

密级状态： 绝密() 秘密() 内部资料() 公开(√)

RK SVEP SR User Guide

(技术部，图形计算平台中心)

文件状态： <input type="checkbox"/> 草稿 <input type="checkbox"/> 正在修改 <input checked="" type="checkbox"/> 正式发布	当前版本：	2.0.5
	作 者：	GPU Team
	完成日期：	2023-08-29
	审 核：	熊伟
	审核日期：	2023-08-29

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

版权所有 © 2023 瑞芯微电子股份有限公司

超越合理使用范畴，非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址：福建省福州市铜盘路软件园 A 区 18 号

网址：www.rock-chips.com

客户服务电话：+86-4007-700-590

客户服务传真：+86-591-83951833

客户服务邮箱：fae@rock-chips.com

前言

本文主要介绍内容如下：

- SVEP (SVEP, Super Vision Enhancement Process, 超级视觉增强处理) 算法模块中的SR (Super Resolution, 超级分辨率) 的技术说明；
- Android 系统显示框架集成SVEP-SR技术对外接口说明。

适用平台

芯片平台	系统平台	系统平台版本
RK3588	Android	12

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

SR版本	HWC版本	作者	更新时间	审核	修订说明
2.0.5	1.5.143	GPU Team	2023/08/29	熊伟	<p>【优化】优化SR用户接口 【修复】修复若干内部软件问题 【修改】补充2.5/2.7章节内容 【修改】更新文档配图</p>
1.9.1	1.5.102	GPU Team	2023/07/11	熊伟	<p>【优化】优化SR算法负载 【修复】修复若干内部软件问题</p>
1.7.7	1.4.11	GPU Team	2022/10/31	熊伟	<p>【优化】优化SR带宽负载 【新增】第三方应用白名单支持应用列表 【新增】OSD-Oneline模式</p>
1.6.2	1.3.41	GPU Team	2022/08/20	熊伟	<p>【新增】运行时开关 【新增】2.7 SR应用黑名单配置章节 【新增】2.6.3 关闭OSD章节 【修改】4.3 如何确认SR授权成功章节</p>
1.6.0	1.3.34	GPU Team	2022/08/13	熊伟	<p>【优化】优化SR效果； 【优化】优化左右对比模式效果； 【新增】FAQ 4.7 问题； 【删除】全局SR模式；</p>
1.5.0	1.3.5	GPU Team	2022/06/20	熊伟	<p>【优化】优化SR系统负载； 【优化】优化SR OSD字幕效果； 【修复】解决若干内部软件问题；</p>
1.4.1	1.3.2	GPU Team	2022/06/09	熊伟	<p>【新增】完成VendorStorage授权逻辑； 【新增】完成OSD字幕修改接口；</p>
1.3.1	1.2.92	GPU Team	2022/05/11	熊伟	<p>【优化】完成SR库内存优化； 【新增】增加SR调试节点； 【优化】修改部分属性名称；</p>
1.2.1	1.2.79	GPU Team	2022/04/11	熊伟	<p>【新增】支持4K SR模型； 【修复】解决若干内部软件问题； 【新增】新增画质增强强度调整接口；</p>
1.1.2	1.2.66	GPU Team	2022/03/25	熊伟	<p>【新增】实现SR授权逻辑； 【修复】解决若干内部实现问题；</p>
1.1.1	1.2.55	GPU Team	2022/03/15	熊伟	<p>【新增】集成实现SR画质增强功能； 【新增】实现SR控制接口属性；</p>

目录

一、概述

1.1 系统集成架构

1.2 数据流处理说明

二、SR 系统功能接口说明

2.1 版本号查询接口

2.2 模式使能接口

2.3 左右对比模式

2.4 强度调整接口

2.5 VendorStorage 授权接口

 2.5.1 VendorStorage 分区介绍

 2.5.2 VendorStorage Write 实例代码

 2.5.3 SR 授权流程

2.6 OSD 字幕接口

 2.6.1 OSD修改方法介绍

 2.6.2 字幕限制信息

 2.6.3 关闭OSD字幕

 2.6.4 OSD-Oneline 模式

2.7 SR 应用黑白名单配置

 2.7.1 通用图层匹配逻辑

 2.7.2 黑名单处理流程说明

 2.7.3 白名单处理流程说明

 2.7.4 配置文件说明

三、对比双线性插值效果展示

四、FAQ

4.1 简单介绍SR具体做了哪些处理?

4.2 SR效果的评价手段和指标有哪些?

4.3 如何确认SR授权成功?

4.4 如何判断是否正确集成并开启SR功能?

4.5 SR强度原理是什么?

4.6 图片进行SR是否支持?

4.7 Android SDK如何使能SR功能?

一、概述

SR (Super Resolution, 超级分辨率) 是一项 **图像放大** 技术，这项技术利用深度神经网络补充图片纹理细节，将原始低分辨率输入重建为清晰高分辨输出，以提升视频清晰度获得更高主观感知。

目前支持的SR模式如下：

SR模式	输入分辨率	输出分辨率	实时帧率 / FPS	备注
360p	640x360	1920x1080	30	
540p	960x540	2880x1620	30	
720p	1280x720	3840x2160	30	
1080p	1920x1080	3840x2160	30	
2160p-4K	3840x2160	3840x2160	30	注意2
2160p-8K	3840x2160	7680x4320	30	注意2

注意1：若 **输入分辨率不满足** 标准SR模型的分辨率，系统端会适配到更大一级的分辨率模型进行画质增强，例如：若输入1024x600分辨率视频，则采用1280x720 SR模型进行处理；

注意2： 2160p-4K / 2160p-8K 模式主要取决于当前屏幕的显示分辨率：

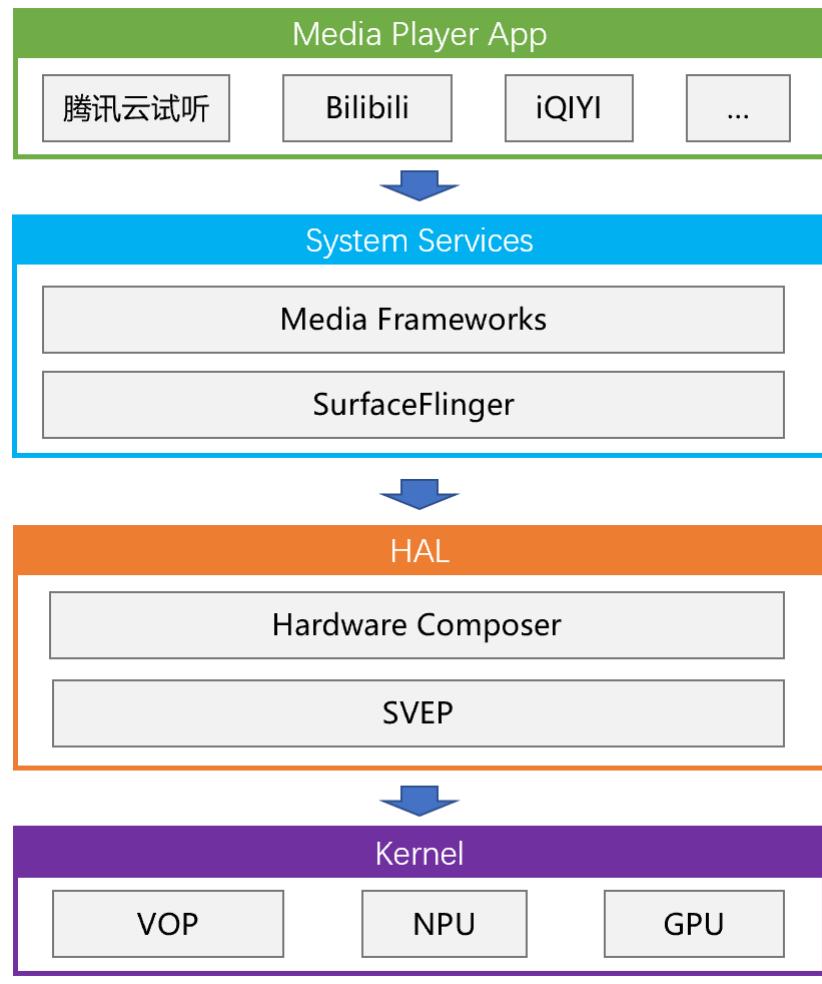
- 2160p-4K：若当前屏幕的分辨率设置为 3840x2160，则SR最大输出分辨率限制为 3840x2160（满足点对点输出），即可实现最佳显示效果；
- 2160p-8K：若当前屏幕的分辨率设置为 7680x4320，则SR最大输出分辨率可达到 7680x4320，即可实现最佳显示效果；

注意3： SR模式目前仅支持SDR片源，HDR片源将不会启动SR进行放大；

1.1 系统集成架构

目前SR集成在 Android 系统显示框架内部，

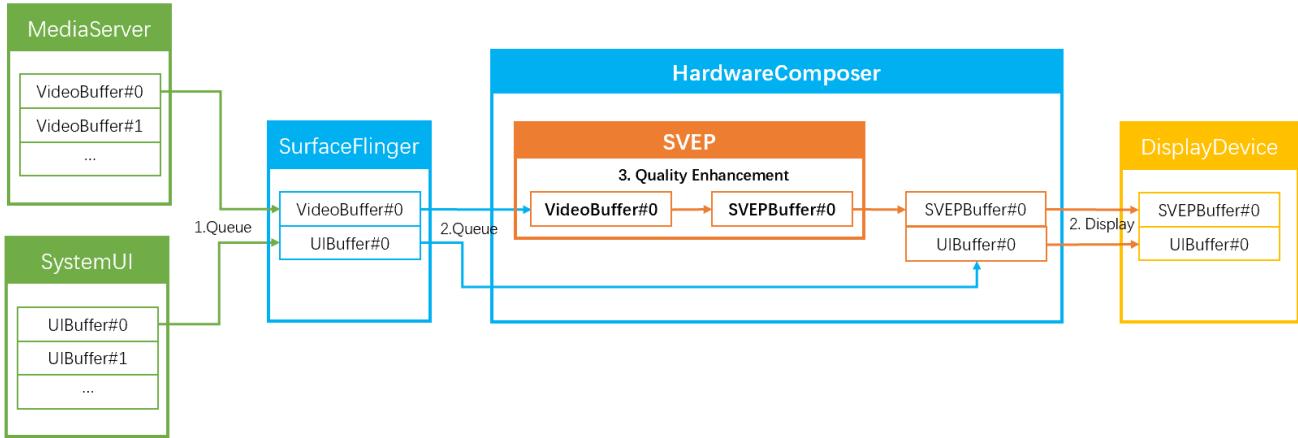
- 整体架构图：



1. 媒体应用请求Media Frameworks播放视频；
2. Media Frameworks向SurfaceFlinger提交视频帧，SurfaceFlinger请求HardwareComposer服务处理视频帧；
3. HardwareComposer服务调用SR库对视频帧执行视频增强处理；
4. HardwareComposer 获得视频画质增强帧，提交屏幕送显；

1.2 数据流处理说明

SR 数据流处理说明如下图所示：



1. 视频播放场景，系统同时提交VideoBuffer#0与UIBuffer#0图像到SurfaceFlinger 服务，请求送显；
2. SurfaceFlinger收集当前应用提交的所有待显示图像（VideoBuffer#0与UIBuffer#0），打包送给 HardwareComposer服务；
3. HardwareComposer接收到VideoBuffer#0利用SR进行画质增强，并且输出SVEPBuffer#0替代 VideoBuffer#0，并且与UIBuffer#0打包成同一帧，提交DisplayDevice送显；
4. DisplayDevice将会在下一个Vsync信号触发后将图像显示到屏幕上；

二、SR 系统功能接口说明

SVEP-SR技术目前已深度集成在系统显示后端服务，应用通过Android Property设置SVEP-SR状态，即可实现SR相关功能的配置：

- 版本查询接口：vendor.svep.version
- 功能模式接口：
 - SVEP-SR 功能开关接口：persist.sys.svep.mode
 - 左右对比模式接口：
 - persist.sys.svep.contrast_mode
 - persist.sys.svep.contrast_offset_ratio
 - SVEP-SR 强度调整接口：vendor.svep.enhancement_rate
 - SVEP-SR 授权ID配置：ro.vendor.svep.vsid

2.1 版本号查询接口

属性名称	典型值	值说明
vendor.svep.version	SR-2.0.3	SR接口库版本号
vendor.ghwc.version	HWC2-1.4.140	HWC版本号

可通过以下命令获取版本信息：

```
adb shell getprop vendor.svep.version
## 输出版本信息为: SR-2.0.3 , 具体版本号等于实际的交付版本号
adb shell getprop vendor.ghwc.version
## 输出版本信息为: HWC2-1.4.140 , 具体版本号等于实际的交付版本号
```

注意：必须要同时确认HWC与SR库版本号，若版本库不匹配可能会导致异常；

2.2 模式使能接口

属性名称	缺省值	值范围	值说明
persist.sys.svep.mode	0	0 / 1	SR模式全局开关（用户使用） 0：关闭SR模式，不进行任何画质增强处理 1：开启SR模式，仅针对视频播放图层
sys.svep.runtime_disable	0	0 / 1	SR运行时强制关闭（系统使用） 0：正常SR处理流程 1：强制关闭SR模式

注意：两项模式使能接口使用建议如下：

- SR模式全局开关：通常作为SR全局开关，例如Setting内部的SR开关选项；
- SR运行时强制关闭：通常提供给系统服务（解码模块）强制关闭SR模式，例如播放60帧片源时建议设置为1

可通过以下命令获取并设置SR使能模式：

```
## 获取:  
adb shell getprop persist.sys.svep.mode  
adb shell getprop sys.svep.runtime_disable  
  
## 设置:  
adb shell setprop persist.sys.svep.mode 1  
adb shell setprop sys.svep.runtime_disable 1
```

SR使能效果如下图：



其中OSD字幕如下图，存于此OSD说明模式正常开启：



详细参数说明：

- RKNPU-SVEP-SR: OSD默认字幕，自定义接口可查看《2.6 OSD字幕接口》进行修改；
- Input: 实际输入片源尺寸，此字幕无法自定义修改；
- Output: 实际增强输出尺寸，此字幕无法自定义修改；

2.3 左右对比模式

属性名称	缺省值	值范围	值说明
persist.sys.svep.contrast_mode	0	0 / 1	图像左右对比模式开关 分割线以左为增强效果 分割线以右为原效果 0：关闭图像对比模式 1：开启图像对比模式
persist.sys.svep.contrast_offset_ratio	0	0 / 10~90	控制分割线的偏移比例 0：开启自动模式：实现扫描效果 50：分割线偏移50%，即居中分割

persist.sys.svep.contrast_mode 说明：

图像左右对比模式开关，将会在图像画面渲染一条分割线，分割线左边为SR输出图像，右边为原始图像，可通过以下命令获取并设置SR使能模式：

```
## 获取：  
adb shell getprop persist.sys.svep.contrast_mode  
## 设置使能SR左右对比模式：  
adb shell setprop persist.sys.svep.contrast_mode 1  
## 设置关闭SR左右对比模式：  
adb shell setprop persist.sys.svep.contrast_mode 0
```

使能SR左右对比模式显示效果如下：



persist.sys.svep.contrast_offset_ratio 说明：

控制SR左右对比模式分割线的偏移比例，值表示SR输出图像比例，可通过以下命令设置：

```
adb shell setprop persist.sys.svep.contrast_offset_ratio 50
```

典型值说明

- 自定义分割线比例: persist.sys.svep.contrast_offset_ratio 值范围为建议 10-90;
- 自动扫描: persist.sys.svep.contrast_offset_ratio = 0 则分割线会依次从 10 递增到 90, 实现扫描效果。

2.4 强度调整接口

SR强度是通过调整SR模型内部权重系数，来实现对纹理细节效果的调整，强度设置越高，细节越清晰。

注意：由于V1.9.1版本算法优化后，强度0的效果相当于原算法强度5效果，故从 V1.9.1版本开始强度默认值修改为0，这样设置有利于降低系统负载；

属性名称	默认值	值范围	值说明
vendor.svep.enhancement_rate	0	0-10	0: 最低增强强度 10: 最高增强强度

SR强度调整接口，产品上可根据实际效果设置强度默认值，可通过以下命令设置SR处理强度：

```
## 设置为SR强度5
adb shell setprop vendor.svep.enhancement_rate 5
```

```
## 设置为SR强度0
adb shell setprop vendor.svep.enhancement_rate 0
```

从左往右依次为强度0/5/10的效果图，强度设置越高，图像细节越清晰：



2.5 VendorStorage 授权接口

SR算法正常执行需要授权激活码，只有授权激活通过的设备才可以体验SR算法。

注意：激活码激活目标设备后即与目标芯片OTP绑定，无法再激活其他芯片，请谨慎保管。

VendorStorage 分区介绍：可参考《Rockchip_Application_Notes_Storage_CN.pdf》文档；

2.5.1 VendorStorage 分区介绍

SR算法激活码分区规划：如下表：

算法模块	Id	分区说明
SR	17	版本小于 V2.0.0 的激活码存放位置。高版本已迁移到 ID=60。
SR	60	版本大于V2.0.0 版本的激活码存放位置。已兼容旧版本，若60未查询到激活码，会从ID=17 查询激活码。

迁移说明： VendorStorage ID 0-31 保留为通用 SDK 功能使用，SR 作为可选功能如果使用 ID=17 可能会与通用SDK 功能冲突，故V2.0.0版本开始，迁移到用于自定义区间 32-65535。

SVEP包含以下两个图像算法：

- SR (Super Resolution, 超级分辨率) ；
- MEMC (Motion Estimation and Motion Compensation, 运动估计与运动补偿) ；

对应的已使用的ID号如下表：

模块	Id	Len	分区说明
SR	17	50	版本小于 V2.0.0 的激活码存放ID。高版本已迁移到 ID=60。
SR	60	50	版本大于V2.0.0 版本的激活码存放ID。已兼容旧版本，若60未查询到激活码，会从ID=17 查询激活码。
MEMC	62	50	MEMC激活码存放ID

2.5.2 VendorStorage Write 实例代码

SR 默认使用 VendorStorage ID=60 分区，如果客户需要定制化修改，可以通过配置如下ro属性：

```
# 可将 ro.vendor.svep.vsid=61 写入 /vendor/build.prop 文件中
# 指定ID=61作为SR激活码存放分区
ro.vendor.svep.vsid
```

规定VendorStorage 长度设置为 50 char

```
// 关键信息: ID=17, Len=50
#define VENDOR_RKNPU_AUTHOR_ID 17
int vendor_storage_write_test (const char* str){
```

```

u32 i;
int ret, sys_fd;
u8 p_buf [2048]; /* malloc req buffer or used extern buffer */
struct rk_vendor_req *req;
req = (struct rk_vendor_req *) p_buf;
sys_fd = open ("/dev/vendor_storage", O_RDWR, 0);
if (sys_fd < 0){
    printf("vendor_storage open fail\n");
    return -1;
}
req->tag = VENDOR_REQ_TAG;
req->id = VENDOR_RKNPU_AUTHOR_ID;
req->len = 50; /* data len */

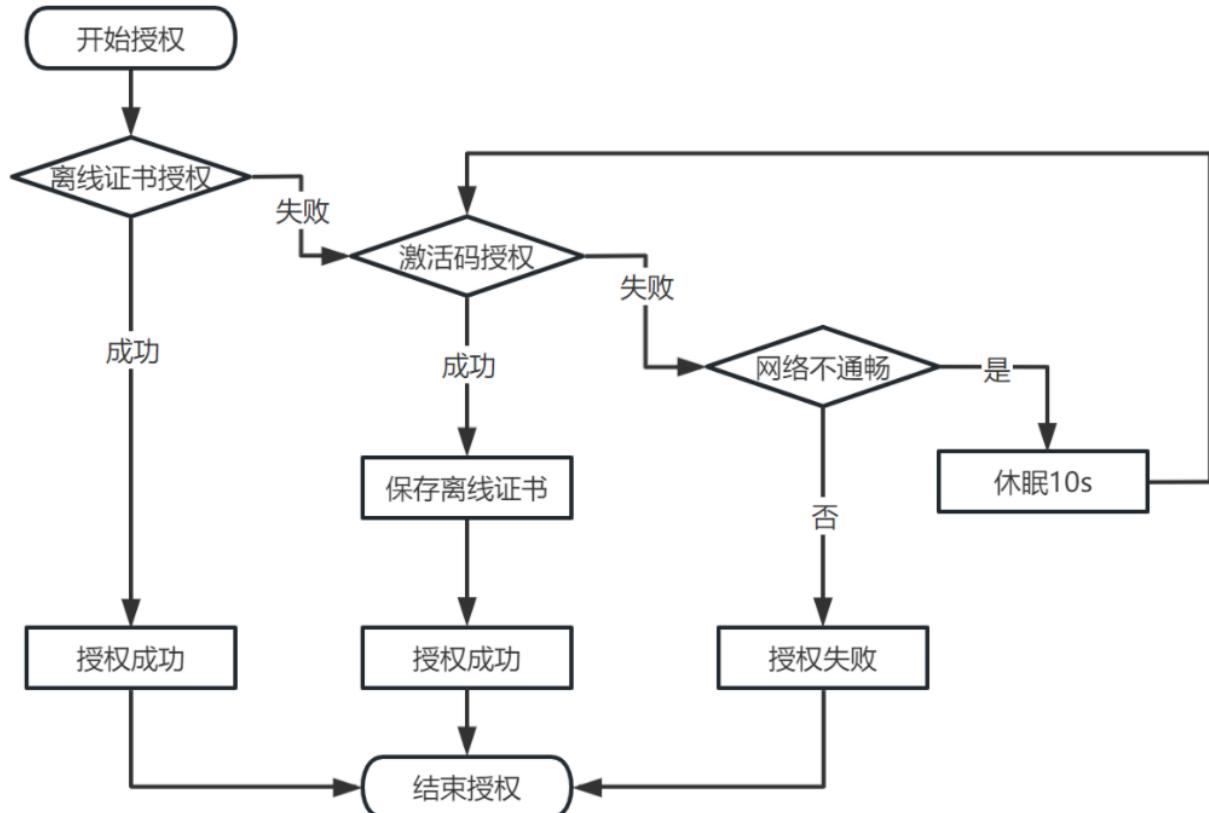
for (i = 0; i < strlen(str); i++)
    req->data [i] = str[i];

print_hex_data ("vendor write:", req, req->len);
ret = ioctl (sys_fd, VENDOR_WRITE_IO, req);
if (ret){
    printf ("vendor write error\n");
    return -1;
}
return 0;
}

```

2.5.3 SR 授权流程

算法授权实现流程图如下：



1. 离线证书授权：初次授权，由于离线证书不存在，则离线证书授权失败，进入激活码授权流程；
2. 激活码授权：获取VendorStorage ID=62 激活码，进行网络授权：
 - 若由于网络原因导致授权失败，则休眠10s再尝试激活码授权；
 - 若激活码无效则直接判断授权失败；
3. 初次授权成功后，系统会将离线授权证书/**/data/system/svep_key.lic**位置，用于下次离线授权；
4. 授权成功；
5. 下次授权直接使用离线授权证书进行授权，离线授权证书为授权激活码与设备识别码生成的唯一加密码，仅可在授权设备上可用；

注意：

部分设备的CPU序列号为0，即未正确设置OTP的芯片，此类芯片可能在授权的过程中存在问题，建议更换有正常的序列号设备进行授权。

2.6 OSD 字幕接口

2.6.1 OSD修改方法介绍

1. 按如下补丁修改：

```
hardware/rockchip/libssvep$ git diff
diff --git a/include/SvepType.h b/include/SvepType.h
index 55238a8..7efc4ab 100644
--- a/include/SvepType.h
+++ b/include/SvepType.h
@@ -38,7 +38,7 @@ namespace android {
#define SVEP_AVG_COST_TIME_NAME      "vendor.svep.avg_cost_time"

// OSD string interface.
-#define SVEP OSD VIDEO STR L"RKNPU-SVEP-SR"
+#define SVEP OSD VIDEO STR L"Rockchips-SVEP-SR"

// Vendor Storage ID.
```

2. 重新编译 hardware/rockchip/hwcomposer/drmhw2 目录：

```
mmm hardware/rockchip/hwcomposer/drmhw2 -j4
```

3. 更新 hwcomposer.rk30board.so 文件，即可修改字幕：

2.6.2 字幕限制信息

长度：自定义OSD对长度存在限制，限制信息如下：

字体类型	大写英文字母	小写英文字母	简体中文	阿拉伯数字
最大支持长度	19	22	14	29

其他：目前仅支持对第一行OSD字幕进行修改，剩余 Input/Output 信息无法修改

2.6.3 关闭OSD字幕

属性名称	缺省值	值范围	值说明
persist.sys.svep.disable_sr_osd	0	0 / 1	0：使能OSD字幕 1：关闭OSD字幕

2.6.4 OSD-Oneline 模式

OSD-Oneline模式即仅保留一行字幕，考虑到部分场景OSD字幕可能会遮挡视频，影响客户观看体验，故提供OSD-Oneline模式减少OSD字幕。相关属性配置如下：

属性名称	默认值	值范围	值说明
persist.sys.svep.enable_oneline_osd	0	0-1	0: 关闭 1: 开启
persist.sys.svep.oneline_osd_wait_second	12	> 0	正常OSD切换到Oneline模式需要等待的秒数

可通过以下命令设置OSD-Oneline模式：

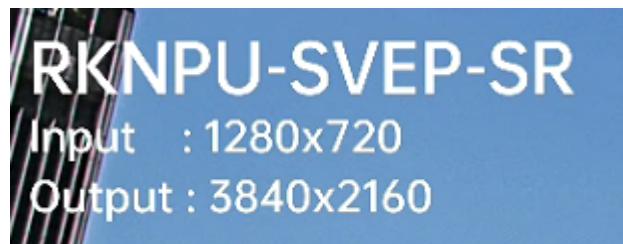
```
## 开启OSD-Oneline模式，并设置5s后进入online模式
adb shell setprop persist.sys.svep.enable_oneline_osd 1
adb shell setprop persist.sys.svep.oneline_osd_wait_second 5

## 关闭OSD-Oneline模式
adb shell setprop persist.sys.svep.enable_oneline_osd 0
```

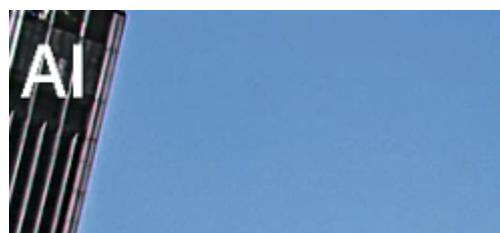
效果展示：配置情况如下：

- persist.sys.svep.enable_oneline_osd=1
- persist.sys.svep.oneline_osd_wait_second=12

1. 播放视频OSD，播放时间 < 12s，OSD显示为正常的三行字幕，如下图：



2. 视频播放时间 > 12s，OSD切换为单行字幕，如下图：



OSD-Oneline字幕自定义方法:

```
hardware/rockchip/libsvep/include/sr$ git diff
diff --git a/include/sr/SrType.h b/include/sr/SrType.h
index c6e8d61..4d53e74 100644
--- a/include/sr/SrType.h
+++ b/include/sr/SrType.h
@@ -33,7 +33,7 @@
 "persist.sys.svep.oneline_osd_wait_second" `

// One line OSD
-#define SR OSD VIDEO ONELINE STR L"AI"
+#define SR OSD VIDEO ONELINE STR L"AI-OnelineMode"
// 30hz, 360 is 12 second.
#define SR OSD VIDEO ONELINE CNT 360
```

注意：只有SR模型切换才会重新计算 WaitTime，举例如下：

- 同为1080p片源切换过程中，则一直保持OnelineMode模式；
- 1080p与720p片源切换过程中，每次切换视频播放，均会经过正常OSD到OnelineMode切换；

最终效果如下：

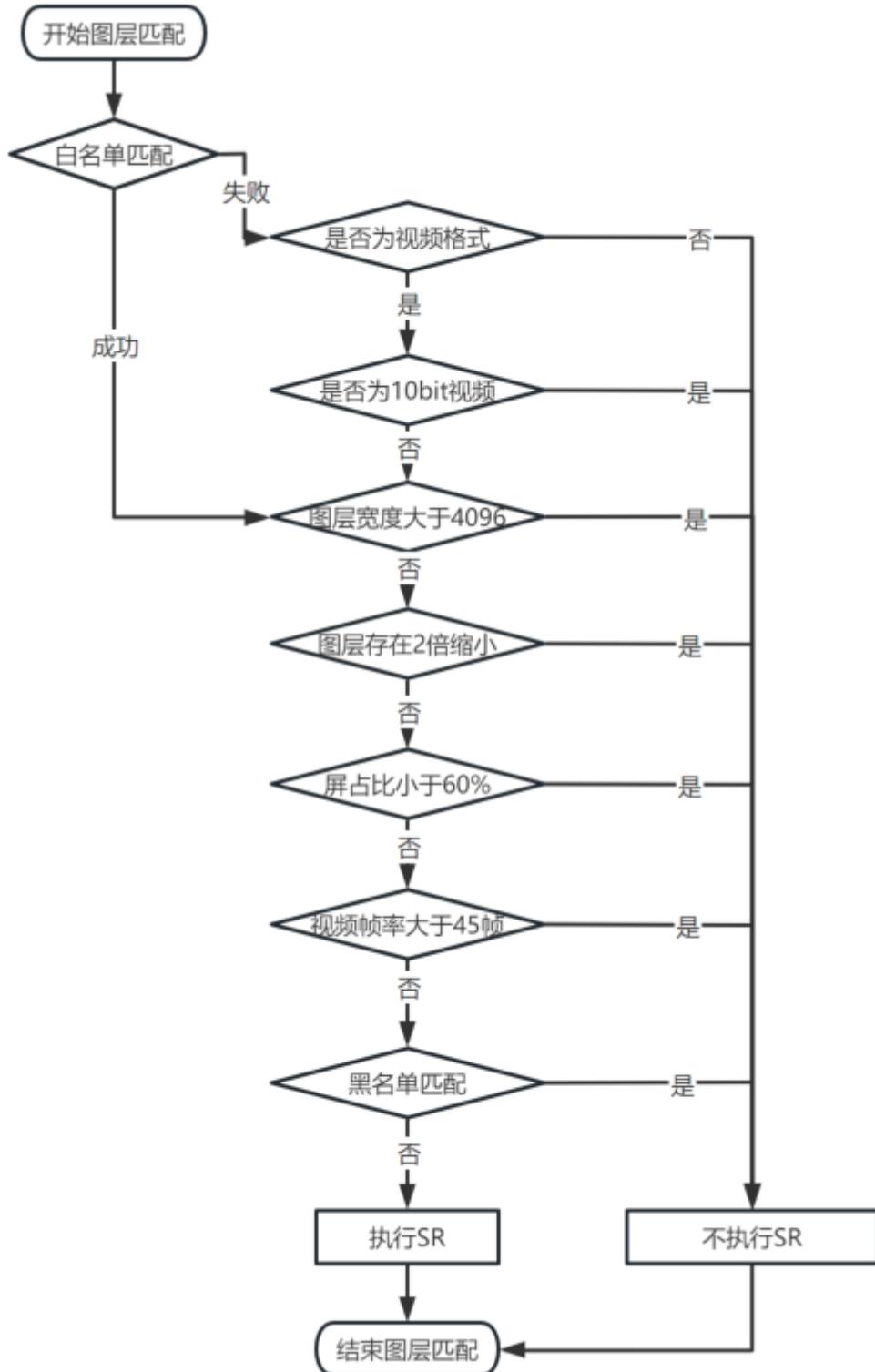


2.7 SR 应用黑白名单配置

目前SR图层匹配流程集成在HardwareComposer服务内部，主要针对刷新率为30帧的视频图层进行MEMC处理，不建议对UI图层进行MEMC操作，对应的处理逻辑如“2.7.1 图层匹配逻辑”介绍。

2.7.1 通用图层匹配逻辑

通用图层匹配逻辑如下图，该逻辑目前集成在 Android HardwareComposer服务内部，目前匹配规则如下图：



1. 视频大于宽度4096，不使用 SR;

2. 非YUV格式，且不在白名单内，不使用SR；
3. 10bit HDR 片源，不使用SR；
4. 图层本身存在2倍缩小，不使用SR；
5. 图像屏占比小于60%，不使用SR；

```
## 此项可通过以下命令修改，修改屏占比边界为 40%，即屏占比小于40% 不使用SR：
adb shell setprop vendor.hwc.disable_svep_dis_area_rate 40
```

6. 刷新率大于45帧，不使用SR；

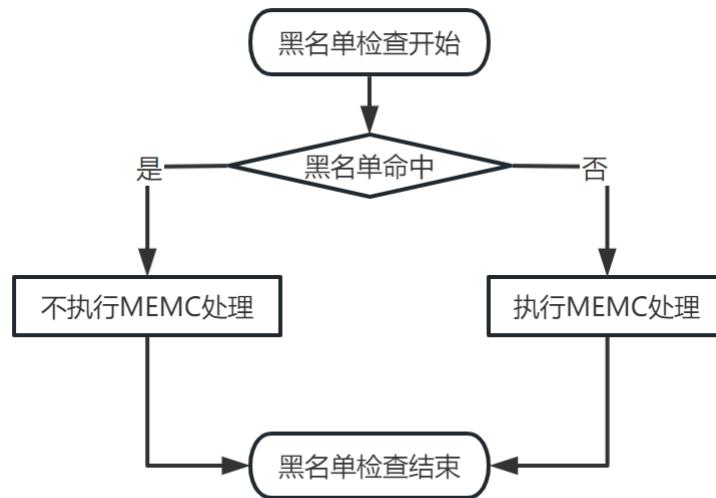
```
# 图层刷新率可通过以下命令查看：
adb shell dumpsys SurfaceFlinger
# 查看 HWC 对应图层的 mFps 参数，如下图：
```

-- HWC2 Version HWC2-1.3-95-1-X-patches by bin.liverock-chips.com --											
DisplayId=0 Connector=400 Type = HDMI-A-1, Connector state = DRM-JPEG_CONNECTED											
NumberLayers=2 activeModeId=3 3840x2160@60.00, colorMode = 0 bStandardSwitchResolution=0											
id	z	sf-type	hwc-type	handle	transform	blnd	source crop (l,t,r,b)	frame	dataspace	mfps	name
0022	1.000	Device	Device	00b40000741101da10	None	None	0.0, 0.0, 864.0, 486.0	0, 0, 1920, 1080	10010000	29.8	SurfaceView[com.ktcp.launcher/com.ktcp.videoview]
0012	1.001	Device	Device	00b40000741102e930	None	Premultipl	0.0, 0.0, 1920.0, 1080.0	0, 0, 1920, 1080	0	0.2	viewRootImpl[DetailCoverActivity]#3(BLAST r3) 0x18c0000045

2.7.2 黑名单处理流程说明

部分应用可能存在无需对视频进行MEMC处理的场景，如要求低延迟的云游戏场景，故提供 /vendor/etc/HwcSvepMemcEnv.xml 配置，允许客户将应用图层名加入黑名单，系统MEMC处理流程检测黑名单上的图层后，不进行MEMC处理增强。

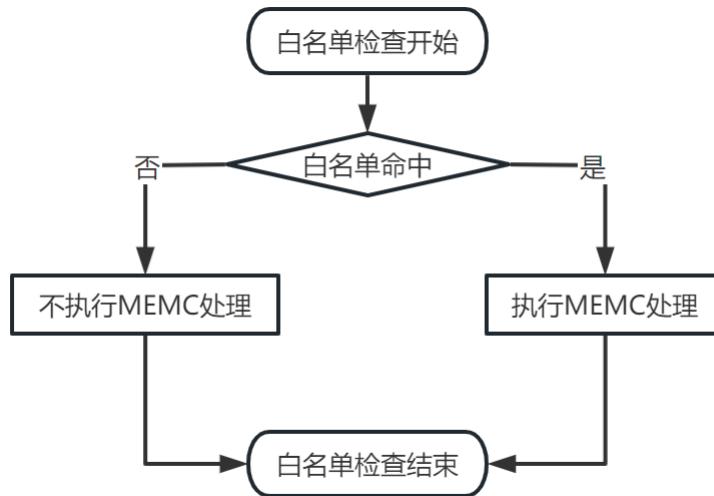
黑名单配置具体处理流程如下图：



2.7.3 白名单处理流程说明

部分客户应用可能不满足通用图层匹配逻辑中的视频格式逻辑，故提供白名单，绕过这部分限制。

白名单配置具体处理流程如下图：



2.7.4 配置文件说明

参考文件目录位于以下目录：

```

# SDK工程路径：
SDK_Project/hardware/rockchip/hwcomposer/drmhwc2/res/HwcSvepEnv.xml

# 设备路径：
/vendor/etc/HwcSvepEnv.xml
  
```

XML文件格式说明如下：

```

<?xml version="1.0" encoding="utf-8"?>
<!-- Svep xml -->
<HwcSvepEnv Version="1.1.1" >

<Blacklist>  <!-- 黑名单头部结构 -->
  <!-- 黑名单的关键词，通常为应用图层名字 -->
  <BlackKeywords>
    SurfaceView[android.rk.RockVideoPlayer/android.rk.RockVideoPlayer.VideoPlayActivity]</BlackKeywords>
  <!-- 可添加多个并列 BlackKeywords -->
  <BlackKeywords> RockVideoPlayer</BlackKeywords>
  <!-- 可添加多个 BlackKeywords -->
  <BlackKeywords> android.rk.RockVideoPlayer.VideoPlayActivity</BlackKeywords>
</Blacklist>

<whitelist>
  <!-- 爱奇艺：标准、Lite、巴布版本适用如下白名单 -->
  <whiteKeywords>SurfaceView[com.qiyi.video]</whiteKeywords>
  <!-- 抖音 -->
  <whiteKeywords>ViewRootImpl[SplashActivity]</whiteKeywords>
  <!-- 优酷视频 -->
  <whiteKeywords>SurfaceView[com.youku.phone/]</whiteKeywords>
  <!-- 哔哩哔哩 -->
  
```

```

<whiteKeywords>SurfaceView[tv.danmaku.bili</whiteKeywords>
<!-- 虎牙直播 -->
<whiteKeywords>SurfaceView[com.duowan.kiwi</whiteKeywords>
<!-- 腾讯课堂 -->
<whiteKeywords>ViewRootImpl[RecruitAvActivity]</whiteKeywords>
<whiteKeywords>ViewRootImpl[FCourseDetailActivity]</whiteKeywords>
<whiteKeywords>ViewRootImpl[webOpenUrlActivity]</whiteKeywords>
<!-- 掌门一对一辅导 -->
<whiteKeywords>ViewRootImpl[RecordedLessonDetailActivity]</whiteKeywords>
</whitelist>
</HwCSvpEnv>

```

应用图层名字获取：

1. 在目标场景按如下命令抓打印日志：

```
adb shell dumpsys SurfaceFlinger
```

2. 输出日志文件的如下红框部分则为目标应用图层名字，如下红框部分：

```

Planner is disabled
h/w composer state:
h/w composer enabled
-- HWC2 Version HWC2-1.3.40 by bin.li@rock-chips.com --
DisplayId=0, Connector 431, Type = DSI-1, Connector state = DRM_MODE_CONNECTED
NumHwLayers=3, activeModeId=16, 1080x1920p60.00, colorMode = 0, bStandardSwitchResolution=0
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | z | sf-type | hwc-type | handle | transform | bind | source crop (l,t,r,b) | frame | dataspace | mFps | name
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
0033 | 000 | Device | Device | 00b400007b560bfa70 | None | None | 0.0, 0.0, 1920.0, 1080.0 | 0, 0, 1080, 1920 | 10001 | 20.8 | SurfaceView[android.rk.RockVideoPlayer/android
0006 | 001 | Client | Client | 00b400007b560bda0 | None | Premultipl | 0.0, 0.0, 1080.0, 48.0 | 0, 0, 1080, 48 | 0 | 0.0 | ViewRootImpl[ScreenDecorOverlay]#0(BLAST Consumer)
0005 | 002 | Client | Client | 00b400007b560b4710 | None | Premultipl | 0.0, 0.0, 1080.0, 48.0 | 0, 1872, 1080, 1920 | 0 | 0.0 | ViewRootImpl[ScreenDecorOverlayBottom]#2(BLAST
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

```

3. 完整应用名称为如下字符串：

```

SurfaceView[android.rk.RockvideoPlayer/android.rk.RockvideoPlayer.VideoPlayActivity]#8(
BLAST Consumer)8
# 其中,
# "#后的字符串"8(BLAST Consumer)8"为Android系统自动添加, 不建议作为识别目标图层的字符串
# "#"以前的字符串可作为目标图层的关键字符串

```

注意：只要应用名称包含 BlackKeywords，则认为黑名单图层命中，则不进行SR画质增强处理，采用直通模式显示；

例如： android.rk.RockVideoPlayer 或 android.rk.RockVideoPlayer.VideoPlayActivity 均可对完整 SurfaceView[android.rk.RockVideoPlayer/android.rk.RockVideoPlayer.VideoPlayActivity] 应用图层名进行黑名单命中；

三、对比双线性插值效果展示

下图为启用SR模型处理后的放大图像：



- 输入分辨率1280x720
- 输出分辨率3840x2160
- 垂直/水平方向放大3倍

细节展示，**左图**为SR处理后的放大图像，**右图**为源视频采用双线性插值放大后的图像：



SR相对比双线性插值：

- 纹理细节更清晰
- 观感更通透

对比总结如下：

对比项目	AI视频画质增强	传统插值
放大原理	卷积神经网络	双线性插值
纹理细节	清晰	模糊
处理单元	NPU/GPU	CPU 或 GPU
计算复杂度	高	低

四、FAQ

4.1 简单介绍SR具体做了哪些处理？

利用 **深度学习** 将低分辨率的图像放大为高分辨率图像，提高在低分辨率图像在高分辨率屏幕上显示效果

4.2 SR效果的评价手段和指标有哪些？

SR效果评测可以利用图像和视频质量的评价指标比如PSNR, SSIM, VMAF等可以参考以下博客：

视频流量在整个互联网流量的占比每年都在高速增长，为降低视频存储成本和数据传输通道的负载，视频压缩标准及算法在不断积极开发和改进。视频质量的评估在其中也起着至关重要的作用，尽管已经发展出了大量视频质量评估方法，但普遍接受度最高、最知名的评价方法还是经典的 PSNR、SSIM 以及 VMAF。

....

原文链接：<https://blog.csdn.net/Visionular/article/details/123423672>

在评价视频质量时，单一指标数值的高低，并不能完全反映视频的画质好坏，与人眼主观感受相比会出现偏差。

所以通常评价视频画质时，要结合多项指标和人眼的主观感受进行综合评价。

4.3 如何确认SR授权成功？

SR模块支持需要申请授权账号并且授权与硬件SOC绑定，一块SOC授权将占用一个授权名额，请谨慎授权；

1. 将授权激活码写入 /dev/vendor_storage 节点，具体可参考《Rockchip_Application_Notes_Storage_CN.pdf》文档自行开发烧写激活码工具，下面示例为自行编写的 vendor-storage-test 用例进行烧写激活码：

```
rk3588_box:/ # vendor-storage-test 9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
## vendor write:
##   tag=0x56524551 id=60 len=50
##   9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
## vendor read:
##   tag=0x56524551 id=60 len=50
##   9106BC0B-BCB6-17CE-9A3E-DD4C462B356F
```

2. 重新开机后，确认联网，可通过以下方式查看激活状态：

```

## 查看是否存在以下日志:
adb shell logcat | grep -s libsr-auth

# 从 VendorStorage ID=60 获取激活码 4ACAFCD0-46FF-144D-E2AF-923196D03D2B
I libsr-auth: ReadAuthCode, line=449 SR-AuthCode: tag=0x56524551 id=60 len=50 Code:
4ACAFCD0-46FF-144D-E2AF-923196D03D2B
# 开机阶段, 由于网络未连接导致授权失败, 进入 try again 流程
E libsr-auth: DoAuth, line=365 rkauth_activate auth code fail because networks err=-8, try
again.
# 激活成功, 离线授权文件写入 /data/system/svep_key.lic 路径
I libsr-auth: DoAuth, line=358 rkauth_activate2 auth code success, Write license to
/data/system/svep_key.lic

# 确认 /data/system/svep_key.lic 离线授权文件输出成功, 即表示授权成功

```

3. 使能 SR 模式, 播放待测片源, 若SR运行正常, 则片源左上角会出现 **RKNPU-SVEP-SR** OSD:

```

## 利用以下命令使能 SR 模式
adb shell setprop persist.sys.svep.mode 1

## 若SR正常初始化, 则通过以下命令可查询SR版本信息
adb shell getprop vendor.svep.version

```

注意: 部分客户由于特殊的产品需求可能会将HWC功能关闭, 导致无法正确进入SR模式, 可检查以下属性是否正确:

```

## 确认 vendor.hwc.compose_policy 属性值为 1
adb shell getprop vendor.hwc.compose_policy

## 若为其他值, 请将属性设置为1
adb shell setprop vendor.hwc.compose_policy 1

```

4.4 如何判断是否正确集成并开启SR功能?

1. 确认SR授权成功, 参照 "FAQ 4.3 如何确认SR授权成功? " 章节确认授权成功
2. 若授权检查成功, 直接播放本地视频或者网络视频, SR正确运行的正常, 即可在视频左上角发现"RKNPU-SVEP-SR" OSD字样, 并且包含输入输出尺寸大小, 如下图:



3. 确认SR模式是否开启, 通过 persist.sys.svep.mode 属性检查, 确保设置值为1

```
## 检查SR模式是否为1  
adb shell getprop persist.sys.svep.mode  
  
## 若不为1，则通过以下命令设置为1  
adb shell setprop persist.sys.svep.mode 1
```

4. 若上述检查均正常，请按以下要求抓打印日志，提供RK工程师分析：

```
adb shell setprop vendor.hwc.log debug  
adb shell setprop vendor.svep.log 1  
adb shell logcat -c  
adb shell logcat > hwc.log
```

若SR正常运行，则会看到视频画面如下，画面左上角出现 "RKNPU-SVEP-SR" OSD字样：



4.5 SR强度原理是什么？

SR强度是通过调整SR模型内部权重系数，来实现对纹理细节效果的调整，强度设置越高，细节越清晰。

4.6 图片进行SR是否支持？

支持，但是需要告知具体需要SR图片的标志。

因为，目前系统显示框架主要是根据应用图层的格式（如YUV）与Surface Type(如SurfaceView) 来判断是否需要使用SR模式，而需要SR的图层是支持通过一些判定来实现使用SR，下面介绍一些方法：

1. 可以修改图片输出格式，例如YUV，或者使用SurfaceView来送显图片；
2. 可以通过设置特殊的LayerName来标识是否使用SR，例如应用图层LayerName加入“SR” 字段来实现；

4.7 Android SDK如何使能SR功能？

SDK工程使能 SR 功能请按照如下步骤：

1. SR接口库位于SDK以下目录：

```
hardware/rockchip/libsvp
```

2. HWC源码位于SDK以下目录：

```
hardware/rockchip/hwcomposer/drmhw2
```

3. 使能SR编译选项：

```
# 修改以下Android.mk文件  
hardware/rockchip/hwcomposer/drmhwc2/Android.mk  
hardware/rockchip/libsvp/Android.mk  
  
# 添加SR编译选项  
BOARD_USES_LIBSVEP := true  
  
# 或者，可在对应product目录下，添加 SR 编译选项，例如BOX工程可直接修改以下mk文件：  
device/rockchip/rk3588/rk3588_box/rk3588_box.mk  
+ BOARD_USES_LIBSVEP := true
```

4. 编译SDK工程；

另外：SR功能要正常使用还需要授权，请向RK获取授权激活码，请参考《2.5 VendorStorage 授权接口》。