长虹音乐播控对接协议

本文档旨在帮助三方音乐服务接入桌面Launche，实现音乐播放控制功能。本文档涵盖了接入准备、接入流程以及注意事项等内容。

1. 接入准备

- 确保您的音乐服务已经使用了Android的MediaSession框架。
- Android MediaSession官方开发手册： https://developer.android.com/reference/android/media/session/MediaSession
- 为达成整体更好的用户体验效果，本协议对接完之后，支持桌面播控。

2.接入流程

2.1 添加依赖和轻松MediaSession配置

为了兼容不同Android版本，需使用androidx中android.support.v4.media包下的XxxCompat类。

```
implementation  'androidx.appcompat:appcompat:xxxx'
implementation  'androidx.media:media:xxxx'
```

MediaBrowser连接路径：com.qs.kugou.tv.service.MediaMusicService

轻松的MediaSession的LogTag：MediaMusicService

示例代码:

```
MediaSession mediaSession = new MediaSession(context, "MediaMusicService");
```

2.2接口callback说明如下：

意义	MediaSession	MediaController.Callback	场景	备注
播放状态	setPlaybackState (PlaybackState)	onPlaybackStateChanged(state: PlaybackState?)	歌曲播放状态变化（同一首歌）	
当前播放音乐	setMetadata (MediaMetadata)	onMetadataChanged(metadata: MediaMetadata?)	歌曲信息变化（切歌，从未播放歌曲 → 播放歌曲）	尽量一次将歌曲信息（歌曲名，歌手，歌曲封面，歌手图片，歌曲总时 长）给到位，不要出现同一首歌因为需要补全信息多次回调此接口
播放队列	setQueue (List<MediaSession. QueueItem>)	onQueueChanged(queue: MutableList<MediaSession. QueueItem>?)	当前播放列表变化	暂未处理此接口回调数据
播放队列标题	setQueueTitle (CharSequence)	onQueueTitleChanged (CharSequence)	/	
额外信息	setExtras(Bundle)	onExtrasChanged(Bundle)	/	自定义参数等场景
自定义事件	sendSessionEvent (String , Bundle)	onSessionEvent(event: String, extras: Bundle?)	自定义回调	目前只用来获取歌曲播放进度，后续歌词也会通过此回调获取
/	/	onSessionDestroyed	session断开	目前接收到此回调后会刷新UI为“暂无歌曲播放”

2.3 歌曲信息变化

客户端可以会从onMetadataChanged回调中回调的metadata获取歌曲信息：

下方为线上版本已使用字段：

对应Key	状态（灵控桌面支持状态）	信息（灵控桌面）	信息（语音）	value类型	是否必须	是否为自定义	备注
MediaMetadataCompat.METADATA_KEY_ARTIST	二期支持	/	// 歌手/演员	String	否	否	语音可能会使用
MediaMetadata.METADATA_KEY_AUTHOR	已支持	歌曲作者/演唱歌手	// 创作/导演	String	是	否	
MediaMetadata.METADATA_KEY_MEDIA_ID	已支持	歌曲ID	/	String	是	否	
MediaMetadata.METADATA_KEY_GENRE	已支持（二期移除song）	歌曲类型(audio)	媒体类型，取值为："audio"	String	是	否	
MediaMetadata.METADATA_KEY_TITLE	已支持	歌曲标题	// 曲目/影视名	String	是	否	
MediaMetadata.METADATA_KEY_DURATION	已支持	歌曲总时长（毫秒）	// 时长	long	是	否	
MediaMetadataCompat.METADATA_KEY_ALBUM	二期支持	/	// 专辑/剧集	String	否	否	
"META_DATA_ALBUM_PIC"	已支持	歌曲专辑封面图uri	/	String	否	是	
MediaMetadata.METADATA_KEY_ALBUM_ART_URI	二期支持	歌曲专辑封面图uri	/	String	否	否	建议后续使用原生Key
"BEIDOU_METADATA_KEY_SINGER_URI"	二期支持	歌曲歌手图uri	/	String	否	是	
MediaMetadata.METADATA_KEY_DISPLAY_ICON_URI	二期支持	cp方标识图(logo图)	/	String	是	否	目前为本地匹配包名
“BEIDOU_METADATA_KEY_IS_TRIAL_MUSIC”	二期支持	是否为试听歌曲	/	long	否	是	不是试听歌曲 ---- 0L 是试听歌曲 ---- 1L
“BEIDOU_METADATA_KEY_IS_VIP_MUSIC”	二期支持	是否为VIP歌曲	/	long	否	是	不是VIP歌曲 ------ 0L 是VIP歌曲 ----- 1L
“BEIDOU_METADATA_KEY_IS_PAY_MUSIC”	二期支持	是否为付费单曲	/	long	否	是	不是付费单曲---- 0L 是付费单曲 ---- 1L
“BEIDOU_METADATA_KET_TRIAL_START”	二期支持	开始试听时间	/	long	否	是	如果传了歌曲为试听，必须要传此参数
“BEIDOU_METADATA_KET_TRIAL_END”	二期支持	结束试听时间	/	long	否	是	如果传了歌曲为试听，必须要传此参数
“BEIDOU_METADATA_KET_HAS_LYRICS”	二期支持	是否有歌词	/	long	是	是	默认为否 有歌词---- 0L 无歌词 ---- 1L
“BEIDOU_METADATA_KET_SLOGAN_TEXT”	cp方slogan	String	/	String	是	是	不传无歌词时展示默认slogan
isLogin	已支持	用户是否登录		String	是	是	未登录--- “0” 已登录 ---- “1”
isVipForSong	已支持	用户是否为音乐会员		String	是	是	非音乐会员---- “0” 音乐会员 ---- “1”

2.4 播控指令说明： onPlaybackStateChanged回调中获取的状态

指令	action（支持语音控制的功能）	onPlaybackStateChanged	是否必须	说明	errorCode扩展	TTS（语音播报反馈）
/	PlaybackStateCompat.STATE_NONE	STATE_NONE	是	默认的播放状态，尚未播放歌曲		
播放	PlaybackStateCompat.ACTION_PLAY	STATE_PLAYING	是	/		
暂停	PlaybackStateCompat.ACTION_PAUSE	STATE_PAUSED	是	/		
停止播放	PlaybackStateCompat.ACTION_STOP	STATE_STOPPED	是	/		
上一首/集	PlaybackStateCompat.ACTION_SKIP_TO_PREVIOUS	/	是	/		
下一首/集	PlaybackStateCompat.ACTION_SKIP_TO_NEXT	/	是	/		
单曲循环	PlaybackStateCompat.REPEAT_MODE_ONE		否	/		
顺序播放	PlaybackStateCompat.REPEAT_MODE_ALL		否	/		
随机播放	PlaybackStateCompat.SHUFFLE_MODE_ALL		否	/		
跳转到任意时间播放	PlaybackStateCompat.ACTION_SEEK_TO		否	/		

2.5 自定义字段

2.5.1 获取当前播放进度

（处于播放状态时，每秒**sendCommand("getPosition")**）

其中当event == “onPosition”时，从extras中获取当前播放进度（单位：秒）

参考代码

```
private void getCurrentPlayTime() {
    MusicLog.d(TAG, "getCurrentPlayTime");
    Bundle bundle = new Bundle();
    bundle.putInt("onPosition", (int) (mPlayer.getCurrentPosition() / 1000));

    MusicLog.d(TAG, "getCurrentPlayTime = " + mPlayer.getCurrentPosition() / 1000);
    mMediaSessionCompat.sendSessionEvent("onPosition", bundle);
}
```

2.5.2 获取当前播放歌曲歌词：

歌词信息通过senndSessionEvent进行同步
sendSessionEvent("onLyricRowChange", bundle)：
时机：即将唱新的一行时回调

字段	类型	字段含义	
lyricContent	ArrayList	歌词内容（第0行是当前即将唱的歌词）	
lyricTime	Long	歌词当前时间	

样例：

```
lyricContent: [{},{},{},{},{}]
```

2.6 一键启播功能

2.6.1 歌单信息，启播参数获取

启动service时，通过参数传递，直接拉取歌单并起播

字段	类型	是否必须	字段含义	备注
playFromMediaId	String	是	歌单ID	
playFromMediaTitle	String	是	歌单标题	
playFromMediaType	String	是	歌单类型	类型定义 2：酷狗歌单 5：歌曲榜单
isContinuePlay	String	是	播放类型	1：恢复播放 0：从第一首播放

参考代码

```
intent1.setComponent(new ComponentName("xxx",
"com.qs.kugou.tv.service.MediaMusicService"));
intent1.putExtra("playFromMediaId", "collection_3_1137485567_117_0");
intent1.putExtra("playFromMediaTitle", "歌单");
intent1.putExtra("playFromMediaType", "2");
intent1.putExtra("isContinuePlay", "0");
startService(intent1);
```

2.7 切换歌单

自定义Action Name = changeMusicPlayListAction
字段跟2.6一键启放的功能参数一致
参考代码

```
Bundle bundle = new Bundle();
bundle.putString("playFromMediaId", "8888");
bundle.putString("playFromMediaType", "5");
bundle.putString("playFromMediaTitle", "500top");
bundle.putString("isCountinuePlay","1");
mController.sendCommand("changeMusicPlayListAction", bundle, null);
```

2.8 桌面直接跳转进播放器界面

Action name: com.kugou.tv.action.current_play