MediaRouter and RemotePlaybackClient

Interest in Casting

- Chromecast
  - Streaming TV device
  - Control Options
    - Web browser
    - iOS device
    - Android device
  - Cheap, buzz-worthy
- Net: interest in Android apps casting to Chromecast

The Cast Vision

- Media Centrally Located
  - Like... YouTube!
- App Playback Options
  - Play it back itself, for cases where there is no Chromecast around or user wants local viewing
  - Tell Chromecast to play it back (and pause, and seek, and stop, and...)
- Term: Remote Playback

Key Questions

- How do we find out about available remote playback devices?
- How do we let the user indicate if they want to "cast" content to such a device?
  - And, if so, which one?
- How do we actually do the casting?
  - And can we avoid any proprietary SDKs?

Enter the MediaRouter

- Added in API Level 16
- Manages media routes
  - Live audio (e.g., Bluetooth speakers)
    - Added API Level 16
  - Live video (e.g., HDMI, MHL, Miracast)
    - Added API Level 17
  - Remote playback (e.g., Chromecast)
    - Added... well, this gets complicated

Your MediaRouter Choices

- android.media.MediaRouter
  - Built into Android
  - Supports live audio and live video
- android.support.v7.media.MediaRouter
  - Ships with Android Support package, in mediarouter-v7 library project
  - Supports live audio, live video, and remote playback
  - Hint: you probably want this one

Interacting with a MediaRouter

- Get an instance (e.g., getInstance())
- Build a MediaRouteSelector
  - Identifies types of routes you are interested in
- Add Callback via addCallback() to find out about changes in route availability
  - Tied to the MediaRouteSelector
  - Key method: onRouteSelected()

MediaRouteActionProvider

- A.k.a., “the cast button”
  - Technically, there's a MediaRouteButton that's the “cast” button, but we'll ignore that for now...
- Roles
  - Tapped, brings up dialog to choose a media route
    - Of categories of relevance to app (e.g., remote playback)
  - Shows highlight to indicate when app is connected to a media route or not

OK, *Which* MediaRouteActionProvider?

- Option #1: android.media
  - Works with native MediaRouter, native action bar
  - Not a great choice
- Option #2: android.support.v7.media
  - Works with Android Support's MediaRouter, AppCompat action bar
- Option #3: CWAC-MediaRouter cross-port
  - Works with Android Support's MediaRouter, native action bar

Wiring Up the Provider

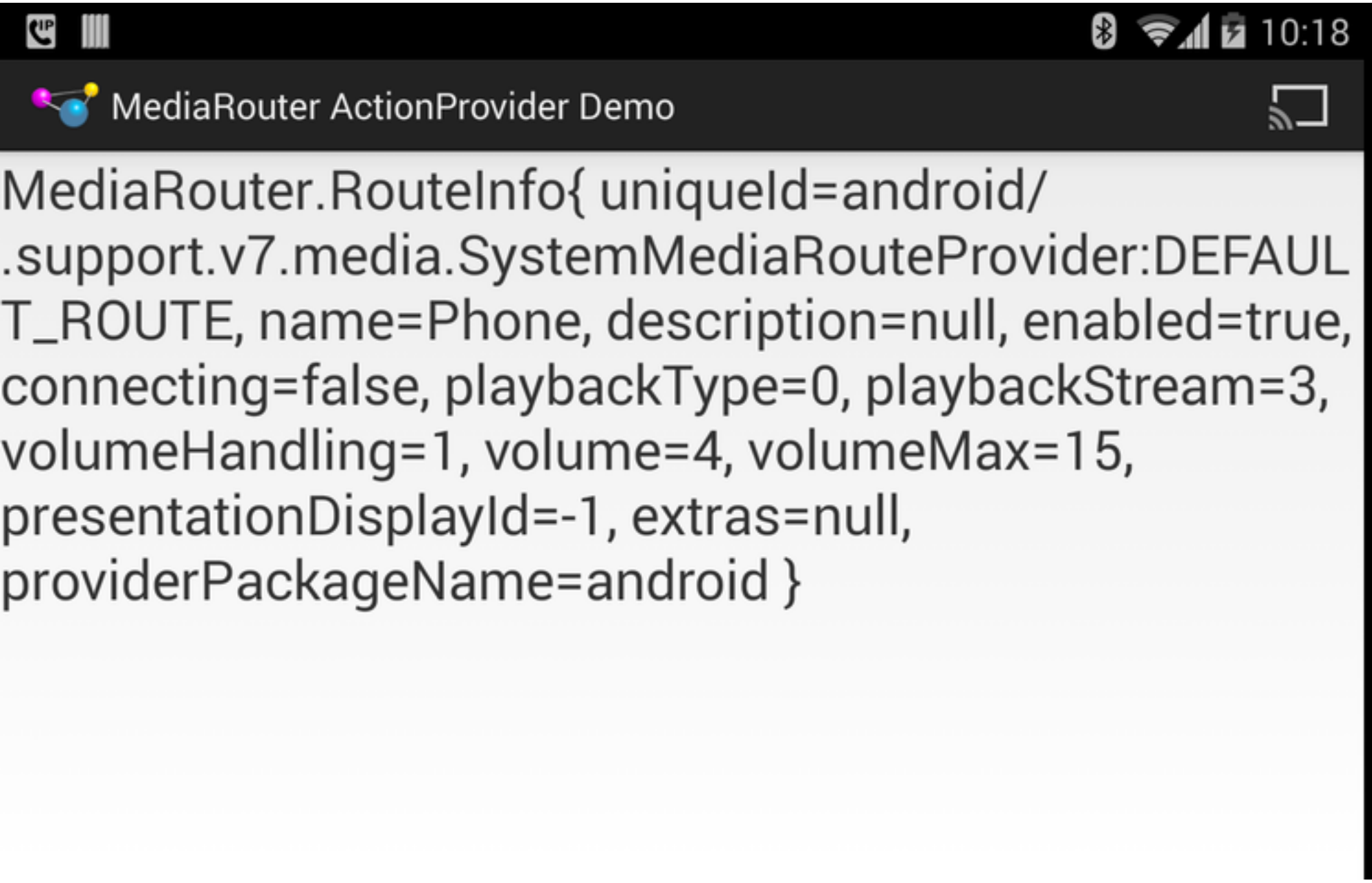
- Add to menu resource
  - Use right MediaRouteActionProvider class given your chosen action bar implementation!
- Call setRouteSelector() on the MediaRouteActionProvider
  - Controls which routes it pays attention to
  - Usually the same MediaRouteSelector that you used with addCallback()

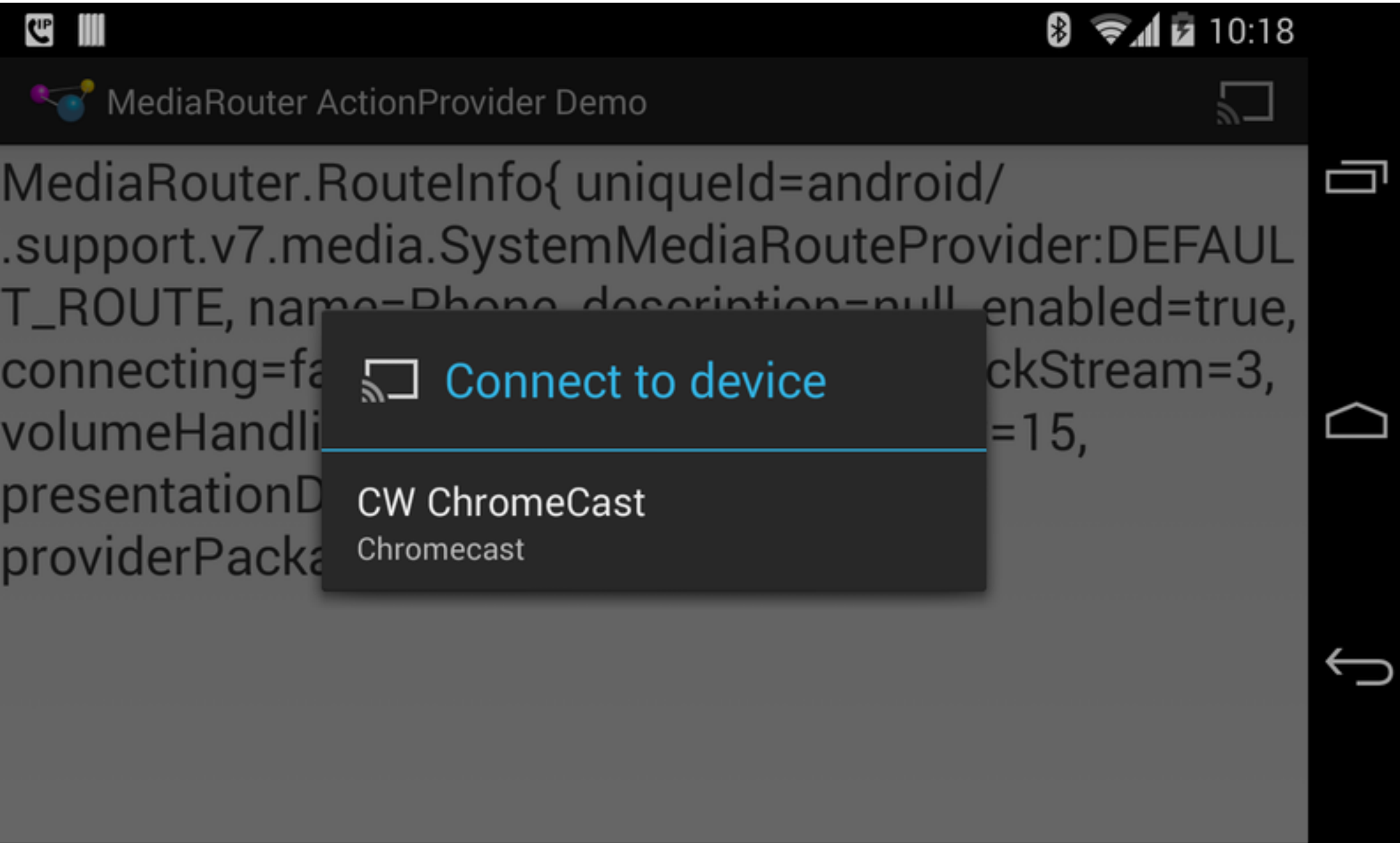
```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto">

  <item
    android:id="@+id/route_provider"
    android:title="@string/route_provider_title"
    app:actionProviderClass="android.support.v7.app.MediaRouteActionProvider"
    app:showAsAction="always"/>

</menu>
```

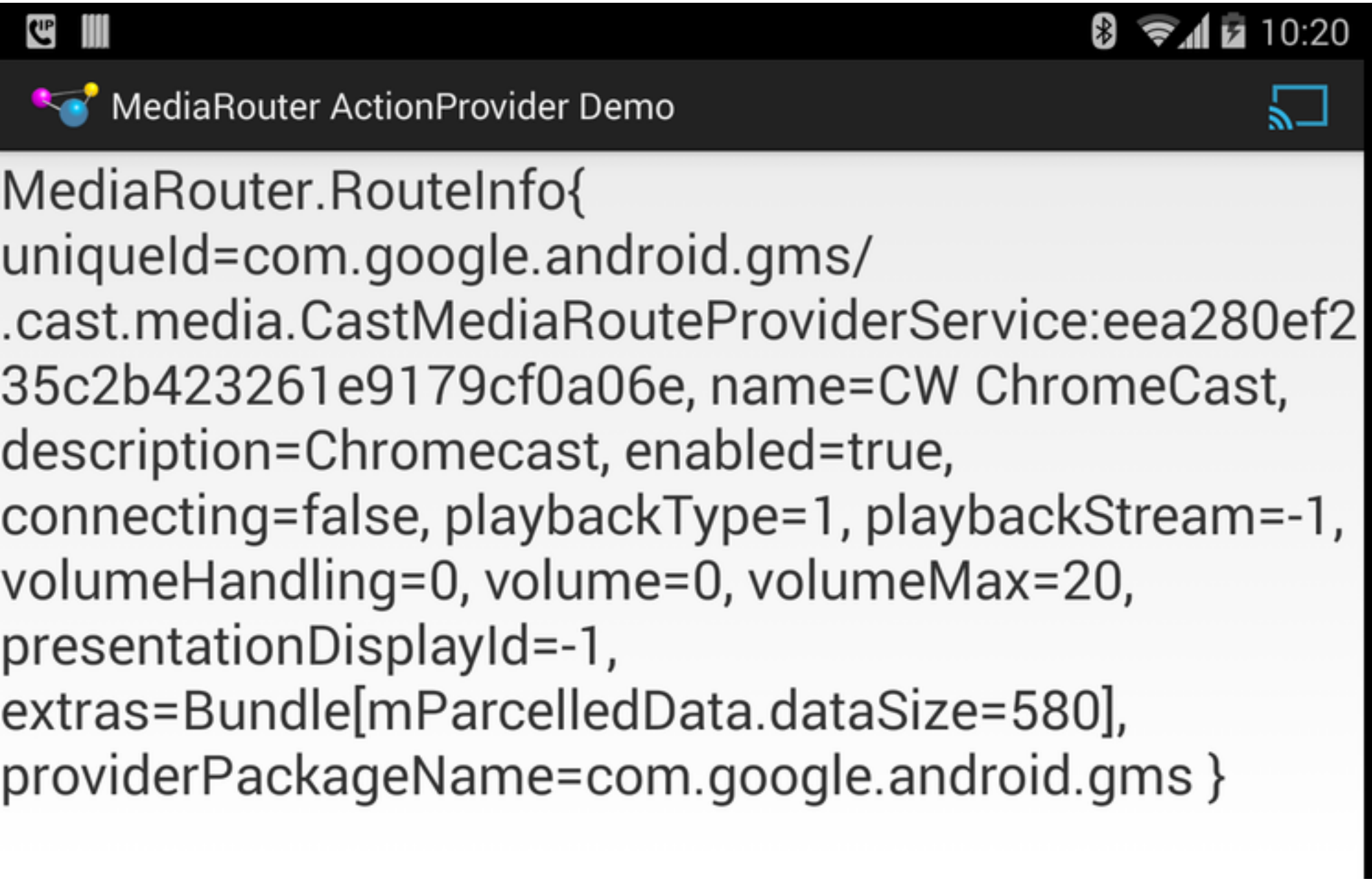
```
public class MainActivity extends ActionBarActivity {
  private MediaRouteSelector selector=null;
  private MediaRouter router=null;
  private TextView selectedRoute=null;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    selectedRoute=(TextView)findViewById(R.id.selected_route);

    router=MediaRouter.getInstance(this);
    selector=
        new MediaRouteSelector.Builder().addControlCategory(MediaControlIntent.CATEGORY_LIVE_AUDIO)
                                        .addControlCategory(MediaControlIntent.CATEGORY_LIVE_VIDEO)
                                        .addControlCategory(MediaControlIntent.CATEGORY_REMOTE_PLAY
                                        .build();

  }
```

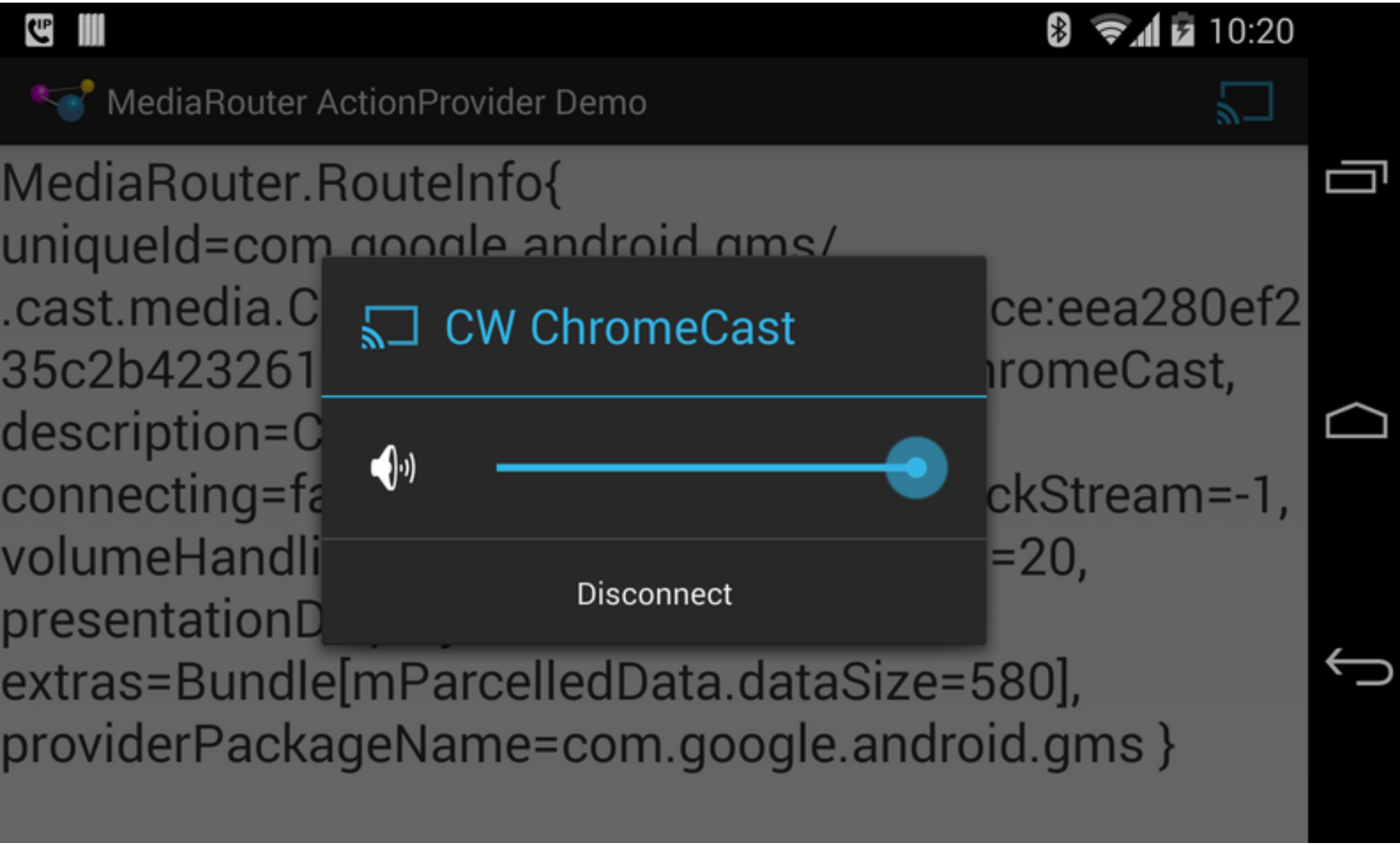
```java
@Override
public void onResume() {
  super.onResume();

  router.addCallback(selector, cb,
                     MediaRouter.CALLBACK_FLAG_REQUEST_DISCOVERY);
}

@Override
public void onPause() {
  router.removeCallback(cb);

  super.onPause();
}
```

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.main, menu);

  MenuItem item=menu.findItem(R.id.route_provider);
  MediaRouteActionProvider provider=
      (MediaRouteActionProvider)MenuItemCompat.getActionProvider(item);

  provider.setRouteSelector(selector);

  return(true);
}

private MediaRouter.Callback cb=new MediaRouter.Callback() {
  @Override
  public void onRouteSelected(MediaRouter router,
                              MediaRouter.RouteInfo route) {
    selectedRoute.setText(route.toString());
  }
};
```


10:18
MediaRouter ActionProvider Demo
MediaRouter.RouteInfo{ uniqueId=android/
.support.v7.media.SystemMediaRouteProvider:DEFAUL
T_ROUTE, name=Phone, description=null, enabled=true,
connecting=false, playbackType=0, playbackStream=3,
volumeHandling=1, volume=4, volumeMax=15,
presentationDisplayId=-1, extras=null,
providerPackageName=android }


10:18
MediaRouter ActionProvider Demo
MediaRouter.RouteInfo{ uniqueId=android/
.support.v7.media.SystemMediaRouteProvider:DEFAUL
T_ROUTE, nam
enabled=true,
connecting=fa
ckStream=3,
volumeHandli
=15,
presentationD
providerPacka
Connect to device
CW ChromeCast
Chromecast

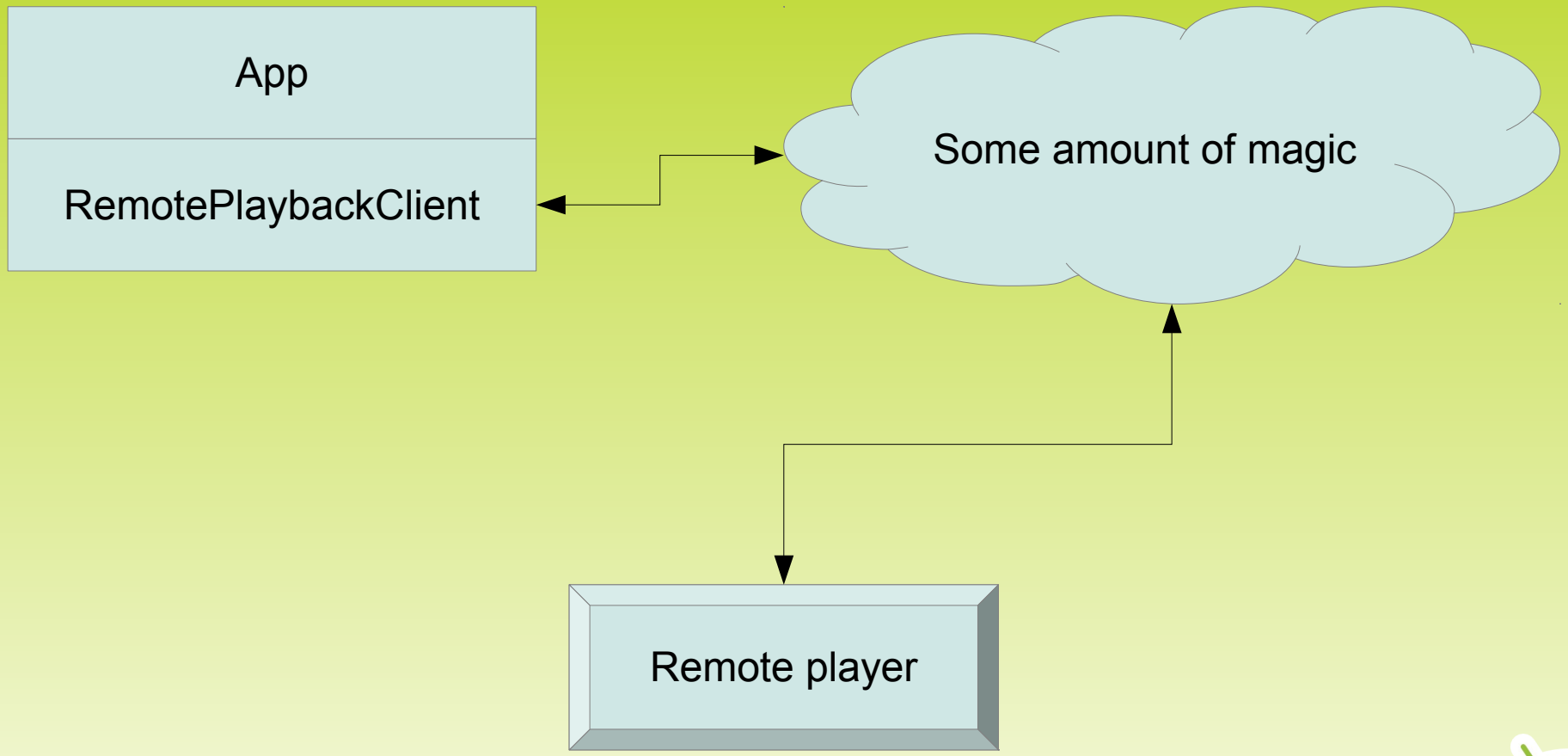

10:20
MediaRouter ActionProvider Demo
MediaRouter.RouteInfo{
uniqueId=com.google.android.gms/
.cast.media.CastMediaRouteProviderService:eea280ef2
35c2b423261e9179cf0a06e, name=CW ChromeCast,
description=Chromecast, enabled=true,
connecting=false, playbackType=1, playbackStream=-1,
volumeHandling=0, volume=0, volumeMax=20,
presentationDisplayId=-1,
extras=Bundle[mParcelledData.dataSize=580],
providerPackageName=com.google.android.gms }


10:20
MediaRouter ActionProvider Demo
MediaRouter.RouteInfo{
uniqueId=com.google.android.gms/
.cast.media.C
ce:eea280ef2
35c2b423261
romeCast,
description=C
connecting=fa
ckStream=-1,
volumeHandli
=20,
presentationD
extras=Bundle[mParcelledData.dataSize=580],
providerPackageName=com.google.android.gms }
CW ChromeCast
Disconnect

RemotePlaybackClient

- Client API for working with remote players
- Basic Mechanics
  - Get an instance
  - Tie to a media route
  - Call methods like play(), pause(), resume(), ...
    - E.g., play() takes URL to play back on remote player

The RemotePlaybackClient Flow



Using RemotePlaybackClient

- Create instance
- Connect to route
- Call control methods
  - play(), pause(), etc.
  - Callbacks to find out success or failure... in theory
- Disconnect and release

```
<item
  android:id="@+id/route_provider"
  android:title="@string/route_provider_title"
  app:actionProviderClass="android.support.v7.app.MediaRouteActionProvider"
  app:showAsAction="always"/>
<item
  android:id="@+id/play"
  android:icon="@android:drawable/ic_media_play"
  android:title="@string/play"
  android:visible="false"
  app:showAsAction="always"/>
<item
  android:id="@+id/pause"
  android:icon="@android:drawable/ic_media_pause"
  android:title="@string/pause"
  android:visible="false"
  app:showAsAction="always"/>
<item
  android:id="@+id/stop"
  android:icon="@drawable/ic_media_stop"
  android:title="@string/stop"
  android:visible="false"
  app:showAsAction="always"/>
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  setRetainInstance(true);
  setHasOptionsMenu(true);
  selector=
      new MediaRouteSelector.Builder().addControlCategory(MediaControlIntent.CATEGORY_REMOTE_PLAYE
                                      .build();
}
```

```java
@Override
public void onAttach(Activity host) {
  super.onAttach(host);

  router=MediaRouter.getInstance(host);
}

@Override
public View onCreateView(LayoutInflater inflater,
                         ViewGroup container,
                         Bundle savedInstanceState) {
  scroll=
      (ScrollView)inflater.inflate(R.layout.activity_main, container,
                                   false);

  transcript=(TextView)scroll.findViewById(R.id.transcript);

  logToTranscript("Started");

  return(scroll);
}
```

```java
@Override
public void onResume() {
  super.onResume();

  router.addCallback(selector, cb,
                     MediaRouter.CALLBACK_FLAG_REQUEST_DISCOVERY);
}

@Override
public void onPause() {
  router.removeCallback(cb);

  super.onPause();
}

@Override
public void onDestroy() {
  disconnect();

  super.onDestroy();
}
```

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
  inflater.inflate(R.menu.main, menu);

  if (client != null) {
    if (isPlaying) {
      menu.findItem(R.id.stop).setVisible(true);

      if (isPaused) {
        menu.findItem(R.id.play).setVisible(true);
      }
      else {
        menu.findItem(R.id.pause).setVisible(true);
      }
    }
    else {
      menu.findItem(R.id.play).setVisible(true);
    }
  }

  MenuItem item=menu.findItem(R.id.route_provider);
  MediaRouteActionProvider provider=
      (MediaRouteActionProvider)MenuItemCompat.getActionProvider(item);

  provider.setRouteSelector(selector);
}
```

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
  switch (item.getItemId()) {
    case R.id.play:
      if (isPlaying && isPaused) {
        resume();
      }
      else {
        play();
      }

      return(true);

    case R.id.stop:
      stop();
      return(true);

    case R.id.pause:
      pause();
      return(true);
  }

  return(super.onOptionsItemSelected(item));
}
```

```
private MediaRouter.Callback cb=new MediaRouter.Callback() {
  @Override
  public void onRouteSelected(MediaRouter router,
                              MediaRouter.RouteInfo route) {
    logToTranscript(getActivity().getString(R.string.route_selected));
    connect(route);
  }

  @Override
  public void onRouteUnselected(MediaRouter router,
                                MediaRouter.RouteInfo route) {
    logToTranscript(getActivity().getString(R.string.route_unselected));
    disconnect();
  }
};
```

```
private void connect(MediaRouter.RouteInfo route) {
  client=
      new RemotePlaybackClient(getActivity().getApplication(), route);

  if (client.isRemotePlaybackSupported()) {
    logToTranscript(getActivity().getString(R.string.connected));

    if (client.isSessionManagementSupported()) {
      client.startSession(null, new SessionActionCallback() {
        @Override
        public void onResult(Bundle data, String sessionId,
                             MediaSessionStatus sessionStatus) {
          logToTranscript(getActivity().getString(R.string.session_started));
          getActivity().supportInvalidateOptionsMenu();
        }

        @Override
        public void onError(String error, int code, Bundle data) {
          logToTranscript(getActivity().getString(R.string.session_failed));
        }
      });
    }
    else {
      getActivity().supportInvalidateOptionsMenu();
    }
  }
  else {
    logToTranscript(getActivity().getString(R.string.remote_playback_not_supported));
```

```
private void play() {
  logToTranscript(getActivity().getString(R.string.play_requested));

  ItemActionCallback playCB=new ItemActionCallback() {
    @Override
    public void onResult(Bundle data, String sessionId,
                         MediaSessionStatus sessionStatus,
                         String itemId, MediaItemStatus itemStatus) {
      logToTranscript(getActivity().getString(R.string.playing));
      isPlaying=true;
      getActivity().supportInvalidateOptionsMenu();
    }

    @Override
    public void onError(String error, int code, Bundle data) {
      logToTranscript(getActivity().getString(R.string.play_error)
          + error);
    }
  };

  client.play(Uri.parse("http://misc.commonsware.com/ed_hd_512kb.mp4"),
              "video/mp4", null, 0, null, playCB);
}
```

```
private void pause() {
  logToTranscript(getActivity().getString(R.string.pause_requested));

  PauseCallback pauseCB=new PauseCallback();

  client.pause(null, pauseCB);
  transcript.postDelayed(pauseCB, 1000);
}

private void resume() {
  logToTranscript(getActivity().getString(R.string.resume_requested));

  ResumeCallback resumeCB=new ResumeCallback();

  client.resume(null, resumeCB);
  transcript.postDelayed(resumeCB, 1000);
}

private void stop() {
  logToTranscript(getActivity().getString(R.string.stop_requested));

  StopCallback stopCB=new StopCallback();

  client.stop(null, stopCB);
  transcript.postDelayed(stopCB, 1000);
}
```

```
abstract class RunnableSessionActionCallback extends
    SessionActionCallback implements Runnable {
  abstract protected void doWork();

  private boolean hasRun=false;

  @Override
  public void onResult(Bundle data, String sessionId,
                       MediaSessionStatus sessionStatus) {
    transcript.removeCallbacks(this);
    run();
  }

  @Override
  public void run() {
    if (!hasRun) {
      hasRun=true;
      doWork();
    }
  }
}
```

```
private class PauseCallback extends RunnableSessionActionCallback {
  @Override
  protected void doWork() {
    isPaused=true;
    getActivity().supportInvalidateOptionsMenu();
    logToTranscript(getActivity().getString(R.string.paused));
  }
}

private class ResumeCallback extends RunnableSessionActionCallback {
  @Override
  protected void doWork() {
    isPaused=false;
    getActivity().supportInvalidateOptionsMenu();
    logToTranscript(getActivity().getString(R.string.resumed));
  }
}

private class StopCallback extends RunnableSessionActionCallback {
  @Override
  protected void doWork() {
    isPlaying=false;
    isPaused=false;
    getActivity().supportInvalidateOptionsMenu();
    logToTranscript(getActivity().getString(R.string.stopped));
  }
}
```

```
private void disconnect() {
  isPlaying=false;
  isPaused=false;

  if (client != null) {
    logToTranscript(getActivity().getString(R.string.session_ending));
    EndSessionCallback endCB=new EndSessionCallback();

    if (client.isSessionManagementSupported()) {
      client.endSession(null, endCB);
    }

    transcript.postDelayed(endCB, 1000);
  }
}
```

```
private class EndSessionCallback extends
    RunnableSessionActionCallback {
  @Override
  protected void doWork() {
    client.release();
    client=null;

    if (getActivity() != null) {
      getActivity().supportInvalidateOptionsMenu();
      logToTranscript(getActivity().getString(R.string.session_ended));
    }
  }
}
```

Other RemotePlaybackClient Capabilities

- Volume control
- Session management
  - Playback queues, managed by different users
- RemoteControlClient
  - Controls on the device's lockscreen