Software

Developing for Android TV and the Nexus Player

인텔코리아
이진용 차장

Software

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

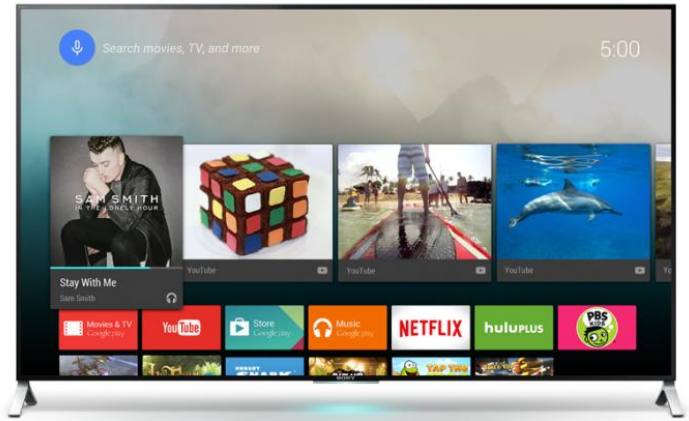
* Ethernet can be added using standard USB adapters

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

Other devices

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

Movies & TV Google play

Store Google play

NETFLIX

Songza

musixmatch

KODI

HUFF POST LIVE

YouTube

Music

Games

Dailymotion

Videos

Sideload

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" />`

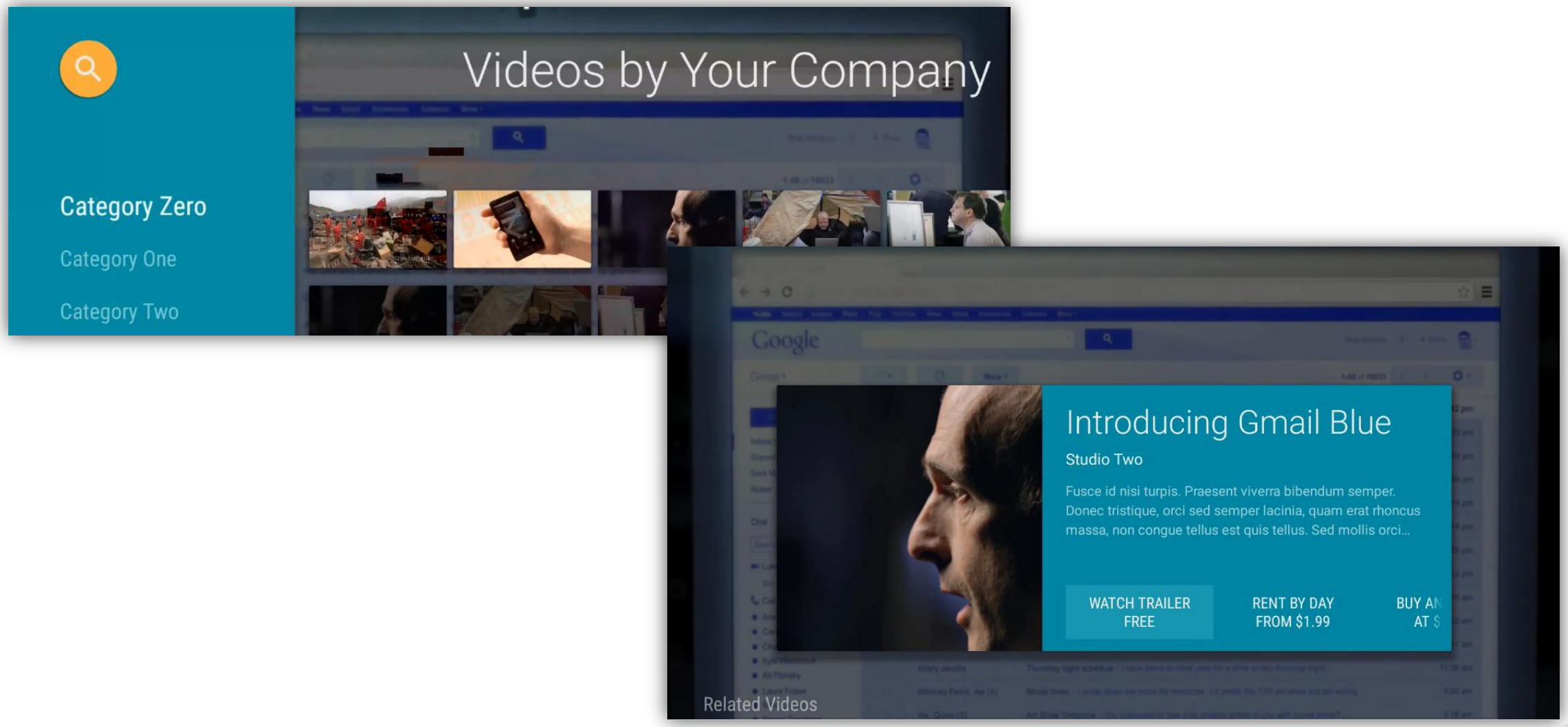
Integrating further with Android TV

Leanback Support library, Recommendations, Search, TV Input Framework

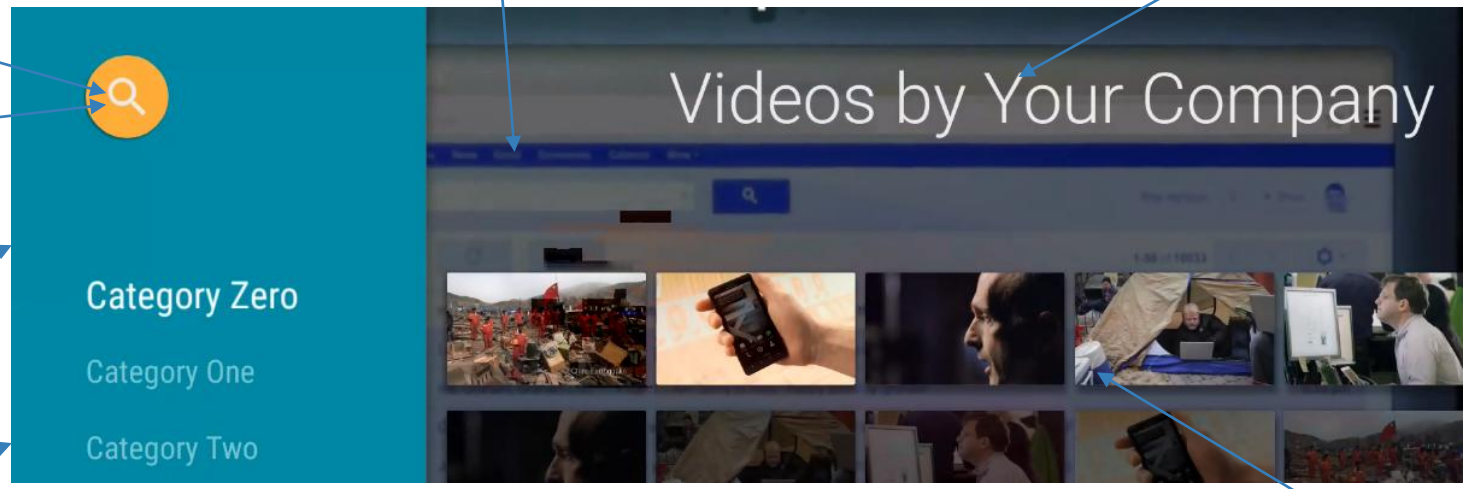
Integrating further with Android TV

Leanback Support library, Recommendations, Search, TV Input Framework

The Leanback support library



The Leanback support library – BrowseFragment



BackgroundManager.getInstance()
.setDrawable()

setTitle()
setBadgeDrawable()

setSearchAffordanceColor()

setOnSearchClickedListener()

setBrandColor()

setHeadersState()

setAdapter(ObjectAdapter)
Presented items have to be *Row*s.

setOnItemViewSelectedListener()
setOnItemViewClickedListener()

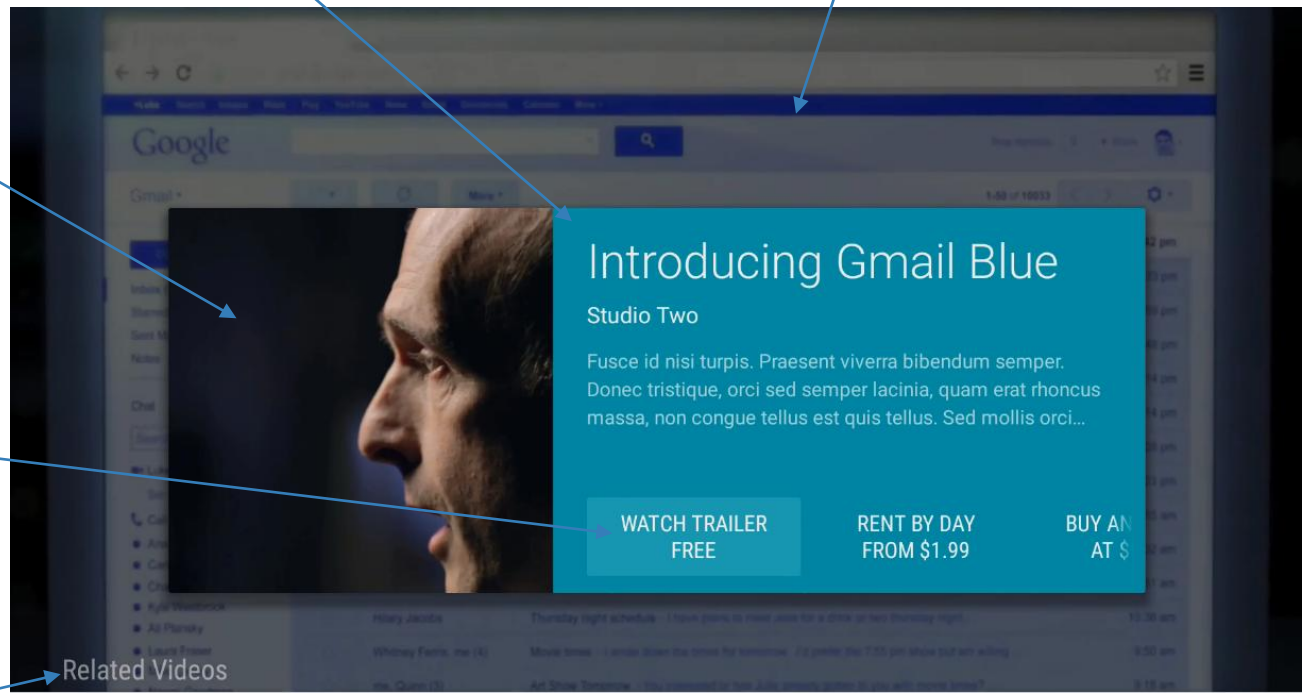
The Leanback support library – DetailsFragment

```
detailsOverviewRowPresenter.setBackgroundColor()
```

BackgroundManager.getInstance()
.setDrawable()

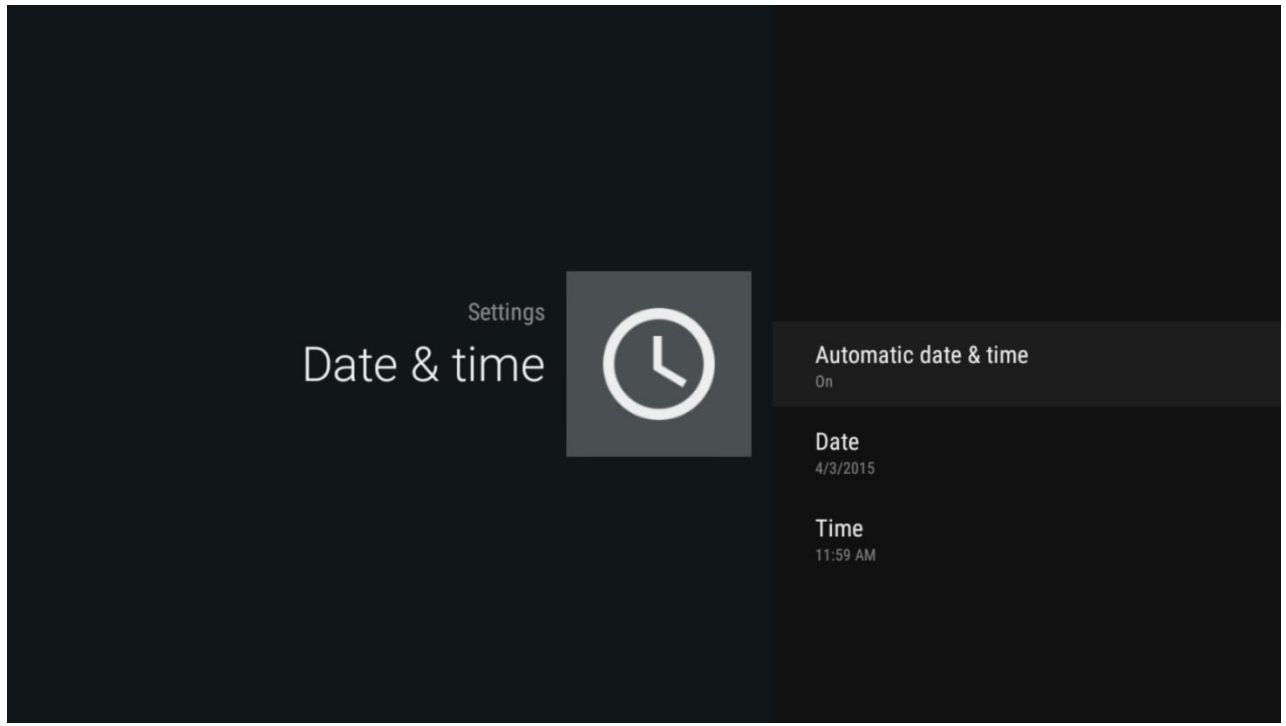
```
detailsOverviewRow.setImageDrawable()
```

```
detailsOverviewRow.addAction()
```




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

The Leanback support library – GuidedStepFragment

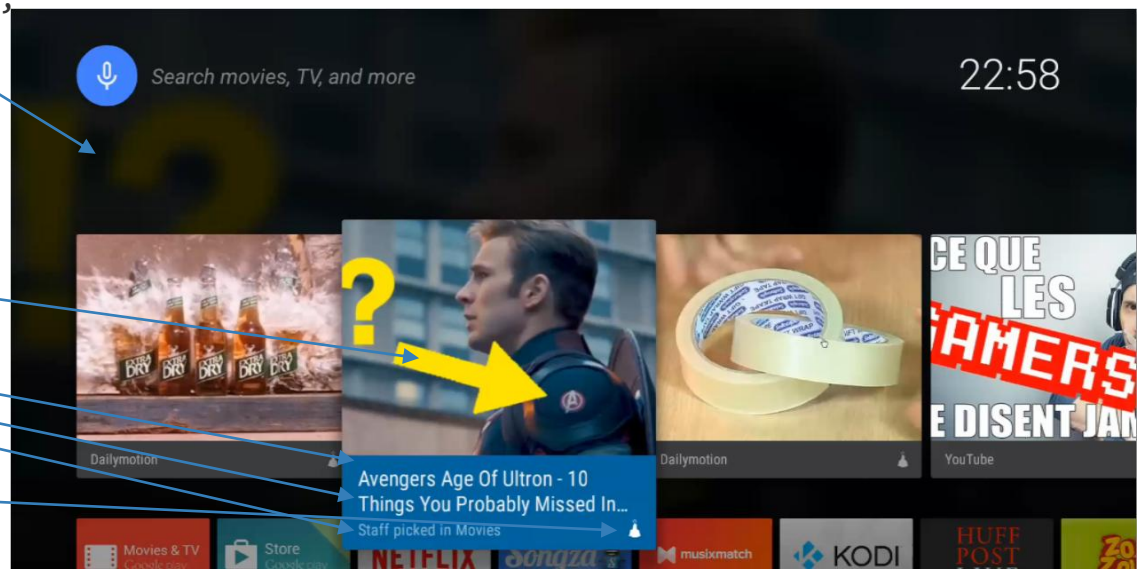


- GuidanceStylist.Guidance(String title, String description, String breadcrumb, Drawable icon)

- 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();
```

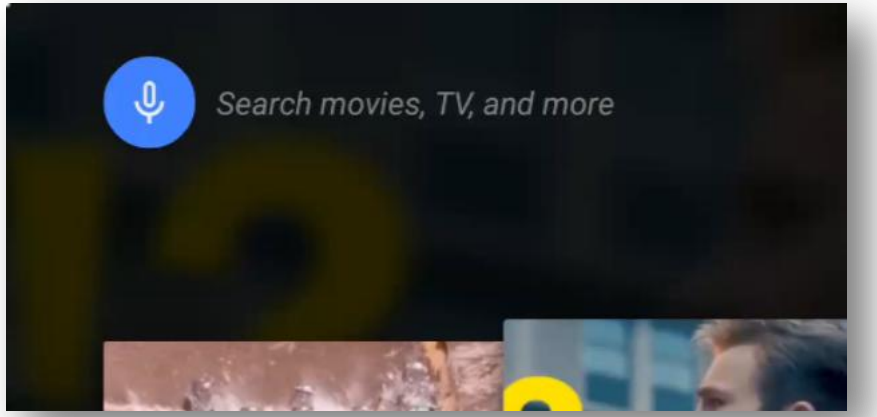


It's advised to update recommendations from a service you can trigger using an AlarmManager that will run it periodically, starting with shorty after boot.

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.



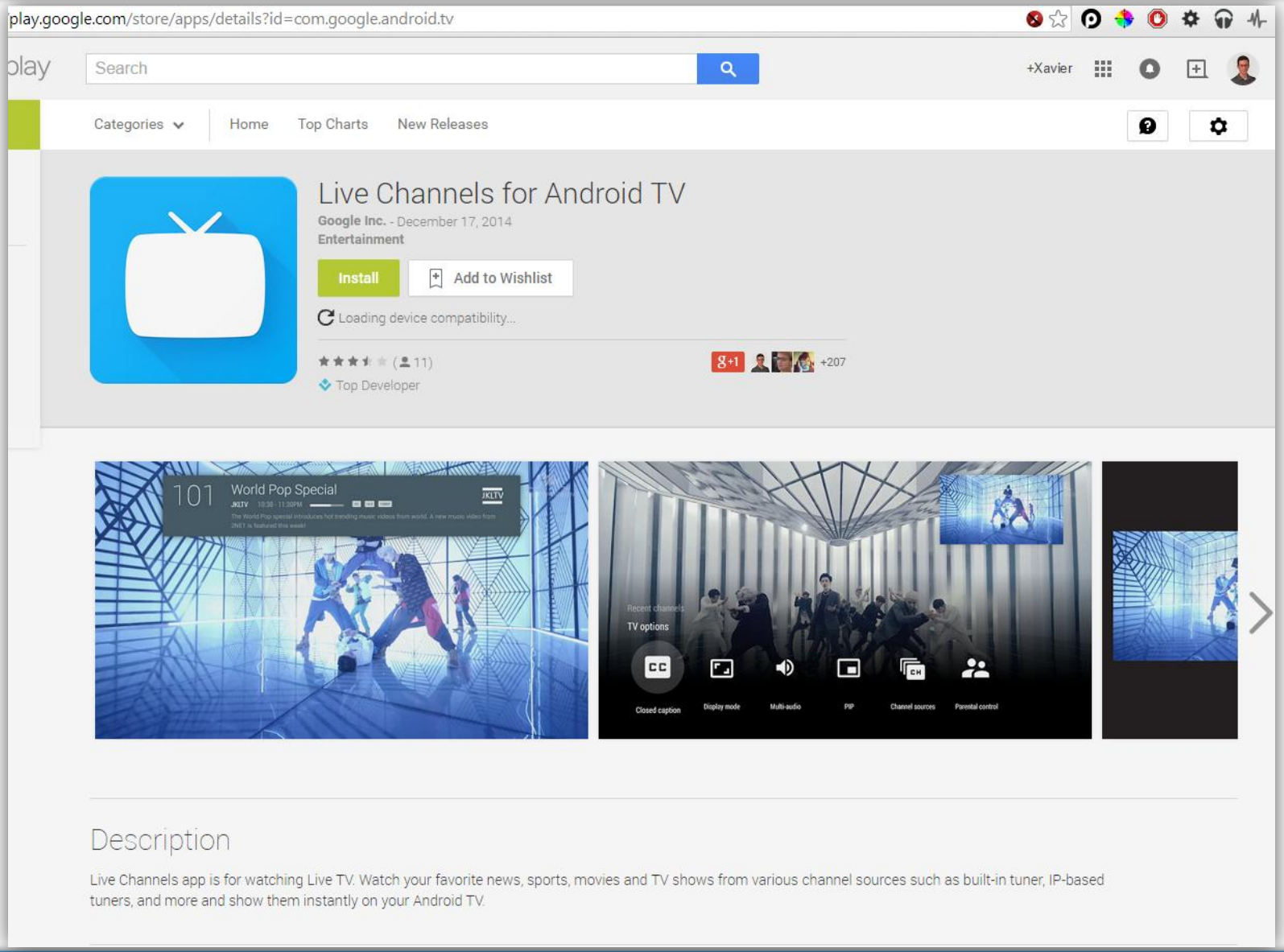
Supporting multiple controllers

- 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



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" />
```

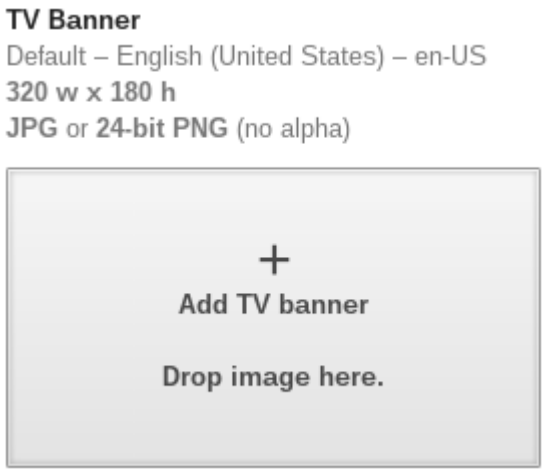
Using permissions with implicit hardware requirements ?

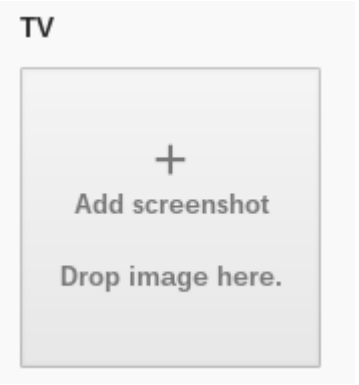
The below features aren't available on every Android TV devices:

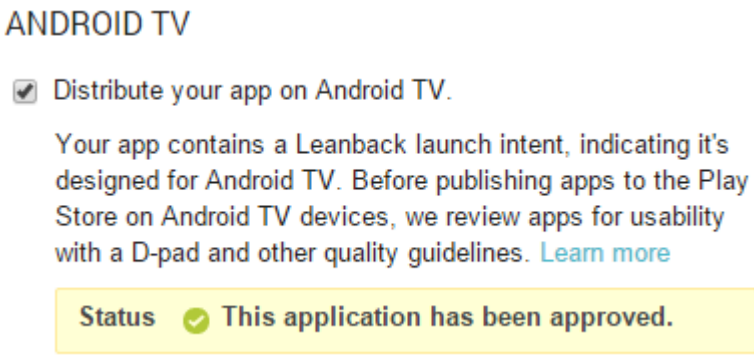
- *android.hardware.location.gps* implied by *android.permission.ACCESS_FINE_LOCATION*
- *android.hardware.camera.autofocus* and *android.hardware.camera* implied by *android.permission.CAMERA*
- *android.hardware.telephony* implied by many telephony-specific permissions
- *android.hardware.microphone* 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

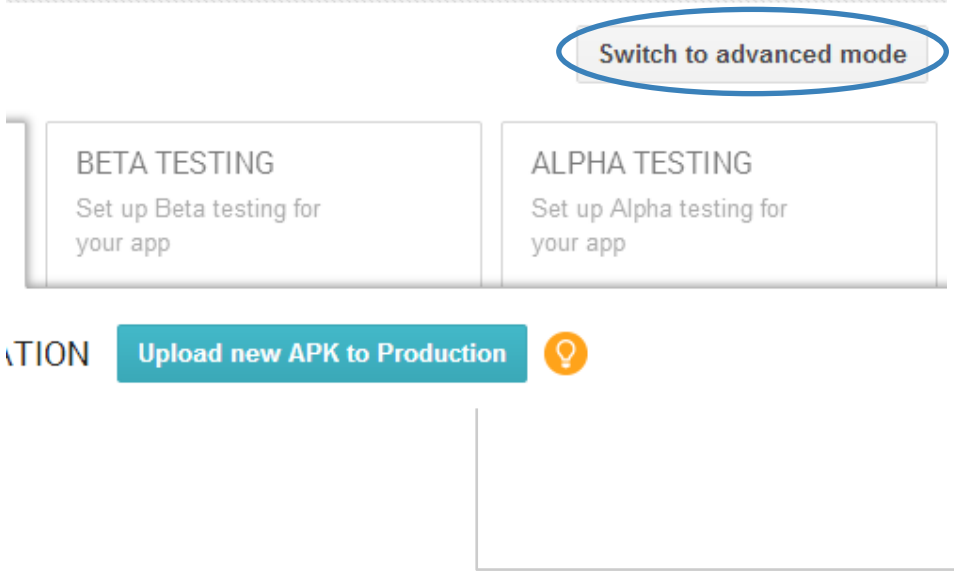


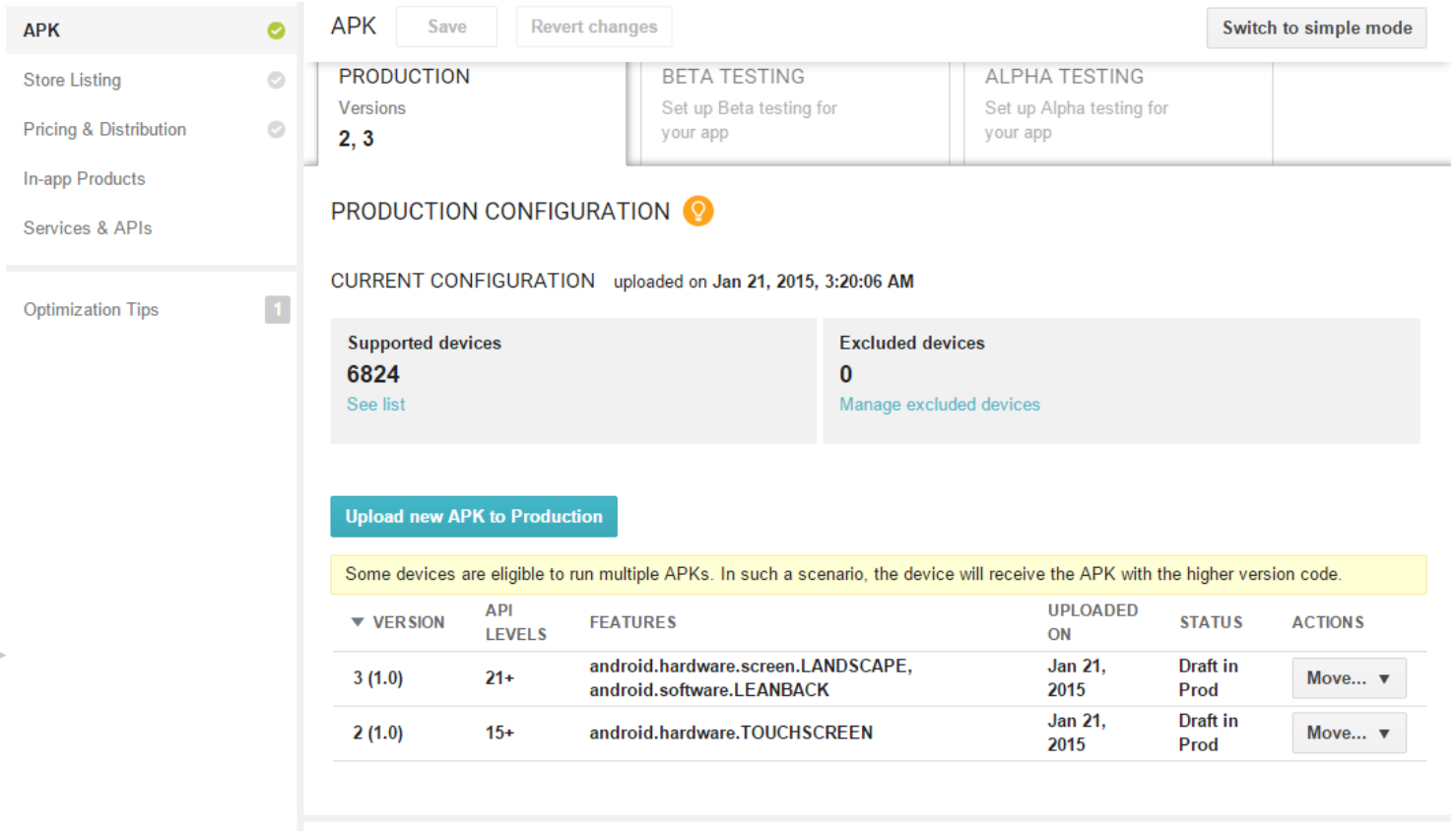




Publishing your TV application as an alternative APK

Switch to Advanced mode before uploading the second APK.





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

Lava* Xolo X900

ASUS Fonepad™ Note FHD - 6”

Intel® Yolo

Samsung* Galaxy™ Tab 3 10.1”

Acer* Liquid C1

ASUS* Fonepad™ 7”

2014…

Asus* Zenfones 4/5/6

KD Interactive Kurio Tablet

Asus* FonePad 7/8

Lenovo* S8

Toshiba Excite Go

Asus* MemoPad 7/8

ASUS* Transformer Pad TF103CG/TF303CL

Acer* Iconia Tab 8 / One 8

Lenovo Yoga Tab 2 8/10/13

Nexus Player

Tesco Hudl 2

2015…

Dell* Venue 8 7840

Asus* Zenfone 2

Lenovo* P90

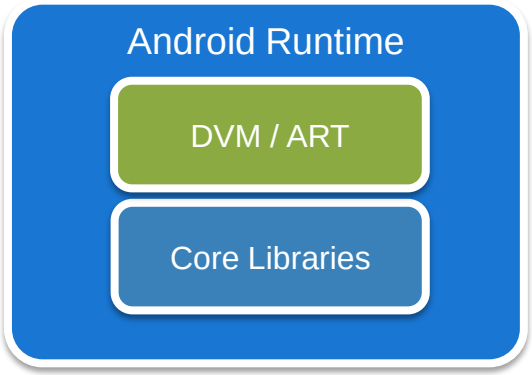
Nokia n1

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

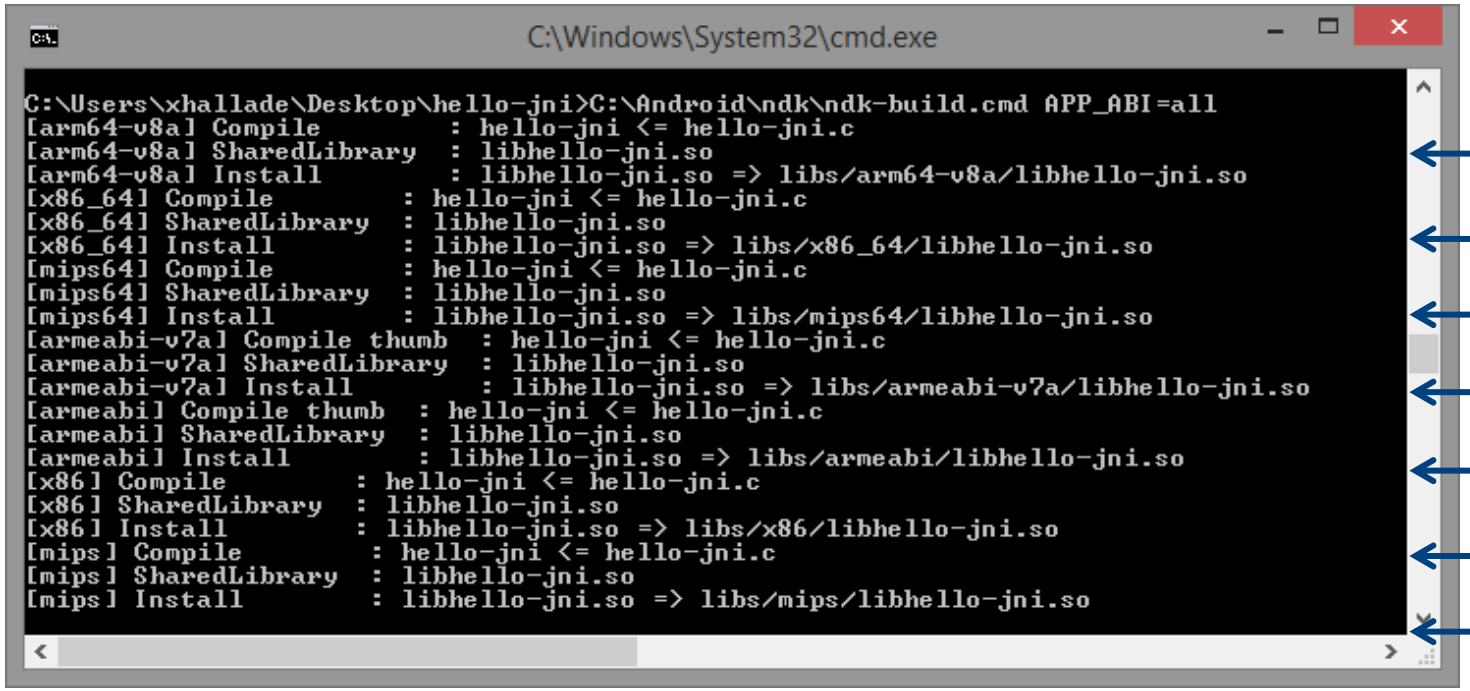
3rd party libraries/engines x86 support

- 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
```



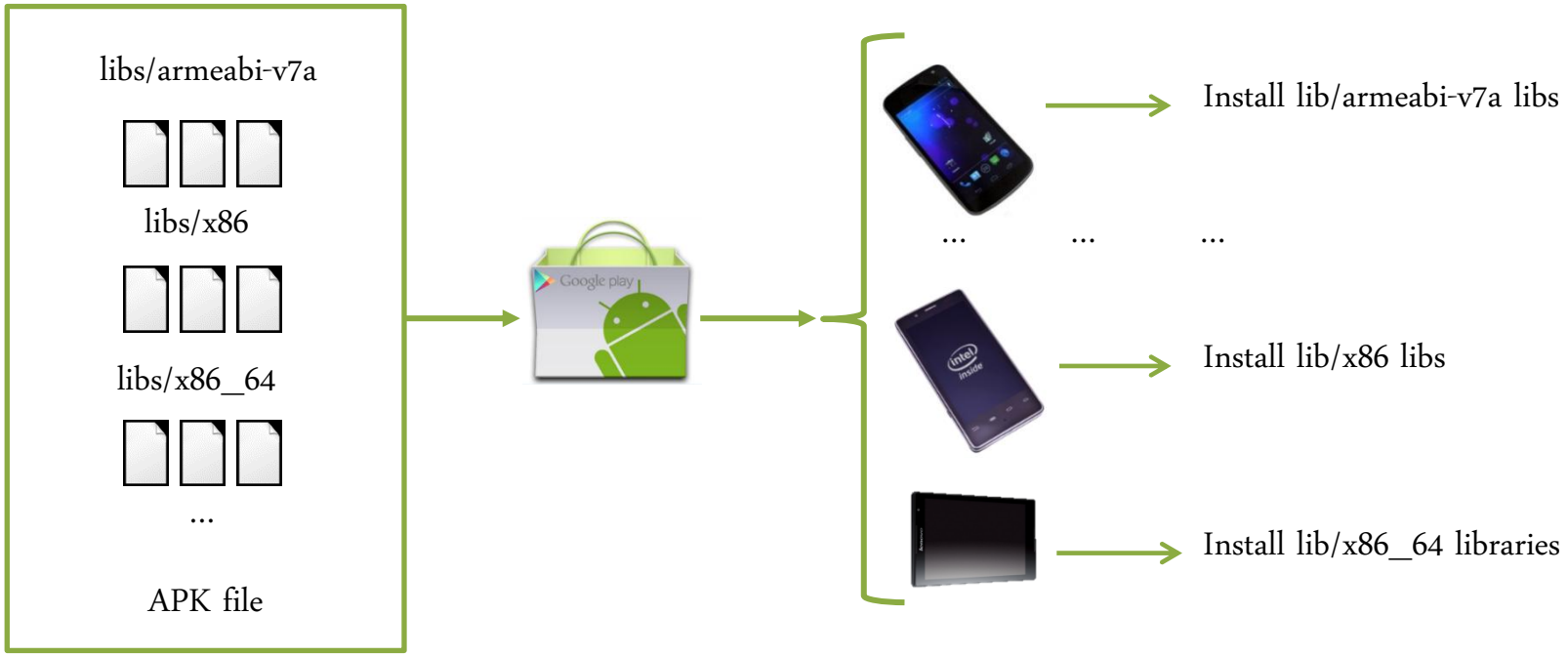
The NDK will generate optimized code for all target ABIs

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.



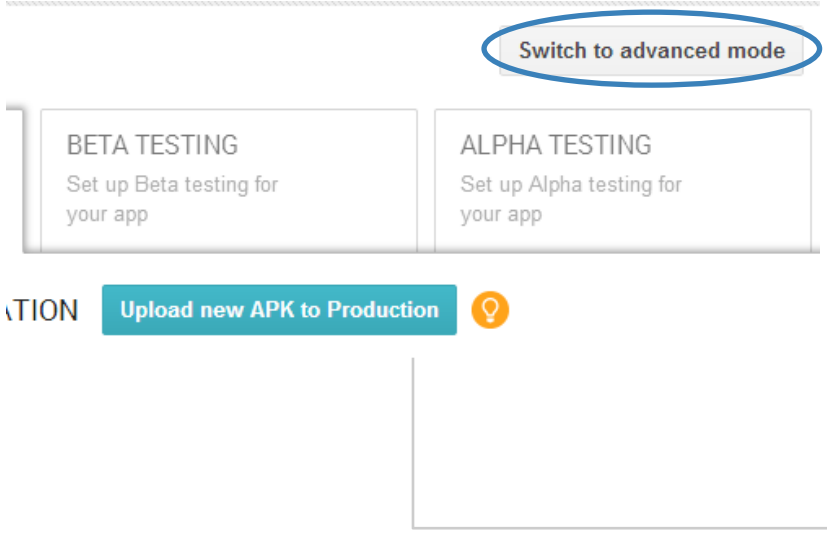
Libs for the selected ABI are installed, the others remain inside the downloaded APK

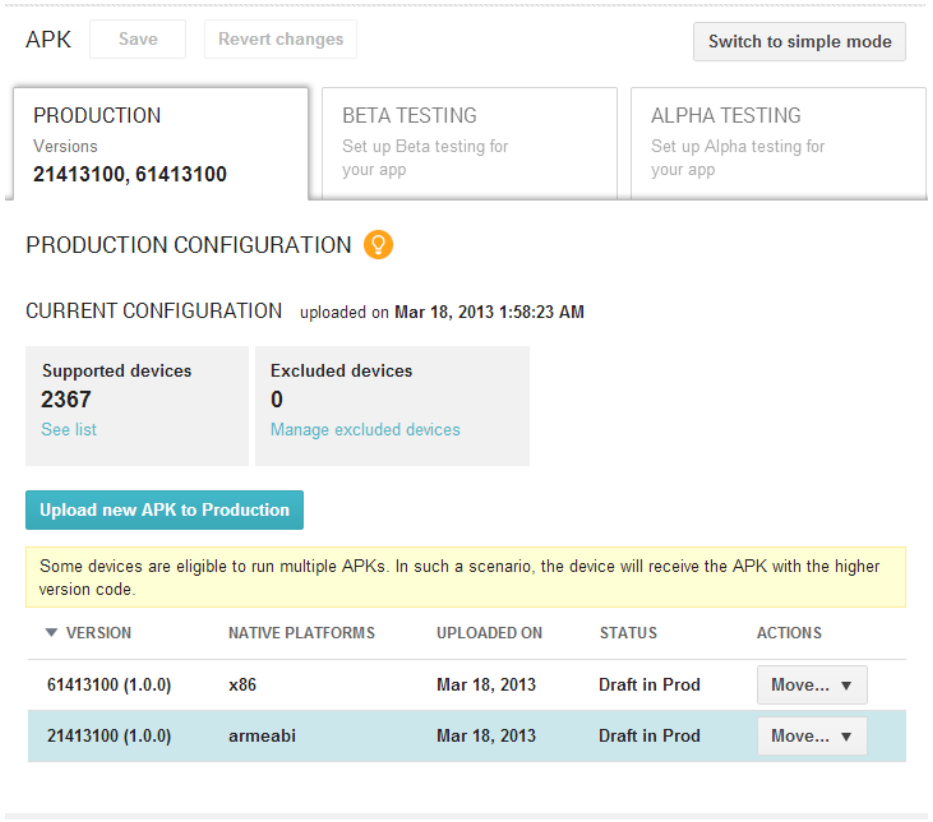
Multiple APKs – clean solution with gradle

```
 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.





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