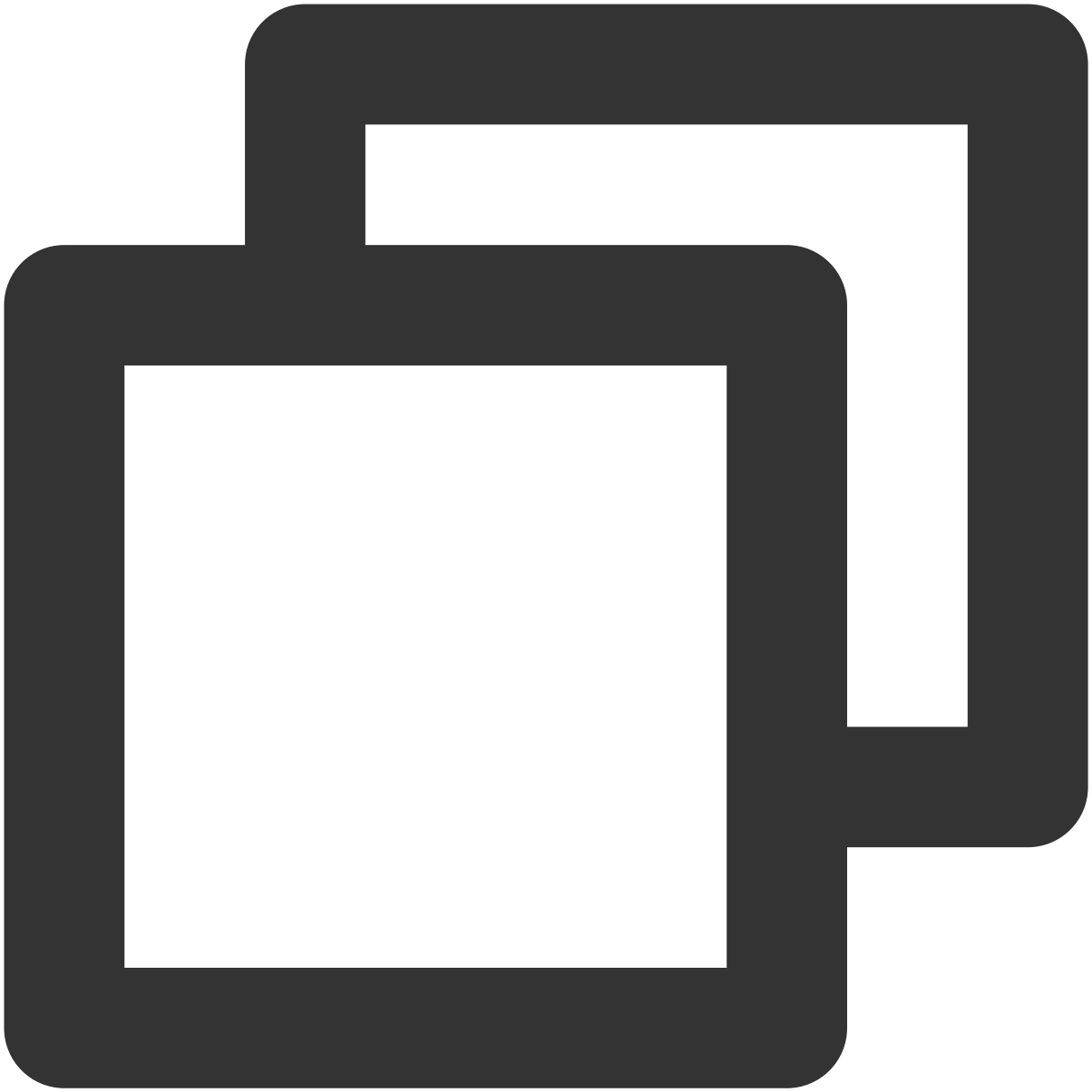
Web美颜特效 SDK 需要浏览器支持并开启硬件加速才能够流畅渲染（小程序端无需判断），因此 SDK 提供了检测方法供业务提前判断，如不支持则进行屏蔽处理。



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

```
if(isWebGLSupported()) {
    const sdk = new ArSdk({
    ...
})
} else {
    // 屏蔽逻辑
}
```

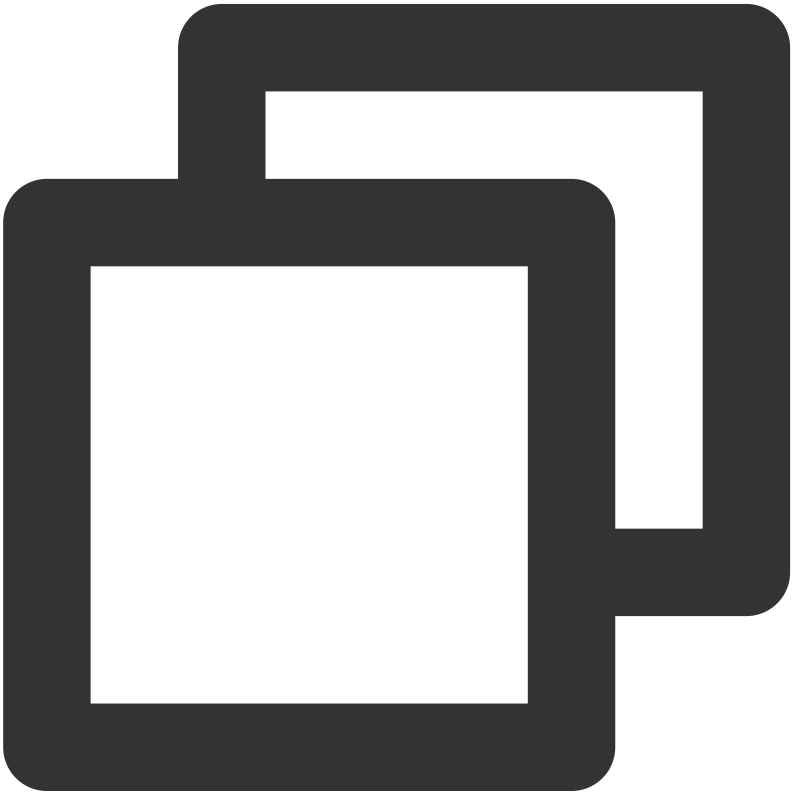
初始化参数

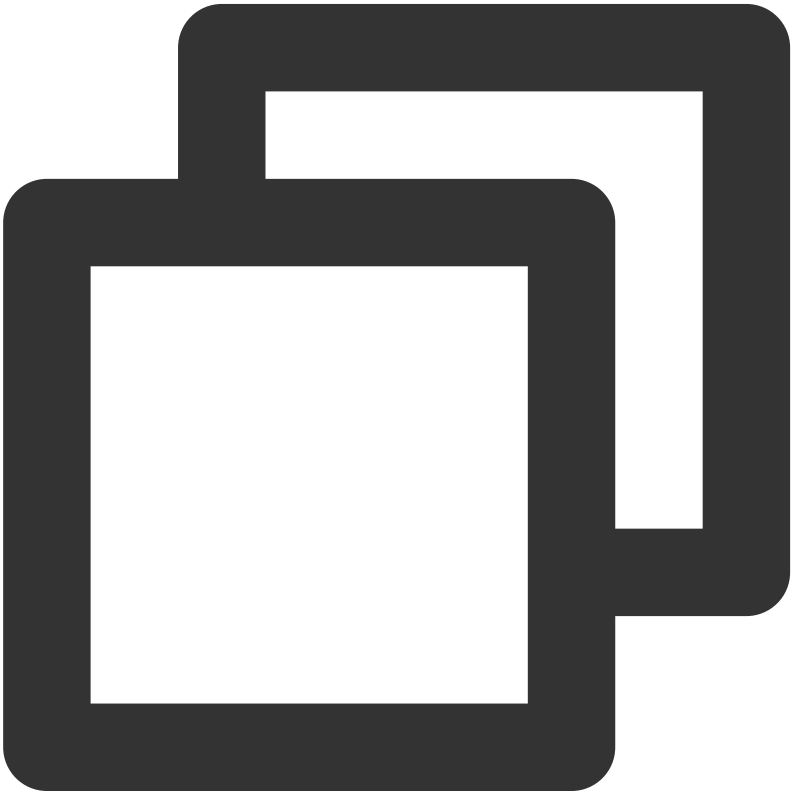


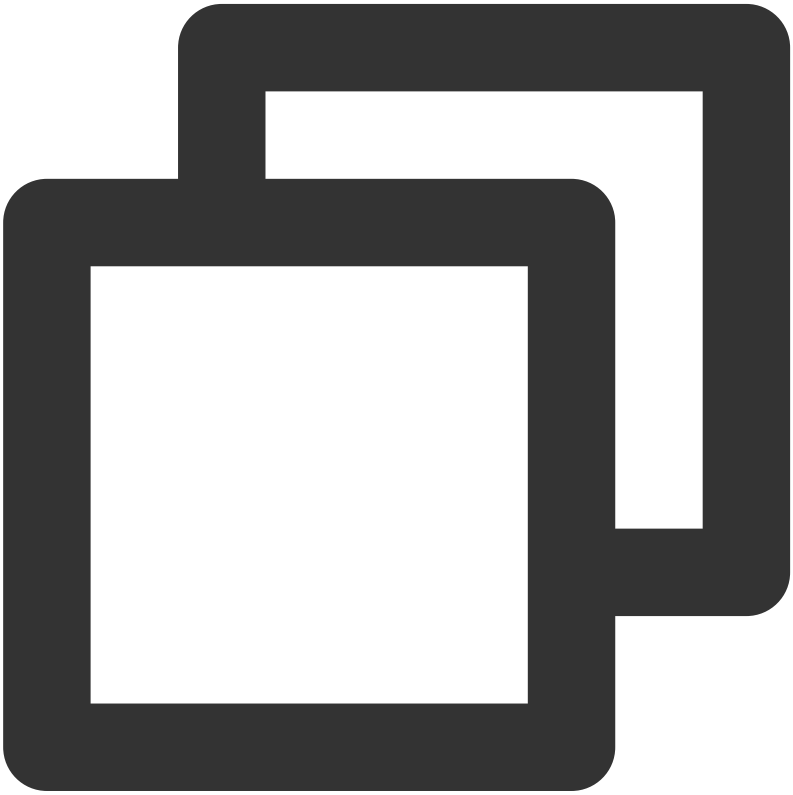
```
import { ArSdk } from 'tencentcloud-webar'
// 初始化SDK
const sdk = new ArSdk({
...
})
```

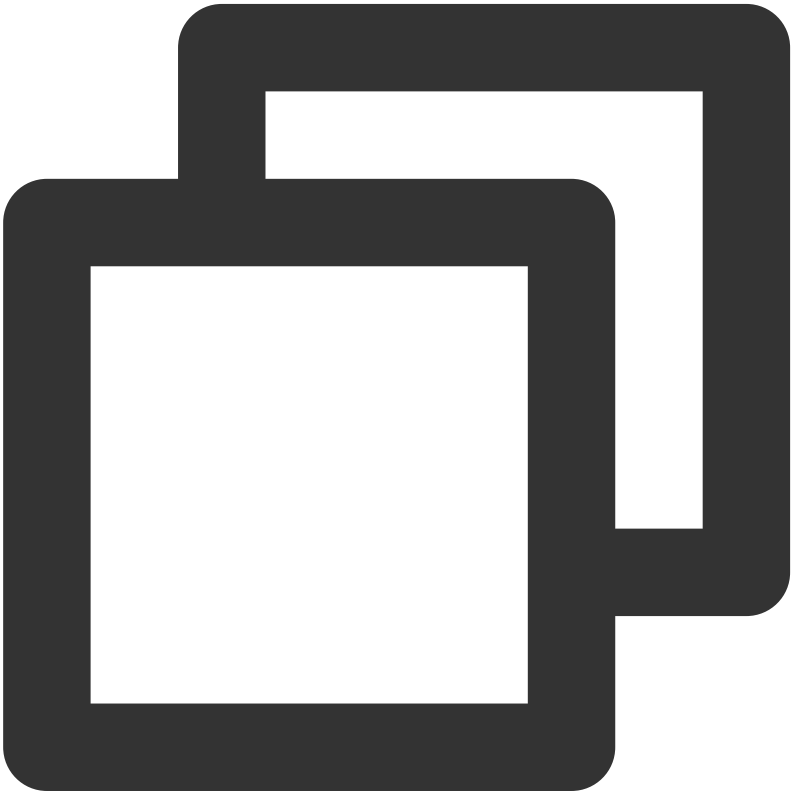
初始化 SDK 的 Config 支持以下参数：

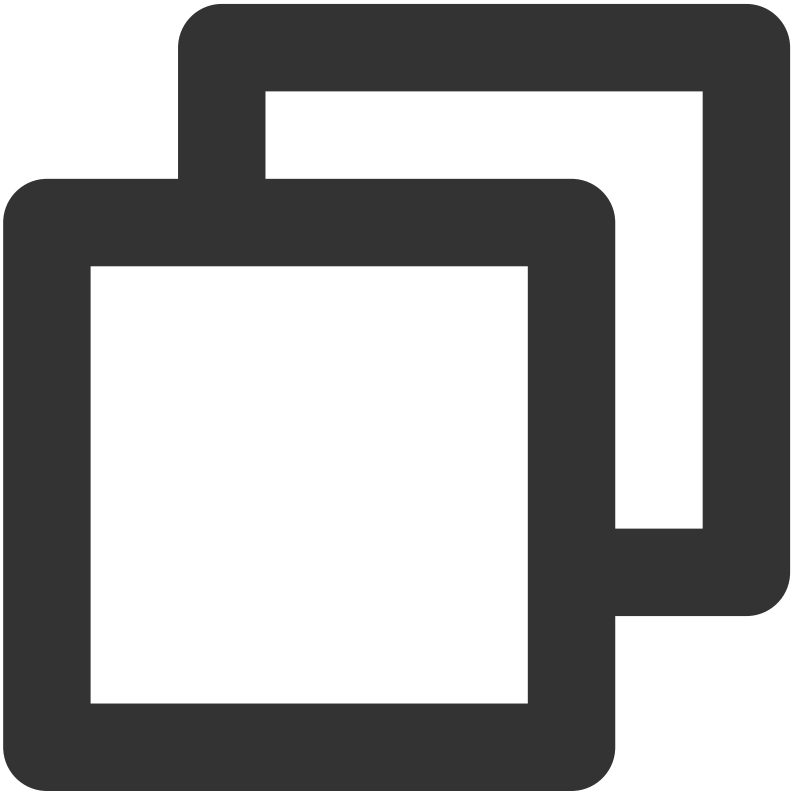
参数	说明	类型	是否必
module	模块配		否，默

	置	 ``` type ModuleConfig = { beautify: boolean // 默认为true segmentation: boolean // 默认为false } ```	{be true segm fals
auth	鉴权参数		是

		 `type AuthConfig = {` `licenseKey: string // 控制台 Web License 管理 获取` `appId: string // 控制台 账号信息 > 基本信息 查看 APPID` `authFunc:() => Promise<{` `signature:string,` `timestamp:string` `}> // 请参见 License 配置使用` `}`	
input	输入源	MediaStream|HTMLImageElement|String	否
camera	内置相机		否

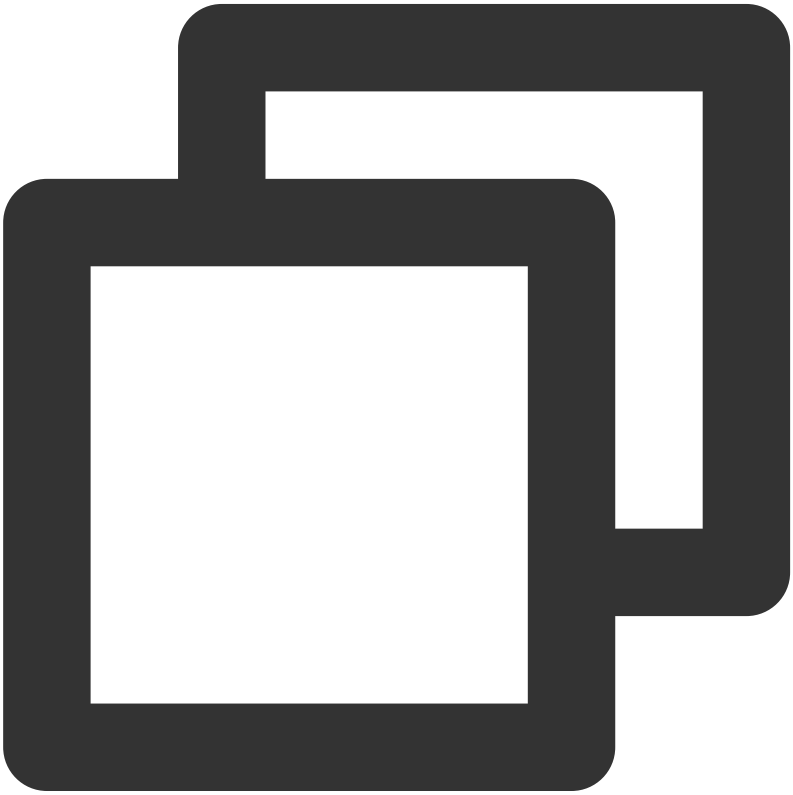
		 ``` type CameraConfig = { width: number, // 拍摄画面宽度 height: number, // 拍摄画面高度 mirror: boolean, // 是否开启左右镜像 frameRate: number // 画面采集帧率 } ```	
beautify	美颜参数		否

		 `type BeautifyOptions = {` `  whiten?: number, // 美白 0-1` `  dermabrasion?: number // 磨皮0-1` `  lift?: number // 瘦脸0-1` `  shave?: number // 削脸0-1` `  eye?: number // 大眼0-1` `  chin?: number // 下巴0-1` `}`	
background	背景参数		否

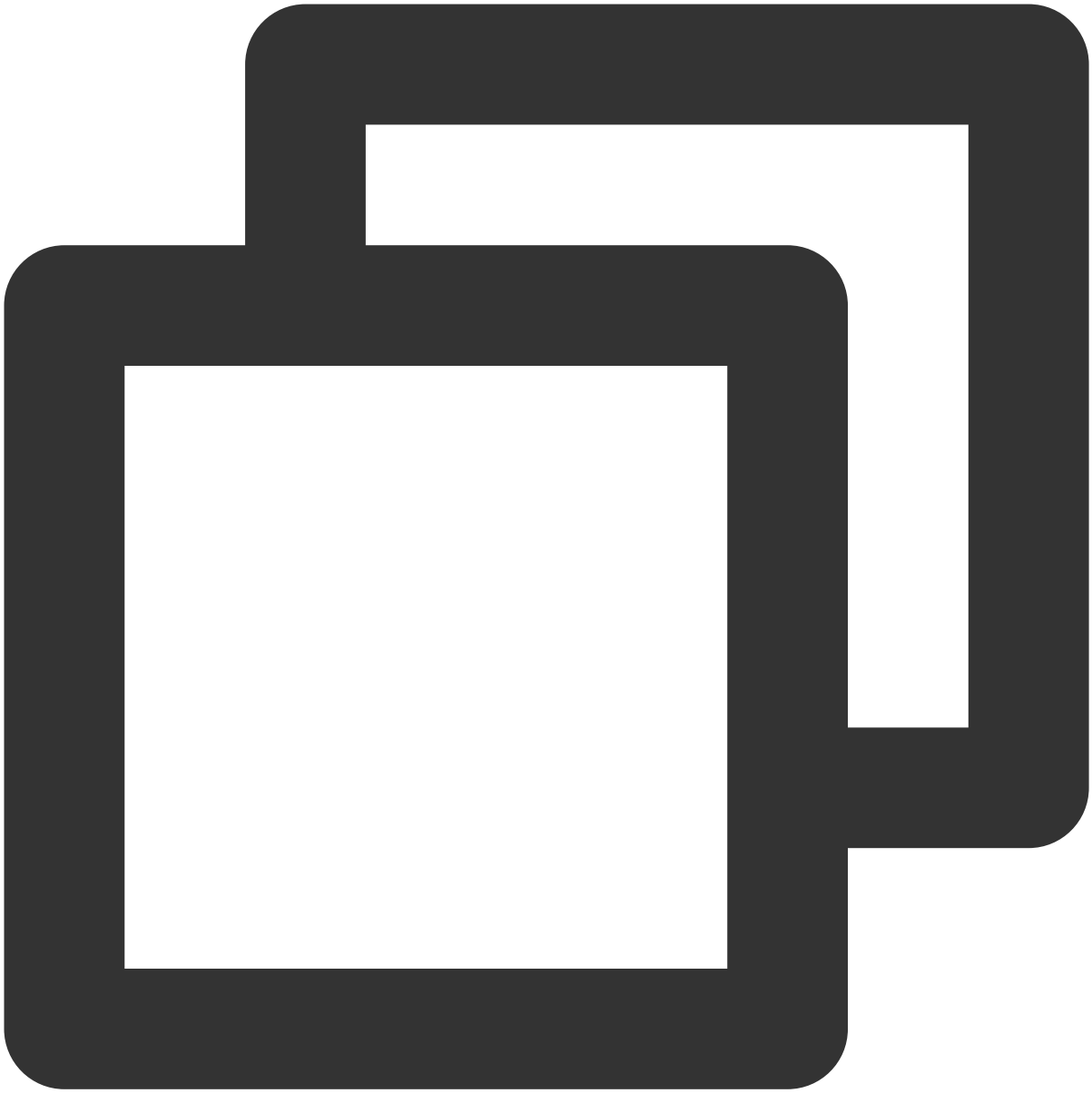


```
type BackgroundOptions = {
    type: 'image' | 'blur' | 'transparent',
    src?: string
}
```

loading	内置 loading icon 配置		

		 `type loadingConfig = { enable: boolean, size?: number lineWidth?: number strokeColor?: number }`	
language	国际化对应(1.0.6版本开始支持），中文（zh）和英文（en）	String: zh | en	否，默

回调事件



```
let effectList = [];
let filterList = [];
// sdk 的回调用法
sdk.on('created', () => {
    // 在 created 回调中拉取特效和滤镜列表供页面展示
    sdk.getEffectList({
        Type: 'Preset',
        Label: '美妆',
    }).then(res => {
        effectList = res
    });
```

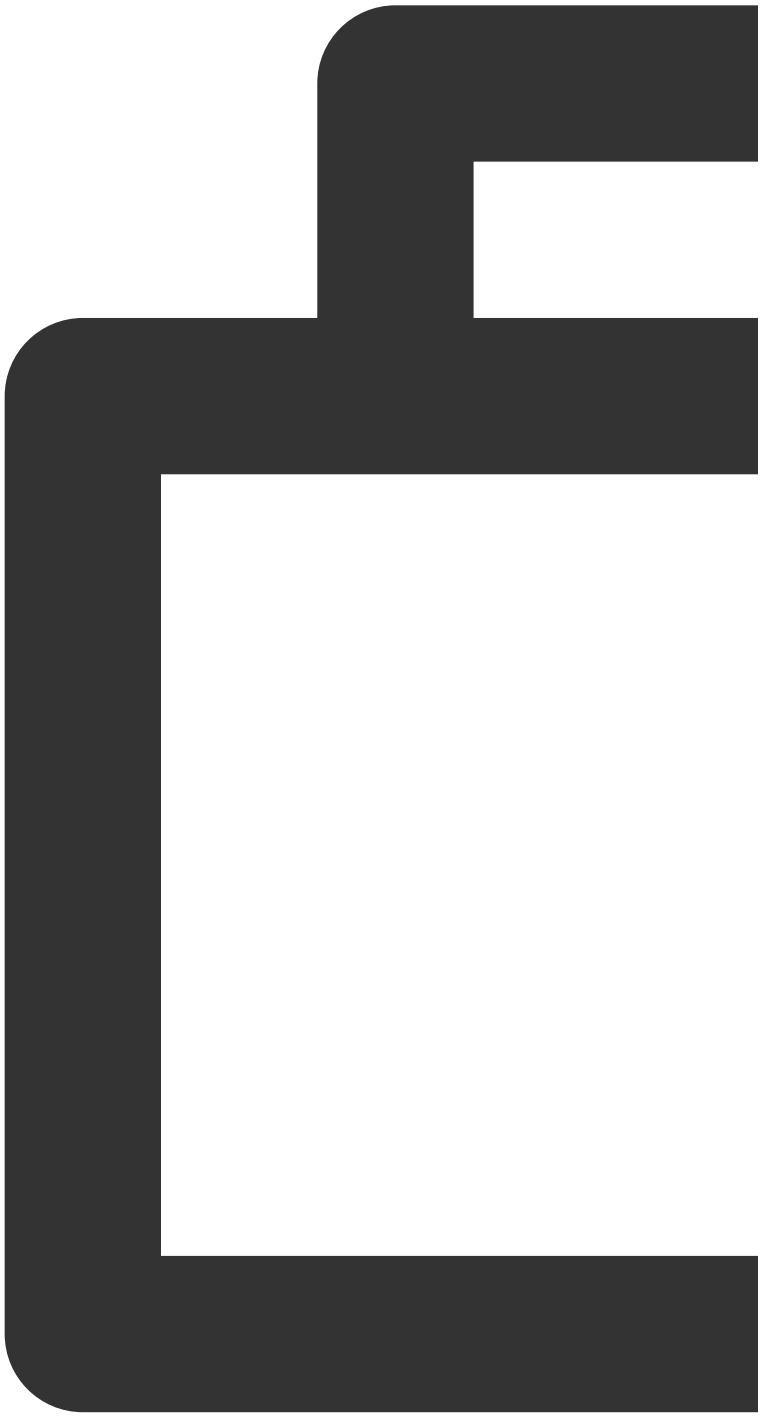
```
    sdk.getCommonFilter().then(res => {
        filterList = res
    })
})
sdk.on('cameraReady', async () => {
    // 在 cameraReady 回调中可以更早地获取输出画面，此时初始化传入的美颜参数还未生效
    // 适用于需要尽早地展示画面，但不要求画面一展示就有美颜的场景
    // 后续美颜生效后无需更新stream，SDK内部已处理
    const arStream = await ar.getOutput();
    // 本地播放
    // localVideo.srcObject = arStream

})
sdk.on('ready', () => {
    // 在 ready 回调中获取输出画面，此时初始化传入的美颜参数已生效
    // 区别上述cameraReady中获取output，适用于画面一展示就要有美颜的场景，但不要求尽早地展示画面
    // 根据自身业务需求选择一种处理方式即可
    const arStream = await ar.getOutput();
    // 本地播放
    // localVideo.srcObject = arStream

    // 在 ready 回调中调用 setBeautify/setEffect/setFilter 等渲染方法
    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 // 0.1.18及以上版本支持单独设置effect中滤镜的强度，不传则默认与
    });
    sdk.setFilter(filterList[0].EffectId, 0.5)
})
```

事件	说明	回调参数
created	SDK 鉴权完成并成功创建实例时触发	-
cameraReady	SDK 的画面生成时触发，此时 output 已有画面但美颜仍无法生效	-
ready	SDK 内部检测初始化完成时触发，此时 output 画面已有美颜，也可以设置新的特效	-

error	SDK 发生错误时触发	error 对象

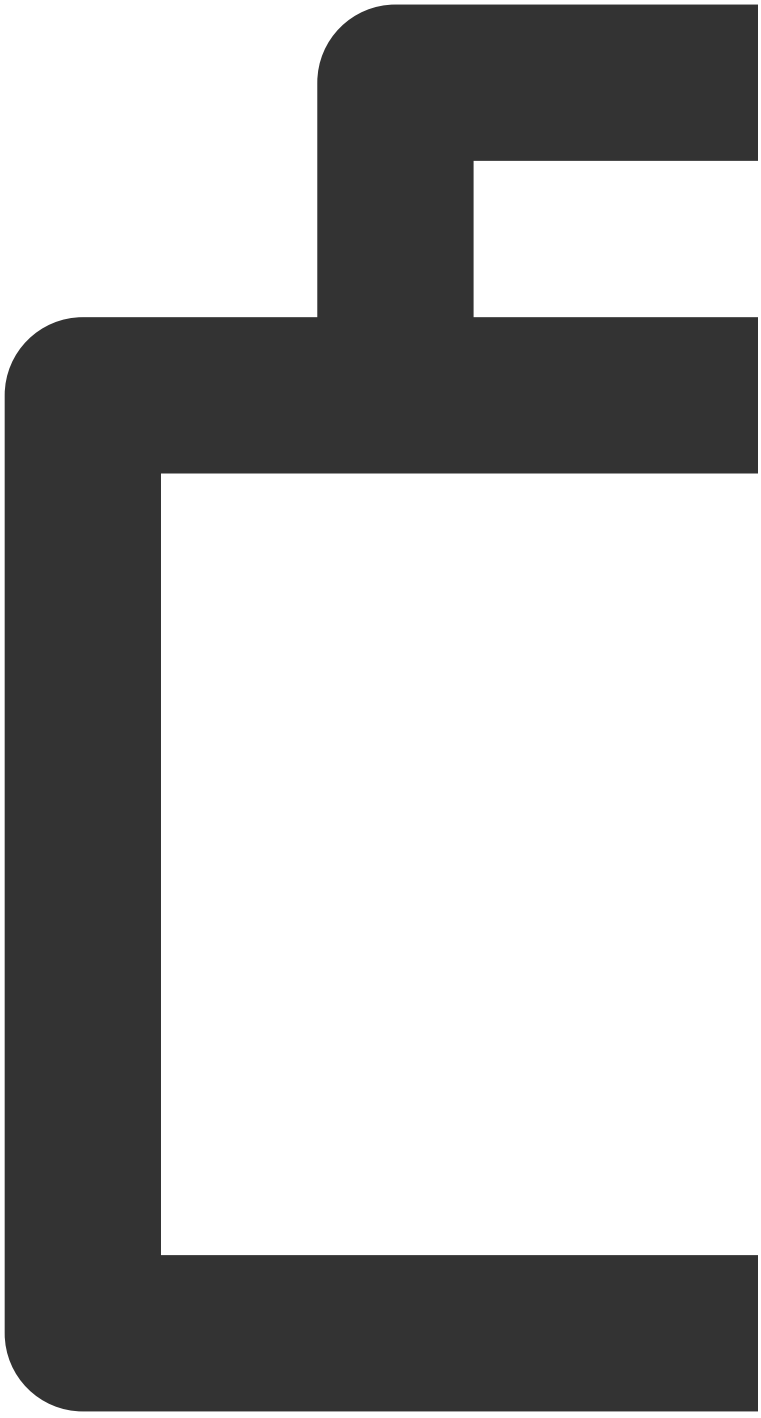
对象方法

接口	参数
async getOutput(fps)	fps：设置输出的媒体流帧率，默认无须设置
setBeautify(options)	

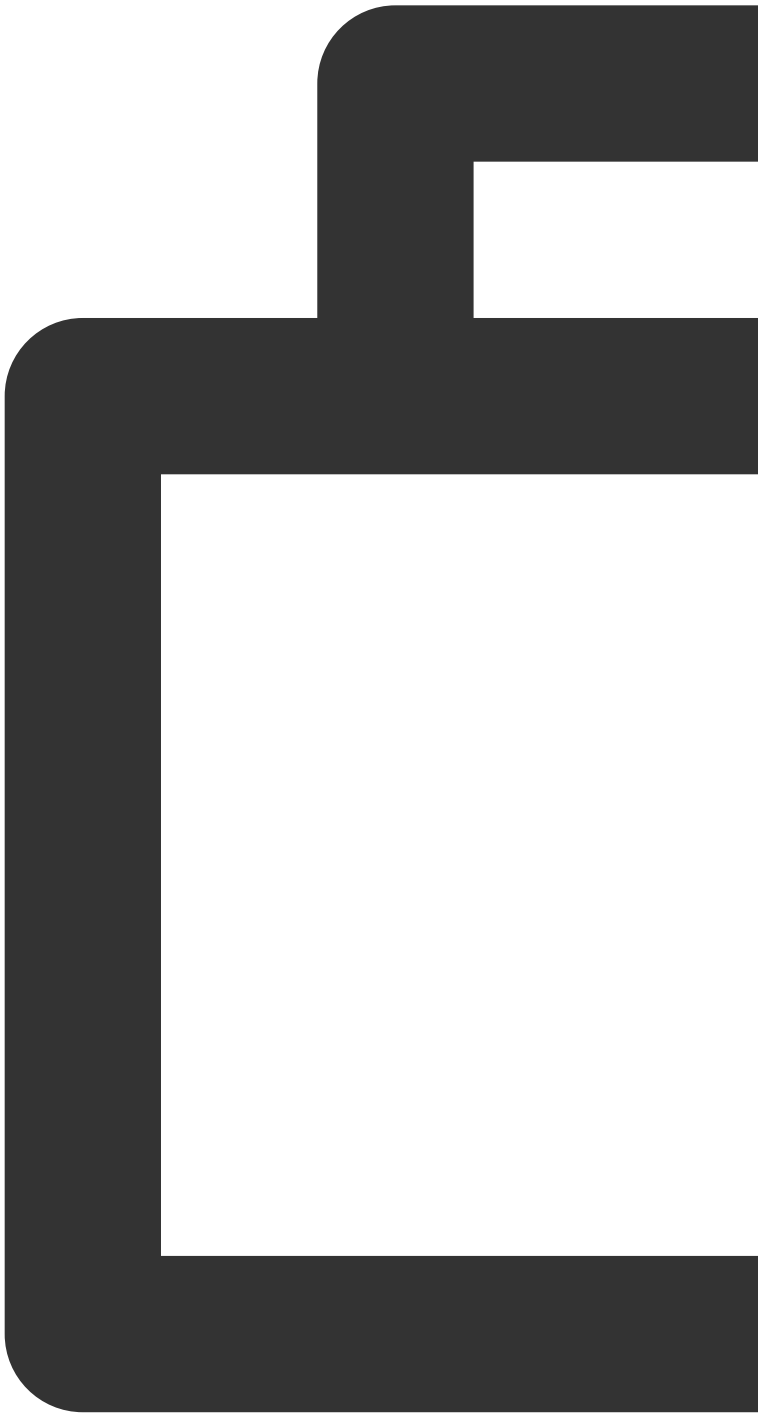


```
type BeautifyOptions = {
  whiten?: number, // 美白 0-1
  dermabrasion?: number // 磨皮0-1
  lift?: number // 瘦脸0-1
  shave?: number // 削脸0-1
```

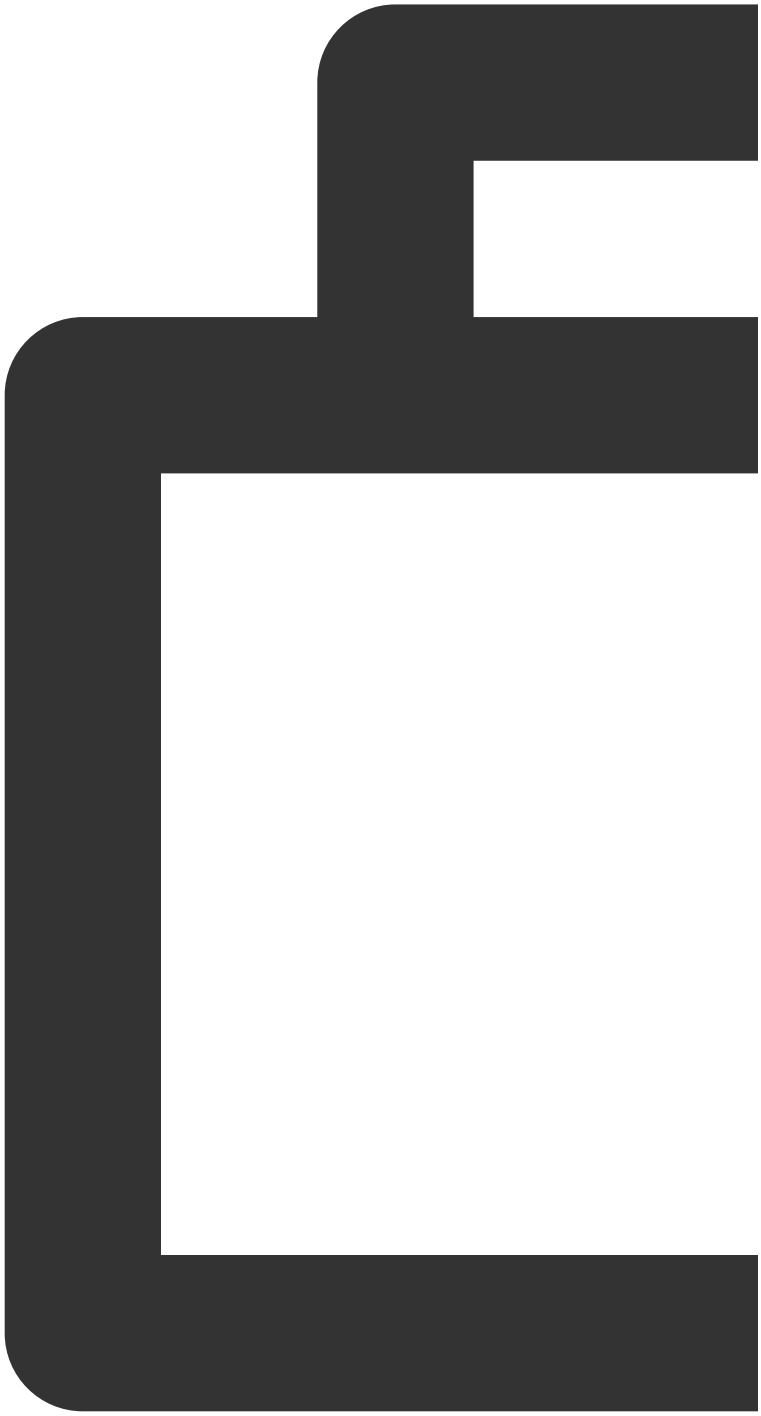
	eye?: number // 大眼0-1 chin?: number // 下巴0-1 }
setEffect(effects, callback)	effects：特效 ID | effect 对象 | 特效 ID / effect 数组



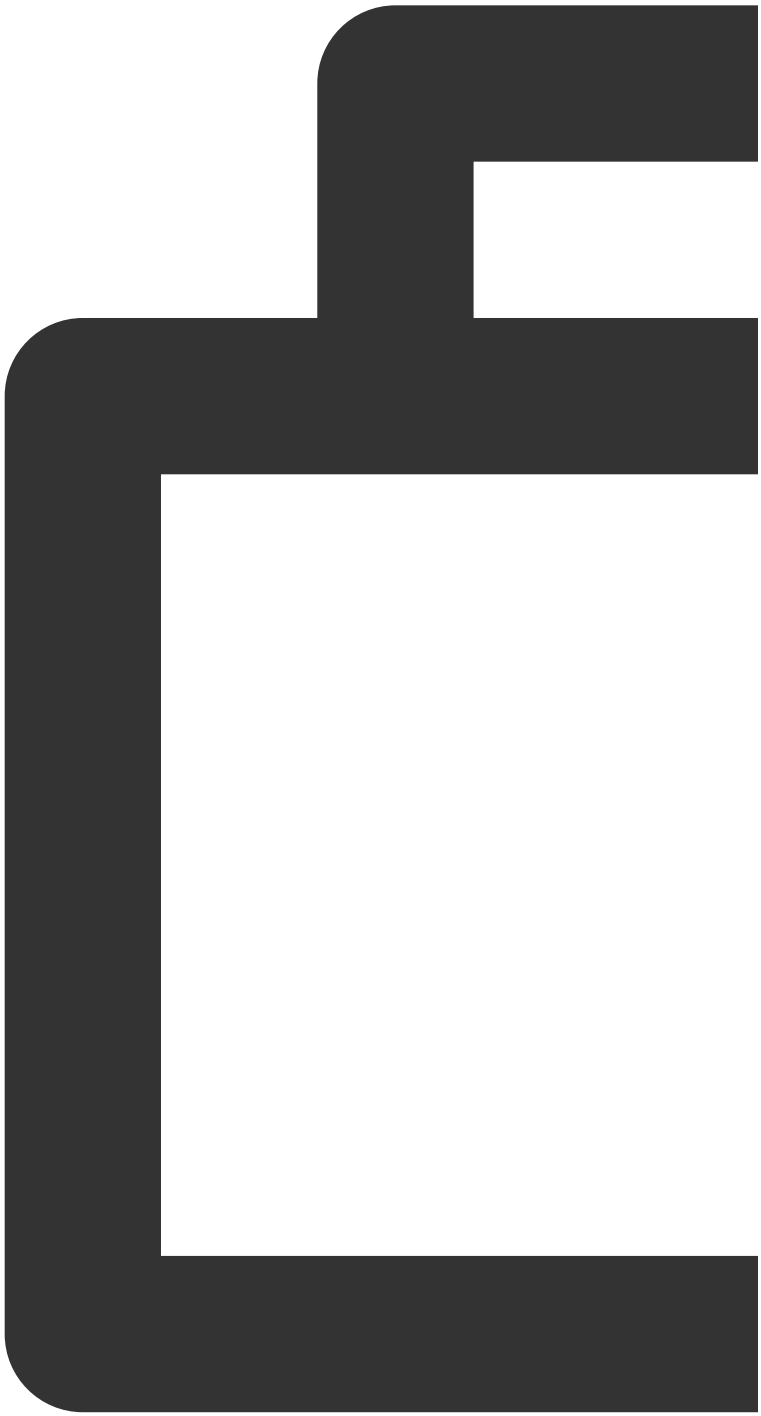
	`` effect:{ id: string, intensity: number, // 特效强度，默认1，范围0-1 filterIntensity: number // 单独控制特效中的滤镜强度 } ``` callback：设置成功的回调
setAvatar(params)	



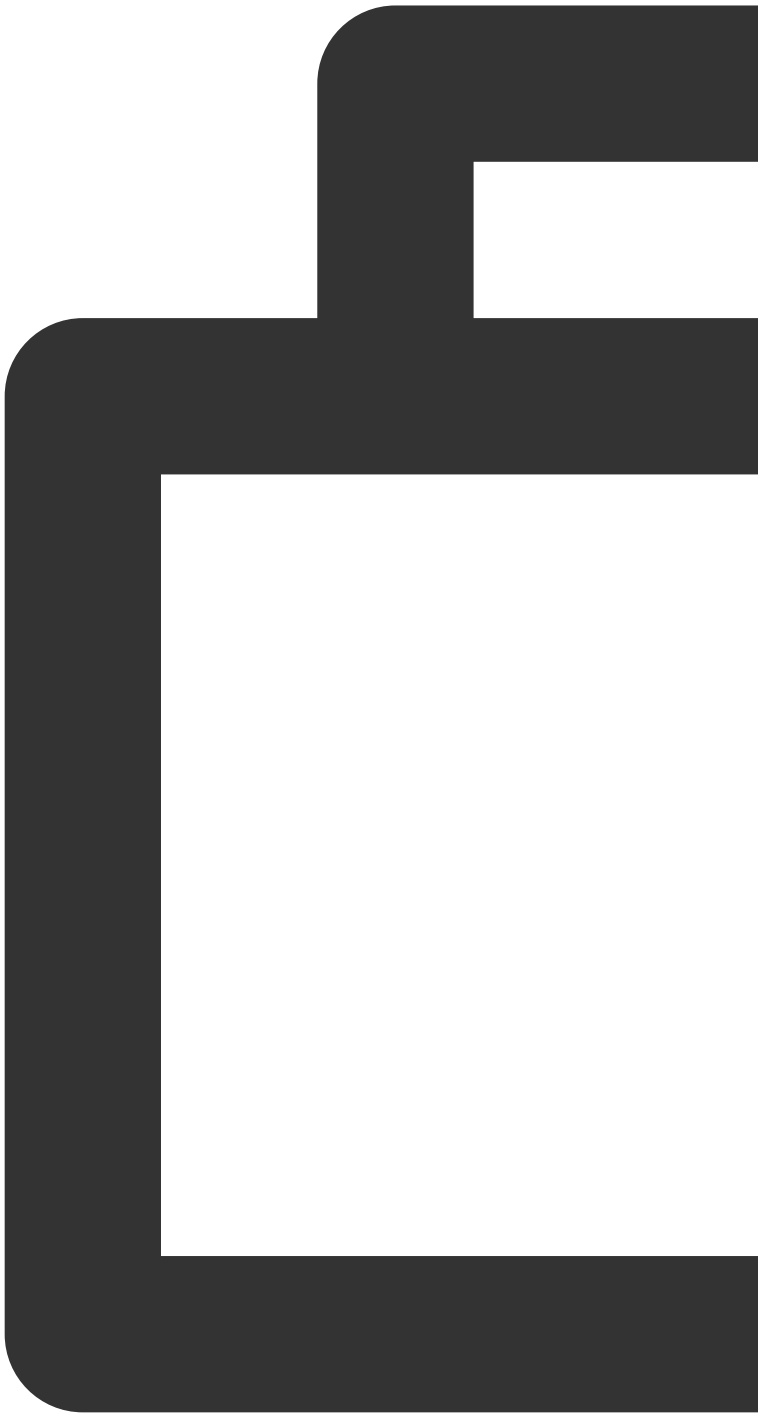
```
{
    mode: 'AR' | 'VR',
    effectId?: string, // 透传effectId使用内置模型
    url?: string, // 透传url使用自定义模型
    backgroundUrl?: string, // 背景图片链接，仅在VR模式
```

	`}`
setBackground(options)	 `{` `type: 'image|blur|transparent',`

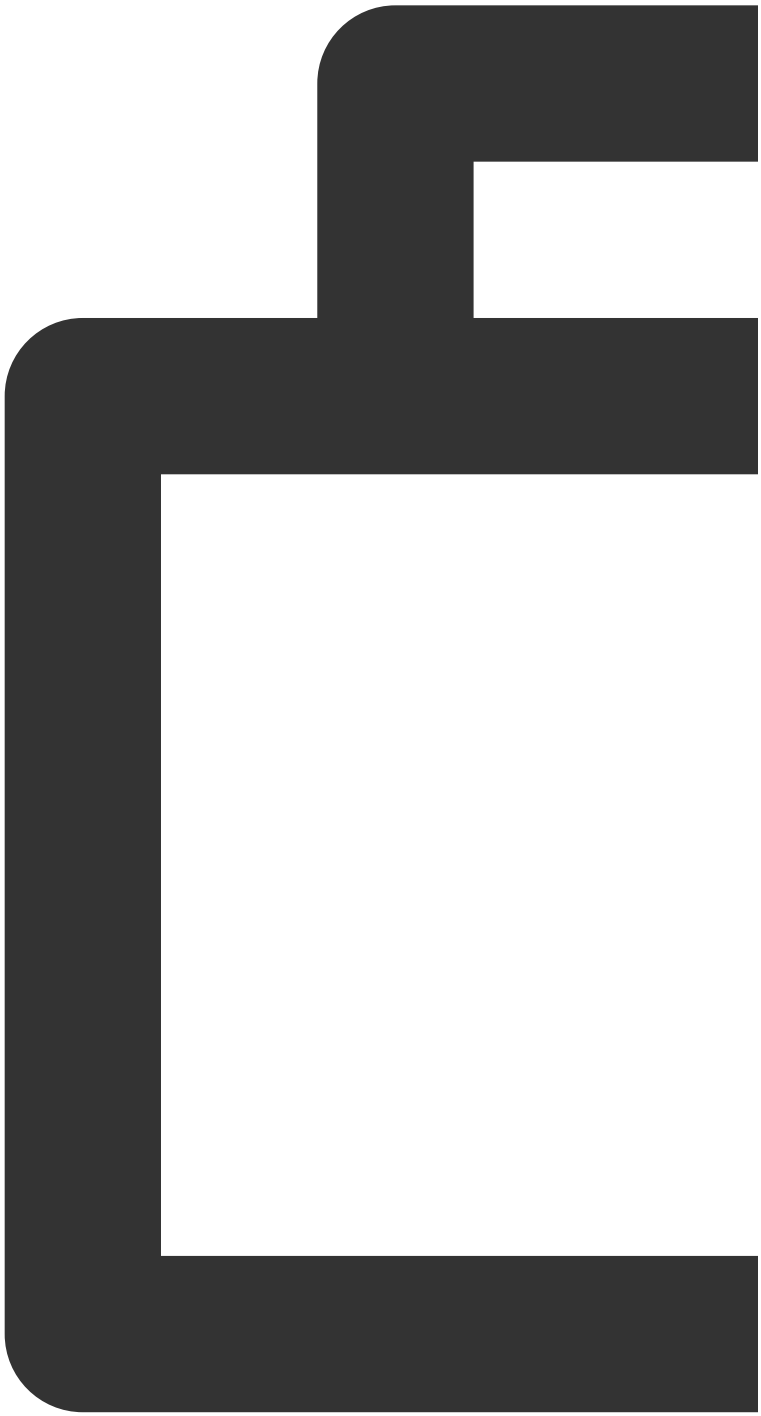
	`` src: string // 仅image类型需要 } ```
setFilter(id, intensity, callback)	id：滤镜 ID intensity：滤镜强度，范围0-1 callback：设置成功回调
getEffectList(params)	



```
{
    PageNumber: number,
    PageSize: number,
    Name: '',
    Label: Array,
```

	Type: 'Custom|Preset' }
getAvatarList(type)	 type = 'AR' | 'VR'

getEffect(effectId)	effectId：特效 ID
getCommonFilter()	-
async updateInputStream(src:MediaStream) **（0.1.19版本后支持）**	src：新的输入流MediaStream
disable()	-
enable()	-
async startRecord()	-
async stopRecord()	

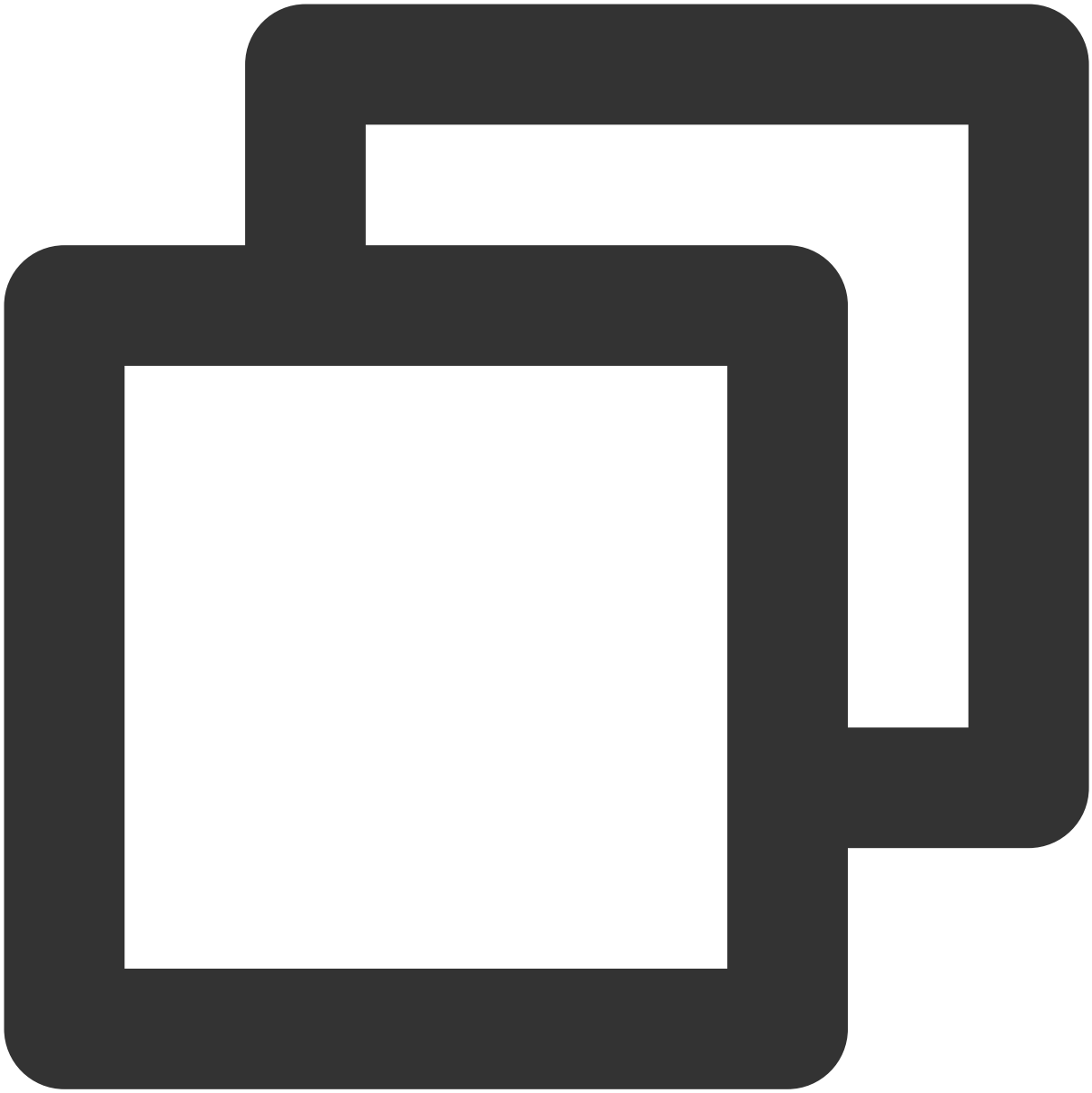


```
{
    useOriginAudio: boolean, // 是否录制视频原声
    musicPath: string, // 背景音乐地址，useOriginAudi
}
```

async takePhoto()	-
destroy()	-

错误处理

在 error 回调返回的对象中包含错误码与错误信息以方便进行错误处理。



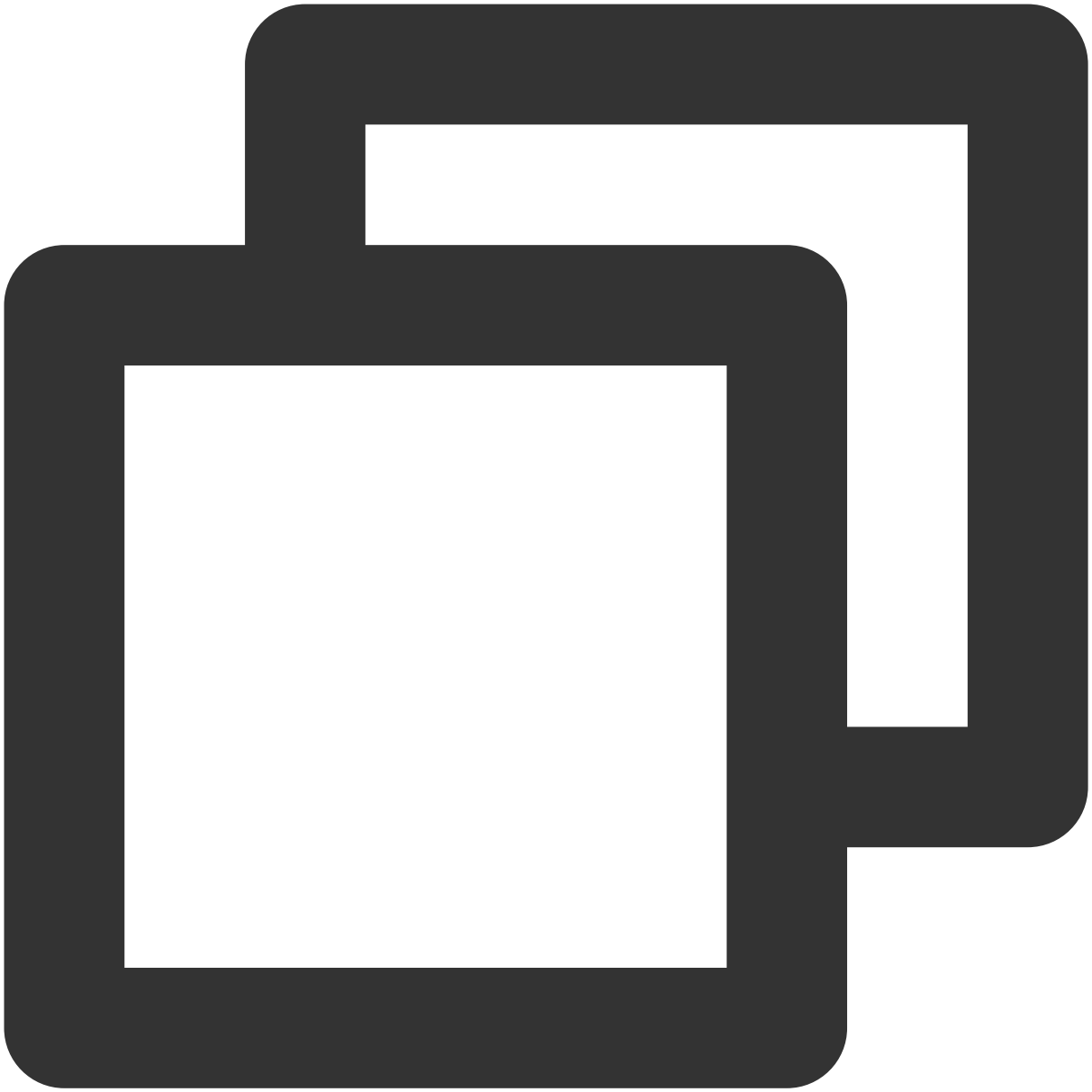
```
sdk.on('error', (error) => {
    // 在 error 回调中处理错误
    const {code, message} = error
    ...
})
```

错误码	含义	备注
10000001	当前浏览器环境不兼容	建议用户使用 Chrome、Firefox、Safari、微信浏览器访问

10000002	当前渲染上下文丢失	-
10000003	渲染耗时长	考虑降低视频分辨率或屏蔽功能
10000005	输入源解析错误	-
10000006	浏览器特性支持不足，可能会出现卡顿情况	建议用户使用 Chrome、Firefox、Safari、微信浏览器访问
10001101	设置特效出错	-
10001102	设置滤镜出错	-
10001103	特效强度参数不正确	-
10001201	调起用户摄像头失败	-
10001202	摄像头中断	-
10001203	没有获取到摄像头权限	需要开启摄像头权限，设置-隐私-相机开启
20002001	缺少鉴权参数	-
20001001	鉴权失败	请确认是否创建 License，请确认签名是否正确
20001002	接口请求失败	回调会回传接口返回的数据，具体信息请参见 接口错误码
20001003	设置特效接口鉴权失败	无权访问的接口，基础版 License 无法使用高级版 License 功能
30000001	小程序 startRecord 失败	-
30000002	小程序 stopRecord 失败	-
40000000	未捕获的异常	-
40000001	当前使用 SDK 版本过低，部分特效无法正确展示，请升级 SDK 版本	-
50000002	分辨率改变导致特效丢失	需要重新设置特效

处理当前渲染上下文丢失

部分 PC 在长期切后台的场景可能触发处理 contextlost 错误，可以调用 `ArSdk.prototype.resetCore(input: MediaStream)` 恢复渲染上下文。



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