Audio Player

One of the most important components of the media framework is the MediaPlayer class. An object of this class can fetch, decode, and play both audio and video with minimal setup. It supports several different media sources such as:

Local resources

Internal URIs, such as one you might obtain from a Content Resolver

External URLs (streaming)

For a list of media formats that Android supports, see the Android Supported Media Formats document.

Here is an example of how to play audio that's available as a local raw resource (saved in your application's res/raw/ directory):

MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound\_file\_1);

mediaPlayer.start(); // no need to call prepare(); create() does that for you

In this case, a "raw" resource is a file that the system does not try to parse in any particular way. However, the content of this resource should not be raw audio. It should be a properly encoded and formatted media file in one of the supported formats.

And here is how you might play from a URI available locally in the system (that you obtained through a Content Resolver, for instance):

Uri myUri = ....; // initialize Uri here

MediaPlayer mediaPlayer = new MediaPlayer();

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

mediaPlayer.setDataSource(getApplicationContext(), myUri);

mediaPlayer.prepare();

mediaPlayer.start();

Playing from a remote URL via HTTP streaming looks like this:

String url = "http://........"; // your URL here

MediaPlayer mediaPlayer = new MediaPlayer();

mediaPlayer.setAudioStreamType(AudioManager.STREAM\_MUSIC);

mediaPlayer.setDataSource(url);

mediaPlayer.prepare(); // might take long! (for buffering, etc)

mediaPlayer.start();

Note: If you're passing a URL to stream an online media file, the file must be capable of progressive download.

Caution: You must either catch or pass IllegalArgumentException and IOException when using setDataSource(), because the file you are referencing might not exist.