MediaPlayer

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public class MediaPlayer`
`extends Object` (https://developer.android.com/reference/java/lang/Object.html) `implements VolumeAutomation` (https://developer.android.com/reference/android/media/VolumeAutomation.html), `AudioRouting` (https://developer.android.com/reference/android/media/AudioRouting.html)

java.lang.Object (https://developer.android.com/reference/java/lang/Object.html)
  ↳ android.media.MediaPlayer

MediaPlayer class can be used to control playback of audio/video files and streams. An example on how to use the methods in this class can be found in `VideoView` (https://developer.android.com/reference/android/widget/VideoView.html).

Topics covered here are:

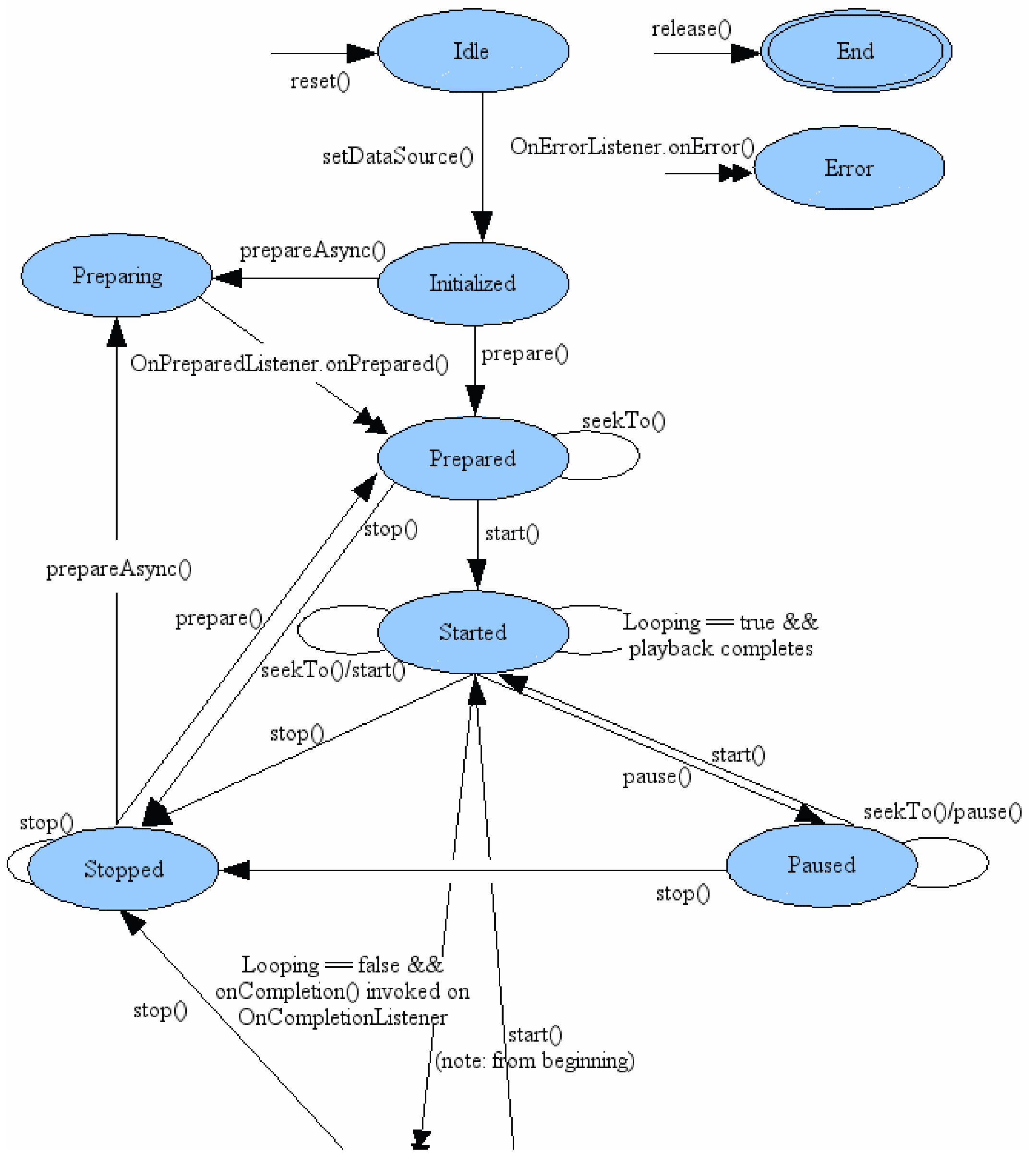
1. State Diagram (https://developer.android.com/reference/android/media/MediaPlayer#StateDiagram)
2. Valid and Invalid States (https://developer.android.com/reference/android/media/MediaPlayer#Valid_and_Invalid_States)
3. Permissions (https://developer.android.com/reference/android/media/MediaPlayer#Permissions)
4. Register informational and error callbacks (https://developer.android.com/reference/android/media/MediaPlayer#Callbacks)

Developer Guides

For more information about how to use MediaPlayer, read the Media Playback (https://developer.android.com/guide/topics/media/mediaplayer.html) developer guide.

State Diagram

Playback control of audio/video files and streams is managed as a state machine. The following diagram shows the life cycle and the states of a MediaPlayer object driven by the supported playback control operations. The ovals represent the states a MediaPlayer object may reside in. The arcs represent the playback control operations that drive the object state transition. There are two types of arcs. The arcs with a single arrow head represent synchronous method calls, while those with a double arrow head represent asynchronous method calls.



From this state diagram, one can see that a MediaPlayer object has the following states:

- When a MediaPlayer object is just created using new or after `reset()` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset()) is called, it is in the *Idle* state; and after `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()) is called, it is in the *End* state. Between these two states is the life cycle of the MediaPlayer object.
  - There is a subtle but important difference between a newly constructed MediaPlayer object and the MediaPlayer object after `reset()` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset()) is called. It is a programming error to invoke methods such as `getCurrentPosition()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getCurrentPosition()), `getDuration()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getDuration()), `getVideoHeight()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getVideoHeight()), `getVideoWidth()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getVideoWidth()), `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes)), `setLooping(boolean)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setLooping(boolean)), `setVolume(float, float)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setVolume(float,%20float)), `pause()` (https://developer.android.com/reference/android/media/MediaPlayer.html#pause()), `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()), `stop()` (https://developer.android.com/reference/android/media/MediaPlayer.html#stop()), `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)), `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) or `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()) in the *Idle* state for both cases. If any of these methods is called right after a MediaPlayer object is constructed, the user supplied callback method OnErrorListener.onError() won't be called by the internal player engine and the object state remains unchanged; but if these methods are called right after `reset()` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset()), the user supplied callback method OnErrorListener.onError() will be invoked by the internal player engine and the object will be transfered to the *Error* state.
  - It is also recommended that once a MediaPlayer object is no longer being used, call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()) immediately so that resources used by the internal player engine associated with the MediaPlayer object can be released immediately. Resource may include singleton resources such as hardware acceleration components and failure to call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()) may cause subsequent instances of MediaPlayer objects to fallback to software implementations or fail altogether. Once the MediaPlayer object is in the *End* state, it can no longer be used and there is no way to bring it back to any other state.

    - Furthermore, the MediaPlayer objects created using `new` is in the *Idle* state, while those created with one of the overloaded convenient `create` methods are *NOT* in the *Idle* state. In fact, the objects are in the *Prepared* state if the creation using `create` method is successful.
- In general, some playback control operation may fail due to various reasons, such as unsupported audio/video format, poorly interleaved audio/video, resolution too high, streaming timeout, and the like. Thus, error reporting and recovery is an important concern under these circumstances. Sometimes, due to programming errors, invoking a playback control operation in an invalid state may also occur. Under all these error conditions, the internal player engine invokes a user supplied OnErrorListener.onError() method if an OnErrorListener has been registered beforehand via `setOnErrorListener(android.media.MediaPlayer.OnErrorListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnErrorListener(android.media.MediaPlayer.OnErrorListener)).
    - It is important to note that once an error occurs, the MediaPlayer object enters the *Error* state (except as noted above), even if an error listener has not been registered by the application.
    - In order to reuse a MediaPlayer object that is in the *Error* state and recover from the error, `reset()` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset()) can be called to restore the object to its *Idle* state.
    - It is good programming practice to have your application register a OnErrorListener to look out for error notifications from the internal player engine.
    - IllegalStateException is thrown to prevent programming errors such as calling `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()), `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()), or one of the overloaded `setDataSource` methods in an invalid state.
- Calling `setDataSource(FileDescriptor)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.io.FileDescriptor)), or `setDataSource(String)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.lang.String)), or `setDataSource(Context, Uri)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.content.Context,%20android.net.Uri)), or `setDataSource(FileDescriptor, long, long)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.io.FileDescriptor,%20long,%20long)), or `setDataSource(MediaDataSource)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.media.MediaDataSource)) transfers a MediaPlayer object in the *Idle* state to the *Initialized* state.
    - An IllegalStateException is thrown if setDataSource() is called in any other state.
    - It is good programming practice to always look out for `IllegalArgumentException` and `IOException` that may be thrown from the overloaded `setDataSource` methods.
- A MediaPlayer object must first enter the *Prepared* state before playback can be started.
    - There are two ways (synchronous vs. asynchronous) that the *Prepared* state can be reached: either a call to `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) (synchronous) which transfers the object to the *Prepared* state once the method call returns, or a call to `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()) (asynchronous) which first transfers the object to the *Preparing* state after the call returns (which occurs almost right way) while the internal player engine continues working on the rest of preparation work until the preparation work completes. When the preparation completes or when `prepare()`

(https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) call returns, the internal player engine then calls a user supplied callback method, onPrepared() of the OnPreparedListener interface, if an OnPreparedListener is registered beforehand via `setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener)).

    - It is important to note that the *Preparing* state is a transient state, and the behavior of calling any method with side effect while a MediaPlayer object is in the *Preparing* state is undefined.
    - An IllegalStateException is thrown if `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) or `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()) is called in any other state.
    - While in the *Prepared* state, properties such as audio/sound volume, screenOnWhilePlaying, looping can be adjusted by invoking the corresponding set methods.
- To start the playback, `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) must be called. After `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) returns successfully, the MediaPlayer object is in the *Started* state. `isPlaying()` (https://developer.android.com/reference/android/media/MediaPlayer.html#isPlaying()) can be called to test whether the MediaPlayer object is in the *Started* state.
    - While in the *Started* state, the internal player engine calls a user supplied OnBufferingUpdateListener.onBufferingUpdate() callback method if a OnBufferingUpdateListener has been registered beforehand via `setOnBufferingUpdateListener(OnBufferingUpdateListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnBufferingUpdateListener(android.media.MediaPlayer.OnBufferingUpdateListener)). This callback allows applications to keep track of the buffering status while streaming audio/video.
    - Calling `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) has not effect on a MediaPlayer object that is already in the *Started* state.
- Playback can be paused and stopped, and the current playback position can be adjusted. Playback can be paused via `pause()` (https://developer.android.com/reference/android/media/MediaPlayer.html#pause()). When the call to `pause()` (https://developer.android.com/reference/android/media/MediaPlayer.html#pause()) returns, the MediaPlayer object enters the *Paused* state. Note that the transition from the *Started* state to the *Paused* state and vice versa happens asynchronously in the player engine. It may take some time before the state is updated in calls to `isPlaying()` (https://developer.android.com/reference/android/media/MediaPlayer.html#isPlaying()), and it can be a number of seconds in the case of streamed content.
    - Calling `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) to resume playback for a paused MediaPlayer object, and the resumed playback position is the same as where it was paused. When the call to `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) returns, the paused MediaPlayer object goes back to the *Started* state.
    - Calling `pause()` (https://developer.android.com/reference/android/media/MediaPlayer.html#pause()) has no effect on a MediaPlayer object that is already in the *Paused* state.
- Calling `stop()` (https://developer.android.com/reference/android/media/MediaPlayer.html#stop()) stops playback and causes a MediaPlayer in the *Started*, *Paused*, *Prepared* or *PlaybackCompleted* state to enter the *Stopped* state.

    - Once in the *Stopped* state, playback cannot be started until `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) or `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()) are called to set the MediaPlayer object to the *Prepared* state again.
    - Calling `stop()` (https://developer.android.com/reference/android/media/MediaPlayer.html#stop()) has no effect on a MediaPlayer object that is already in the *Stopped* state.
- The playback position can be adjusted with a call to `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)).
    - Although the asynchronuous `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) call returns right away, the actual seek operation may take a while to finish, especially for audio/video being streamed. When the actual seek operation completes, the internal player engine calls a user supplied OnSeekComplete.onSeekComplete() if an OnSeekCompleteListener has been registered beforehand via `setOnSeekCompleteListener(OnSeekCompleteListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSeekCompleteListener(android.media.MediaPlayer.OnSeekCompleteListener)).
    - Please note that `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) can also be called in the other states, such as *Prepared*, *Paused* and *PlaybackCompleted* state. When `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) is called in those states, one video frame will be displayed if the stream has video and the requested position is valid.
    - Furthermore, the actual current playback position can be retrieved with a call to `getCurrentPosition()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getCurrentPosition()), which is helpful for applications such as a Music player that need to keep track of the playback progress.
- When the playback reaches the end of stream, the playback completes.
    - If the looping mode was being set to ***true***with `setLooping(boolean)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setLooping(boolean)), the MediaPlayer object shall remain in the *Started* state.
    - If the looping mode was set to ***false*** , the player engine calls a user supplied callback method, OnCompletion.onCompletion(), if a OnCompletionListener is registered beforehand via `setOnCompletionListener(OnCompletionListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnCompletionListener(android.media.MediaPlayer.OnCompletionListener)). The invoke of the callback signals that the object is now in the *PlaybackCompleted* state.
    - While in the *PlaybackCompleted* state, calling `start()` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) can restart the playback from the beginning of the audio/video source.

Valid and invalid states

Method Name	Valid Sates	Invalid States	Comments
attachAuxEffect	{Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Idle, Error}	This method must be called after setDataSource. Calling it does not change the object state.
getAudioSessionId	any	{}	This method can be called in any state and calling it does not change the object state.
getCurrentPosition	{Idle, Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
getDuration	{Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
getVideoHeight	{Idle, Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
getVideoWidth	{Idle, Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
isPlaying	{Idle, Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
pause	{Started, Paused, PlaybackCompleted}	{Idle, Initialized, Prepared, Stopped, Error}	Successful invoke of this method in a valid state transfers the object to the *Paused* state. Calling this method in an invalid state transfers the object to the *Error* state.
prepare	{Initialized, Stopped}	{Idle, Prepared, Started, Paused, PlaybackCompleted, Error}	Successful invoke of this method in a valid state transfers the object to the *Prepared* state. Calling this method in an invalid state throws an IllegalStateException.
prepareAsync	{Initialized, Stopped}	{Idle, Prepared, Started, Paused, PlaybackCompleted, Error}	Successful invoke of this method in a valid state transfers the object to the *Preparing* state. Calling this method in an invalid state throws an IllegalStateException.
release	any	{}	After `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()), the object is no longer available.

reset	{Idle, Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted, Error}	{}	After `reset()` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset()), the object is like being just created.
seekTo	{Prepared, Started, Paused, PlaybackCompleted}	{Idle, Initialized, Stopped, Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
setAudioAttributes	{Idle, Initialized, Stopped, Prepared, Started, Paused, PlaybackCompleted}	{Error}	Successful invoke of this method does not change the state. In order for the target audio attributes type to become effective, this method must be called before prepare() or prepareAsync().
setAudioSessionId	{Idle}	{Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted, Error}	This method must be called in idle state as the audio session ID must be known before calling setDataSource. Calling it does not change the object state.
setAudioStreamType (deprecated)	{Idle, Initialized, Stopped, Prepared, Started, Paused, PlaybackCompleted}	{Error}	Successful invoke of this method does not change the state. In order for the target audio stream type to become effective, this method must be called before prepare() or prepareAsync().
setAuxEffectSendLevel	any	{}	Calling this method does not change the object state.
setDataSource	{Idle}	{Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted, Error}	Successful invoke of this method in a valid state transfers the object to the *Initialized* state. Calling this method in an invalid state throws an IllegalStateException.
setDisplay	any	{}	This method can be called in any state and calling it does not change the object state.
setSurface	any	{}	This method can be called in any state and calling it does not change the object state.
setVideoScalingMode	{Initialized, Prepared, Started, Paused, Stopped, PlaybackCompleted}	{Idle, Error}	Successful invoke of this method does not change the state.
setLooping	{Idle, Initialized, Stopped, Prepared, Started, Paused, PlaybackCompleted}	{Error}	Successful invoke of this method in a valid state does not change the state. Calling this method in an invalid state transfers the object to the *Error* state.
isLooping	any	{}	This method can be called in any state and calling it does not change the object state.
setOnBufferingUpdateListener	any	{}	This method can be called in any state and calling it does not change the object state.
setOnCompletionListener	any	{}	This method can be called in any state and calling it does not change the object state.

setOnErrorListener	any	{}	This method can be called in any state and calling it does not change the object state.
setOnPreparedListener	any	{}	This method can be called in any state and calling it does not change the object state.
setOnSeekCompleteListener	any	{}	This method can be called in any state and calling it does not change the object state.
setPlaybackParams	{Initialized, Prepared, Started, Paused, PlaybackCompleted, Error}	{Idle, Stopped}	This method will change state in some cases, depending on when it's called.
setScreenOnWhilePlaying	any	{}	This method can be called in any state and calling it does not change the object state.
setVolume	{Idle, Initialized, Stopped, Prepared, Started, Paused, PlaybackCompleted}	{Error}	Successful invoke of this method does not change the state.
setWakeMode	any	{}	This method can be called in any state and calling it does not change the object state.
start	{Prepared, Started, Paused, PlaybackCompleted}	{Idle, Initialized, Stopped, Error}	Successful invoke of this method in a valid state transfers the object to the *Started* state. Calling this method in an invalid state transfers the object to the *Error* state.
stop	{Prepared, Started, Stopped, Paused, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method in a valid state transfers the object to the *Stopped* state. Calling this method in an invalid state transfers the object to the *Error* state.
getTrackInfo	{Prepared, Started, Stopped, Paused, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method does not change the state.
addTimedTextSource	{Prepared, Started, Stopped, Paused, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method does not change the state.
selectTrack	{Prepared, Started, Stopped, Paused, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method does not change the state.
deselectTrack	{Prepared, Started, Stopped, Paused, PlaybackCompleted}	{Idle, Initialized, Error}	Successful invoke of this method does not change the state.

Permissions

One may need to declare a corresponding WAKE_LOCK permission `<uses-permission>` (https://developer.android.com/reference/android/R.styleable.html#AndroidManifestUsesPermission) element.

This class requires the `Manifest.permission.INTERNET` (https://developer.android.com/reference/android/Manifest.permission.html#INTERNET) permission when used with network-based content.

Callbacks

Applications may want to register for informational and error events in order to be informed of some internal state update and possible runtime errors during playback or streaming. Registration for these events is done by properly setting the appropriate listeners (via calls to `setOnPreparedListener(OnPreparedListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener))setOnPreparedListener, `setOnVideoSizeChangedListener(OnVideoSizeChangedListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnVideoSizeChangedListener(android.media.MediaPlayer.OnVideoSizeChangedListener)) setOnVideoSizeChangedListener, `setOnSeekCompleteListener(OnSeekCompleteListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSeekCompleteListener(android.media.MediaPlayer.OnSeekCompleteListener)) setOnSeekCompleteListener, `setOnCompletionListener(OnCompletionListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnCompletionListener(android.media.MediaPlayer.OnCompletionListener))setOnCompletionListener, `setOnBufferingUpdateListener(OnBufferingUpdateListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnBufferingUpdateListener(android.media.MediaPlayer.OnBufferingUpdateListener)) setOnBufferingUpdateListener, `setOnInfoListener(OnInfoListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnInfoListener(android.media.MediaPlayer.OnInfoListener))setOnInfoListener, `setOnErrorListener(OnErrorListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnErrorListener(android.media.MediaPlayer.OnErrorListener)) setOnErrorListener, etc). In order to receive the respective callback associated with these listeners, applications are required to create MediaPlayer objects on a thread with its own Looper running (main UI thread by default has a Looper running).

Summary

Nested classes

`class`	`MediaPlayer.DrmInfo` (https://developer.android.com/reference/android/media/MediaPlayer.DrmInfo.html) Encapsulates the DRM properties of the source.

`class`	`MediaPlayer.MetricsConstants` (https://developer.android.com/reference/android/media/MediaPlayer.MetricsConstants.html)
`class`	`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html) Thrown when a DRM method is called before preparing a DRM scheme through prepareDrm().
`interface`	`MediaPlayer.OnBufferingUpdateListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnBufferingUpdateListener.html) Interface definition of a callback to be invoked indicating buffering status of a media resource being streamed over the network.
`interface`	`MediaPlayer.OnCompletionListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnCompletionListener.html) Interface definition for a callback to be invoked when playback of a media source has completed.
`interface`	`MediaPlayer.OnDrmConfigHelper` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmConfigHelper.html) Interface definition of a callback to be invoked when the app can do DRM configuration (get/set properties) before the session is opened.
`interface`	`MediaPlayer.OnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmInfoListener.html) Interface definition of a callback to be invoked when the DRM info becomes available
`interface`	`MediaPlayer.OnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListener.html) Interface definition of a callback to notify the app when the DRM is ready for key request/response
`interface`	`MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html) Interface definition of a callback to be invoked when there has been an error during an asynchronous operation (other errors will throw exceptions at method call time).
`interface`	`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html) Interface definition of a callback to be invoked to communicate some info and/or warning about the media or its playback.
`interface`	`MediaPlayer.OnMediaTimeDiscontinuityListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnMediaTimeDiscontinuityListener.html) Interface definition of a callback to be invoked when discontinuity in the normal progression of the media time is detected.
`interface`	`MediaPlayer.OnPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnPreparedListener.html) Interface definition for a callback to be invoked when the media source is ready for playback.
`interface`	`MediaPlayer.OnSeekCompleteListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSeekCompleteListener.html) Interface definition of a callback to be invoked indicating the completion of a seek operation.

`interface`	`MediaPlayer.OnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSubtitleDataListener.html) Interface definition of a callback to be invoked when a player subtitle track has new subtitle data available.
`interface`	`MediaPlayer.OnTimedMetaDataAvailableListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedMetaDataAvailableListener.html) Interface definition of a callback to be invoked when a track has timed metadata available.
`interface`	`MediaPlayer.OnTimedTextListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedTextListener.html) Interface definition of a callback to be invoked when a timed text is available for display.
`interface`	`MediaPlayer.OnVideoSizeChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnVideoSizeChangedListener.html) Interface definition of a callback to be invoked when the video size is first known or updated
`class`	`MediaPlayer.ProvisioningNetworkErrorException` (https://developer.android.com/reference/android/media/MediaPlayer.ProvisioningNetworkErrorException.html) Thrown when the device requires DRM provisioning but the provisioning attempt has failed due to a network error (Internet reachability, timeout, etc.).
`class`	`MediaPlayer.ProvisioningServerErrorException` (https://developer.android.com/reference/android/media/MediaPlayer.ProvisioningServerErrorException.html) Thrown when the device requires DRM provisioning but the provisioning attempt has failed due to the provisioning server denying the request.
`class`	`MediaPlayer.TrackInfo` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html) Class for MediaPlayer to return each audio/video/subtitle track's metadata.

Constants

`int`	`MEDIA_ERROR_IO` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_IO) File or network related operation errors.
`int`	`MEDIA_ERROR_MALFORMED` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_MALFORMED) Bitstream is not conforming to the related coding standard or file spec.
`int`	`MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK)

	The video is streamed and its container is not valid for progressive playback i.e the video's index (e.g moov atom) is not at the start of the file.
`int`	`MEDIA_ERROR_SERVER_DIED` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_SERVER_DIED) Media server died.
`int`	`MEDIA_ERROR_TIMED_OUT` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_TIMED_OUT) Some operation takes too long to complete, usually more than 3-5 seconds.
`int`	`MEDIA_ERROR_UNKNOWN` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_UNKNOWN) Unspecified media player error.
`int`	`MEDIA_ERROR_UNSUPPORTED` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_ERROR_UNSUPPORTED) Bitstream is conforming to the related coding standard or file spec, but the media framework does not support the feature.
`int`	`MEDIA_INFO_AUDIO_NOT_PLAYING` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_AUDIO_NOT_PLAYING) Informs that audio is not playing.
`int`	`MEDIA_INFO_BAD_INTERLEAVING` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_BAD_INTERLEAVING) Bad interleaving means that a media has been improperly interleaved or not interleaved at all, e.g has all the video samples first then all the audio ones.
`int`	`MEDIA_INFO_BUFFERING_END` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_BUFFERING_END) MediaPlayer is resuming playback after filling buffers.
`int`	`MEDIA_INFO_BUFFERING_START` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_BUFFERING_START) MediaPlayer is temporarily pausing playback internally in order to buffer more data.
`int`	`MEDIA_INFO_METADATA_UPDATE` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_METADATA_UPDATE) A new set of metadata is available.
`int`	`MEDIA_INFO_NOT_SEEKABLE` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_NOT_SEEKABLE) The media cannot be seeked (e.g live stream)
`int`	`MEDIA_INFO_STARTED_AS_NEXT` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_STARTED_AS_NEXT) The player was started because it was used as the next player for another player, which just completed playback.
`int`	`MEDIA_INFO_SUBTITLE_TIMED_OUT` (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_SUBTITLE_TIMED_OUT) Reading the subtitle track takes too long.

int	MEDIA_INFO_UNKNOWN (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_UNKNOWN) Unspecified media player info.
int	MEDIA_INFO_UNSUPPORTED_SUBTITLE (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_UNSUPPORTED_SUBTITLE) Subtitle track was not supported by the media framework.
int	MEDIA_INFO_VIDEO_NOT_PLAYING (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_VIDEO_NOT_PLAYING) Informs that video is not playing.
int	MEDIA_INFO_VIDEO_RENDERING_START (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_VIDEO_RENDERING_START) The player just pushed the very first video frame for rendering.
int	MEDIA_INFO_VIDEO_TRACK_LAGGING (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_INFO_VIDEO_TRACK_LAGGING) The video is too complex for the decoder: it can't decode frames fast enough.
String (https://developer.android.com/reference/java/lang/String.html)	MEDIA_MIMETYPE_TEXT_SUBRIP (https://developer.android.com/reference/android/media/MediaPlayer.html#MEDIA_MIMETYPE_TEXT_SUBRIP) *This constant was deprecated in API level 28. use* MediaFormat.MIMETYPE_TEXT_SUBRIP (https://developer.android.com/reference/android/media/MediaFormat.html#MIMETYPE_TEXT_SUBRIP)
int	PREPARE_DRM_STATUS_PREPARATION_ERROR (https://developer.android.com/reference/android/media/MediaPlayer.html#PREPARE_DRM_STATUS_PREPARATION_ERROR) The DRM preparation has failed .
int	PREPARE_DRM_STATUS_PROVISIONING_NETWORK_ERROR (https://developer.android.com/reference/android/media/MediaPlayer.html#PREPARE_DRM_STATUS_PROVISIONING_NETWORK_ERROR) The device required DRM provisioning but couldn't reach the provisioning server.
int	PREPARE_DRM_STATUS_PROVISIONING_SERVER_ERROR (https://developer.android.com/reference/android/media/MediaPlayer.html#PREPARE_DRM_STATUS_PROVISIONING_SERVER_ERROR) The device required DRM provisioning but the provisioning server denied the request.
int	PREPARE_DRM_STATUS_SUCCESS (https://developer.android.com/reference/android/media/MediaPlayer.html#PREPARE_DRM_STATUS_SUCCESS) The status codes for MediaPlayer.OnDrmPreparedListener.onDrmPrepared(MediaPlayer, int) (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListener.html#onDrmPrepared(android.media.MediaPlayer,%20int)) listener.

`int`	`SEEK_CLOSEST` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_CLOSEST) This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a frame (not necessarily a key frame) associated with a data source that is located closest to or at the given time.
`int`	`SEEK_CLOSEST_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_CLOSEST_SYNC) This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located closest to (in time) or at the given time.
`int`	`SEEK_NEXT_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_NEXT_SYNC) This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located right after or at the given time.
`int`	`SEEK_PREVIOUS_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_PREVIOUS_SYNC) This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located right before or at the given time.
`int`	`VIDEO_SCALING_MODE_SCALE_TO_FIT` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT) Specifies a video scaling mode.
`int`	`VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING) Specifies a video scaling mode.

Fields

`protected AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html)	`mAttributes` (https://developer.android.com/reference/android/media/MediaPlayer.html#mAttributes)
`protected float`	`mAuxEffectSendLevel` (https://developer.android.com/reference/android/media/MediaPlayer.html#mAuxEffectSendLevel)
`protected float`	`mLeftVolume` (https://developer.android.com/reference/android/media/MediaPlayer.html#mLeftVolume)

`protected float`	`mRightVolume` (https://developer.android.com/reference/android/media/MediaPlayer.html#mRightVolume)

Public constructors

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html#MediaPlayer())`()`

Default constructor.

Public methods

`void`	`addOnRoutingChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#addOnRoutingChangedListener(android.media.AudioRouting.OnRoutingChangedListener,%20android.os.Handler))`(AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Adds an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) to receive notifications of routing changes on this MediaPlayer.
`void`	`addTimedTextSource` (https://developer.android.com/reference/android/media/MediaPlayer.html#addTimedTextSource(java.io.FileDescriptor,%20java.lang.String))`(FileDescriptor` (https://developer.android.com/reference/java/io/FileDescriptor.html) `fd, String` (https://developer.android.com/reference/java/lang/String.html) `mimeType)` Adds an external timed text source file (FileDescriptor).
`void`	`addTimedTextSource` (https://developer.android.com/reference/android/media/MediaPlayer.html#addTimedTextSource(java.lang.String,%20java.lang.String))`(String` (https://developer.android.com/reference/java/lang/String.html) `path, String` (https://developer.android.com/reference/java/lang/String.html) `mimeType)` Adds an external timed text source file.
`void`	`addTimedTextSource` (https://developer.android.com/reference/android/media/MediaPlayer.html#addTimedTextSource(java.io.FileDescriptor,%20long,%20long,%20java.lang.String))

	(FileDescriptor (https://developer.android.com/reference/java/io/FileDescriptor.html) fd, long offset, long length, String (https://developer.android.com/reference/java/lang/String.html) mime) Adds an external timed text file (FileDescriptor).
void	addTimedTextSource (https://developer.android.com/reference/android/media/MediaPlayer.html#addTimedTextSource(android.content.Context,%20android.net.Uri,%20java.lang.String)) (Context (https://developer.android.com/reference/android/content/Context.html) context, Uri (https://developer.android.com/reference/android/net/Uri.html) uri, String (https://developer.android.com/reference/java/lang/String.html) mimeType) Adds an external timed text source file (Uri).
void	attachAuxEffect (https://developer.android.com/reference/android/media/MediaPlayer.html#attachAuxEffect(int))(int effectId) Attaches an auxiliary effect to the player.
void	clearOnMediaTimeDiscontinuityListener (https://developer.android.com/reference/android/media/MediaPlayer.html#clearOnMediaTimeDiscontinuityListener())() Clears the listener previously set with setOnMediaTimeDiscontinuityListener(OnMediaTimeDiscontinuityListener) (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener)) or setOnMediaTimeDiscontinuityListener(OnMediaTimeDiscontinuityListener, Handler) (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener,%20android.os.Handler))
void	clearOnSubtitleDataListener (https://developer.android.com/reference/android/media/MediaPlayer.html#clearOnSubtitleDataListener())() Clears the listener previously set with setOnSubtitleDataListener(OnSubtitleDataListener) (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener)) or setOnSubtitleDataListener(OnSubtitleDataListener, Handler) (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener,%20android.os.Handler)) .
static MediaPlayer (https://developer.android.com/reference/android/media/MediaPlayer.html)	create (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri,%20android.view.SurfaceHolder,%20android.media.AudioAttributes,%20int)) (Context (https://developer.android.com/reference/android/content/Context.html) context, Uri (https://developer.android.com/reference/android/net/Uri.html) uri, SurfaceHolder (https://developer.android.com/reference/android/view/SurfaceHolder.html) holder, AudioAttributes (https://developer.android.com/reference/android/media/AudioAttributes.html) audioAttributes, int audioSessionId)

	Same factory method as `create(Context, Uri, SurfaceHolder)` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri,%20android.view.SurfaceHolder)) but that lets you specify the audio attributes and session ID to be used by the new MediaPlayer instance.
`static MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	`create` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20int,%20android.media.AudioAttributes,%20int)) `(Context` (https://developer.android.com/reference/android/content/Context.html) `context, int resid, AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) `audioAttributes, int audioSessionId)` Same factory method as `create(Context, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20int)) but that lets you specify the audio attributes and session ID to be used by the new MediaPlayer instance.
`static MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	`create` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri,%20android.view.SurfaceHolder)) `(Context` (https://developer.android.com/reference/android/content/Context.html) `context, Uri` (https://developer.android.com/reference/android/net/Uri.html) `uri, SurfaceHolder` (https://developer.android.com/reference/android/view/SurfaceHolder.html) `holder)` Convenience method to create a MediaPlayer for a given Uri.
`static MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	`create` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20int)) `(Context` (https://developer.android.com/reference/android/content/Context.html) `context, int resid)` Convenience method to create a MediaPlayer for a given resource id.
`static MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	`create` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri)) `(Context` (https://developer.android.com/reference/android/content/Context.html) `context, Uri` (https://developer.android.com/reference/android/net/Uri.html) `uri)` Convenience method to create a MediaPlayer for a given Uri.
`VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html)	`createVolumeShaper` (https://developer.android.com/reference/android/media/MediaPlayer.html#createVolumeShaper(android.media.VolumeShaper.Configuration)) `(VolumeShaper.Configuration` (https://developer.android.com/reference/android/media/VolumeShaper.Configuration.html) `configuration)` Returns a `VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html) object that can be used modify the volume envelope of the player or track.
`static void`	`deprecateStreamTypeForPlayback`

	(https://developer.android.com/reference/android/media/MediaPlayer.html#deprecateStreamTypeForPlayback(int,%20java.lang.String,%20java.lang.String)) (int streamType, String (https://developer.android.com/reference/java/lang/String.html) className, String (https://developer.android.com/reference/java/lang/String.html) opName) Use to generate warning or exception in legacy code paths that allowed passing stream types to qualify audio playback.
void	deselectTrack (https://developer.android.com/reference/android/media/MediaPlayer.html#deselectTrack(int))(int index) Deselect a track.
int	getAudioSessionId (https://developer.android.com/reference/android/media/MediaPlayer.html#getAudioSessionId())() Returns the audio session ID.
int	getCurrentPosition (https://developer.android.com/reference/android/media/MediaPlayer.html#getCurrentPosition())() Gets the current playback position.
MediaPlayer.DrmInfo (https://developer.android.com/reference/android/media/MediaPlayer.DrmInfo.html)	getDrmInfo (https://developer.android.com/reference/android/media/MediaPlayer.html#getDrmInfo())() Retrieves the DRM Info associated with the current source
String (https://developer.android.com/reference/java/lang/String.html)	getDrmPropertyString (https://developer.android.com/reference/android/media/MediaPlayer.html#getDrmPropertyString(java.lang.String))(String (https://developer.android.com/reference/java/lang/String.html) propertyName) Read a DRM engine plugin String property value, given the property name string.
int	getDuration (https://developer.android.com/reference/android/media/MediaPlayer.html#getDuration())() Gets the duration of the file.
MediaDrm.KeyRequest (https://developer.android.com/reference/android/media/MediaDrm.KeyRequest.html)	getKeyRequest (https://developer.android.com/reference/android/media/MediaPlayer.html#getKeyRequest(byte[],%20byte[],%20java.lang.String,%20int,%20java.util.Map%3Cjava.lang.String,%20java.lang.String%3E)) (byte[] keySetId, byte[] initData, String (https://developer.android.com/reference/java/lang/String.html) mimeType, int keyType, Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.android.com/reference/java/lang/String.html)> optionalParameters) A key request/response exchange occurs between the app and a license server to obtain or release keys used to decrypt encrypted content.
PersistableBundle (https://developer.android.com/reference/android/os/PersistableBundle.html)	getMetrics (https://developer.android.com/reference/android/media/MediaPlayer.html#getMetrics())() Return Metrics data about the current player.

`PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html)	`getPlaybackParams` (https://developer.android.com/reference/android/media/MediaPlayer.html#getPlaybackParams()) `()` Gets the playback params, containing the current playback rate.
`AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)	`getPreferredDevice` (https://developer.android.com/reference/android/media/MediaPlayer.html#getPreferredDevice()) `()` Returns the selected output specified by `setPreferredDevice(AudioDeviceInfo)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setPreferredDevice(android.media.AudioDeviceInfo)).
`AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)	`getRoutedDevice` (https://developer.android.com/reference/android/media/MediaPlayer.html#getRoutedDevice()) `()` Returns an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) identifying the current routing of this MediaPlayer Note: The query is only valid if the MediaPlayer is currently playing.
`int`	`getSelectedTrack` (https://developer.android.com/reference/android/media/MediaPlayer.html#getSelectedTrack(int)) `(int trackType)` Returns the index of the audio, video, or subtitle track currently selected for playback, The return value is an index into the array returned by `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()), and can be used in calls to `selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int)) or `deselectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#deselectTrack(int)).
`SyncParams` (https://developer.android.com/reference/android/media/SyncParams.html)	`getSyncParams` (https://developer.android.com/reference/android/media/MediaPlayer.html#getSyncParams()) `()` Gets the A/V sync mode.
`MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html)	`getTimestamp` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTimestamp()) `()` Get current playback position as a `MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html).
`TrackInfo[]` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html)	`getTrackInfo` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) `()` Returns an array of track information.
`int`	`getVideoHeight` (https://developer.android.com/reference/android/media/MediaPlayer.html#getVideoHeight()) `()` Returns the height of the video.
`int`	`getVideoWidth` (https://developer.android.com/reference/android/media/MediaPlayer.html#getVideoWidth()) `()`

	Returns the width of the video.
`boolean`	`isLooping` (https://developer.android.com/reference/android/media/MediaPlayer.html#isLooping())`()` Checks whether the MediaPlayer is looping or non-looping.
`boolean`	`isPlaying` (https://developer.android.com/reference/android/media/MediaPlayer.html#isPlaying())`()` Checks whether the MediaPlayer is playing.
`void`	`pause` (https://developer.android.com/reference/android/media/MediaPlayer.html#pause())`()` Pauses playback.
`void`	`prepare` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare())`()` Prepares the player for playback, synchronously.
`void`	`prepareAsync` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync())`()` Prepares the player for playback, asynchronously.
`void`	`prepareDrm` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareDrm(java.util.UUID))`(UUID` (https://developer.android.com/reference/java/util/UUID.html) `uuid)` Prepares the DRM for the current source If `OnDrmConfigHelper` is registered, it will be called during preparation to allow configuration of the DRM properties before opening the DRM session.
`byte[]`	`provideKeyResponse` (https://developer.android.com/reference/android/media/MediaPlayer.html#provideKeyResponse(byte[],%20byte[]))`(byte[] keySetId, byte[] response)` A key response is received from the license server by the app, then it is provided to the DRM engine plugin using provideKeyResponse.
`void`	`release` (https://developer.android.com/reference/android/media/MediaPlayer.html#release())`()` Releases resources associated with this MediaPlayer object.
`void`	`releaseDrm` (https://developer.android.com/reference/android/media/MediaPlayer.html#releaseDrm())`()` Releases the DRM session The player has to have an active DRM session and be in stopped, or prepared state before this call is made.
`void`	`removeOnRoutingChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#removeOnRoutingChangedListener(android.media.AudioRouting.OnRoutingChangedListener))`(AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) `listener)`

	Removes an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) which has been previously added to receive rerouting notifications.
`void`	`reset` (https://developer.android.com/reference/android/media/MediaPlayer.html#reset())`()` Resets the MediaPlayer to its uninitialized state.
`void`	`restoreKeys` (https://developer.android.com/reference/android/media/MediaPlayer.html#restoreKeys(byte[]))`(byte[] keySetId)` Restore persisted offline keys into a new session.
`void`	`seekTo` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(int))`(int msec)` Seeks to specified time position.
`void`	`seekTo` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int))`(long msec, int mode)` Moves the media to specified time position by considering the given mode.
`void`	`selectTrack` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int))`(int index)` Selects a track.
`void`	`setAudioAttributes` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes))`(AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) `attributes)` Sets the audio attributes for this MediaPlayer.
`void`	`setAudioSessionId` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioSessionId(int))`(int sessionId)` Sets the audio session ID.
`void`	`setAudioStreamType` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioStreamType(int))`(int streamtype)` *This method was deprecated in API level 26. use* `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes))
`void`	`setAuxEffectSendLevel` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAuxEffectSendLevel(float))`(float level)` Sets the send level of the player to the attached auxiliary effect.
`void`	`setDataSource` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.content.res.AssetFileDescriptor))`(AssetFileDescriptor` (https://developer.android.com/reference/android/content/res/AssetFileDescriptor.html) `afd)` Sets the data source (AssetFileDescriptor) to use.
`void`	`setDataSource` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.io.FileDescriptor))`(FileDescriptor`

	(https://developer.android.com/reference/java/io/FileDescriptor.html) fd) Sets the data source (FileDescriptor) to use.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.io.FileDescriptor,%20long,%20long)) (FileDescriptor (https://developer.android.com/reference/java/io/FileDescriptor.html) fd, long offset, long length) Sets the data source (FileDescriptor) to use.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.lang.String)) (String (https://developer.android.com/reference/java/lang/String.html) path) Sets the data source (file-path or http/rtsp URL) to use.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.content.Context,%20android.net.Uri,%20java.util.Map%3Cjava.lang.String,%20java.lang.String%3E,%20java.util.List%3Cjava.net.HttpCookie%3E)) (Context (https://developer.android.com/reference/android/content/Context.html) context, Uri (https://developer.android.com/reference/android/net/Uri.html) uri, Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.android.com/reference/java/lang/String.html)> headers, List (https://developer.android.com/reference/java/util/List.html)<HttpCookie (https://developer.android.com/reference/java/net/HttpCookie.html)> cookies) Sets the data source as a content Uri.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.content.Context,%20android.net.Uri,%20java.util.Map%3Cjava.lang.String,%20java.lang.String%3E)) (Context (https://developer.android.com/reference/android/content/Context.html) context, Uri (https://developer.android.com/reference/android/net/Uri.html) uri, Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.android.com/reference/java/lang/String.html)> headers) Sets the data source as a content Uri.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.media.MediaDataSource)) (MediaDataSource (https://developer.android.com/reference/android/media/MediaDataSource.html) dataSource) Sets the data source (MediaDataSource) to use.
void	setDataSource (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(android.content.Context,%20android.net.Uri)) (Context (https://developer.android.com/reference/android/content/Context.html) context, Uri (https://developer.android.com/reference/android/net/Uri.html) uri) Sets the data source as a content Uri.
void	setDisplay (https://developer.android.com/reference/android/media/MediaPlayer.html#setDisplay(android.view.SurfaceHolder)) (SurfaceHolder

	(https://developer.android.com/reference/android/view/SurfaceHolder.html) `sh)` Sets the `SurfaceHolder` (https://developer.android.com/reference/android/view/SurfaceHolder.html) to use for displaying the video portion of the media.
`void`	`setDrmPropertyString` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDrmPropertyString(java.lang.String,%20java.lang.String))`(String` (https://developer.android.com/reference/java/lang/String.html) `propertyName, String` (https://developer.android.com/reference/java/lang/String.html) `value)` Set a DRM engine plugin String property value.
`void`	`setLooping` (https://developer.android.com/reference/android/media/MediaPlayer.html#setLooping(boolean))`(boolean looping)` Sets the player to be looping or non-looping.
`void`	`setNextMediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html#setNextMediaPlayer(android.media.MediaPlayer)) `(MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html) `next)` Set the MediaPlayer to start when this MediaPlayer finishes playback (i.e.
`void`	`setOnBufferingUpdateListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnBufferingUpdateListener(android.media.MediaPlayer.OnBufferingUpdateListener)) `(MediaPlayer.OnBufferingUpdateListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnBufferingUpdateListener.html) `listener)` Register a callback to be invoked when the status of a network stream's buffer has changed.
`void`	`setOnCompletionListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnCompletionListener(android.media.MediaPlayer.OnCompletionListener)) `(MediaPlayer.OnCompletionListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnCompletionListener.html) `listener)` Register a callback to be invoked when the end of a media source has been reached during playback.
`void`	`setOnDrmConfigHelper` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnDrmConfigHelper(android.media.MediaPlayer.OnDrmConfigHelper)) `(MediaPlayer.OnDrmConfigHelper` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmConfigHelper.html) `listener)` Register a callback to be invoked for configuration of the DRM object before the session is created.
`void`	`setOnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnDrmInfoListener(android.media.MediaPlayer.OnDrmInfoListener)) `(MediaPlayer.OnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmInfoListener.html) `listener)` Register a callback to be invoked when the DRM info is known.

`void`	`setOnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnDrmInfoListener(android.media.MediaPlayer.OnDrmInfoListener,%20android.os.Handler)) (`MediaPlayer.OnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmInfoListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Register a callback to be invoked when the DRM info is known.
`void`	`setOnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnDrmPreparedListener(android.media.MediaPlayer.OnDrmPreparedListener,%20android.os.Handler)) (`MediaPlayer.OnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Register a callback to be invoked when the DRM object is prepared.
`void`	`setOnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnDrmPreparedListener(android.media.MediaPlayer.OnDrmPreparedListener)) (`MediaPlayer.OnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListener.html) `listener)` Register a callback to be invoked when the DRM object is prepared.
`void`	`setOnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnErrorListener(android.media.MediaPlayer.OnErrorListener)) (`MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html) `listener)` Register a callback to be invoked when an error has happened during an asynchronous operation.
`void`	`setOnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnInfoListener(android.media.MediaPlayer.OnInfoListener)) (`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html) `listener)` Register a callback to be invoked when an info/warning is available.
`void`	`setOnMediaTimeDiscontinuityListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener,%20android.os.Handler)) (`MediaPlayer.OnMediaTimeDiscontinuityListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnMediaTimeDiscontinuityListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Sets the listener to be invoked when a media time discontinuity is encountered.

`void`	`setOnMediaTimeDiscontinuityListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener)) (`MediaPlayer.OnMediaTimeDiscontinuityListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnMediaTimeDiscontinuityListener.html) `listener)` Sets the listener to be invoked when a media time discontinuity is encountered.
`void`	`setOnPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener)) (`MediaPlayer.OnPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnPreparedListener.html) `listener)` Register a callback to be invoked when the media source is ready for playback.
`void`	`setOnSeekCompleteListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSeekCompleteListener(android.media.MediaPlayer.OnSeekCompleteListener)) (`MediaPlayer.OnSeekCompleteListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSeekCompleteListener.html) `listener)` Register a callback to be invoked when a seek operation has been completed.
`void`	`setOnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener)) (`MediaPlayer.OnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSubtitleDataListener.html) `listener)` Sets the listener to be invoked when a subtitle track has new data available.
`void`	`setOnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener,%20android.os.Handler)) (`MediaPlayer.OnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSubtitleDataListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Sets the listener to be invoked when a subtitle track has new data available.
`void`	`setOnTimedMetaDataAvailableListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnTimedMetaDataAvailableListener(android.media.MediaPlayer.OnTimedMetaDataAvailableListener)) (`MediaPlayer.OnTimedMetaDataAvailableListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedMetaDataAvailableListener.html) `listener)` Register a callback to be invoked when a selected track has timed metadata available.

`void`	`setOnTimedTextListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnTimedTextListener(android.media.MediaPlayer.OnTimedTextListener)) (`MediaPlayer.OnTimedTextListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedTextListener.html) `listener)` Register a callback to be invoked when a timed text is available for display.
`void`	`setOnVideoSizeChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnVideoSizeChangedListener(android.media.MediaPlayer.OnVideoSizeChangedListener)) (`MediaPlayer.OnVideoSizeChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnVideoSizeChangedListener.html) `listener)` Register a callback to be invoked when the video size is known or updated.
`void`	`setPlaybackParams` (https://developer.android.com/reference/android/media/MediaPlayer.html#setPlaybackParams(android.media.PlaybackParams)) (`PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html) `params)` Sets playback rate using `PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html).
`boolean`	`setPreferredDevice` (https://developer.android.com/reference/android/media/MediaPlayer.html#setPreferredDevice(android.media.AudioDeviceInfo)) (`AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) `deviceInfo)` Specifies an audio device (via an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) object) to route the output from this MediaPlayer.
`void`	`setScreenOnWhilePlaying` (https://developer.android.com/reference/android/media/MediaPlayer.html#setScreenOnWhilePlaying(boolean))`(boolean screenOn)` Control whether we should use the attached SurfaceHolder to keep the screen on while video playback is occurring.
`void`	`setSurface` (https://developer.android.com/reference/android/media/MediaPlayer.html#setSurface(android.view.Surface))(`Surface` (https://developer.android.com/reference/android/view/Surface.html) `surface)` Sets the `Surface` (https://developer.android.com/reference/android/view/Surface.html) to be used as the sink for the video portion of the media.
`void`	`setSyncParams` (https://developer.android.com/reference/android/media/MediaPlayer.html#setSyncParams(android.media.SyncParams))(`SyncParams` (https://developer.android.com/reference/android/media/SyncParams.html) `params)` Sets A/V sync mode.
`void`	`setVideoScalingMode` (https://developer.android.com/reference/android/media/MediaPlayer.html#setVideoScalingMode(int))`(int mode)` Sets video scaling mode.
`void`	`setVolume` (https://developer.android.com/reference/android/media/MediaPlayer.html#setVolume(float,%20float))`(float leftVolume, float rightVolume)`

	Sets the volume on this player.
`void`	`setWakeMode` (https://developer.android.com/reference/android/media/MediaPlayer.html#setWakeMode(android.content.Context,%20int)) (`Context` (https://developer.android.com/reference/android/content/Context.html) `context, int mode`) Set the low-level power management behavior for this MediaPlayer.
`void`	`start` (https://developer.android.com/reference/android/media/MediaPlayer.html#start()) () Starts or resumes playback.
`void`	`stop` (https://developer.android.com/reference/android/media/MediaPlayer.html#stop()) () Stops playback after playback has been started or paused.

Protected methods

`void`	`baseRegisterPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html#baseRegisterPlayer()) () Call from derived class when instantiation / initialization is successful
`void`	`finalize` (https://developer.android.com/reference/android/media/MediaPlayer.html#finalize()) () Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`int`	`getStartDelayMs` (https://developer.android.com/reference/android/media/MediaPlayer.html#getStartDelayMs()) ()

Inherited methods

From class `java.lang.Object` (https://developer.android.com/reference/java/lang/Object.html)

`Object` (https://developer.android.com/reference/java/lang/Object.html)	`clone` (https://developer.android.com/reference/java/lang/Object.html#clone()) () Creates and returns a copy of this object.
`boolean`	`equals` (https://developer.android.com/reference/java/lang/Object.html#equals(java.lang.Object)) (`Object` (https://developer.android.com/reference/java/lang/Object.html) `obj`) Indicates whether some other object is "equal to" this one.

`void`	`finalize` (https://developer.android.com/reference/java/lang/Object.html#finalize()) `()` Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`final Class` (https://developer.android.com/reference/java/lang/Class.html)`<?>`	`getClass` (https://developer.android.com/reference/java/lang/Object.html#getClass()) `()` Returns the runtime class of this `Object`.
`int`	`hashCode` (https://developer.android.com/reference/java/lang/Object.html#hashCode()) `()` Returns a hash code value for the object.
`final void`	`notify` (https://developer.android.com/reference/java/lang/Object.html#notify()) `()` Wakes up a single thread that is waiting on this object's monitor.
`final void`	`notifyAll` (https://developer.android.com/reference/java/lang/Object.html#notifyAll()) `()` Wakes up all threads that are waiting on this object's monitor.
`String` (https://developer.android.com/reference/java/lang/String.html)	`toString` (https://developer.android.com/reference/java/lang/Object.html#toString()) `()` Returns a string representation of the object.
`final void`	`wait` (https://developer.android.com/reference/java/lang/Object.html#wait(long,%20int)) `(long millis, int nanos)` Causes the current thread to wait until another thread invokes the `notify()` (https://developer.android.com/reference/java/lang/Object.html#notify()) method or the `notifyAll()` (https://developer.android.com/reference/java/lang/Object.html#notifyAll()) method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed.
`final void`	`wait` (https://developer.android.com/reference/java/lang/Object.html#wait(long)) `(long millis)` Causes the current thread to wait until either another thread invokes the `notify()` (https://developer.android.com/reference/java/lang/Object.html#notify()) method or the `notifyAll()` (https://developer.android.com/reference/java/lang/Object.html#notifyAll()) method for this object, or a specified amount of time has elapsed.
`final void`	`wait` (https://developer.android.com/reference/java/lang/Object.html#wait()) `()` Causes the current thread to wait until another thread invokes the `notify()` (https://developer.android.com/reference/java/lang/Object.html#notify()) method or the `notifyAll()` (https://developer.android.com/reference/java/lang/Object.html#notifyAll()) method for this object.

From interface `android.media.VolumeAutomation` (https://developer.android.com/reference/android/media/VolumeAutomation.html)

`abstract VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html)	`createVolumeShaper` (https://developer.android.com/reference/android/media/VolumeAutomation.html#createVolumeShaper(android.media.VolumeShaper.Configuration)) `(VolumeShaper.Configuration` (https://developer.android.com/reference/android/media/VolumeShaper.Configuration.html) `configuration)` Returns a `VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html) object that can be used modify the volume envelope of the player or track.

From interface `android.media.AudioRouting` (https://developer.android.com/reference/android/media/AudioRouting.html)

`abstract void`	`addOnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.html#addOnRoutingChangedListener(android.media.AudioRouting.OnRoutingChangedListener,%20android.os.Handler)) `(AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) `listener, Handler` (https://developer.android.com/reference/android/os/Handler.html) `handler)` Adds an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) to receive notifications of routing changes on this AudioTrack/AudioRecord.
`abstract AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)	`getPreferredDevice` (https://developer.android.com/reference/android/media/AudioRouting.html#getPreferredDevice()) `()`

	Returns the selected output/input specified by `setPreferredDevice(AudioDeviceInfo)` (https://developer.android.com/reference/android/media/AudioRouting.html#setPreferredDevice(android.media.AudioDeviceInfo)) .
`abstract AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)	`getRoutedDevice` (https://developer.android.com/reference/android/media/AudioRouting.html#getRoutedDevice()) `()` Returns an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) identifying the current routing of this AudioTrack/AudioRecord.
`abstract void`	`removeOnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.html#removeOnRoutingChangedListener(android.media.AudioRouting.OnRoutingChangedListener)) `(AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) `listener)` Removes an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) which has been previously added to receive rerouting notifications.
`abstract boolean`	`setPreferredDevice` (https://developer.android.com/reference/android/media/AudioRouting.html#setPreferredDevice(android.media.AudioDeviceInfo)) `(AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) `deviceInfo)` Specifies an audio device (via an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) object) to route the output/input to/from.

Constants

MEDIA_ERROR_IO

added in API level 17 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_IO
```

File or network related operation errors.

Constant Value: -1004 (0xfffffc14)

MEDIA_ERROR_MALFORMED

added in API level 17 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_MALFORMED
```

Bitstream is not conforming to the related coding standard or file spec.

Constant Value: -1007 (0xfffffc11)

MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_NOT_VALID_FOR_PROGRESSIVE_PLAYBACK
```

The video is streamed and its container is not valid for progressive playback i.e the video's index (e.g moov atom) is not at the start of the file.

See also:

`MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html)

Constant Value: 200 (0x000000c8)

MEDIA_ERROR_SERVER_DIED

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_SERVER_DIED
```

Media server died. In this case, the application must release the MediaPlayer object and instantiate a new one.

See also:

`MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html)

Constant Value: 100 (0x00000064)

MEDIA_ERROR_TIMED_OUT

added in API level 17 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_TIMED_OUT
```

Some operation takes too long to complete, usually more than 3-5 seconds.

Constant Value: -110 (0xffffff92)

MEDIA_ERROR_UNKNOWN

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_UNKNOWN
```

Unspecified media player error.

See also:

`MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html)

Constant Value: 1 (0x00000001)

MEDIA_ERROR_UNSUPPORTED

added in API level 17 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_ERROR_UNSUPPORTED
```

Bitstream is conforming to the related coding standard or file spec, but the media framework does not support the feature.

Constant Value: -1010 (0xfffffc0e)

MEDIA_INFO_AUDIO_NOT_PLAYING

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_AUDIO_NOT_PLAYING
```

Informs that audio is not playing. Note that playback of the video is not interrupted.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 804 (0x00000324)

MEDIA_INFO_BAD_INTERLEAVING

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_BAD_INTERLEAVING
```

Bad interleaving means that a media has been improperly interleaved or not interleaved at all, e.g has all the video samples first then all the audio ones. Video is playing but a lot of disk seeks may be happening.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 800 (0x00000320)

MEDIA_INFO_BUFFERING_END

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_BUFFERING_END
```

MediaPlayer is resuming playback after filling buffers.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 702 (0x000002be)

MEDIA_INFO_BUFFERING_START

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_BUFFERING_START
```

MediaPlayer is temporarily pausing playback internally in order to buffer more data.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 701 (0x000002bd)

MEDIA_INFO_METADATA_UPDATE

added in API level 5 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_METADATA_UPDATE
```

A new set of metadata is available.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 802 (0x00000322)

MEDIA_INFO_NOT_SEEKABLE

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_NOT_SEEKABLE
```

The media cannot be seeked (e.g live stream)

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 801 (0x00000321)

MEDIA_INFO_STARTED_AS_NEXT

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_STARTED_AS_NEXT
```

The player was started because it was used as the next player for another player, which just completed playback.

See also:

`setNextMediaPlayer(MediaPlayer)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setNextMediaPlayer(android.media.MediaPlayer))

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 2 (0x00000002)

MEDIA_INFO_SUBTITLE_TIMED_OUT

added in API level 19 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_SUBTITLE_TIMED_OUT
```

Reading the subtitle track takes too long.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 902 (0x00000386)

MEDIA_INFO_UNKNOWN

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_UNKNOWN
```

Unspecified media player info.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 1 (0x00000001)

MEDIA_INFO_UNSUPPORTED_SUBTITLE

added in API level 19 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_UNSUPPORTED_SUBTITLE
```

Subtitle track was not supported by the media framework.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 901 (0x00000385)

MEDIA_INFO_VIDEO_NOT_PLAYING

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_VIDEO_NOT_PLAYING
```

Informs that video is not playing. Note that playback of the audio is not interrupted.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 805 (0x00000325)

MEDIA_INFO_VIDEO_RENDERING_START

added in API level 17 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_VIDEO_RENDERING_START
```

The player just pushed the very first video frame for rendering.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 3 (0x00000003)

MEDIA_INFO_VIDEO_TRACK_LAGGING

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int MEDIA_INFO_VIDEO_TRACK_LAGGING
```

The video is too complex for the decoder: it can't decode frames fast enough. Possibly only the audio plays fine at this stage.

See also:

`MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html)

Constant Value: 700 (0x000002bc)

MEDIA_MIMETYPE_TEXT_SUBRIP

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final String` (https://developer.android.com/reference/java/lang/String.html) `MEDIA_MIMETYPE_TEXT_SUBRIP`

error**This constant was deprecated in API level 28.**

use **`MediaFormat.MIMETYPE_TEXT_SUBRIP`** (https://developer.android.com/reference/android/media/MediaFormat.html#MIMETYPE_TEXT_SUBRIP)

MIME type for SubRip (SRT) container. Used in addTimedTextSource APIs.

Constant Value: "application/x-subrip"

PREPARE_DRM_STATUS_PREPARATION_ERROR

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int PREPARE_DRM_STATUS_PREPARATION_ERROR
```

The DRM preparation has failed .

Constant Value: 3 (0x00000003)

PREPARE_DRM_STATUS_PROVISIONING_NETWORK_ERROR

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int PREPARE_DRM_STATUS_PROVISIONING_NETWORK_ERROR`

The device required DRM provisioning but couldn't reach the provisioning server.

Constant Value: 1 (0x00000001)

PREPARE_DRM_STATUS_PROVISIONING_SERVER_ERROR

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int PREPARE_DRM_STATUS_PROVISIONING_SERVER_ERROR`

The device required DRM provisioning but the provisioning server denied the request.

Constant Value: 2 (0x00000002)

PREPARE_DRM_STATUS_SUCCESS

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int PREPARE_DRM_STATUS_SUCCESS`

The status codes for `MediaPlayer.OnDrmPreparedListener.onDrmPrepared(MediaPlayer, int)` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListener.html#onDrmPrepared(android.media.MediaPlayer,%20int)) listener.

DRM preparation has succeeded.

Constant Value: 0 (0x00000000)

SEEK_CLOSEST

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int SEEK_CLOSEST`

This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a frame (not necessarily a key frame) associated with a data source that is located closest to or at the given time.

See also:

`seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int))

Constant Value: 3 (0x00000003)

SEEK_CLOSEST_SYNC

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int SEEK_CLOSEST_SYNC
```

This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located closest to (in time) or at the given time.

See also:

`seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int))

Constant Value: 2 (0x00000002)

SEEK_NEXT_SYNC

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int SEEK_NEXT_SYNC
```

This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located right after or at the given time.

See also:

`seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int))

Constant Value: 1 (0x00000001)

SEEK_PREVIOUS_SYNC

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static final int SEEK_PREVIOUS_SYNC
```

This mode is used with `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) to move media position to a sync (or key) frame associated with a data source that is located right before or at the given time.

See also:

`seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int))

Constant Value: 0 (0x00000000)

VIDEO_SCALING_MODE_SCALE_TO_FIT

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int VIDEO_SCALING_MODE_SCALE_TO_FIT`

Specifies a video scaling mode. The content is stretched to the surface rendering area. When the surface has the same aspect ratio as the content, the aspect ratio of the content is maintained; otherwise, the aspect ratio of the content is not maintained when video is being rendered. Unlike `VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING), there is no content cropping with this video scaling mode.

Constant Value: 1 (0x00000001)

VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static final int VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING`

Specifies a video scaling mode. The content is scaled, maintaining its aspect ratio. The whole surface area is always used. When the aspect ratio of the content is the same as the surface, no content is cropped; otherwise, content is cropped to fit the surface.

Constant Value: 2 (0x00000002)

Fields

mAttributes

`protected AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) `mAttributes`

mAuxEffectSendLevel

```
protected float mAuxEffectSendLevel
```

mLeftVolume

```
protected float mLeftVolume
```

mRightVolume

```
protected float mRightVolume
```

Public constructors

MediaPlayer

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public MediaPlayer ()
```

Default constructor. Consider using one of the create() methods for synchronously instantiating a MediaPlayer from a Uri or resource.

When done with the MediaPlayer, you should call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()), to free the resources. If not released, too many MediaPlayer instances may result in an exception.

Public methods

addOnRoutingChangedListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void addOnRoutingChangedListener (`AudioRouting.OnRoutingChangedListener (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingCh
    Handler (https://developer.android.com/reference/android/os/Handler.html) `handler)`

Adds an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) to receive notifications of routing changes on this MediaPlayer.

Parameters

`listener`	`AudioRouting.OnRoutingChangedListener`: The `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) interface to receive notifications of rerouting events.
`handler`	`Handler`: Specifies the `Handler` (https://developer.android.com/reference/android/os/Handler.html) object for the thread on which to execute the callback. If `null`, the handler on the main looper will be used.

addTimedTextSource

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void addTimedTextSource (`FileDescriptor (https://developer.android.com/reference/java/io/FileDescriptor.html) `fd,`
    String (https://developer.android.com/reference/java/lang/String.html) `mimeType)`

Adds an external timed text source file (FileDescriptor). It is the caller's responsibility to close the file descriptor. It is safe to do so as soon as this call returns. Currently supported format is SubRip. Note that a single external timed text source may contain multiple tracks in it. One can find the total number of available tracks using `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to see what additional tracks become available after this method call.

Parameters

`fd`	`FileDescriptor`: the FileDescriptor for the file you want to play
`mimeType`	`String`: The mime type of the file. Must be one of the mime types listed above.

Throws

`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if the mimeType is not supported.
`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

addTimedTextSource

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void addTimedTextSource (String (https://developer.android.com/reference/java/lang/String.html) path,
                String (https://developer.android.com/reference/java/lang/String.html) mimeType)
```

Adds an external timed text source file. Currently supported format is SubRip with the file extension .srt, case insensitive. Note that a single external timed text source may contain multiple tracks in it. One can find the total number of available tracks using `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to see what additional tracks become available after this method call.

Parameters

`path`	`String`: The file path of external timed text source file.
`mimeType`	`String`: The mime type of the file. Must be one of the mime types listed above.

Throws

`IOException`	if the file cannot be accessed or is corrupted.

(https://developer.android.com/reference/java/io/IOException.html)	
IllegalArgumentException (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if the mimeType is not supported.
IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

addTimedTextSource

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void addTimedTextSource (FileDescriptor (https://developer.android.com/reference/java/io/FileDescriptor.html) fd,
                long offset,
                long length,
                String (https://developer.android.com/reference/java/lang/String.html) mime)
```

Adds an external timed text file (FileDescriptor). It is the caller's responsibility to close the file descriptor. It is safe to do so as soon as this call returns. Currently supported format is SubRip. Note that a single external timed text source may contain multiple tracks in it. One can find the total number of available tracks using `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to see what additional tracks become available after this method call.

Parameters

`fd`	`FileDescriptor`: the FileDescriptor for the file you want to play
`offset`	`long`: the offset into the file where the data to be played starts, in bytes
`length`	`long`: the length in bytes of the data to be played
`mime`	`String`: The mime type of the file. Must be one of the mime types listed above.

Throws

`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if the mimeType is not supported.
`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

addTimedTextSource

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void addTimedTextSource (Context (https://developer.android.com/reference/android/content/Context.html) context,
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri,
                String (https://developer.android.com/reference/java/lang/String.html) mimeType)
```

Adds an external timed text source file (Uri). Currently supported format is SubRip with the file extension .srt, case insensitive. Note that a single external timed text source may contain multiple tracks in it. One can find the total number of available tracks using `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to see what additional tracks become available after this method call.

Parameters

`context`	`Context`: the Context to use when resolving the Uri
`uri`	`Uri`: the Content URI of the data you want to play
`mimeType`	`String`: The mime type of the file. Must be one of the mime types listed above.

Throws

`IOException` (https://developer.android.com/reference/java/io/IOException.html)	if the file cannot be accessed or is corrupted.
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if the mimeType is not supported.
`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

attachAuxEffect

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void attachAuxEffect (int effectId)
```

Attaches an auxiliary effect to the player. A typical auxiliary effect is a reverberation effect which can be applied on any sound source that directs a certain amount of its energy to this effect. This amount is defined by setAuxEffectSendLevel(). See `setAuxEffectSendLevel(float)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAuxEffectSendLevel(float)).

After creating an auxiliary effect (e.g. `EnvironmentalReverb` (https://developer.android.com/reference/android/media/audiofx/EnvironmentalReverb.html)), retrieve its ID with `AudioEffect.getId()` (https://developer.android.com/reference/android/media/audiofx/AudioEffect.html#getId()) and use it when calling this method to attach the player to the effect.

To detach the effect from the player, call this method with a null effect id.

This method must be called after one of the overloaded `setDataSource` methods.

Parameters

`effectId`	`int`: system wide unique id of the effect to attach

clearOnMediaTimeDiscontinuityListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void clearOnMediaTimeDiscontinuityListener ()
```

Clears the listener previously set with `setOnMediaTimeDiscontinuityListener(OnMediaTimeDiscontinuityListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener)) or `setOnMediaTimeDiscontinuityListener(OnMediaTimeDiscontinuityListener, Handler)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnMediaTimeDiscontinuityListener(android.media.MediaPlayer.OnMediaTimeDiscontinuityListener,%20android.os.Handler))

clearOnSubtitleDataListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void clearOnSubtitleDataListener ()
```

Clears the listener previously set with `setOnSubtitleDataListener(OnSubtitleDataListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener)) or `setOnSubtitleDataListener(OnSubtitleDataListener, Handler)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnSubtitleDataListener(android.media.MediaPlayer.OnSubtitleDataListener,%20android.os.Handler)).

create

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static MediaPlayer (https://developer.android.com/reference/android/media/MediaPlayer.html) create (Context (https://developer.android.com/reference/android/content/C
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri,
                SurfaceHolder (https://developer.android.com/reference/android/view/SurfaceHolder.html) holder,
                AudioAttributes (https://developer.android.com/reference/android/media/AudioAttributes.html) audioAttributes,
                int audioSessionId)
```

Same factory method as `create(Context, Uri, SurfaceHolder)` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20android.net.Uri,%20android.view.SurfaceHolder)) but that lets you specify the audio attributes and session ID to be used by the new MediaPlayer instance.

Parameters

`context`	`Context`: the Context to use
`uri`	`Uri`: the Uri from which to get the datasource
`holder`	`SurfaceHolder`: the SurfaceHolder to use for displaying the video, may be null.
`audioAttributes`	`AudioAttributes`: the `AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) to be used by the media player.
`audioSessionId`	`int`: the audio session ID to be used by the media player, see `AudioManager.generateAudioSessionId()` (https://developer.android.com/reference/android/media/AudioManager.html#generateAudioSessionId()) to obtain a new session.

Returns

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	a MediaPlayer object, or null if creation failed

create

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static MediaPlayer (https://developer.android.com/reference/android/media/MediaPlayer.html) create (Context (https://developer.android.com/reference/android/content/Co
                int resid,
                AudioAttributes (https://developer.android.com/reference/android/media/AudioAttributes.html) audioAttributes,
                int audioSessionId)
```

Same factory method as `create(Context, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#create(android.content.Context,%20int)) but that lets you specify the audio attributes and session ID to be used by the new MediaPlayer instance.

Parameters

`context`	`Context`: the Context to use
`resid`	`int`: the raw resource id (***R.raw.<something>***) for the resource to use as the datasource
`audioAttributes`	`AudioAttributes`: the `AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) to be used by the media player.
`audioSessionId`	`int`: the audio session ID to be used by the media player, see `AudioManager.generateAudioSessionId()` (https://developer.android.com/reference/android/media/AudioManager.html#generateAudioSessionId()) to obtain a new session.

Returns

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	a MediaPlayer object, or null if creation failed

create

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static MediaPlayer (https://developer.android.com/reference/android/media/MediaPlayer.html) create (Context (https://developer.android.com/reference/android/content/C
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri,
                SurfaceHolder (https://developer.android.com/reference/android/view/SurfaceHolder.html) holder)
```

Convenience method to create a MediaPlayer for a given Uri. On success, `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) will already have been called and must not be called again.

When done with the MediaPlayer, you should call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()), to free the resources. If not released, too many MediaPlayer instances will result in an exception.

Note that since `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) is called automatically in this method, you cannot change the audio session ID (see `setAudioSessionId(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioSessionId(int))) or audio attributes (see `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes)) of the new MediaPlayer.

Parameters

`context`	`Context`: the Context to use
`uri`	`Uri`: the Uri from which to get the datasource
`holder`	`SurfaceHolder`: the SurfaceHolder to use for displaying the video

Returns

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	a MediaPlayer object, or null if creation failed

create

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public static MediaPlayer (https://developer.android.com/reference/android/media/MediaPlayer.html) create (Context (https://developer.android.com/reference/android/content/C
                int resid)
```

Convenience method to create a MediaPlayer for a given resource id. On success, `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) will already have been called and must not be called again.

When done with the MediaPlayer, you should call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()), to free the resources. If not released, too many MediaPlayer instances will result in an exception.

Note that since `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) is called automatically in this method, you cannot change the audio session ID (see `setAudioSessionId(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioSessionId(int))) or audio attributes (see `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes)) of the new MediaPlayer.

Parameters

`context`	`Context`: the Context to use
`resid`	`int`: the raw resource id (***R.raw.<something>***) for the resource to use as the datasource

Returns

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	a MediaPlayer object, or null if creation failed

create

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public static` `MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html) `create (``Context` (https://developer.android.com/reference/android/content/C
`Uri` (https://developer.android.com/reference/android/net/Uri.html) `uri)`

Convenience method to create a MediaPlayer for a given Uri. On success, `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) will already have been called and must not be called again.

When done with the MediaPlayer, you should call `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release()), to free the resources. If not released, too many MediaPlayer instances will result in an exception.

Note that since `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) is called automatically in this method, you cannot change the audio session ID (see `setAudioSessionId(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioSessionId(int))) or audio attributes (see `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes)) of the new MediaPlayer.

Parameters

`context`	`Context`: the Context to use
`uri`	`Uri`: the Uri from which to get the datasource

Returns

`MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html)	a MediaPlayer object, or null if creation failed

createVolumeShaper

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html) `createVolumeShaper (VolumeShaper.Configuration` (https://developer.andro

Returns a `VolumeShaper` (https://developer.android.com/reference/android/media/VolumeShaper.html) object that can be used modify the volume envelope of the player or track.

Parameters

`configuration`	`VolumeShaper.Configuration`: the `configuration` (https://developer.android.com/reference/android/media/VolumeShaper.Configuration.html) that specifies the curve and duration to use. This value must never be `null`.

Returns

`VolumeShaper`	a `VolumeShaper` object This value will never be `null`.

(https://developer.android.com/reference/android/media/VolumeShaper.html)

deprecateStreamTypeForPlayback

```
public static void deprecateStreamTypeForPlayback (int streamType,
                String (https://developer.android.com/reference/java/lang/String.html) className,
                String (https://developer.android.com/reference/java/lang/String.html) opName)
```

Use to generate warning or exception in legacy code paths that allowed passing stream types to qualify audio playback.

Parameters

`streamType`	`int`: the stream type to check
`className`	`String`
`opName`	`String`

Throws

`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)

deselectTrack

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void deselectTrack (int index)
```

Deselect a track.

Currently, the track must be a timed text track and no audio or video tracks can be deselected. If the timed text track identified by index has not been selected before, it throws an exception.

Parameters

`index`	`int`: the index of the track to be deselected. The valid range of the index is 0..total number of tracks - 1. The total number of tracks as well as the type of each individual track can be found by calling `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) method.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

See also:

`getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo())

getAudioSessionId

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public int getAudioSessionId ()
```

Returns the audio session ID.

Returns

`int`	the audio session ID. Note that the audio session ID is 0 only if a problem occured when the MediaPlayer was contructed.

getCurrentPosition

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public int getCurrentPosition ()`

Gets the current playback position.

Returns

`int`	the current position in milliseconds

getDrmInfo

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public MediaPlayer.DrmInfo` (https://developer.android.com/reference/android/media/MediaPlayer.DrmInfo.html) `getDrmInfo ()`

Retrieves the DRM Info associated with the current source

Returns

`MediaPlayer.DrmInfo` (https://developer.android.com/reference/android/media/MediaPlayer.DrmInfo.html)	

Throws

`IllegalStateException`	if called before prepare()

(https://developer.android.com/reference/java/lang/IllegalStateException.html)

getDrmPropertyString

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public String` (https://developer.android.com/reference/java/lang/String.html) `getDrmPropertyString (String` (https://developer.android.com/reference/java/lang/String.html) `prope`

Read a DRM engine plugin String property value, given the property name string.

Parameters

`propertyName`	`String`: the property name Standard fields names are: `MediaDrm.PROPERTY_VENDOR` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VENDOR), `MediaDrm.PROPERTY_VERSION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VERSION), `MediaDrm.PROPERTY_DESCRIPTION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_DESCRIPTION), `MediaDrm.PROPERTY_ALGORITHMS` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_ALGORITHMS) This value must never be `null`. Value is `PROPERTY_VENDOR` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VENDOR), `PROPERTY_VERSION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VERSION), `PROPERTY_DESCRIPTION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_DESCRIPTION) or `PROPERTY_ALGORITHMS` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_ALGORITHMS).

Returns

`String` (https://developer.android.com/reference/java/lang/String.html)	This value will never be `null`.

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)	

getDuration

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public int getDuration ()
```

Gets the duration of the file.

Returns

`int`	the duration in milliseconds, if no duration is available (for example, if streaming live content), -1 is returned.

getKeyRequest

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public MediaDrm.KeyRequest (https://developer.android.com/reference/android/media/MediaDrm.KeyRequest.html) getKeyRequest (byte[] keySetId,
                byte[] initData,
                String (https://developer.android.com/reference/java/lang/String.html) mimeType,
                int keyType,
                Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.an
```

A key request/response exchange occurs between the app and a license server to obtain or release keys used to decrypt encrypted content.

getKeyRequest() is used to obtain an opaque key request byte array that is delivered to the license server. The opaque key request byte array is returned in KeyRequest.data. The recommended URL to deliver the key request to is returned in KeyRequest.defaultUrl.

After the app has received the key request response from the server, it should deliver to the response to the DRM engine plugin using the method `provideKeyResponse(byte[], byte[])` (https://developer.android.com/reference/android/media/MediaPlayer.html#provideKeyResponse(byte[],%20byte[])).

Parameters

`keySetId`	`byte`: is the key-set identifier of the offline keys being released when keyType is `MediaDrm.KEY_TYPE_RELEASE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_RELEASE). It should be set to null for other key requests, when keyType is `MediaDrm.KEY_TYPE_STREAMING` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_STREAMING) or `MediaDrm.KEY_TYPE_OFFLINE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_OFFLINE).
`initData`	`byte`: is the container-specific initialization data when the keyType is `MediaDrm.KEY_TYPE_STREAMING` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_STREAMING) or `MediaDrm.KEY_TYPE_OFFLINE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_OFFLINE). Its meaning is interpreted based on the mime type provided in the mimeType parameter. It could contain, for example, the content ID, key ID or other data obtained from the content metadata that is required in generating the key request. When the keyType is `MediaDrm.KEY_TYPE_RELEASE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_RELEASE), it should be set to null.
`mimeType`	`String`: identifies the mime type of the content This value may be `null`.
`keyType`	`int`: specifies the type of the request. The request may be to acquire keys for streaming, `MediaDrm.KEY_TYPE_STREAMING` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_STREAMING), or for offline content `MediaDrm.KEY_TYPE_OFFLINE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_OFFLINE), or to release previously acquired keys (`MediaDrm.KEY_TYPE_RELEASE` (https://developer.android.com/reference/android/media/MediaDrm.html#KEY_TYPE_RELEASE)), which are identified by a keySetId.
`optionalParameters`	`Map`: are included in the key request message to allow a client application to provide additional message parameters to the server. This may be `null` if no additional parameters are to be sent.

Returns

`MediaDrm.KeyRequest`	This value will never be `null`.

(https://developer.android.com/reference/android/media/MediaDrm.KeyRequest.html)	

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)	if there is no active DRM session

getMetrics

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public` `PersistableBundle` (https://developer.android.com/reference/android/os/PersistableBundle.html) `getMetrics ()`

Return Metrics data about the current player.

Returns

`PersistableBundle` (https://developer.android.com/reference/android/os/PersistableBundle.html)	a `PersistableBundle` (https://developer.android.com/reference/android/os/PersistableBundle.html) containing the set of attributes and values available for the media being handled by this instance of MediaPlayer The attributes are descibed in `MediaPlayer.MetricsConstants` (https://developer.android.com/reference/android/media/MediaPlayer.MetricsConstants.html). Additional vendor-specific fields may also be present in the return value.

getPlaybackParams

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public` `PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html) `getPlaybackParams ()`

Gets the playback params, containing the current playback rate.

Returns

`PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html)	the playback params. This value will never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized.

getPreferredDevice

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) `getPreferredDevice ()`

Returns the selected output specified by `setPreferredDevice(AudioDeviceInfo)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setPreferredDevice(android.media.AudioDeviceInfo)). Note that this is not guaranteed to correspond to the actual device being used for playback.

Returns

`AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)

getRoutedDevice

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) `getRoutedDevice ()`

Returns an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) identifying the current routing of this MediaPlayer Note: The query is only valid if the MediaPlayer is currently playing. If the player is not playing, the returned device can be null or correspond to previously selected device when the player was last active.

Returns

`AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html)	

getSelectedTrack

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public int getSelectedTrack (int trackType)`

Returns the index of the audio, video, or subtitle track currently selected for playback, The return value is an index into the array returned by `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()), and can be used in calls to `selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int)) or `deselectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#deselectTrack(int)).

Parameters

`trackType`	`int`: should be one of `MediaPlayer.TrackInfo.MEDIA_TRACK_TYPE_VIDEO` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#MEDIA_TRACK_TYPE_VIDEO), `MediaPlayer.TrackInfo.MEDIA_TRACK_TYPE_AUDIO` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#MEDIA_TRACK_TYPE_AUDIO), or

	`MediaPlayer.TrackInfo.MEDIA_TRACK_TYPE_SUBTITLE` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#MEDIA_TRACK_TYPE_SUBTITLE)

Returns

`int`	index of the audio, video, or subtitle track currently selected for playback; a negative integer is returned when there is no selected track for `trackType` or when `trackType` is not one of audio, video, or subtitle.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called after `release()` (https://developer.android.com/reference/android/media/MediaPlayer.html#release())

See also:

`getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo())

`selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int))

`deselectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#deselectTrack(int))

getSyncParams

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public SyncParams` (https://developer.android.com/reference/android/media/SyncParams.html) `getSyncParams ()`

Gets the A/V sync mode.

Returns

`SyncParams` (https://developer.android.com/reference/android/media/SyncParams.html)	the A/V sync params This value will never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized.

getTimestamp

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html) `getTimestamp ()`

Get current playback position as a `MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html).

The MediaTimestamp represents how the media time correlates to the system time in a linear fashion using an anchor and a clock rate. During regular playback, the media time moves fairly constantly (though the anchor frame may be rebased to a current system time, the linear correlation stays steady). Therefore, this method does not need to be called often.

To help users get current playback position, this method always anchors the timestamp to the current `system time` (https://developer.android.com/reference/java/lang/System.html#nanoTime()), so `MediaTimestamp.getAnchorMediaTimeUs()` (https://developer.android.com/reference/android/media/MediaTimestamp.html#getAnchorMediaTimeUs()) can be used as current playback position.

Returns

`MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html)	a MediaTimestamp object if a timestamp is available, or `null` if no timestamp is available, e.g. because the media player has not been initialized.

See also:

`MediaTimestamp` (https://developer.android.com/reference/android/media/MediaTimestamp.html)

getTrackInfo

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public TrackInfo[]` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html) `getTrackInfo ()`

Returns an array of track information.

Returns

`TrackInfo[]` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html)	Array of track info. The total number of tracks is the array length. Must be called again if an external timed text source has been added after any of the addTimedTextSource methods are called.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state.

getVideoHeight

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public int getVideoHeight ()`

Returns the height of the video.

Returns

`int`	the height of the video, or 0 if there is no video, no display surface was set, or the height has not been determined yet. The OnVideoSizeChangedListener can be registered via `setOnVideoSizeChangedListener(OnVideoSizeChangedListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnVideoSizeChangedListener(android.media.MediaPlayer.OnVideoSizeChangedListener)) to provide a notification when the height is available.

getVideoWidth

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public int getVideoWidth ()
```

Returns the width of the video.

Returns

`int`	the width of the video, or 0 if there is no video, no display surface was set, or the width has not been determined yet. The OnVideoSizeChangedListener can be registered via `setOnVideoSizeChangedListener(OnVideoSizeChangedListener)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setOnVideoSizeChangedListener(android.media.MediaPlayer.OnVideoSizeChangedListener)) to provide a notification when the width is available.

isLooping

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public boolean isLooping ()
```

Checks whether the MediaPlayer is looping or non-looping.

Returns

`boolean`	true if the MediaPlayer is currently looping, false otherwise

isPlaying

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public boolean isPlaying ()
```

Checks whether the MediaPlayer is playing.

Returns

`boolean`	true if currently playing, false otherwise

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized or has been released.

pause

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void pause ()
```

Pauses playback. Call start() to resume.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized.

prepare

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void prepare ()
```

Prepares the player for playback, synchronously. After setting the datasource and the display surface, you need to either call prepare() or prepareAsync(). For files, it is OK to call prepare(), which blocks until MediaPlayer is ready for playback.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	

prepareAsync

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void prepareAsync ()
```

Prepares the player for playback, asynchronously. After setting the datasource and the display surface, you need to either call prepare() or prepareAsync(). For streams, you should call prepareAsync(), which returns immediately, rather than blocking until enough data has been buffered.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state

prepareDrm

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void prepareDrm (UUID` (https://developer.android.com/reference/java/util/UUID.html) `uuid)`

Prepares the DRM for the current source

If `OnDrmConfigHelper` is registered, it will be called during preparation to allow configuration of the DRM properties before opening the DRM session. Note that the callback is called synchronously in the thread that called `prepareDrm`. It should be used only for a series of `getDrmPropertyString` and `setDrmPropertyString` calls and refrain from any lengthy operation.

If the device has not been provisioned before, this call also provisions the device which involves accessing the provisioning server and can take a variable time to complete depending on the network connectivity. If `OnDrmPreparedListener` is registered, prepareDrm() runs in non-blocking mode by launching the provisioning in the background and returning. The listener will be called when provisioning and preparation has finished. If a `OnDrmPreparedListener` is not registered, prepareDrm() waits till provisioning and preparation has finished, i.e., runs in blocking mode.

If `OnDrmPreparedListener` is registered, it is called to indicate the DRM session being ready. The application should not make any assumption about its call sequence (e.g., before or after prepareDrm returns), or the thread context that will execute the listener (unless the listener is registered with a handler thread).

Parameters

`uuid`	`UUID`: The UUID of the crypto scheme. If not known beforehand, it can be retrieved from the source through `getDrmInfo` or registering a `onDrmInfoListener`. This value must never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called before prepare(), or the DRM was prepared already
`UnsupportedSchemeException` (https://developer.android.com/reference/android/media/UnsupportedSchemeException.html)	if the crypto scheme is not supported
`ResourceBusyException` (https://developer.android.com/reference/android/media/ResourceBusyException.html)	if required DRM resources are in use
`MediaPlayer.ProvisioningNetworkErrorException` (https://developer.android.com/reference/android/media/MediaPlayer.ProvisioningNetworkErrorException.html)	if provisioning is required but failed due to a network error
`MediaPlayer.ProvisioningServerErrorException` (https://developer.android.com/reference/android/media/MediaPlayer.ProvisioningServerErrorException.html)	if provisioning is required but failed due to the request denied by the provisioning server

provideKeyResponse

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public byte[] provideKeyResponse (byte[] keySetId,
                byte[] response)
```

A key response is received from the license server by the app, then it is provided to the DRM engine plugin using provideKeyResponse. When the response is for an offline key request, a key-set identifier is returned that can be used to later restore the keys to a new session with the method {@ link # restoreKeys}. When the response is for a streaming or release request, null is returned.

Parameters

`keySetId`	`byte`: When the response is for a release request, keySetId identifies the saved key associated with the release request (i.e., the same keySetId passed to the earlier {@ link # getKeyRequest} call. It MUST be null when the response is for either streaming or offline key requests.
`response`	`byte`: the byte array response from the server This value must never be `null`.

Returns

`byte[]`

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)	if there is no active DRM session
`DeniedByServerException` (https://developer.android.com/reference/android/media/DeniedByServerException.html)	if the response indicates that the server rejected the request

release

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void release ()
```

Releases resources associated with this MediaPlayer object. It is considered good practice to call this method when you're done using the MediaPlayer. In particular, whenever an Activity of an application is paused (its onPause() method is called), or stopped (its onStop() method is called), this method should be invoked to release the MediaPlayer object, unless the application has a special need to keep the object around. In addition to unnecessary resources (such as memory and instances of codecs) being held, failure to call this method immediately if a MediaPlayer object is no longer needed may also lead to continuous battery consumption for mobile devices, and playback failure for other applications if no multiple instances of the same codec are supported on a device. Even if multiple instances of the same codec are supported, some performance degradation may be expected when unnecessary multiple instances are used at the same time.

releaseDrm

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void releaseDrm ()
```

Releases the DRM session

The player has to have an active DRM session and be in stopped, or prepared state before this call is made. A `reset()` call will release the DRM session implicitly.

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)	if there is no active DRM session to release

removeOnRoutingChangedListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void removeOnRoutingChangedListener (AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutin

Removes an `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) which has been previously added to receive rerouting notifications.

Parameters

`listener`	`AudioRouting.OnRoutingChangedListener`: The previously added `AudioRouting.OnRoutingChangedListener` (https://developer.android.com/reference/android/media/AudioRouting.OnRoutingChangedListener.html) interface to remove.

reset

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void reset ()`

Resets the MediaPlayer to its uninitialized state. After calling this method, you will have to initialize it again by setting the data source and calling prepare().

restoreKeys

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void restoreKeys (byte[] keySetId)`

Restore persisted offline keys into a new session. keySetId identifies the keys to load, obtained from a prior call to `provideKeyResponse(byte[], byte[])` (https://developer.android.com/reference/android/media/MediaPlayer.html#provideKeyResponse(byte[],%20byte[])).

Parameters

`keySetId`	`byte`: identifies the saved key set to restore This value must never be `null`.

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)	

seekTo

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void seekTo (int msec)
```

Seeks to specified time position. Same as `seekTo(long, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#seekTo(long,%20int)) with `mode = SEEK_PREVIOUS_SYNC`.

Parameters

`msec`	`int`: the offset in milliseconds from the start to seek to

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized

seekTo

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void seekTo (long msec,
                int mode)
```

Moves the media to specified time position by considering the given mode.

When seekTo is finished, the user will be notified via OnSeekComplete supplied by the user. There is at most one active seekTo processed at any time. If there is a to-be-completed seekTo, new seekTo requests will be queued in such a way that only the last request is kept. When current seekTo is completed, the queued request will be processed if that request is different from just-finished seekTo operation, i.e., the requested position or mode is different.

Parameters

`msec`	`long`: the offset in milliseconds from the start to seek to. When seeking to the given time position, there is no guarantee that the data source has a frame located at the position. When this happens, a frame nearby will be rendered. If msec is negative, time position zero will be used. If msec is larger than duration, duration will be used.
`mode`	`int`: the mode indicating where exactly to seek to. Use `SEEK_PREVIOUS_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_PREVIOUS_SYNC) if one wants to seek to a sync frame that has a timestamp earlier than or the same as msec. Use `SEEK_NEXT_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_NEXT_SYNC) if one wants to seek to a sync frame that has a timestamp later than or the same as msec. Use `SEEK_CLOSEST_SYNC` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_CLOSEST_SYNC) if one wants to seek to a sync frame that has a timestamp closest to or the same as msec. Use `SEEK_CLOSEST` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_CLOSEST) if one wants to seek to a frame that may or may not be a sync frame but is closest to or the same as msec. `SEEK_CLOSEST` (https://developer.android.com/reference/android/media/MediaPlayer.html#SEEK_CLOSEST) often has larger performance overhead compared to the other options if there is no sync frame located at msec.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if the mode is invalid.

selectTrack

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void selectTrack (int index)
```

Selects a track.

If a MediaPlayer is in invalid state, it throws an IllegalStateException exception. If a MediaPlayer is in *Started* state, the selected track is presented immediately. If a MediaPlayer is not in Started state, it just marks the track to be played.

In any valid state, if it is called multiple times on the same type of track (ie. Video, Audio, Timed Text), the most recent one will be chosen.

The first audio and video tracks are selected by default if available, even though this method is not called. However, no timed text track will be selected until this function is called.

Currently, only timed text, subtitle or audio tracks can be selected via this method. In addition, the support for selecting an audio track at runtime is pretty limited in that an audio track can only be selected in the *Prepared* state.

Parameters

`index`	`int`: the index of the track to be selected. The valid range of the index is 0..total number of track - 1. The total number of tracks as well as the type of each individual track can be found by calling `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) method.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if called in an invalid state.

See also:

`getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo())

setAudioAttributes

added in API level 21 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setAudioAttributes (AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) `attributes)`

Sets the audio attributes for this MediaPlayer. See `AudioAttributes` (https://developer.android.com/reference/android/media/AudioAttributes.html) for how to build and configure an instance of this class. You must call this method before `prepare()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepare()) or `prepareAsync()` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareAsync()) in order for the audio attributes to become effective thereafter.

Parameters

`attributes`	`AudioAttributes`: a non-null set of audio attributes

Throws

`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)

setAudioSessionId

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setAudioSessionId (int sessionId)`

Sets the audio session ID.

Parameters

`sessionId`	`int`: the audio session ID. The audio session ID is a system wide unique identifier for the audio stream played by this MediaPlayer instance. The primary use of the audio session ID is to associate audio effects to a particular instance of MediaPlayer: if an audio session ID is provided when creating an audio effect, this effect will be applied only to the audio content of media players within the same audio session and not to the output mix. When created, a MediaPlayer instance automatically generates its own audio session ID. However, it is possible to force this player to be part of an already existing audio session by calling this method. This method must be called before one of the overloaded `setDataSource` methods.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	

setAudioStreamType

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setAudioStreamType (int streamtype)
```

error**This method was deprecated in API level 26.**
use `setAudioAttributes(AudioAttributes)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setAudioAttributes(android.media.AudioAttributes))

Sets the audio stream type for this MediaPlayer. See `AudioManager` (https://developer.android.com/reference/android/media/AudioManager.html) for a list of stream types. Must call this method before prepare() or prepareAsync() in order for the target stream type to become effective thereafter.

Parameters

`streamtype`	`int`: the audio stream type

See also:

AudioManager (https://developer.android.com/reference/android/media/AudioManager.html)

setAuxEffectSendLevel

added in API level 9 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setAuxEffectSendLevel (float level)
```

Sets the send level of the player to the attached auxiliary effect. See `attachAuxEffect(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#attachAuxEffect(int)). The level value range is 0 to 1.0.

By default the send level is 0, so even if an effect is attached to the player this method must be called for the effect to be applied.

Note that the passed level value is a raw scalar. UI controls should be scaled logarithmically: the gain applied by audio framework ranges from -72dB to 0dB, so an appropriate conversion from linear UI input x to level is: x == 0 -> level = 0 0 < x <= R -> level = 10^(72*(x-R)/20/R)

Parameters

`level`	`float`: send level scalar

setDataSource

added in API level 24 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDataSource (AssetFileDescriptor` (https://developer.android.com/reference/android/content/res/AssetFileDescriptor.html) `afd)`

Sets the data source (AssetFileDescriptor) to use. It is the caller's responsibility to close the file descriptor. It is safe to do so as soon as this call returns.

Parameters

`afd`	`AssetFileDescriptor`: the AssetFileDescriptor for the file you want to play This value must never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if afd is not a valid AssetFileDescriptor
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	if afd can not be read

setDataSource

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDataSource (FileDescriptor` (https://developer.android.com/reference/java/io/FileDescriptor.html) `fd)`

Sets the data source (FileDescriptor) to use. It is the caller's responsibility to close the file descriptor. It is safe to do so as soon as this call returns.

Parameters

`fd`	`FileDescriptor`: the FileDescriptor for the file you want to play

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if fd is not a valid FileDescriptor
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	if fd can not be read

setDataSource

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setDataSource (FileDescriptor (https://developer.android.com/reference/java/io/FileDescriptor.html) fd,
                long offset,
                long length)
```

Sets the data source (FileDescriptor) to use. The FileDescriptor must be seekable (N.B. a LocalSocket is not seekable). It is the caller's responsibility to close the file descriptor. It is safe to do so as soon as this call returns.

Parameters

`fd`	`FileDescriptor`: the FileDescriptor for the file you want to play
`offset`	`long`: the offset into the file where the data to be played starts, in bytes
`length`	`long`: the length in bytes of the data to be played

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if fd is not a valid FileDescriptor
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	if fd can not be read

setDataSource

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDataSource (String` (https://developer.android.com/reference/java/lang/String.html) `path)`

Sets the data source (file-path or http/rtsp URL) to use.

When `path` refers to a local file, the file may actually be opened by a process other than the calling application. This implies that the pathname should be an absolute path (as any other process runs with unspecified current working directory), and that the pathname should reference a world-readable file. As an alternative, the application could first open the file for reading, and then use the file descriptor form `setDataSource(FileDescriptor)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDataSource(java.io.FileDescriptor)).

Parameters

`path`	`String`: the path of the file, or the http/rtsp URL of the stream you want to play

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	
`SecurityException` (https://developer.android.com/reference/java/lang/SecurityException.html)	

setDataSource

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setDataSource (Context (https://developer.android.com/reference/android/content/Context.html) context,
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri,
                Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.an
                List (https://developer.android.com/reference/java/util/List.html)<HttpCookie (https://developer.android.com/reference/java/net/HttpCookie.html)> cookies)
```

Sets the data source as a content Uri. To provide cookies for the subsequent HTTP requests, you can install your own default cookie handler and use other variants of setDataSource APIs instead. Alternatively, you can use this API to pass the cookies as a list of HttpCookie. If the app has not installed a CookieHandler already, this API creates a CookieManager and populates its CookieStore with the provided cookies. If the app has installed its own handler already, this API requires the handler to be of CookieManager type such that the API can update the manager’s CookieStore.

Note that the cross domain redirection is allowed by default, but that can be changed with key/value pairs through the headers parameter with "android-allow-cross-domain-redirect" as the key and "0" or "1" as the value to disallow or allow cross domain redirection.

Parameters

`context`	`Context`: the Context to use when resolving the Uri This value must never be `null`.
`uri`	`Uri`: the Content URI of the data you want to play This value must never be `null`.
`headers`	`Map`: the headers to be sent together with the request for the data The headers must not include cookies. Instead, use the cookies param. This value may be `null`.
`cookies`	`List`: the cookies to be sent together with the request This value may be `null`.

Throws

`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if cookies are provided and the installed handler is not a CookieManager
`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`NullPointerException` (https://developer.android.com/reference/java/lang/NullPointerException.html)	if context or uri is null
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	if uri has a file scheme and an I/O error occurs

setDataSource

added in API level 14 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setDataSource (Context (https://developer.android.com/reference/android/content/Context.html) context,
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri,
                Map (https://developer.android.com/reference/java/util/Map.html)<String (https://developer.android.com/reference/java/lang/String.html), String (https://developer.an
```

Sets the data source as a content Uri.

Note that the cross domain redirection is allowed by default, but that can be changed with key/value pairs through the headers parameter with "android-allow-cross-domain-redirect" as the key and "0" or "1" as the value to disallow or allow cross domain redirection.

Parameters

`context`	`Context`: the Context to use when resolving the Uri This value must never be `null`.
`uri`	`Uri`: the Content URI of the data you want to play This value must never be `null`.
`headers`	`Map`: the headers to be sent together with the request for the data This value may be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	
`IllegalArgumentException`	

(https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	
`SecurityException` (https://developer.android.com/reference/java/lang/SecurityException.html)	

setDataSource

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDataSource (MediaDataSource` (https://developer.android.com/reference/android/media/MediaDataSource.html) `dataSource)`

Sets the data source (MediaDataSource) to use.

Parameters

`dataSource`	`MediaDataSource`: the MediaDataSource for the media you want to play

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if dataSource is not a valid MediaDataSource

setDataSource

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setDataSource (Context (https://developer.android.com/reference/android/content/Context.html) context, 
                Uri (https://developer.android.com/reference/android/net/Uri.html) uri)
```

Sets the data source as a content Uri.

Parameters

`context`	`Context`: the Context to use when resolving the Uri This value must never be `null`.
`uri`	`Uri`: the Content URI of the data you want to play This value must never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state
`IOException` (https://developer.android.com/reference/java/io/IOException.html)	
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	
`SecurityException` (https://developer.android.com/reference/java/lang/SecurityException.html)	

setDisplay

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDisplay (SurfaceHolder` (https://developer.android.com/reference/android/view/SurfaceHolder.html) `sh)`

Sets the `SurfaceHolder` (https://developer.android.com/reference/android/view/SurfaceHolder.html) to use for displaying the video portion of the media. Either a surface holder or surface must be set if a display or video sink is needed. Not calling this method or `setSurface(Surface)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setSurface(android.view.Surface)) when playing back a video will result in only the audio track being played. A null surface holder or surface will result in only the audio track being played.

Parameters

`sh`	`SurfaceHolder`: the SurfaceHolder to use for video display

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized or has been released.

setDrmPropertyString

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setDrmPropertyString (String` (https://developer.android.com/reference/java/lang/String.html) `propertyName,`
`String` (https://developer.android.com/reference/java/lang/String.html) `value)`

Set a DRM engine plugin String property value.

Parameters

`propertyName`	`String`: the property name

	This value must never be `null`. Value is `PROPERTY_VENDOR` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VENDOR), `PROPERTY_VERSION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VERSION), `PROPERTY_DESCRIPTION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_DESCRIPTION) or `PROPERTY_ALGORITHMS` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_ALGORITHMS).
`value`	`String`: the property value Standard fields names are: `MediaDrm.PROPERTY_VENDOR` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VENDOR), `MediaDrm.PROPERTY_VERSION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_VERSION), `MediaDrm.PROPERTY_DESCRIPTION` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_DESCRIPTION), `MediaDrm.PROPERTY_ALGORITHMS` (https://developer.android.com/reference/android/media/MediaDrm.html#PROPERTY_ALGORITHMS) This value must never be `null`.

Throws

`MediaPlayer.NoDrmSchemeException` (https://developer.android.com/reference/android/media/MediaPlayer.NoDrmSchemeException.html)

setLooping

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setLooping (boolean looping)
```

Sets the player to be looping or non-looping.

Parameters

`looping`	`boolean`: whether to loop or not

setNextMediaPlayer

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setNextMediaPlayer (MediaPlayer` (https://developer.android.com/reference/android/media/MediaPlayer.html) `next)`

Set the MediaPlayer to start when this MediaPlayer finishes playback (i.e. reaches the end of the stream). The media framework will attempt to transition from this player to the next as seamlessly as possible. The next player can be set at any time before completion, but shall be after setDataSource has been called successfully. The next player must be prepared by the app, and the application should not call start() on it. The next MediaPlayer must be different from 'this'. An exception will be thrown if next == this. The application may call setNextMediaPlayer(null) to indicate no next player should be started at the end of playback. If the current player is looping, it will keep looping and the next player will not be started.

Parameters

`next`	`MediaPlayer`: the player to start after this one completes playback.

setOnBufferingUpdateListener

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnBufferingUpdateListener (MediaPlayer.OnBufferingUpdateListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnBufferingU

Register a callback to be invoked when the status of a network stream's buffer has changed.

Parameters

`listener`	`MediaPlayer.OnBufferingUpdateListener`: the callback that will be run.

setOnCompletionListener

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnCompletionListener (MediaPlayer.OnCompletionListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnCompletionListener.html

Register a callback to be invoked when the end of a media source has been reached during playback.

Parameters

`listener`	`MediaPlayer.OnCompletionListener`: the callback that will be run

setOnDrmConfigHelper

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnDrmConfigHelper (MediaPlayer.OnDrmConfigHelper` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmConfigHelper.html) `lister`

Register a callback to be invoked for configuration of the DRM object before the session is created. The callback will be invoked synchronously during the execution of `prepareDrm(UUID)` (https://developer.android.com/reference/android/media/MediaPlayer.html#prepareDrm(java.util.UUID)).

Parameters

`listener`	`MediaPlayer.OnDrmConfigHelper`: the callback that will be run

setOnDrmInfoListener

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnDrmInfoListener (MediaPlayer.OnDrmInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmInfoListener.html) `listene`

Register a callback to be invoked when the DRM info is known.

Parameters

`listener`	`MediaPlayer.OnDrmInfoListener`: the callback that will be run

setOnDrmInfoListener

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setOnDrmInfoListener (MediaPlayer.OnDrmInfoListener (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmInfoListener.html) listene
                Handler (https://developer.android.com/reference/android/os/Handler.html) handler)
```

Register a callback to be invoked when the DRM info is known.

Parameters

`listener`	`MediaPlayer.OnDrmInfoListener`: the callback that will be run
`handler`	`Handler`

setOnDrmPreparedListener

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setOnDrmPreparedListener (MediaPlayer.OnDrmPreparedListener (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListene
                Handler (https://developer.android.com/reference/android/os/Handler.html) handler)
```

Register a callback to be invoked when the DRM object is prepared.

Parameters

`listener`	`MediaPlayer.OnDrmPreparedListener`: the callback that will be run
`handler`	`Handler`: the Handler that will receive the callback

setOnDrmPreparedListener

added in API level 26 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnDrmPreparedListener (MediaPlayer.OnDrmPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnDrmPreparedListene

Register a callback to be invoked when the DRM object is prepared.

Parameters

`listener`	`MediaPlayer.OnDrmPreparedListener`: the callback that will be run

setOnErrorListener

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnErrorListener (MediaPlayer.OnErrorListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnErrorListener.html) `listener)`

Register a callback to be invoked when an error has happened during an asynchronous operation.

Parameters

`listener`	`MediaPlayer.OnErrorListener`: the callback that will be run

setOnInfoListener

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnInfoListener (MediaPlayer.OnInfoListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnInfoListener.html) `listener)`

Register a callback to be invoked when an info/warning is available.

Parameters

`listener`	`MediaPlayer.OnInfoListener`: the callback that will be run

setOnMediaTimeDiscontinuityListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setOnMediaTimeDiscontinuityListener (MediaPlayer.OnMediaTimeDiscontinuityListener (https://developer.android.com/reference/android/media/MediaP
                Handler (https://developer.android.com/reference/android/os/Handler.html) handler)
```

Sets the listener to be invoked when a media time discontinuity is encountered.

Parameters

`listener`	`MediaPlayer.OnMediaTimeDiscontinuityListener`: the listener called after a discontinuity This value must never be `null`.
`handler`	`Handler`: the `Handler` (https://developer.android.com/reference/android/os/Handler.html) that receives the listener events This value must never be `null`.

setOnMediaTimeDiscontinuityListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setOnMediaTimeDiscontinuityListener (MediaPlayer.OnMediaTimeDiscontinuityListener (https://developer.android.com/reference/android/media/MediaP
```

Sets the listener to be invoked when a media time discontinuity is encountered. The listener will be called on the same thread as the one in which the MediaPlayer was created.

Parameters

`listener`	`MediaPlayer.OnMediaTimeDiscontinuityListener`: the listener called after a discontinuity This value must never be `null`.

setOnPreparedListener

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnPreparedListener (MediaPlayer.OnPreparedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnPreparedListener.html) `list`

Register a callback to be invoked when the media source is ready for playback.

Parameters

`listener`	`MediaPlayer.OnPreparedListener`: the callback that will be run

setOnSeekCompleteListener

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnSeekCompleteListener (MediaPlayer.OnSeekCompleteListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSeekCompleteList

Register a callback to be invoked when a seek operation has been completed.

Parameters

`listener`	`MediaPlayer.OnSeekCompleteListener`: the callback that will be run

setOnSubtitleDataListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnSubtitleDataListener (MediaPlayer.OnSubtitleDataListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnSubtitleDataListen

Sets the listener to be invoked when a subtitle track has new data available. The subtitle data comes from a subtitle track previously selected with `selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int)). Use `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to determine which tracks are subtitles (of type `MediaPlayer.TrackInfo.MEDIA_TRACK_TYPE_SUBTITLE` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#MEDIA_TRACK_TYPE_SUBTITLE)), Subtitle track encodings can be determined by `MediaPlayer.TrackInfo.getFormat()` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#getFormat())).
See `SubtitleData` (https://developer.android.com/reference/android/media/SubtitleData.html) for an example of querying subtitle encoding.
The listener will be called on the same thread as the one in which the MediaPlayer was created.

Parameters

`listener`	`MediaPlayer.OnSubtitleDataListener`: the listener called when new data is available This value must never be `null`.

setOnSubtitleDataListener

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setOnSubtitleDataListener (MediaPlayer.OnSubtitleDataListener (https://developer.android.com/reference/android/media/MediaPlayer.OnSubtitleDataListen
                Handler (https://developer.android.com/reference/android/os/Handler.html) handler)
```

Sets the listener to be invoked when a subtitle track has new data available. The subtitle data comes from a subtitle track previously selected with `selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int)). Use `getTrackInfo()` (https://developer.android.com/reference/android/media/MediaPlayer.html#getTrackInfo()) to determine which tracks are subtitles (of type `MediaPlayer.TrackInfo.MEDIA_TRACK_TYPE_SUBTITLE` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#MEDIA_TRACK_TYPE_SUBTITLE)), Subtitle track encodings can be determined by `MediaPlayer.TrackInfo.getFormat()` (https://developer.android.com/reference/android/media/MediaPlayer.TrackInfo.html#getFormat())).
See `SubtitleData` (https://developer.android.com/reference/android/media/SubtitleData.html) for an example of querying subtitle encoding.

Parameters

`listener`	`MediaPlayer.OnSubtitleDataListener`: the listener called when new data is available

	This value must never be `null`.
`handler`	`Handler`: the `Handler` (https://developer.android.com/reference/android/os/Handler.html) that receives the listener events This value must never be `null`.

setOnTimedMetaDataAvailableListener

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnTimedMetaDataAvailableListener (MediaPlayer.OnTimedMetaDataAvailableListener` (https://developer.android.com/reference/android/media/MediaP

Register a callback to be invoked when a selected track has timed metadata available.

Currently only HTTP live streaming data URI's embedded with timed ID3 tags generates `TimedMetaData` (https://developer.android.com/reference/android/media/TimedMetaData.html).

Parameters

`listener`	`MediaPlayer.OnTimedMetaDataAvailableListener`: the callback that will be run

See also:

`selectTrack(int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#selectTrack(int))

`MediaPlayer.OnTimedMetaDataAvailableListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedMetaDataAvailableListener.html)

`TimedMetaData` (https://developer.android.com/reference/android/media/TimedMetaData.html)

setOnTimedTextListener

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnTimedTextListener (MediaPlayer.OnTimedTextListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnTimedTextListener.html) `l`

Register a callback to be invoked when a timed text is available for display.

Parameters

`listener`	`MediaPlayer.OnTimedTextListener`: the callback that will be run

setOnVideoSizeChangedListener

added in API level 3 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setOnVideoSizeChangedListener (MediaPlayer.OnVideoSizeChangedListener` (https://developer.android.com/reference/android/media/MediaPlayer.OnVideoSi

Register a callback to be invoked when the video size is known or updated.

Parameters

`listener`	`MediaPlayer.OnVideoSizeChangedListener`: the callback that will be run

setPlaybackParams

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setPlaybackParams (PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html) `params)`

Sets playback rate using `PlaybackParams` (https://developer.android.com/reference/android/media/PlaybackParams.html). The object sets its internal PlaybackParams to the input, except that the object remembers previous speed when input speed is zero. This allows the object to resume at previous speed when start() is called. Calling it before the object is prepared does not change the object state. After the object is prepared, calling it with zero speed is equivalent to calling pause(). After the object is prepared, calling it with non-zero speed is equivalent to calling start().

Parameters

`params`	`PlaybackParams`: the playback params. This value must never be `null`.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized or has been released.
`IllegalArgumentException` (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if params is not supported.

setPreferredDevice

added in API level 28 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public boolean setPreferredDevice (AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) `deviceInfo)`

Specifies an audio device (via an `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) object) to route the output from this MediaPlayer.

Parameters

`deviceInfo`	`AudioDeviceInfo`: The `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) specifying the audio sink or source. If deviceInfo is null, default routing is restored.

Returns

`boolean`	true if succesful, false if the specified `AudioDeviceInfo` (https://developer.android.com/reference/android/media/AudioDeviceInfo.html) is non-null and does not correspond to a valid audio device.

setScreenOnWhilePlaying

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setScreenOnWhilePlaying (boolean screenOn)`

Control whether we should use the attached SurfaceHolder to keep the screen on while video playback is occurring. This is the preferred method over `setWakeMode(Context, int)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setWakeMode(android.content.Context,%20int)) where possible, since it doesn't require that the application have permission for low-level wake lock access.

Parameters

`screenOn`	`boolean`: Supply true to keep the screen on, false to allow it to turn off.

setSurface

added in API level 14 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setSurface (Surface` (https://developer.android.com/reference/android/view/Surface.html) `surface)`

Sets the `Surface` (https://developer.android.com/reference/android/view/Surface.html) to be used as the sink for the video portion of the media. This is similar to `setDisplay(SurfaceHolder)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDisplay(android.view.SurfaceHolder)), but does not support `setScreenOnWhilePlaying(boolean)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setScreenOnWhilePlaying(boolean)). Setting a Surface will un-set any Surface or SurfaceHolder that was previously set. A null surface will result in only the audio track being played. If the Surface sends frames to a `SurfaceTexture` (https://developer.android.com/reference/android/graphics/SurfaceTexture.html), the timestamps returned from `SurfaceTexture.getTimestamp()` (https://developer.android.com/reference/android/graphics/SurfaceTexture.html#getTimestamp()) will have an unspecified zero point. These timestamps cannot be directly compared between different media sources, different instances of the same media source, or multiple runs of the same program. The timestamp is normally monotonically increasing and is unaffected by time-of-day adjustments, but it is reset when the position is set.

Parameters

`surface`	`Surface`: The `Surface` (https://developer.android.com/reference/android/view/Surface.html) to be used for the video portion of the media.

Throws

IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized or has been released.

setSyncParams

added in API level 23 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`public void setSyncParams (SyncParams` (https://developer.android.com/reference/android/media/SyncParams.html) `params)`

Sets A/V sync mode.

Parameters

`params`	`SyncParams`: the A/V sync params to apply This value must never be `null`.

Throws

IllegalStateException (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized.
IllegalArgumentException (https://developer.android.com/reference/java/lang/IllegalArgumentException.html)	if params are not supported.

setVideoScalingMode

added in API level 16 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setVideoScalingMode (int mode)
```

Sets video scaling mode. To make the target video scaling mode effective during playback, this method must be called after data source is set. If not called, the default video scaling mode is `VIDEO_SCALING_MODE_SCALE_TO_FIT` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT).

The supported video scaling modes are:

- `VIDEO_SCALING_MODE_SCALE_TO_FIT` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT)
- `VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING)

Parameters

`mode`	`int`: target video scaling mode. Must be one of the supported video scaling modes; otherwise, IllegalArgumentException will be thrown.

See also:

`VIDEO_SCALING_MODE_SCALE_TO_FIT` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT)

`VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING` (https://developer.android.com/reference/android/media/MediaPlayer.html#VIDEO_SCALING_MODE_SCALE_TO_FIT_WITH_CROPPING)

setVolume

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setVolume (float leftVolume,
                float rightVolume)
```

Sets the volume on this player. This API is recommended for balancing the output of audio streams within an application. Unless you are writing an application to control user settings, this API should be used in preference to `AudioManager.setStreamVolume(int, int, int)` (https://developer.android.com/reference/android/media/AudioManager.html#setStreamVolume(int,%20int,%20int)) which sets the volume of ALL streams of a particular type. Note that the passed volume values are raw scalars in range 0.0 to 1.0. UI controls should be scaled logarithmically.

Parameters

`leftVolume`	`float`: left volume scalar
`rightVolume`	`float`: right volume scalar

setWakeMode

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void setWakeMode (Context (https://developer.android.com/reference/android/content/Context.html) context, 
                int mode)
```

Set the low-level power management behavior for this MediaPlayer. This can be used when the MediaPlayer is not playing through a SurfaceHolder set with `setDisplay(SurfaceHolder)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setDisplay(android.view.SurfaceHolder)) and thus can use the high-level `setScreenOnWhilePlaying(boolean)` (https://developer.android.com/reference/android/media/MediaPlayer.html#setScreenOnWhilePlaying(boolean)) feature.

This function has the MediaPlayer access the low-level power manager service to control the device's power usage while playing is occurring. The parameter is a combination of `PowerManager` (https://developer.android.com/reference/android/os/PowerManager.html) wake flags. Use of this method requires `Manifest.permission.WAKE_LOCK` (https://developer.android.com/reference/android/Manifest.permission.html#WAKE_LOCK) permission. By default, no attempt is made to keep the device awake during playback.

Parameters

`context`	`Context`: the Context to use
`mode`	`int`: the power/wake mode to set

See also:

`PowerManager` (https://developer.android.com/reference/android/os/PowerManager.html)

start

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void start ()
```

Starts or resumes playback. If playback had previously been paused, playback will continue from where it was paused. If playback had been stopped, or never started before, playback will start at the beginning.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if it is called in an invalid state

stop

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

```
public void stop ()
```

Stops playback after playback has been started or paused.

Throws

`IllegalStateException` (https://developer.android.com/reference/java/lang/IllegalStateException.html)	if the internal player engine has not been initialized.

Protected methods

baseRegisterPlayer

`protected void baseRegisterPlayer ()`

Call from derived class when instantiation / initialization is successful

finalize

added in API level 1 (https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels)

`protected void finalize ()`

Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. A subclass overrides the `finalize` method to dispose of system resources or to perform other cleanup.

The general contract of `finalize` is that it is invoked if and when the Java™ virtual machine has determined that there is no longer any means by which this object can be accessed by any thread that has not yet died, except as a result of an action taken by the finalization of some other object or class which is ready to be finalized. The `finalize` method may take any action, including making this object available again to other threads; the usual purpose of `finalize`, however, is to perform cleanup actions before the object is irrevocably discarded. For example, the finalize method for an object that represents an input/output connection might perform explicit I/O transactions to break the connection before the object is permanently discarded.

The `finalize` method of class `Object` performs no special action; it simply returns normally. Subclasses of `Object` may override this definition.

The Java programming language does not guarantee which thread will invoke the `finalize` method for any given object. It is guaranteed, however, that the thread that invokes finalize will not be holding any user-visible synchronization locks when finalize is invoked. If an uncaught exception is thrown by the finalize method, the exception is ignored and finalization of that object terminates.

After the `finalize` method has been invoked for an object, no further action is taken until the Java virtual machine has again determined that there is no longer any means by which this object can be accessed by any thread that has not yet died, including possible actions by other objects or classes which are ready to be finalized, at which point the object may be discarded.

The `finalize` method is never invoked more than once by a Java virtual machine for any given object.

Any exception thrown by the `finalize` method causes the finalization of this object to be halted, but is otherwise ignored.

getStartDelayMs

`protected int getStartDelayMs ()`

Returns

`int`