Media Center Tutorial

1. Overview

The media center supports playback of multiple streams, what can be started/paused/resumed/stopped, supports pre-recording for finding aircraft, and supports obtaining real-time stream information.

2. Flow Charts

Flow charts for playback starting/stopping and pre-recording.

3. Media Center API Description

3.1 Multi-stream Player

3.1.1 Overview

Multiple streams (IR/wide camera) can be played at the same time, and playback can be started, paused, resumed and stopped.

3.1.2 API Description

APIs for calling: **AutelPlayerManager** (single instance), **AutelPlayer**, **AutelPlayerView**

AutelPlayerManager.java

```
/**
 * Singleton class object of AutelPlayerManager
 */
public synchronized static AutelPlayerManager getInstance()

/**
 * initialises
 * @param cxt  Context
 * @param bDroneCenter the boolean value
 */
public void init(Context cxt,boolean bDroneCenter)

/**
 * Registers stream data listener
 */
public void registerStreamDataListener()

/**
* Unregisters  stream data listeners
*/
public void unregisterStreamDataListener()

/**
 * starts streaming video
 * @param port stream channel Id
```

```
 */
public void startStreamChannel(int port)

/**
 * ends streaming video
 * @param port stream channel Id
 */
public void endStreamChannel(int port)

/**
 * Adds item to mAutelPlayerList
 *
 * @param player AutelPlayer instance to be added
 */
public void addAutelPlayer(AutelPlayer player)

/**
* Releases/removes all players
*/
public void release()

/**
* add mediacodec listeners
* @param key
* @param callbackWith
*/
public void addCodecListeners(String key, final OnRenderFrameInfoListener
callbackWith)
```

AutelPlayer.java

```
/**
* construction method
* @param playerID the player id
*/
public AutelPlayer(int playerID)

/**
* adds view for video
* @param view the Autelplayerview
*/
public void addVideoView(AutelPlayerView view)

/**
* removing the view for video
*/
public void removeVideoView()

/**
 * sets listener for video stream listener
 * @param ls the iVideoStreamListener
 */
public void setVideoInfoListener(IVideoStreamListener ls)

/**
* Pauses decoding and displays gray interface.
```

```
*/
public void pauseVideo()

/**
* Resumes decoding and displays multicolor interface.
*/
public void resuemeVideo()

/**
* Releases resources of each thread.
*/
public void releasePlayer()
```

AutelPlayerView.java

```
/**
 * Creates AutelCodecView.
 *
 * @param context The context to associate this view with.
 */
public AutelPlayerView(Context context)

/**
* Creates AutelCodecView.
*
* @param context The context to associate this view with.
* @param attrs  The attributes of the XML tag that is inflating the view.
*/
public AutelPlayerView(Context context, AttributeSet attrs)
```

3.1.3 Calling An API For Multi-stream Player

```
//Initializes an API for player management.
AutelPlayerManager.getInstance().init(ctx, false)
//Registers listening of stream receival.
override fun onResume() {
    ......
    AutelPlayerManager.getInstance().registerStreamDataListener()
    ......
}
//Enables the channel for receiving streams.
AutelPlayerManager.getInstance().startStreamChannel(getPlayerId())
//Creates a TextureView object for playback.
val codecView = AutelPlayerView(activity)
uiBinding.root.addView(codecView, 0)
//Creates a player named AutelPlayer.
mAutelPlayer = AutelPlayer(getPlayerId())
//Passes TextureView to the player.
mAutelPlayer?.addVideoView(codecView)
//Sets the stream information callback API.
mAutelPlayer?.setVideoInfoListener(mVideoStreamListener)
//Adds the created player to the player list.
AutelPlayerManager.getInstance().addAutelPlayer(mAutelPlayer)
//Starts thread resources of the player.
mAutelPlayer?.startPlayer()
```

```
//Pauses playback.
override fun onPause() {
    ......
    mAutelPlayer?.pauseVideo()
    ......
}
//Resumes playback.
override fun onResume() {
    ......
    mAutelPlayer?.resuemeVideo()
    ......
}

//Removes TextureView from the player.
mAutelPlayer?.removeVideoView()
//Disables the channel for receiving streams.
AutelPlayerManager.getInstance().endStreamChannel(getPlayerId())
//Releases the thread resources of the player.
mAutelPlayer?.releasePlayer()
//Removes the player from the player list.
AutelPlayerManager.getInstance().removeAutelPlayer(mAutelPlayer)

//Cancels listening of stream receival and removes all players from the list.
override fun onDestroy() {
    ......
    AutelPlayerManager.getInstance().unregisterStreamDataListener()
    AutelPlayerManager.getInstance().release()
    ......
}
```

3.2 Pre-recording For Finding Aircraft

3.2.1 Overview

When the pre-recording feature is enabled, the recording of every 30 seconds will be saved with a fixed name and overwrite the previously saved file, so that the latest 30 seconds of the aircraft's video content can always be kept. If the aircraft gets lost, the content can be used to find the aircraft.

3.2.2 API Description

API for calling: **AutelPlayerManager** (single instance)

```
/**
 * Starts pre-recording with the specified camera (wide/night/IR). The maximum
length is about 30s. If true is returned, the operation is successful; if false
is returned, the operation fails due to invalid parameters.
 * @param videoType Video type.
 * @param cacheDuration: Uses ms.*/
public boolean openVideoCache(VideoType videoType, long cacheDuration)

/**
 * Stops pre-recording with the specified camera (wide/night/IR). If true is
returned, the operation is successful; if false is returned, the operation fails
due to invalid parameters.
 * @param videoType Video type.
```

```
 */
public boolean closeVideoCache(VideoType videoType)

/**
 * Sets the path to save pre-recordings. If left unspecified, the files will be
saved to the root directory.
 * @param path the video path
*/
public void setCachePath(String path)

/**
 * Whether to start pre-recording with the specified camera.
 */
public boolean isOpenVideoCache(VideoType videoType)
```

3.2.3 Sample Code

```
//Starts 30s pre-recording.
AutelPlayerManager.getInstance().openVideoCache(VideoType.VISIBLE_LIGHT, 30_000)
//Stops 30s pre-recording.
AutelPlayerManager.getInstance().closeVideoCache(VideoType.VISIBLE_LIGHT)
//Sets the path to save pre-recordings. If left unspecified, the files will be
saved to the root directory.
AutelPlayerManager.getInstance().setCachePath(AutelDirPathUtils.getLookFlightVid
eoCachePath())
//Whether to start 30s pre-recording.
AutelPlayerManager.getInstance().isOpenVideoCache(VideoType.VISIBLE_LIGHT)
```

3.3 Obtaining Real-Time Stream Information

3.3.1 Overview

Real-time stream information can be obtained for fault locating.

3.3.2 API Description

APIs for calling: **AutelPlayer**

```
//Decode frame rate
public int getVideoFps()
//Receive frame rate
public int getRecvFps()
//Refresh frame rate
public int getRenderFps()
//Receive bitrate
public int getRecvBitrate()
//Number of keyframes received in 60s
public int getKeyFrameNum()
//Number of keyframe requests in 60s
public int getKeyFrameReqNum()
//Number of lost packets in 1s
public int getPktLossNum()
//Video width
public int getVideoWidth()
//Video height
public int getVideoHeigh()
```