三方应用媒体信息展示

日期 2022-11-07

修订记录

*类别：**A**–增加 **M**–修改 **D**–删除

日期	版本号	修订类型	作者	描述
2020-11-16	1.0	A	李刚	制定初稿
2022-09-21	1.1	M	李刚	新增进度协议要求
2022-11-7	1.2	M	李刚	新增音频/视频模式扩展协议
2022-11-11	1.3	M	李刚	细化进度协议要求
2022-11-14	1.4	M	李刚	新增扩展媒体控制协议
2023-10-23	1.5	M	李刚	新增点击跳转控制协议

目录

1 引言

1.1 编写目的

本文档简单介绍三方应用媒体应用如何展示媒体信息到小鹏车机媒体中心，同时如何支持媒体中心基础播放控制。

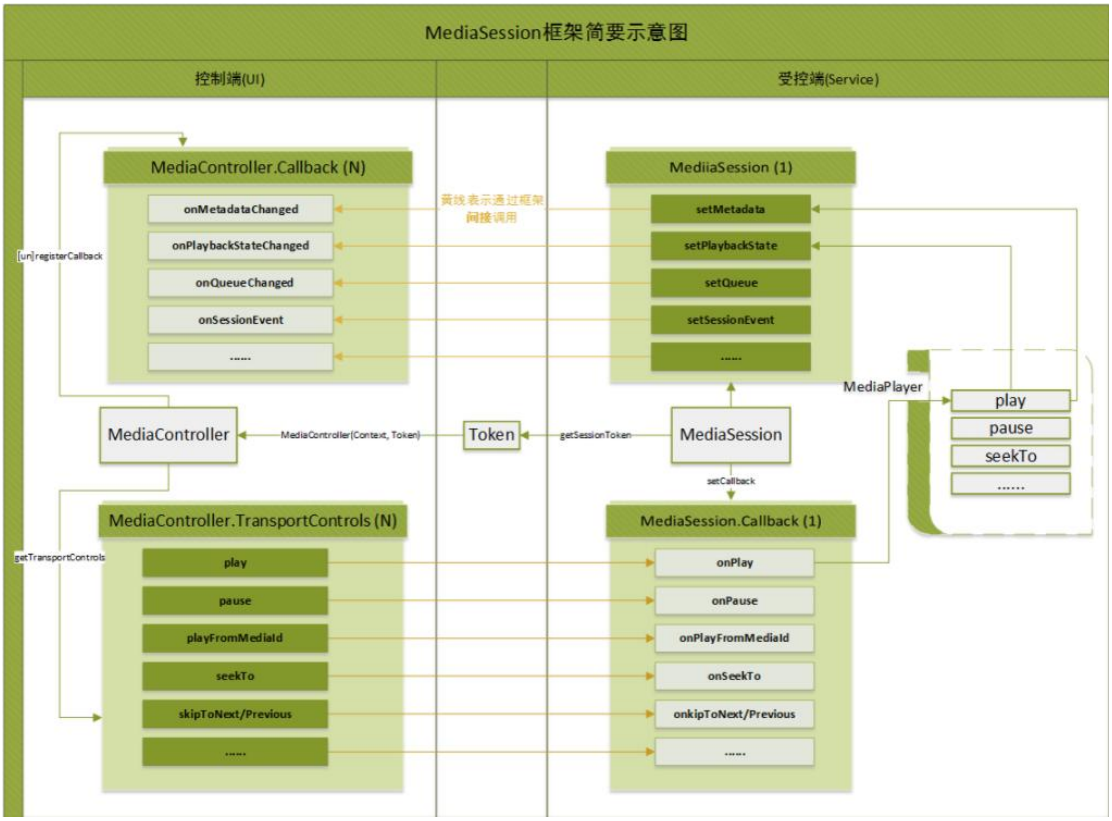
1.2 编写背景

为保证视觉风格的统一和可控，小鹏车机有统一的媒体信息展示入口和控制入口。

2 协议介绍和支持

目前三方的媒体信息暴露和控制统一通过安卓系统标准的 MediaSession 协议来实现。

2.1 基本框架



2.2 协议实现

从 MediaSession 框架中可以理解，三方媒体应用需要在 Service 实现 MediaSession 类。

三方应用需要调用 MediaSession 对象更新媒体信息和播放状态。

1、通过设置 MediaSession.Callback 回调来接收媒体控制器发送的控制指令。

2、通过 setMetadata 来通知媒体信息更新。

3、通过 setPlaybackState 来通知播放状态更新。

2.3 示例代码

三方应用要暴露媒体信息，接受 MediaButton 的控制需要在 service 里面实现

MediaSession 类。应用必须在其清单中声明带有 Intent 过滤器的 MediaBrowserService

```
<!--
Declare the common MediaBrowserService for use in the mobile app, including
with the Android Auto app.
-->
<service
    android:name=".media.MusicService"
    android:enabled="true"
    android:exported="true"
    tools:ignore="ExportedService">

    <intent-filter>
        <action android:name="android.media.browse.MediaBrowserService" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.media.action.MEDIA_PLAY_FROM_SEARCH" />
    </intent-filter>
</service>
```

2.3.1 初始化 MediaSession

在 service 声明周期 onCreate() 回调方法时，执行 MediaSession 的初始化。

1、创建和初始化 MediaSession

2、设置 MediaSession 回调

3、设置 MediaSession 令牌

示例代码

```java
public class MediaPlaybackService extends MediaBrowserService {
    private MediaSession mSession;

    @Override
    public void onCreate() {
        super.onCreate();
        // Start a new MediaSession
        mSession = new MediaSession( context: this,  tag: "MediaPlaybackService");
        // Enable callbacks from MediaButtons and TransportControls
        mSession.setFlags(MediaSession.FLAG_HANDLES_MEDIA_BUTTONS
                | MediaSession.FLAG_HANDLES_TRANSPORT_CONTROLS);
        // Set an initial PlaybackState with ACTION_PLAY, so media buttons can start the player
        PlaybackState.Builder stateBuilder = new PlaybackState.Builder().setActions(
                PlaybackState.ACTION_PLAY | PlaybackState.ACTION_PLAY_PAUSE);
        mSession.setPlaybackState(stateBuilder.build());
        // MediaSessionCallback() has methods that handle callbacks from a media controller
        mSession.setCallback(new MediaSessionCallback());
        // Set the session's token so that client activities can communicate with it.
        setSessionToken(mSession.getSessionToken());
    }
```

2.3.2 实现 Playback.Callback，接受 MediaButton 的控制

```
private void setCallback(){
    //注册callback
    mSession.setCallback(new MediaSessionCallback());
}
//回调事件通过mediaButton转换过来
private final class MediaSessionCallback extends MediaSession.Callback {
    @Override
    public void onPlay() {
        //播放
        super.onPlay();
    }

    @Override
    public void onPause() {
        //暂停
        super.onPause();
    }

    @Override
    public void onSkipToNext() {
        //下一首
        super.onSkipToNext();
    }

    @Override
    public void onSkipToPrevious() {
        //上一首
        super.onSkipToPrevious();
    }
```

2.3.3 通过 MediaSession 通知媒体信息更新

```
/**
 * 更新媒体信息
 */
private void updateMetadata() {
    MediaMetadata track = new MediaMetadata.Builder().
            putString(MediaMetadata.METADATA_KEY_TITLE, "沉默是金")
            .putString(MediaMetadata.METADATA_KEY_ARTIST, "张国荣")
            .putBitmap(MediaMetadata.METADATA_KEY_DISPLAY_ICON, null).
                    build();
    mSession.setMetadata(track);
}
```

2.3.4 通过 MediaSession 通知播放状态更新

```
/**
 * 更新播放状态
 */
private void updatePlaybackState() {
    //播放中
    PlaybackState state = new PlaybackState.Builder().
            setState(PlaybackState.STATE_PLAYING,  position: 0,  playbackSpeed: 1.0f).build();
    mSession.setPlaybackState(state);
}
```

2.3.5 通过 MediaSession 通知进度变化

目前小鹏快速 widget 支持进度信息的展示：

进度信息的计算公式如下：

```
if (state.getPosition() == PlaybackState.PLAYBACK_POSITION_UNKNOWN) {
    LogUtil.d(TAG,  message: "PLAYBACK_POSITION_UNKNOWN");
    return PlaybackState.PLAYBACK_POSITION_UNKNOWN;
}
long timeDiff = SystemClock.elapsedRealtime() - state.getLastPositionUpdateTime();
float speed = state.getPlaybackSpeed();
if (state.getState() == PlaybackState.STATE_PAUSED
        || state.getState() == PlaybackState.STATE_STOPPED) {
    // This guards against apps who don't keep their playbackSpeed to spec (b/62375164)
    speed = 0f;
}
long posDiff = (long) (timeDiff * speed);
return Math.min(posDiff + state.getPosition(), duration);
```

三方如果期望正常展示进度信息，需要使用 PlaybackState 的 setState 方法传递相关参数：

```
     *
     * @param state The current state of playback.
     * @param position The position in the current item in ms.
     * @param playbackSpeed The current speed of playback as a multiple of
     *            normal playback.
     * @param updateTime The time in the {@link SystemClock#elapsedRealtime}
     *            timebase that the position was updated at.
     * @return this
     */
    public Builder setState(@State int state, long position, float playbackSpeed,
            long updateTime) {
        mState = state;
        mPosition = position;
        mUpdateTime = updateTime;
        mSpeed = playbackSpeed;
        return this;
    }
```

参数说明：

1、state：当前状态，支持的参数可以查看 PlayState 里面的 State 定义，比如 STATE_PLAYING，STATE_STOPPED，STATE_PAUSED 等

2、position：当前的进度信息

3、playbackSpeed：当前的倍速，默认是 1

4、updateTime：是指更新状态的时间，默认是 SystemClock.elapsedRealtime，这个时间可以让业务层去做校验逻辑

2.3.6 通过 MediaSession 标定音/视频模式

目前官方协议里面未定义该字段，但是官方协议支持扩展字段。

在创建 MetaData 的时候，标记应用音/视频模式，示例如下：

```
private void createMediaItem(){
    MediaMetadataCompat.Builder builder = new MediaMetadataCompat.Builder();
    builder.putString("playMode", "Video");
}
```

注意：视频应用默认在后台会被小鹏 AMS 回收掉，只有标记为 Music 模式才能后台存活。

进入卡片的系统默认都是音频模式，只有标记为 video 模式的才会执行 video 相关的逻辑，

比如到后台系统进行进程回收，屏蔽部分快捷播控操作。

2.3.7 媒体播控扩展协议

Android MediaController 提供了扩展协议的接口和能力。
方法：

```
@Override
public void onCustomAction(String action, Bundle extras) {
    super.onCustomAction(action, extras);
}
```

action 定义如下：
com.xiaopeng.action_media_session_custom
Bundle 数据定义如下
KEY_CONTROL : Control
目前具体指令包括：setMode （播放模式）
KEY_TEXT_VALUE：TextValue（指令中包含的文本参数）
KEY_LONG_VALUE：LongValue （指令中包含的整型参数）
比如业务端处理播放模式调整的代码如下：

```
@Override
public void onCustomAction(String action, Bundle extras) {
    super.onCustomAction(action, extras);
    if (action.equals("com.xiaopeng.action_media_session_custom")){
        String command = extras.getString( key: "Control",  defaultValue: "");
        if (!TextUtils.isEmpty(command) && command.equals("setMode")){
            String mode = extras.getString( key: "mode");
            toSetMode(mode);
        }
    }
}

private void toSetMode(String mode){

}
```

2.3.8 媒体组件点击跳转

桌面媒体小组件点击需要跳转到当前 session 应用的页面

目前 Android Media Session 框架提供了对应的能力和接口：

```java
/**
 * Set an intent for launching UI for this Session. This can be used as a
 * quick link to an ongoing media screen. The intent should be for an
 * activity that may be started using {@link Activity#startActivity(Intent)}.
 *
 * @param pi The intent to launch to show UI for this Session.
 */
public void setSessionActivity(@Nullable PendingIntent pi) {
    try {
        mBinder.setLaunchPendingIntent(pi);
    } catch (RemoteException e) {
        Log.wtf(TAG, "Failure in setLaunchPendingIntent.", e);
    }
}
```

示例代码：

```java
mSession.setPlaybackState(mPlaybackState);
mSession.setActive(true);

mSession.setSessionActivity(PendingIntent.getActivity( context: this,  requestCode: 0,
        new Intent( action: "define by self")
        , PendingIntent.FLAG_UPDATE_CURRENT
        ));
```

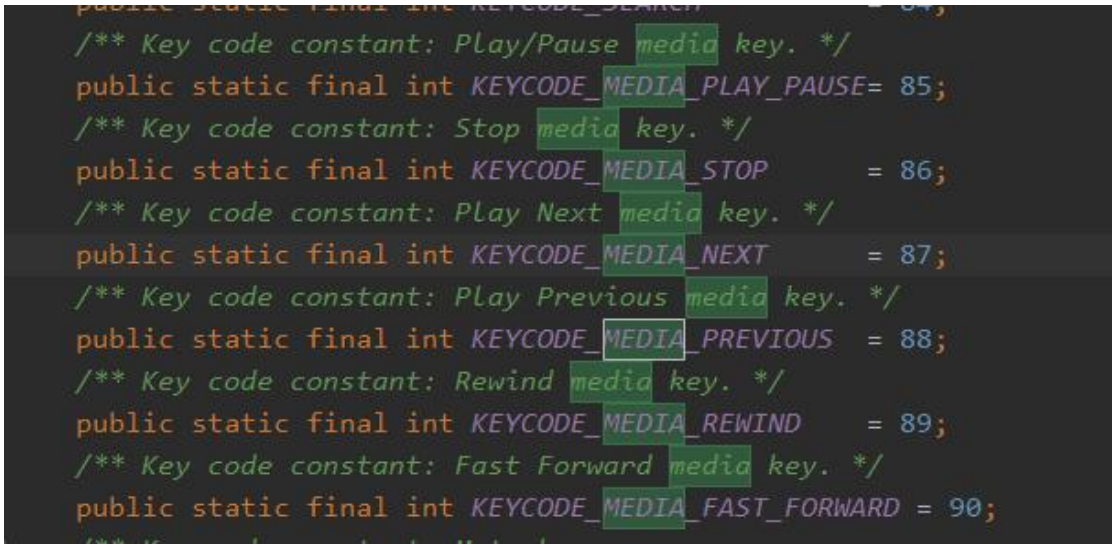
3 示范代码

MediaPlaybackService.java

4 测试方法

控制指令可以通过模拟媒体按键来测试

常用的 KeyCode 如下



示例：模拟暂停和播放

5 参考文档

https://developer.android.com/guide/topics/media-apps/audio-app/building-a-mediabrowserservice?hl=zh-cn

https://developer.android.com/guide/topics/media-apps/working-with-a-media-session?hl=zh-cn