



# Developing for Android TV and the Nexus Player

인텔코리아  
이진용 차장





# Developing for Android TV and the Nexus Player

인텔코리아  
이진용 차장



# Developing for Android TV and the Nexus Player

- What is Android TV
- How to make your app or game compatible with Android TV
- How to integrate it further
- Publishing your application

# Android TV and the Nexus Player

# Android TV and the Nexus Player

# Android TV and the Nexus Player

- It's Android
- it's also Chromecast
- Apps (streaming and others)
- Games (casual and more)
- AOSP compliant
- Leanback Launcher, Google Apps and Play Store



# Nexus Player

- First Android TV device / The only Nexus
- Quad-Core 64bit Intel Silvermont CPU @1.83Ghz
- PowerVR™ Series 6 G6430 GPU - OpenGL ES 3.1
- 64bit, WiFi 802.11ac\*
- 1GB ram, 8GB flash, USB-OTG
- Gamepad sold separately\*\*
- Australia, Austria, Canada, Denmark, Finland, France, Germany, Italy, Japan, **Korea**, Norway, Spain, Sweden, Switzerland, United Kingdom, United States



\* Ethernet can be added using standard USB adapters

\*\* Android TV supports almost any USB/Bluetooth HID Gamepads

# Other devices

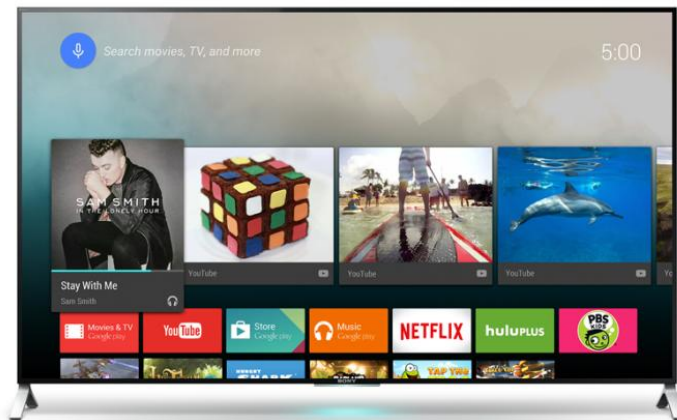
freebox mini 4K  
Notre petite dernière avec Android TV™



NVIDIA\* Shield

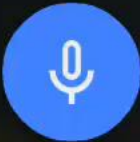


Razer\* Forge TV



Smart TVs from Sony\*, Philips\*, Sharp\*...





Search movies, TV, and more

22:58



Dailymotion



Avengers Age Of Ultron - 10 Things You Probably Missed In...

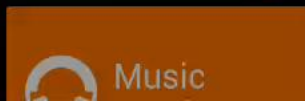
Staff picked in Movies



Dailymotion



YouTube



# Adapting your app for Android TV

# Adapting your app for Android TV

# Adapting your app for Android TV

1. Add/reuse a TV-compatible activity that will receive the Leanback intent
  2. Integrate TV-specific assets
  3. Support non-touchscreen input
  4. Adapt the UX of your app
- No need to create a separate *Application*.
  - Still possible to provide an alternative APK for TV.

# 1. The Leanback Intent

```
<activity android:name=".TvMainActivity" >  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category  
      android:name="android.intent.category.LEANBACK_LAUNCHER" />  
  </intent-filter>  
</activity>
```

## 2. The Banner



```
<activity or <application
```

```
...
```

```
    android:banner="@drawable/ic_banner"
```

```
...
```

```
>
```

- Include the localized name of your application
- No transparency
- Size:160x90dp -> 320x180px in drawable-xhdpi

## 3. Supporting non-touchscreen input

```
<uses-feature  
    android:name="android.hardware.touchscreen"  
    android:required="false" />
```

### Adjust D-PAD navigation:

```
android:focusable="true", <requestFocus /> / .requestFocus()  
android:nextFocusDown="@+id/whatever1"  
android:nextFocusUp="@id/whatever2"
```

### For custom Views:

```
KeyEvent.KEYCODE_DPAD_ (UP | DOWN | LEFT | RIGHT | CENTER)
```

## 4. Adapting the UX

Start from `android:Theme.NoTitleBar`

or `Theme.Leanback` from the Leanback support library:

```
compile "com.android.support:leanback-v17:21.0.+"
```

Add overscan margins to your views: (Leanback Views and Fragments already have these)

```
android:layout_marginTop="27dp"  
android:layout_marginLeft="48dp"  
android:layout_marginRight="48dp"  
android:layout_marginBottom="27dp"
```

Leanback support library requires API 17 but you can still support lower API levels:

- Use `Theme.Leanback` from `-v21+` resources, use Leanback classes only from TV-part.
- Set `<uses-sdk tools:overrideLibrary="android.support.v17.leanback" />`

# Integrating further with Android TV

*Leanback Support library, Recommendations, Search, TV Input Framework*

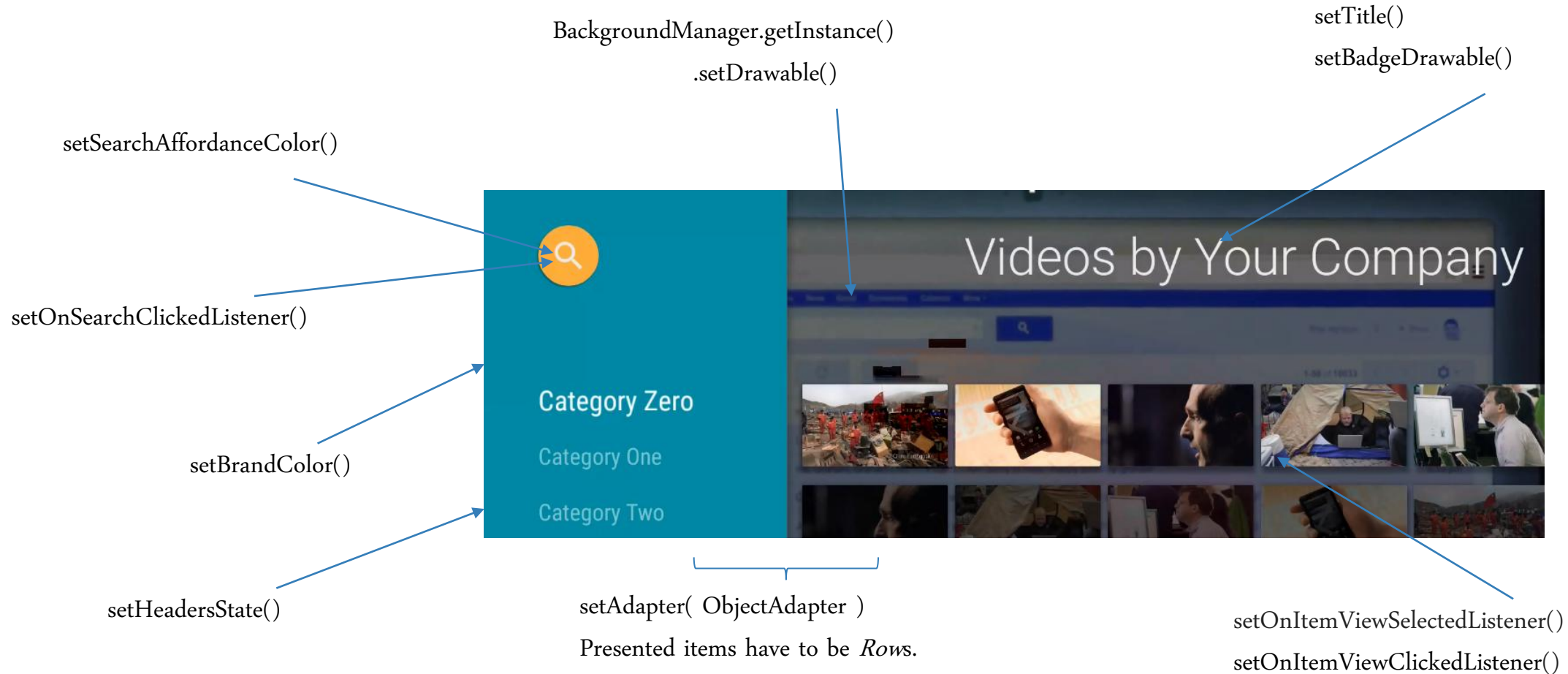
# Integrating further with Android TV

*Leanback Support library, Recommendations, Search, TV Input Framework*

# The Leanback support library

The image displays a user interface for a video library. On the left, a teal sidebar contains a search icon and three category options: 'Category Zero', 'Category One', and 'Category Two'. The main area shows a grid of video thumbnails under the heading 'Videos by Your Company'. A larger, detailed view of a video is shown in the foreground, featuring a profile of a man and a teal overlay with the text 'Introducing Gmail Blue' and 'Studio Two'. Below the title, there is a paragraph of placeholder text and three buttons: 'WATCH TRAILER FREE', 'RENT BY DAY FROM \$1.99', and 'BUY AN AT \$'. The background of the detailed view shows a blurred screenshot of a Gmail inbox.

# The Leanback support library – BrowseFragment



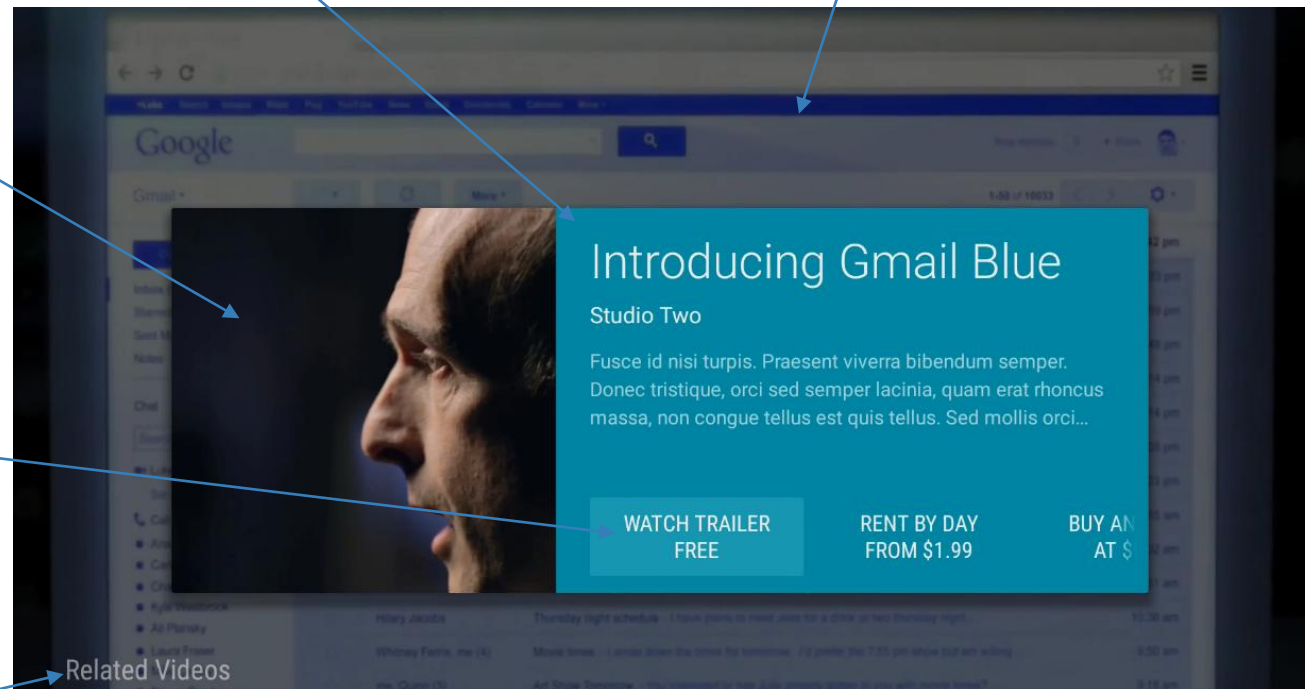
# The Leanback support library – DetailsFragment

`detailsOverviewRowPresenter.setBackgroundColor()` `BackgroundManager.getInstance().setDrawable()`

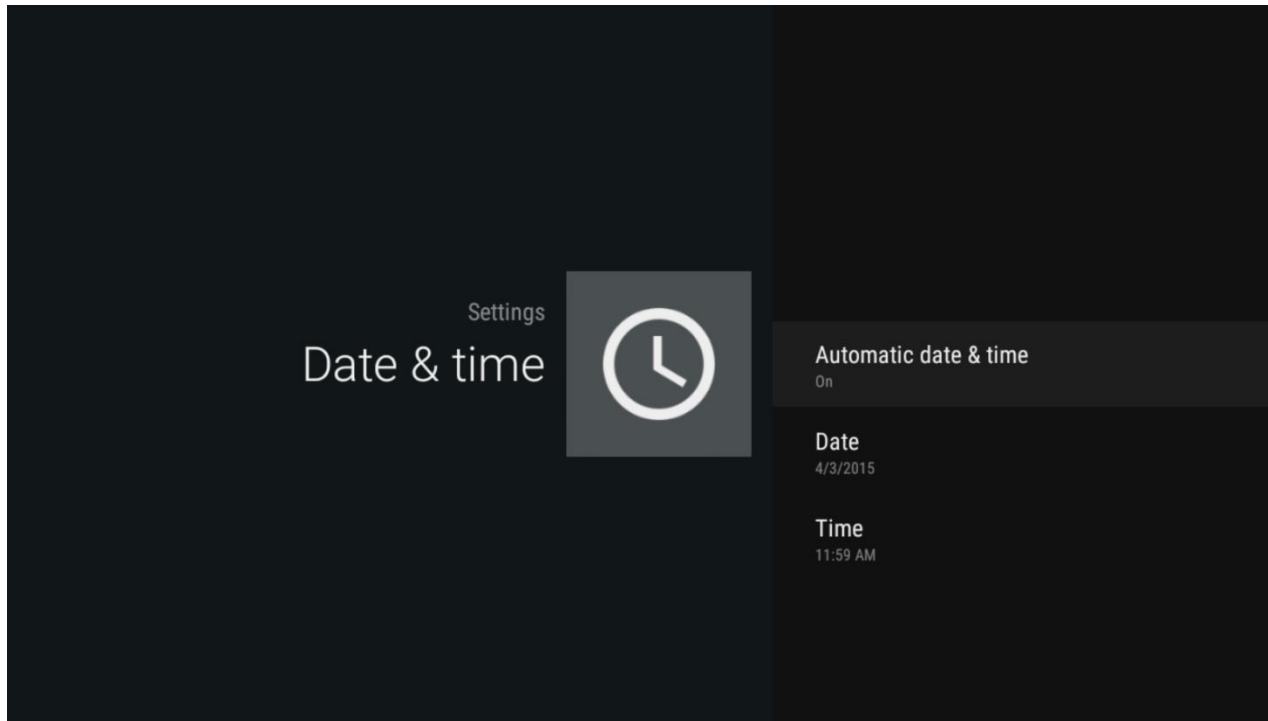
`detailsOverviewRow.setImageDrawable()`

`detailsOverviewRow.addAction()`

`mAdapter.add(new ListRow(header, listRowAdapter))`



# The Leanback support library – GuidedStepFragment

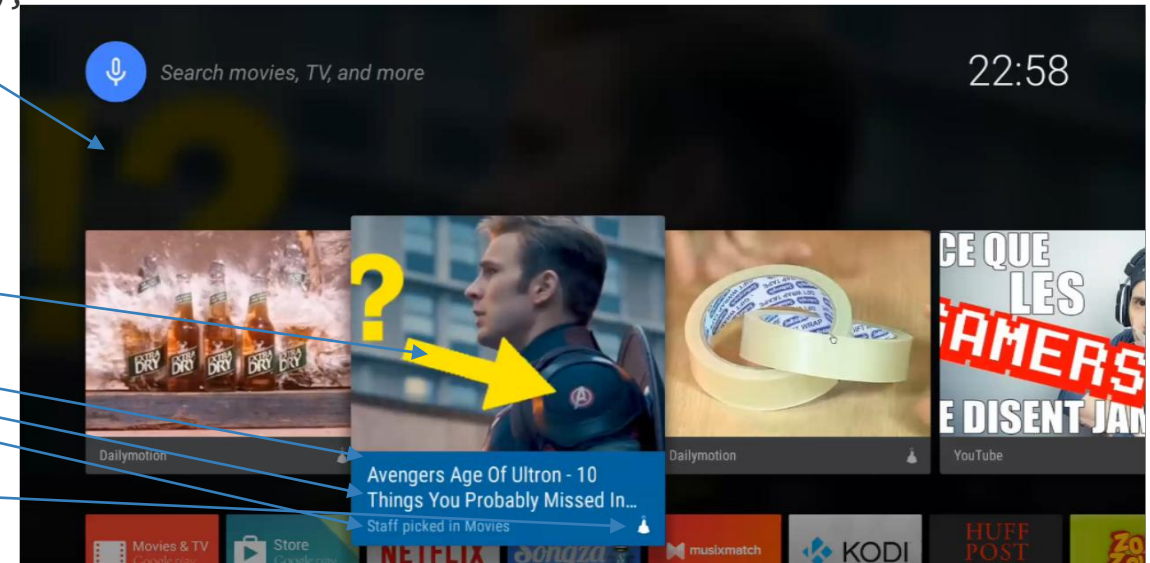


- [GuidanceStylist.Guidance](#)([String](#) title, [String](#) description, [String](#) breadcrumb, [Drawable](#) icon)
- [onCreateGuidance\(Bundle\)](#)
- [onCreateActions\(List, Bundle\)](#)
- [onGuidedActionClicked\(GuidedAction\)](#)

# Recommendation System

```
Bundle extras = new Bundle();  
extras.putString(Notification.EXTRA_BACKGROUND_IMAGE_URI, backgroundUri);
```

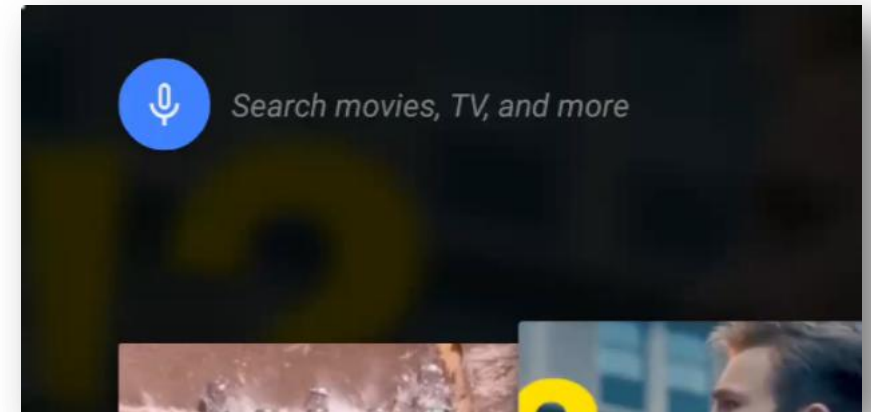
```
Notification notification = new NotificationCompat.BigPictureStyle(  
    new NotificationCompat.Builder(mContext)  
        .setLargeIcon(bitmap)  
        .setColor(color)  
        .setContentTitle(title)  
        .setContentText(description)  
        .setLocalOnly(true)  
        .setOngoing(true)  
        .setCategory(Notification.CATEGORY_RECOMMENDATION)  
        .setSmallIcon(mSmallIcon)  
        .setContentIntent(mIntent)  
        .setExtras(extras))  
    .build();
```



It's advised to update recommendations from a service you can trigger using an `AlarmManager` that will run it periodically, starting with shorty after boot.

# Search System

- Global Search
- Implement a Content Provider
- Declare android.app.searchable meta-data
- <https://developer.android.com/training/tv/discovery/searchable.html>
- Search Activity:
- Use SpeechRecognizer
- Can use Leanback SearchFragment.



# Supporting multiple controllers

- `int KeyEvent.getDeviceId()`
- `String KeyEvent.getDevice().getDescriptor()`  
API Level 16+
- Nearby Connection API  
Google Play Services 7.0+



# TV Input Framework and Live Channels

# TV Input Framework and Live Channels

# TV Input Framework and Live Channels

play.google.com/store/apps/details?id=com.google.android.tv

Search

Categories | Home | Top Charts | New Releases

## Live Channels for Android TV

Google Inc. - December 17, 2014  
Entertainment

**Install** | Add to Wishlist

Loading device compatibility...

★★★★☆ (11) | +207

Top Developer

**101 World Pop Special**  
JKTV 10:50 - 11:30PM  
The World Pop special introduces hot trending music videos from around the world. A new music video from each is featured this week!

Recent channels  
TV options

Closed caption | Display mode | Multi-audio | PIP | Channel sources | Parental control

### Description

Live Channels app is for watching Live TV. Watch your favorite news, sports, movies and TV shows from various channel sources such as built-in tuner, IP-based tuners, and more and show them instantly on your Android TV.

# TV Input Framework – Live Channels

```
public class MyTvInputService extends TvInputService {  
    ...  
    @Override  
    public Session onCreateSession(String inputId) {  
        return new MyTvInputServiceSession(MyTvInputService.this); //ServiceSession implementation is on next slide  
    }  
}
```

AndroidManifest.xml

```
<service  
    android:name=".MyTvInputService"  
    android:permission="android.permission.BIND_TV_INPUT" >  
    <intent-filter>  
        <action android:name="android.media.tv.TvInputService" />  
    </intent-filter>  
    <meta-data android:name="android.media.tv.input" android:resource="@xml/my_tv_input" />  
</service>
```

my\_tv\_input.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<tv-input xmlns:android="http://schemas.android.com/apk/res/android"  
    android:settingsActivity="com.xh.tvinputservicetest.SettingsActivity"  
    android:setupActivity="com.xh.tvinputservicetest.SetupActivity" />
```

# TV Input Framework – Live Channels

```
public class MyTvInputServiceSession extends TvInputService.Session {
    Surface mSurface;

    @Override
    public boolean onSetSurface(Surface surface) {
        mSurface = surface;
        return true;
    }

    @Override
    public boolean onTune(Uri channelUri) {
        notifyVideoUnavailable(TvInputManager.VIDEO_UNAVAILABLE_REASON_BUFFERING);
        //tune to channel and change draws to surface in a render thread, then fire notifyVideoAvailable()
        return true;
    }

    @Override
    public void onSetCaptionEnabled(boolean enabled) { }
    ...
}
```

# Live Channel – Setup and Content upgrade

```
String inputId = TvContract.buildInputId(new ComponentName(this, TvInputServiceTest.class));

ContentValues channelsValues = new ContentValues();
channelsValues.put(TvContract.Channels.COLUMN_DISPLAY_NAME, "Channel display name");
channelsValues.put(TvContract.Channels.COLUMN_DESCRIPTION, "Channel description");
channelsValues.put(TvContract.Channels.COLUMN_DISPLAY_NUMBER, "1");
channelsValues.put(TvContract.Channels.COLUMN_INPUT_ID, inputId);
channelsValues.put(TvContract.Channels.COLUMN_INTERNAL_PROVIDER_DATA, "internal channel service data");
channelsValues.put(TvContract.Channels.COLUMN_NETWORK_AFFILIATION, "Network Affiliation");
channelsValues.put(TvContract.Channels.COLUMN_ORIGINAL_NETWORK_ID, "Test Network ID");
channelsValues.put(TvContract.Channels.COLUMN_PACKAGE_NAME, "com.xh.tvinputserVICETest");
channelsValues.put(TvContract.Channels.COLUMN_SEARCHABLE, true);
channelsValues.put(TvContract.Channels.COLUMN_SERVICE_ID, 123456);
channelsValues.put(TvContract.Channels.COLUMN_SERVICE_TYPE, TvContract.Channels.SERVICE_TYPE_AUDIO_VIDEO);
channelsValues.put(TvContract.Channels.COLUMN_VIDEO_FORMAT, TvContract.Channels.VIDEO_FORMAT_1080P);
channelsValues.put(TvContract.Channels.COLUMN_VERSION_NUMBER, 0);
channelsValues.put(TvContract.Channels.COLUMN_TRANSPORT_STREAM_ID, 0);
channelsValues.put(TvContract.Channels.COLUMN_TYPE, TvContract.Channels.TYPE_OTHER);

Uri channelUri = getContentResolver().insert(TvContract.Channels.CONTENT_URI, channelsValues);
long channelId = ContentUris.parseId(channelUri);

Uri channelLogoUri = TvContract.buildChannelLogoUri(channelId);
AssetFileDescriptor fd = getContentResolver().openAssetFileDescriptor(channelLogoUri, "rw");
```

# Live Channel – Programs update

```
ContentValues programsValues = new ContentValues();
programsValues.put(TvContract.Programs.COLUMN_TITLE, "program title");
programsValues.put(TvContract.Programs.COLUMN_AUDIO_LANGUAGE, "fr");
programsValues.put(TvContract.Programs.COLUMN_BROADCAST_GENRE, "movie/drama"); // ATSC A/65 or Content Descriptor of ETSI EN
300 468,
programsValues.put(TvContract.Programs.COLUMN_CANONICAL_GENRE,
TvContract.Programs.Genres.encode(TvContract.Programs.Genres.DRAMA, TvContract.Programs.Genres.COMEDY));
programsValues.put(TvContract.Programs.COLUMN_CHANNEL_ID, channelId);
programsValues.put(TvContract.Programs.COLUMN_PACKAGE_NAME, "com.xh.tvinputservicetest");
programsValues.put(TvContract.Programs.COLUMN_INTERNAL_PROVIDER_DATA, "internal program service data");
programsValues.put(TvContract.Programs.COLUMN_CONTENT_RATING, TvContentRating.createRating("com.android.tv", "US_TV",
"US_TV_PG", "US_TV_L", "US_TV_D").flattenToString());
programsValues.put(TvContract.Programs.COLUMN_START_TIME.UTC_MILLIS, System.currentTimeMillis() - 3600 * 1000);
programsValues.put(TvContract.Programs.COLUMN_END_TIME.UTC_MILLIS, System.currentTimeMillis() + 3600 * 1000);
programsValues.put(TvContract.Programs.COLUMN_EPISODE_NUMBER, 1);
programsValues.put(TvContract.Programs.COLUMN_SEASON_NUMBER, 3);
programsValues.put(TvContract.Programs.COLUMN_EPISODE_TITLE, "episode title");
programsValues.put(TvContract.Programs.COLUMN_SHORT_DESCRIPTION, "short description");
programsValues.put(TvContract.Programs.COLUMN_LONG_DESCRIPTION, "long description");
programsValues.put(TvContract.Programs.COLUMN_POSTER_ART_URI, );
programsValues.put(TvContract.Programs.COLUMN_THUMBNAIL_URI, );
programsValues.put(TvContract.Programs.COLUMN_VERSION_NUMBER, 0);
programsValues.put(TvContract.Programs.COLUMN_VIDEO_WIDTH, 1920);
programsValues.put(TvContract.Programs.COLUMN_VIDEO_HEIGHT, 1080);

getContentResolver().insert(TvContract.Programs.CONTENT_URI, programsValues);
//TODO: update instead of delete and insert.
```

# Publishing apps to the Android TV Play Store

# Publishing apps to the Android TV Play Store

# Publishing apps to the Android TV Play Store

- Adjust features requirements and meta-data
- Upload your updated/additional APK
- Upload banner and screenshots
- Opt-in for TV distribution

# Adjusting Feature Requirements and meta-data

## Is your app a Game ?

```
<application ... android:isGame="true" ...>
```

## Supporting Gamepads ?

```
<uses-feature android:name="android.hardware.gamepad" android:required="false" />
```

## Supporting only Android TV ?

```
<uses-feature android:name="android.software.leanback" android:required="true" />
```

## Features you shouldn't require:

- *android.hardware.location.gps, android.hardware.camera.\*, android.hardware.telephony, android.hardware.sensor.\*, android.hardware.nfc, android.hardware.touchscreen, android.hardware.microphone*

# A classic trap: implicitly required features

Having portrait activities declared inside your app ?

Need to specify that your app can be used on hardware that doesn't support "portrait" mode:

```
<uses-feature android:name="android.hardware.screen.portrait"  
            android:required="false" />
```

Using permissions with implicit hardware requirements ?

The below features aren't available on every Android TV devices:

- *android.hardware.location.gps* implied by *android.permission.ACCESS\_FINE\_LOCATION*
- *android.hardware.camera.autofocus* and *android.hardware.camera* implied by *android.permission.CAMERA*
- *android.hardware.telephony* implied by many telephony-specific permissions
- *android.hardware.microphone* implied by *android.permission.RECORD\_AUDIO*

# Submitting your application

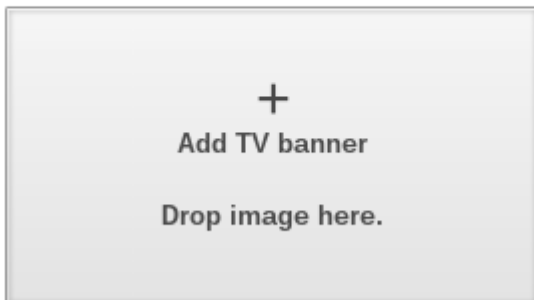
1. Check you're compliant with all the guidelines
2. Upload your APK
3. Upload your banner and a screenshot
4. Opt-in for distribution to the Play Store on Android TV

## TV Banner

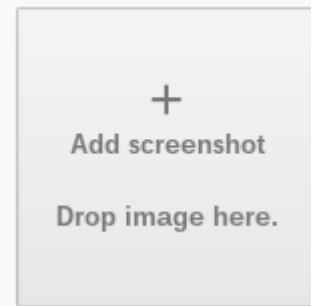
Default – English (United States) – en-US

320 w x 180 h

JPG or 24-bit PNG (no alpha)




## TV



## ANDROID TV

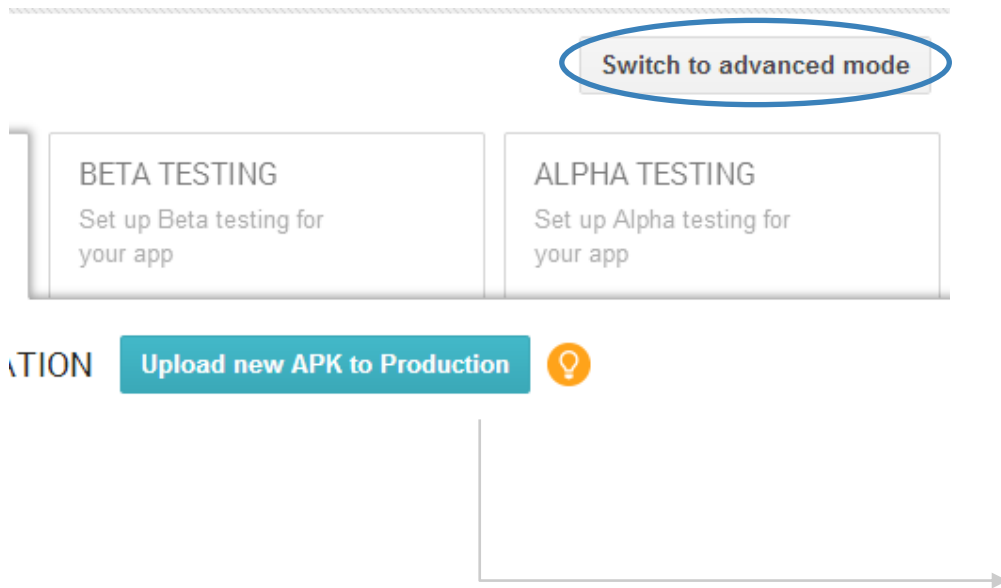
- Distribute your app on Android TV.

Your app contains a Leanback launch intent, indicating it's designed for Android TV. Before publishing apps to the Play Store on Android TV devices, we review apps for usability with a D-pad and other quality guidelines. [Learn more](#)

Status  This application has been approved.

# Publishing your TV application as an alternative APK

Switch to Advanced mode before uploading the second APK.



The screenshot shows the Google Play Console interface for configuring an APK. The left sidebar contains navigation options: APK (checked), Store Listing, Pricing & Distribution, In-app Products, Services & APIs, and Optimization Tips (with a '1' notification). The main content area shows the 'APK' configuration page with 'Save' and 'Revert changes' buttons. The 'PRODUCTION' tab is active, showing 'Versions 2, 3'. There are tabs for 'BETA TESTING' and 'ALPHA TESTING'. Below this is the 'PRODUCTION CONFIGURATION' section, which includes a 'CURRENT CONFIGURATION' note (uploaded on Jan 21, 2015, 3:20:06 AM) and two summary boxes: 'Supported devices 6824' and 'Excluded devices 0'. A teal button 'Upload new APK to Production' is visible. Below this is a yellow warning box: 'Some devices are eligible to run multiple APKs. In such a scenario, the device will receive the APK with the higher version code.' A table lists the current configurations:

VERSION	API LEVELS	FEATURES	UPLOADED ON	STATUS	ACTIONS
3 (1.0)	21+	android.hardware.screen.LANDSCAPE, android.software.LEANBACK	Jan 21, 2015	Draft in Prod	Move... ▼
2 (1.0)	15+	android.hardware.TOUCHSCREEN	Jan 21, 2015	Draft in Prod	Move... ▼

Nexus Player and x86 support from apps?

Nexus Player and x86 support from apps?

# Android\* Devices with Intel Inside

(Some of them – there are more than hundred, not all could be listed here)

2012, 2013...

Orange\* San Diego



ZTE\* Geek



Motorola\* RAZR i



ASUS\* MeMO Pad FHD 10



ZTE\* Grand X IN



Dell\* Venue 7/8



Megafon\* Mint



Lenovo\* K900



Lenovo\* K800



ASUS Fonepad™ Note FHD - 6"



Lava\* Xolo X900



Intel® Yolo



Samsung\* Galaxy™ Tab 3 10.1"



Acer\* Liquid C1



Lenovo Yoga Tab 2 8/10/13



ASUS\* Fonepad™ 7"



2014...

Asus\* Zenfones 4/5/10



Asus\* FonePad 7/8



Toshiba Excite Go



ASUS\* Transformer Pad TF103CG/TF303CL



Lenovo Yoga Tab 2 8/10/13



Tesco Hudl 2



KD Interactive Kurio Tablet



Lenovo\* S8



Asus\* MemoPad 7/8

Acer\* Iconia Tab 8 / One 8



Nexus Player



2015...



Dell\* Venue 8 7840



Asus\* Zenfone 2

Lenovo\* P90



Nokia n1

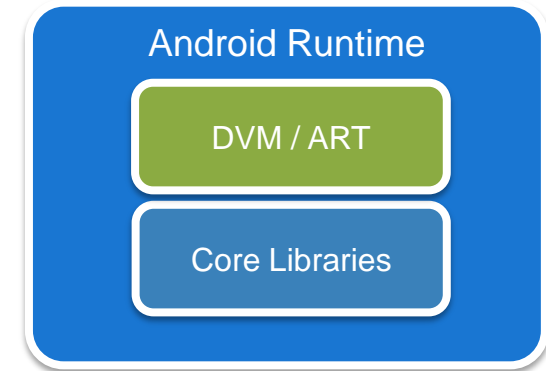


Dell\* Venue 10 7040

# These devices are all fully compatible with ARM\* ecosystem

## Android\* SDK apps

- These will directly work. We're optimizing the Runtimes for Intel® platforms.



## Android\* NDK apps

- Most will run without any recompilation on consumer platforms.
- Android NDK provides an x86 toolchain since 2011
- A simple recompile using the Android NDK yields the best performance
- If there is specific processor dependent code, porting may be necessary

# 3rd party libraries/engines x86 support

- Game engines/libraries with x86 support:
- Unity: default
- libgdx: default
- NexPlayer SDK: default
- Unreal Engine 3: available
- Marmalade: available
- Cocos2Dx: available
- Havok Anarchy SDK: available
- FMOD: available
- AppGameKit: available
- AppPortable: available
- Adobe Air: available
- ... custom NDK code: set NDK variable APP\_ABI to 'all', inside *Application.mk*

# Configuring NDK Target ABIs

Include all ABIs by setting APP\_ABI to all in jni/**Application.mk**:

```
APP_ABI=all
```

```
C:\Windows\System32\cmd.exe
C:\Users\xhallade\Desktop\hello-jni>C:\Android\ndk\ndk-build.cmd APP_ABI=all
[arm64-v8a] Compile      : hello-jni <= hello-jni.c
[arm64-v8a] SharedLibrary : libhello-jni.so
[arm64-v8a] Install     : libhello-jni.so => libs/arm64-v8a/libhello-jni.so
[x86_64] Compile      : hello-jni <= hello-jni.c
[x86_64] SharedLibrary : libhello-jni.so
[x86_64] Install     : libhello-jni.so => libs/x86_64/libhello-jni.so
[mips64] Compile      : hello-jni <= hello-jni.c
[mips64] SharedLibrary : libhello-jni.so
[mips64] Install     : libhello-jni.so => libs/mips64/libhello-jni.so
[armeabi-v7a] Compile thumb : hello-jni <= hello-jni.c
[armeabi-v7a] SharedLibrary : libhello-jni.so
[armeabi-v7a] Install     : libhello-jni.so => libs/armeabi-v7a/libhello-jni.so
[armeabi] Compile thumb : hello-jni <= hello-jni.c
[armeabi] SharedLibrary : libhello-jni.so
[armeabi] Install     : libhello-jni.so => libs/armeabi/libhello-jni.so
[x86] Compile      : hello-jni <= hello-jni.c
[x86] SharedLibrary : libhello-jni.so
[x86] Install     : libhello-jni.so => libs/x86/libhello-jni.so
[mips] Compile      : hello-jni <= hello-jni.c
[mips] SharedLibrary : libhello-jni.so
[mips] Install     : libhello-jni.so => libs/mips/libhello-jni.so
```

Build ARM64 libs

Build x86\_64 libs

Build mips64 libs

Build ARMv7a libs

Build ARMv5 libs

Build x86 libs

Build mips libs

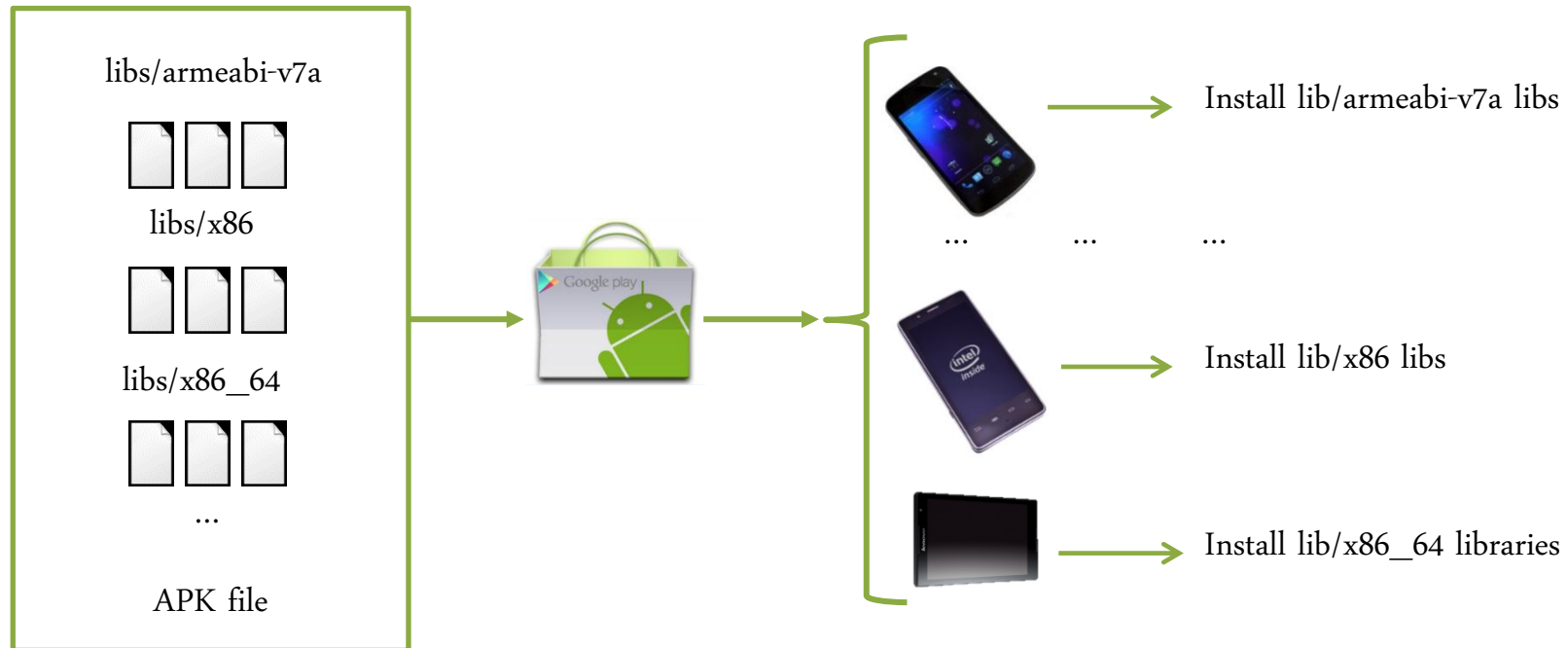
The NDK will generate optimized code for all target ABIs

You can also pass APP\_ABI variable to ndk-build, and specify each ABI:

```
ndk-build APP_ABI=x86 #all32 and all64 are also possible values.
```

# “Fat” APKs

By default, an APK contains the .so files for every supported ABIs.



Libs for the selected ABI are installed, the others remain inside the downloaded APK

# Multiple APKs – clean solution with gradle

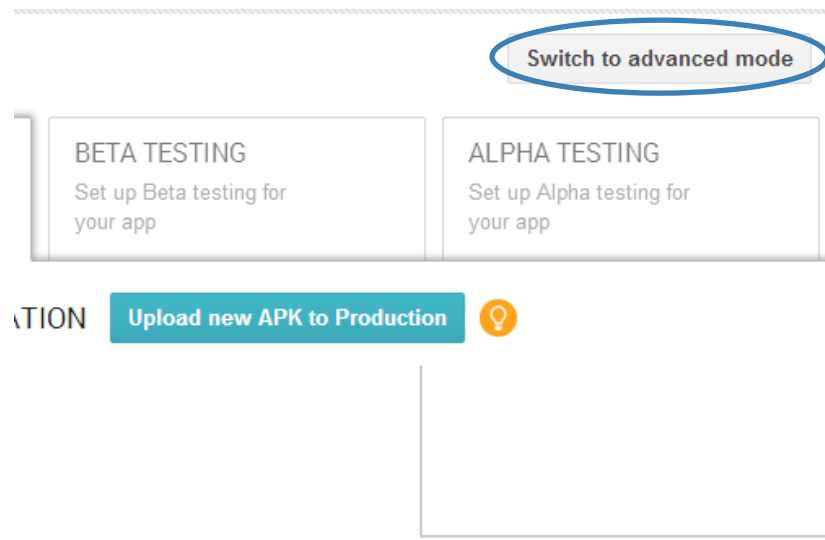
```
splits {
    abi {
        enable true
        reset()
        include 'x86', 'x86_64', 'armeabi-v7a', 'arm64-v8a'
        universalApk true
    }
}

// map for the version code
project.ext.versionCodes = ['armeabi': 1, 'armeabi-v7a': 2, 'arm64-v8a': 3, 'mips': 5, 'mips64': 6,
'x86': 8, 'x86_64': 9]

android.applicationVariants.all { variant ->
    // assign different version code for each output
    variant.outputs.each { output ->
        output.versionCodeOverride =
            project.ext.versionCodes.get(output.abiFilter, 0) * 1000000 +
android.defaultConfig.versionCode
    }
}
```

# Uploading Multiple APKs to the store

Switch to Advanced mode before uploading the second APK.



APK Save Revert changes Switch to simple mode

**PRODUCTION**  
Versions  
**21413100, 61413100**

**BETA TESTING**  
Set up Beta testing for your app

**ALPHA TESTING**  
Set up Alpha testing for your app

**PRODUCTION CONFIGURATION** ⓘ

**CURRENT CONFIGURATION** uploaded on **Mar 18, 2013 1:58:23 AM**

**Supported devices**  
**2367**  
[See list](#)

**Excluded devices**  
**0**  
[Manage excluded devices](#)

Upload new APK to Production

Some devices are eligible to run multiple APKs. In such a scenario, the device will receive the APK with the higher version code.

VERSION	NATIVE PLATFORMS	UPLOADED ON	STATUS	ACTIONS
61413100 (1.0.0)	x86	Mar 18, 2013	Draft in Prod	<span>Move...</span>
21413100 (1.0.0)	armeabi	Mar 18, 2013	Draft in Prod	<span>Move...</span>

# Summary



- Apps submission started only last November: good opportunity to bring more visibility to your apps and games
- Adding Android TV support isn't necessarily much work
- No need for maintaining a separate APK
- When targeting the Nexus Player, a better x86 apps compatibility is nice to have for your apps/libs/engines