Monetizing Android apps on the Play Store

(With In-app Billing V3 & Google Play Developer API)

Yash Prabhu

Why are we here?

7 things we are going to learn today

The Big Picture

Product Types

In-app Billing v3

Monetization

Testing

Google Play Developer API

Security

Monetization

Revenue models

Digital goods

- Freemium
  - In-app products
  - Subscription
- Paid
- AdMob

Physical goods

- Android Pay

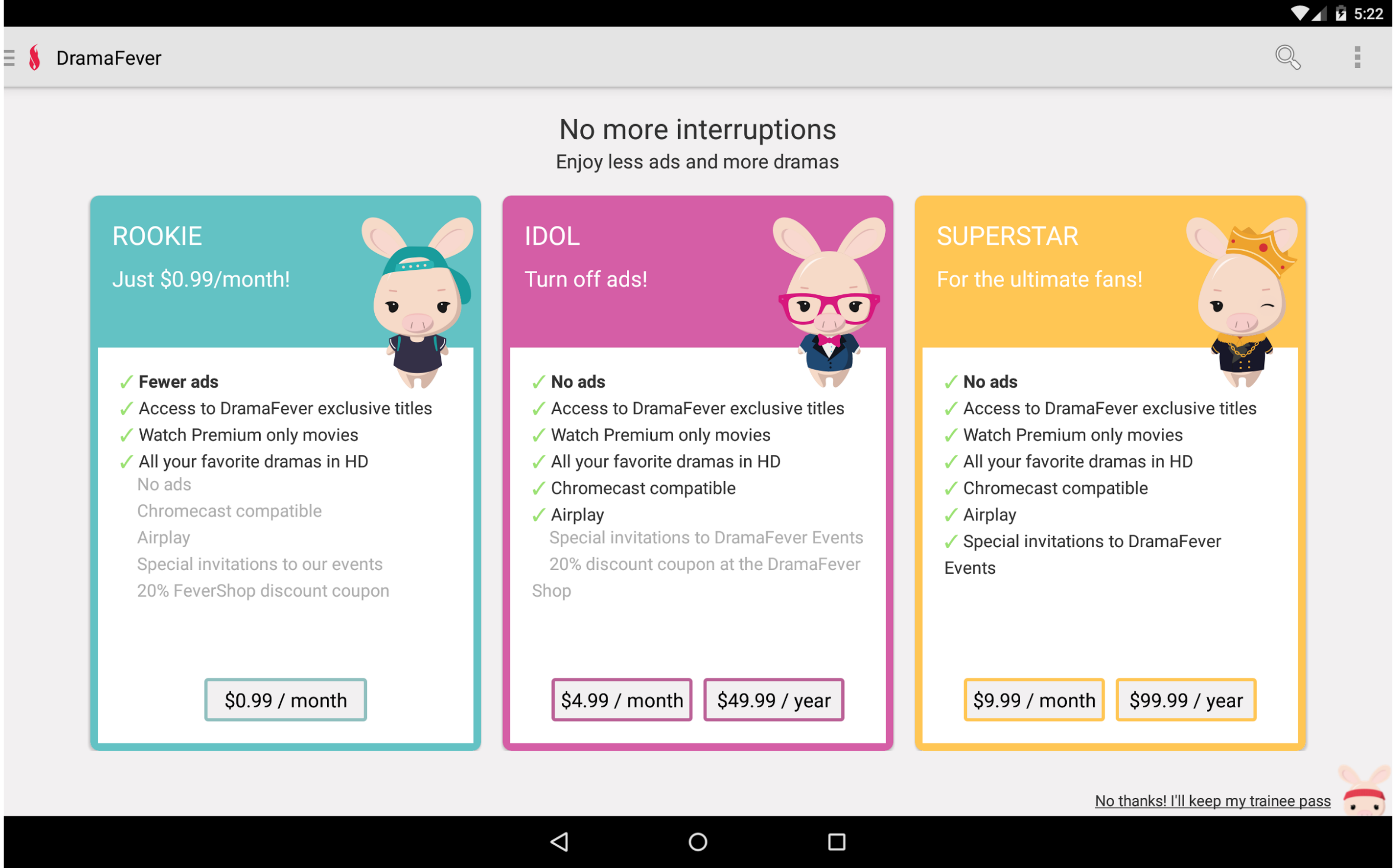
Product Types

$ Managed

- For one time purchases
- Non-Consumable
  - Can only be purchased once
  - Permanent benefit
  - Premium upgrade, level pack
- Consumable
  - Available for purchase multiple times
  - Temporary benefit
  - Gold coins, extra life

Subscriptions

- For automated, recurring billing
- Weekly, monthly, annual, seasonal
- Free trials (7 to 30 days)
- Manual renewal
- Subscription upgrade/downgrade
- Deferred Billing


5:22

DramaFever

No more interruptions

Enjoy less ads and more dramas

ROOKIE

Just $0.99/month!

✓ **Fewer ads**
✓ Access to DramaFever exclusive titles
✓ Watch Premium only movies
✓ All your favorite dramas in HD
No ads
Chromecast compatible
Airplay
Special invitations to our events
20% FeverShop discount coupon

$0.99 / month

IDOL

Turn off ads!

✓ **No ads**
✓ Access to DramaFever exclusive titles
✓ Watch Premium only movies
✓ All your favorite dramas in HD
✓ Chromecast compatible
✓ Airplay
Special invitations to DramaFever Events
20% discount coupon at the DramaFever Shop

$4.99 / month | $49.99 / year

SUPERSTAR

For the ultimate fans!

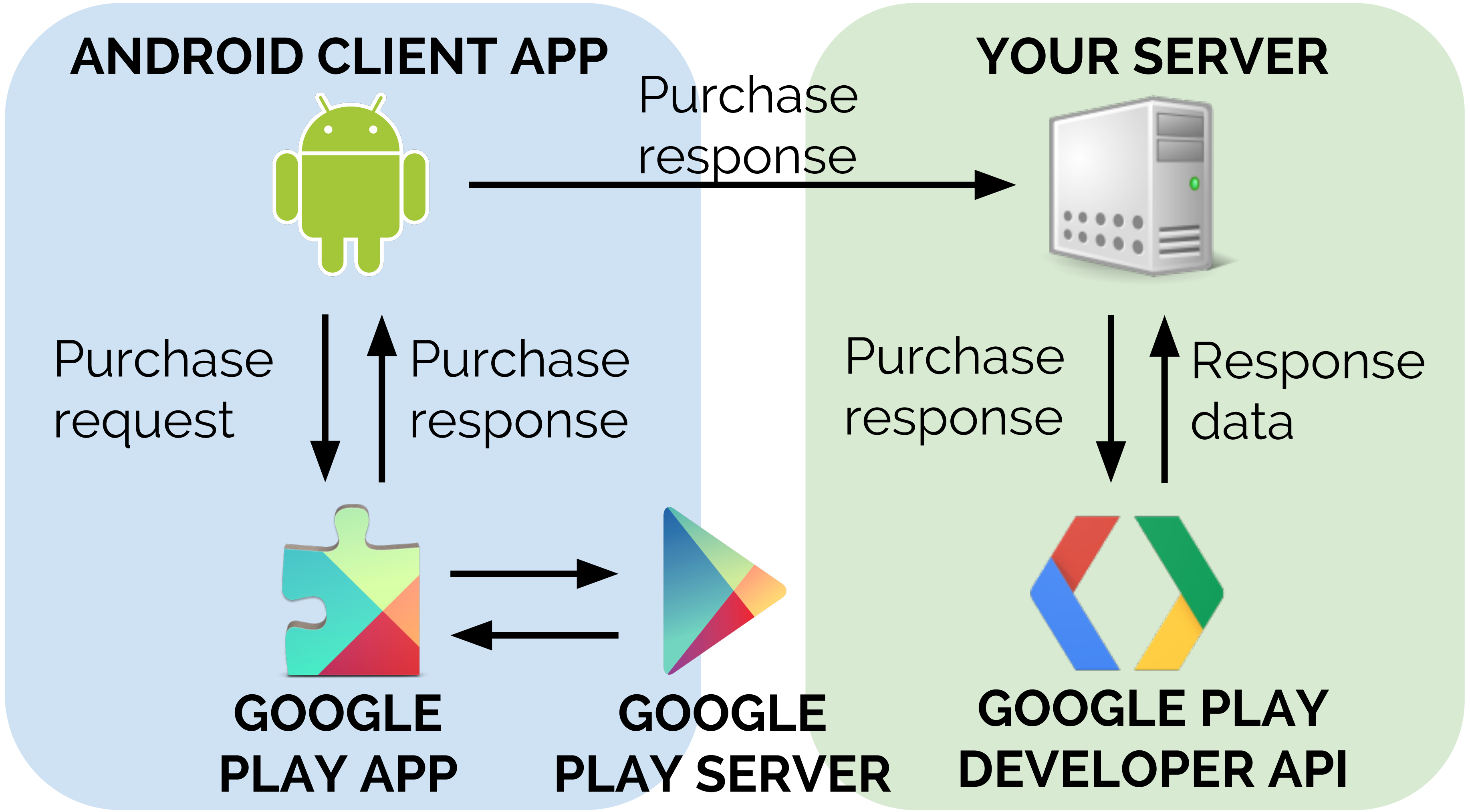
✓ **No ads**
✓ Access to DramaFever exclusive titles
✓ Watch Premium only movies
✓ All your favorite dramas in HD
✓ Chromecast compatible
✓ Airplay
✓ Special invitations to DramaFever Events

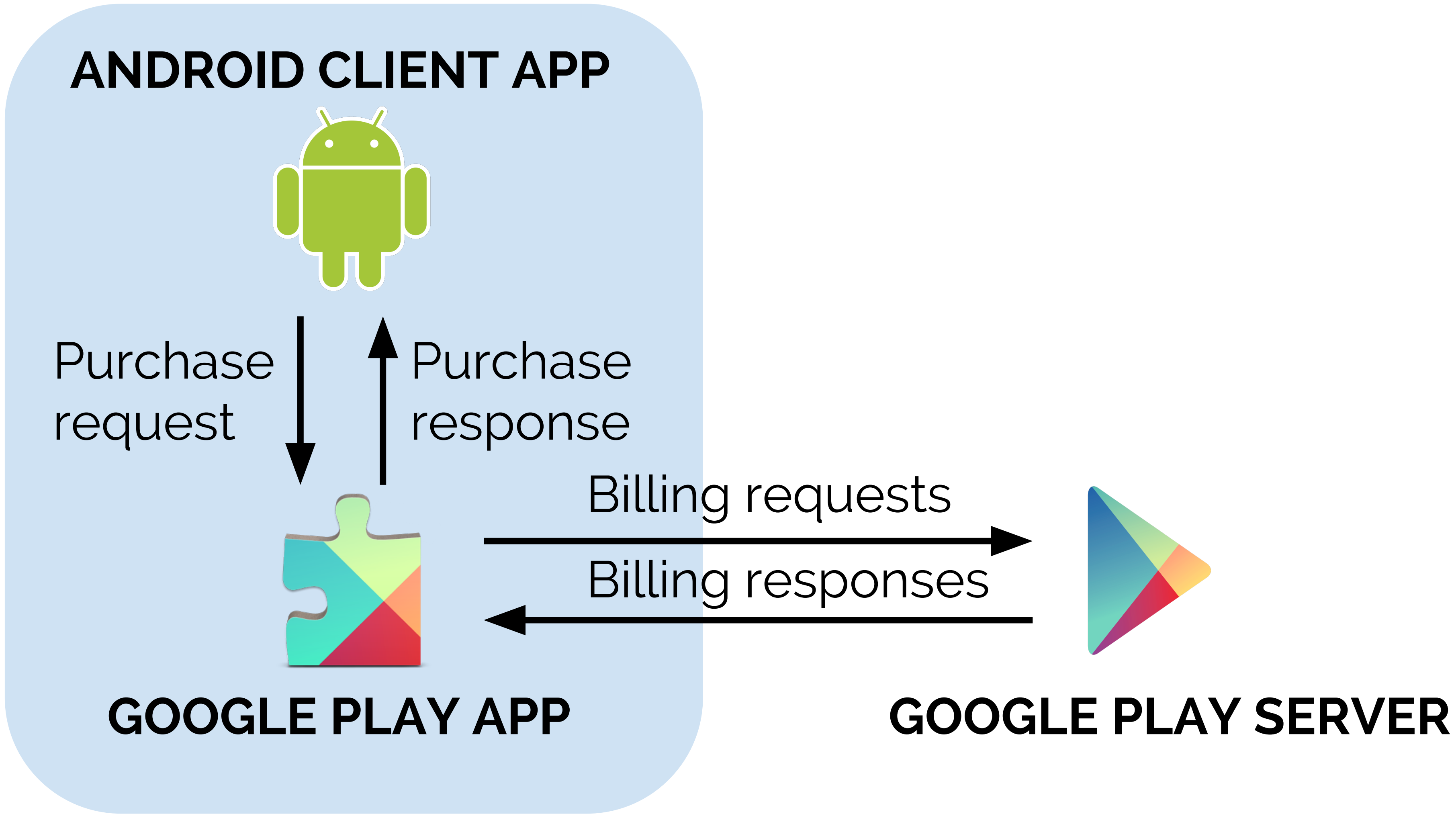
$9.99 / month | $99.99 / year

No thanks! I'll keep my trainee pass

The Big Picture

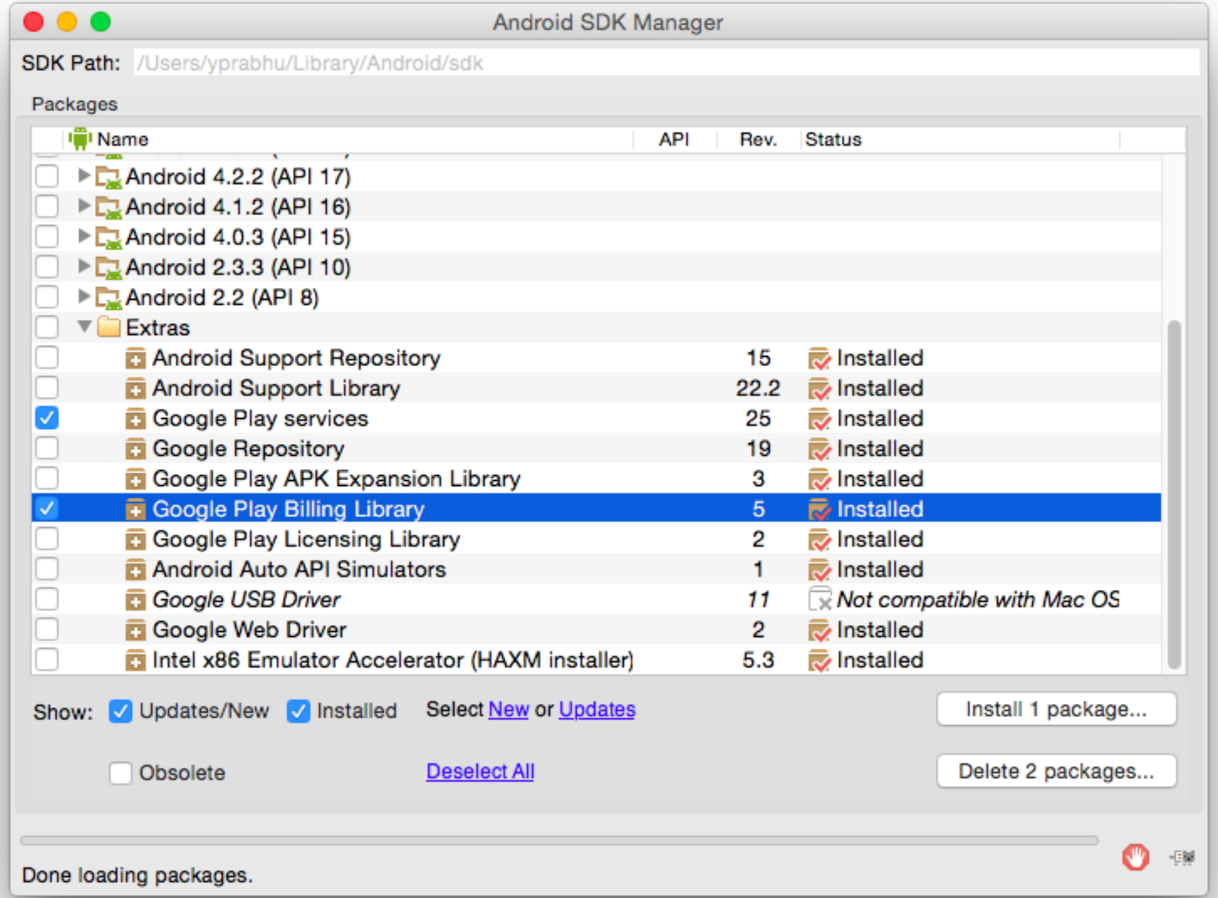
Purchase workflow


ANDROID CLIENT APP
YOUR SERVER
Purchase response
Purchase request
Purchase response
Purchase response
Response data
GOOGLE PLAY APP
GOOGLE PLAY SERVER
GOOGLE PLAY DEVELOPER API


ANDROID CLIENT APP
Purchase request
Purchase response
Billing requests
Billing responses
GOOGLE PLAY APP
GOOGLE PLAY SERVER

In-app Billing v3

Client side workflow



Android SDK Manager

SDK Path: /Users/yprabhu/Library/Android/sdk

Packages

	Name	API	Rev.	Status
☐	▶ Android 4.2.2 (API 17)			
☐	▶ Android 4.1.2 (API 16)			
☐	▶ Android 4.0.3 (API 15)			
☐	▶ Android 2.3.3 (API 10)			
☐	▶ Android 2.2 (API 8)			
☐	▼ Extras			
☐	Android Support Repository		15	Installed
☐	Android Support Library		22.2	Installed
☑	Google Play services		25	Installed
☐	Google Repository		19	Installed
☐	Google Play APK Expansion Library		3	Installed
☑	Google Play Billing Library		5	Installed
☐	Google Play Licensing Library		2	Installed
☐	Android Auto API Simulators		1	Installed
☐	*Google USB Driver*		*11*	*Not compatible with Mac OS*
☐	Google Web Driver		2	Installed
☐	Intel x86 Emulator Accelerator (HAXM installer)		5.3	Installed

Show: ☑ Updates/New ☑ Installed Select New or Updates [Install 1 package...]

☐ Obsolete Deselect All [Delete 2 packages...]

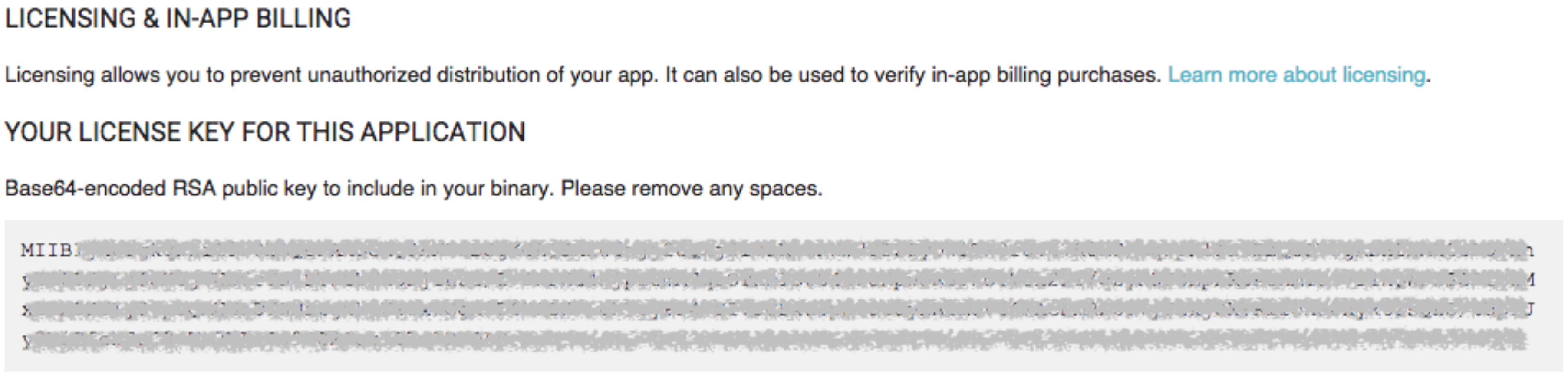
Done loading packages.

Setting up In-app billing

- Go to sdk/extras/google/play_billing
- Copy into your project
  - IInAppBillingService.aidl
  - All Java files in trivialdrivesample/util
- Add billing permission into your manifest

```
<uses-permission
  android:name= "com.android.vending.BILLING" />
```

Copy your license key into util/IabHelper.java



LICENSING & IN-APP BILLING

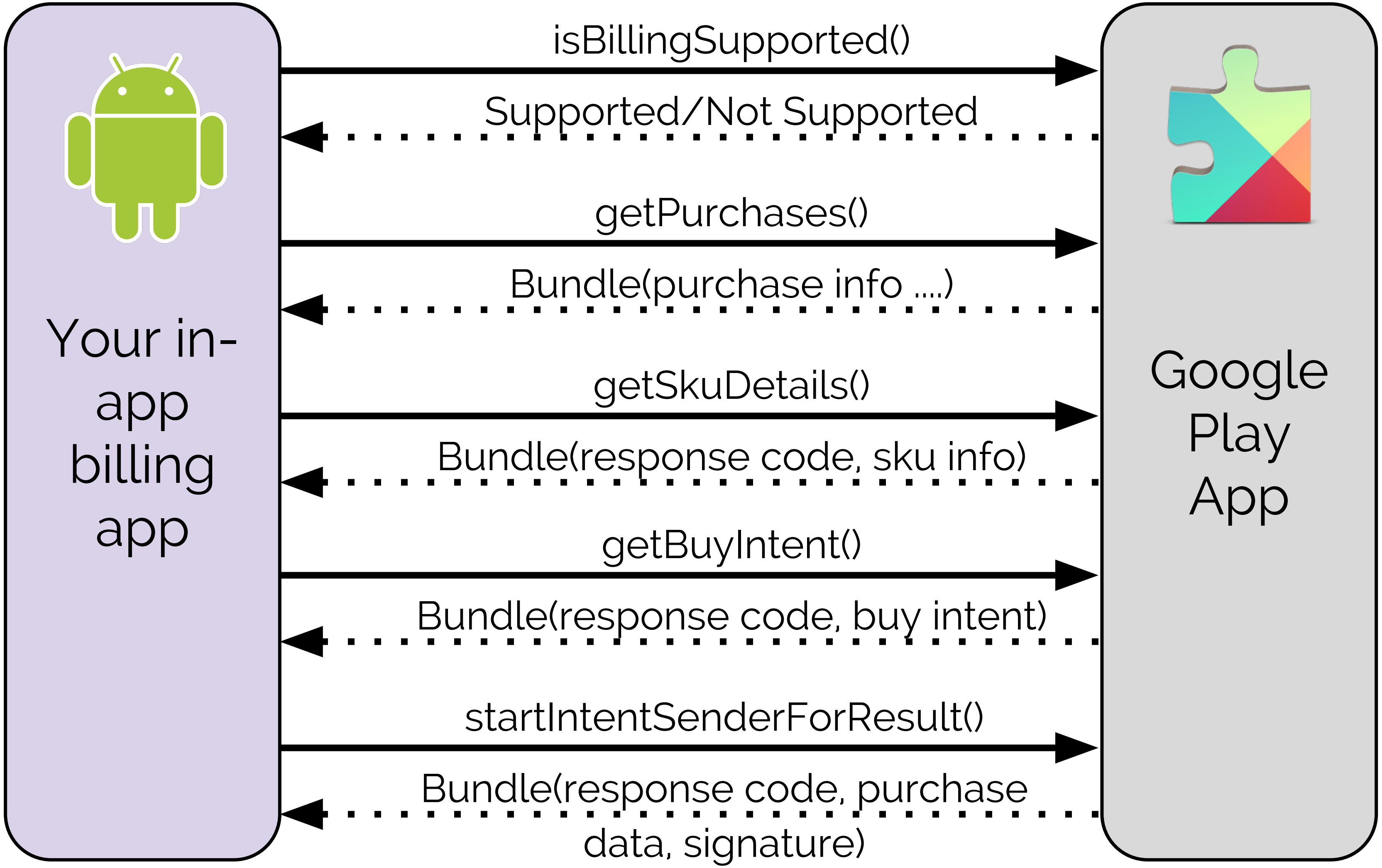
Licensing allows you to prevent unauthorized distribution of your app. It can also be used to verify in-app billing purchases. Learn more about licensing.

YOUR LICENSE KEY FOR THIS APPLICATION

Base64-encoded RSA public key to include in your binary. Please remove any spaces.

```
MIIB[redacted]
```

Iab => In-app billing


Your in-app billing app
isBillingSupported()
Supported/Not Supported
getPurchases()
Bundle(purchase info ....)
getSkuDetails()
Bundle(response code, sku info)
getBuyIntent()
Bundle(response code, buy intent)
startIntentSenderForResult()
Bundle(response code, purchase data, signature)
Google Play App

Is billing supported?

```
public void onServiceConnected(ComponentName name, IBinder service) {
    mService = IInAppBillingService.Stub.asInterface(service);
    int response = mService.isBillingSupported(3, packageName, itemType);
    if (response == BILLING_RESPONSE_RESULT_OK) {
    // billing is supported
    } else {
    // billing is not supported
    }
}
```

Query user's in-app purchases

```
Bundle ownedItems =
    mService.getPurchases(3, mContext.getPackageName(),
        itemType, continueToken);
```

Query in-app product details

```
Bundle skuDetails =
  mService.getSkuDetails(3, mContext.getPackageName(),
    itemType, querySkus);
```

When user wants to purchase

```
public void launchPurchaseFlow(Activity act, String sku,
    String itemType, int requestCode,
    OnlabPurchaseFinishedListener listener, String extraData) {
    ....
    Bundle buyIntentBundle =
        mService.getBuyIntent(3, mContext.getPackageName(),
        sku, itemType, extraData);
    ....
}
```

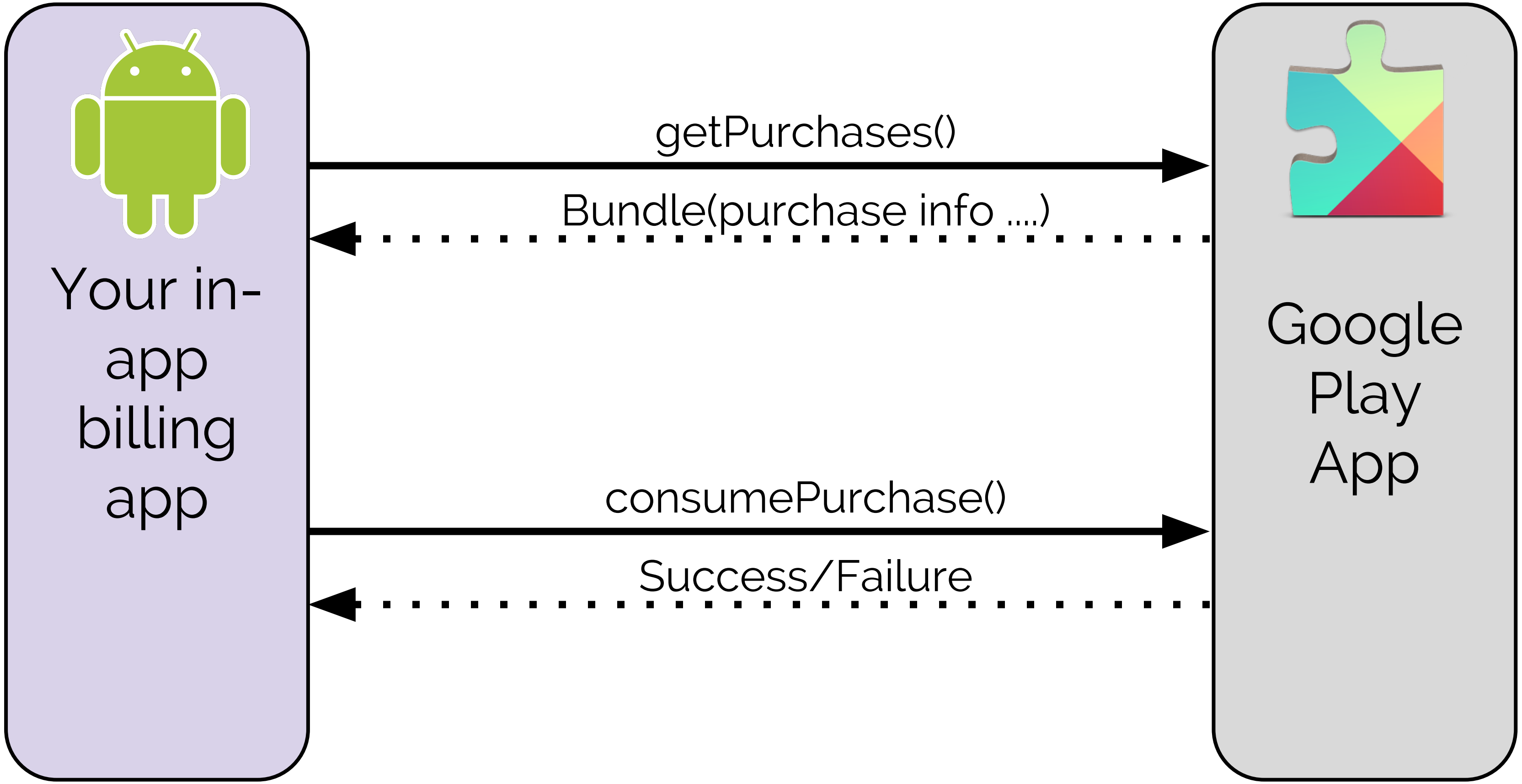
App launches the pending intent

```
PendingIntent pendingIntent =        buyIntentBundle.getParcelable
(RESPONSE_BUY_INTENT);


activity.startIntentSenderForResult(
    pendingIntent.getIntentSender(), requestCode,
    new Intent(), flagsMask, flagsValues, extraFlags);
```

Get purchase information

```
public boolean handleActivityResult(int requestCode, int resultCode, Intent
data) {
    int responseCode = getResponseCodeFromIntent(data);
    String purchaseData =   data.getStringExtra
(RESPONSE_INAPP_PURCHASE_DATA);
    String dataSignature =     data.getStringExtra
(RESPONSE_INAPP_SIGNATURE);
     return true;
}
```


Your in-app billing app
getPurchases()
Bundle(purchase info ....)
consumePurchase()
Success/Failure
Google Play App

Consume a purchase

```
void consume(Purchase itemInfo) throws IabException {
    String token = itemInfo.getToken();
    String sku = itemInfo.getSku();
    int response = mService.consumePurchase(3,
        mContext.getPackageName(), token);
    if (response == BILLING_RESPONSE_RESULT_OK) {
        // sku consumed successfully
    }
}
```

Recap

On startup

- isBillingSupported?
- getPurchases()

When user wants to purchase

- getBuyIntent()

After user bought the purchase

- handleActivityResult()
- if purchase successful consume()

Purchase data

```
{ "orderId": "12999111111111111.14533335334137..0",
  "packageName": "com.sample.app",
  "productId": "test_inapp_upgrade",
  "purchaseTime": 1384834368656,
  "purchaseState": 0,
  "developerPayload": "abcdef",
  "purchaseToken": "fdgdghdjdp86dg",
  "autoRenewing":true }
```

Signature & Response code

Signature: jHCzy6MGITNtOuF0nfYyiEyGw==

Response codes

0: Purchased

1: Canceled

2: Refunded

Testing

Client side workflow

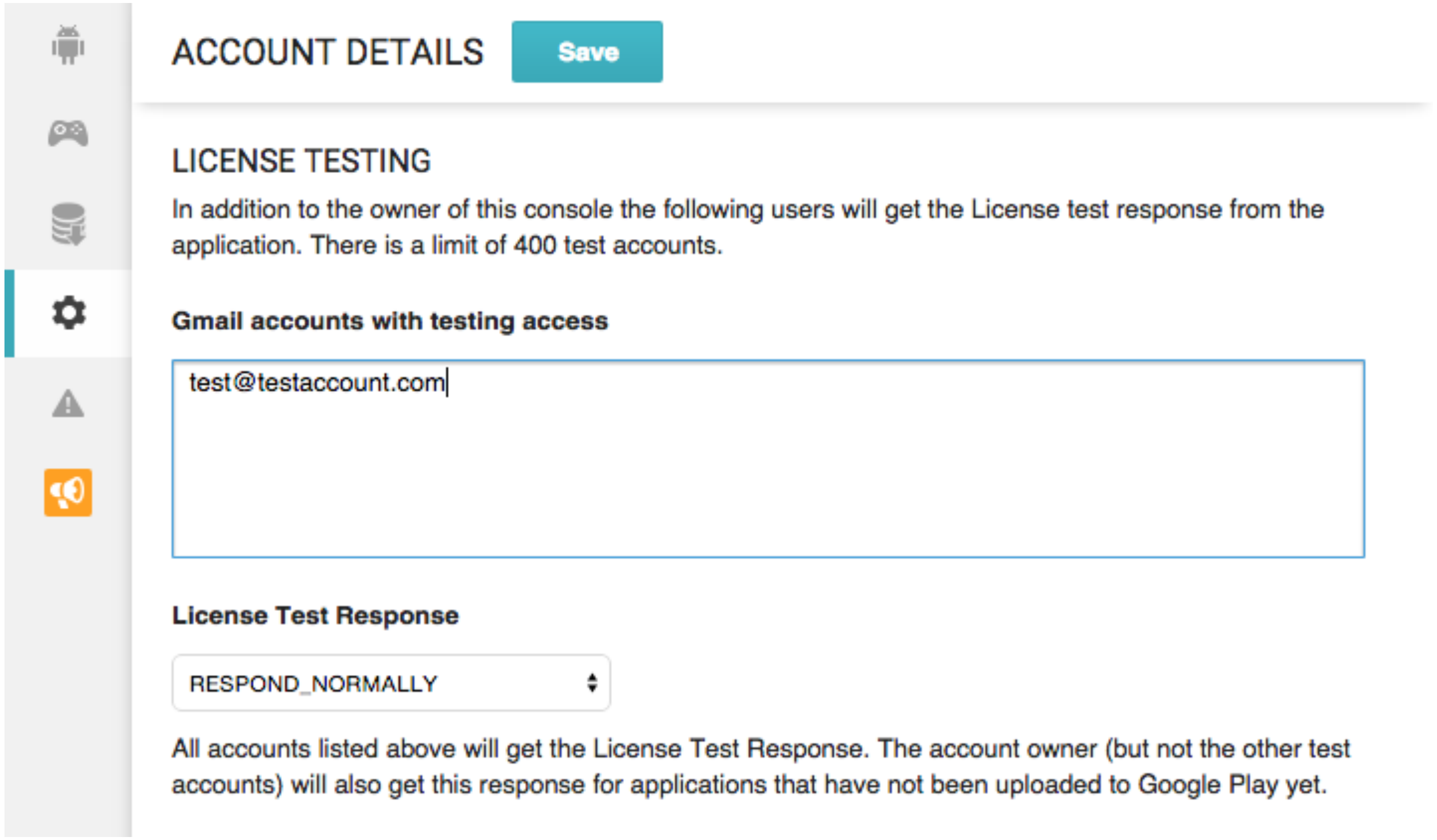
Testing

Test purchases

- Add licensed test users to Google Play developer console
- No charges
- Alpha/beta release groups or published

Real purchases

- Regular users who can download your app from the Play Store
- Actual charges
- Alpha/beta release groups or published


ACCOUNT DETAILS

LICENSE TESTING

In addition to the owner of this console the following users will get the License test response from the application. There is a limit of 400 test accounts.

Gmail accounts with testing access

test@testaccount.com

License Test Response

RESPOND_NORMALLY

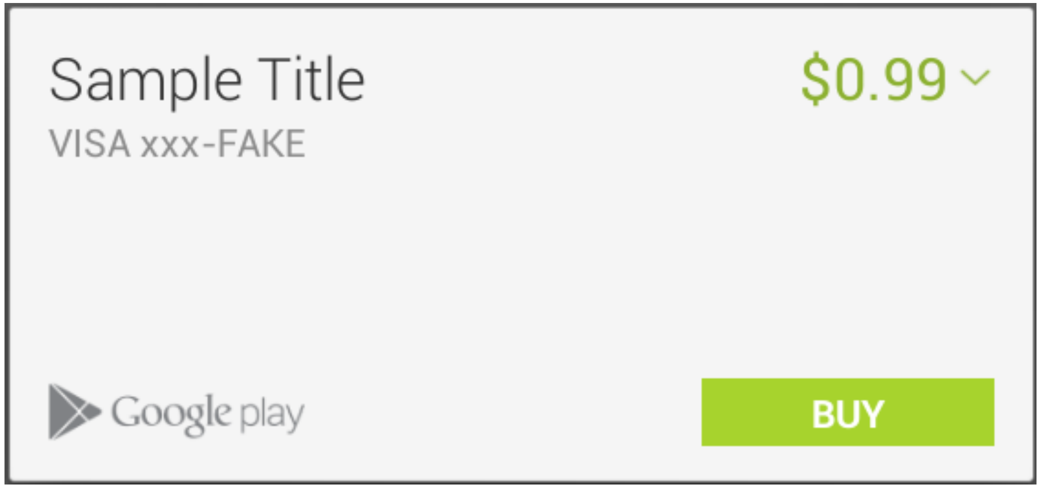
All accounts listed above will get the License Test Response. The account owner (but not the other test accounts) will also get this response for applications that have not been uploaded to Google Play yet.

Reserved product IDs

- android.test.purchased
- android.test.canceled
- android.test.refunded
- android.test.item_unavailable

Testing with reserved product IDs

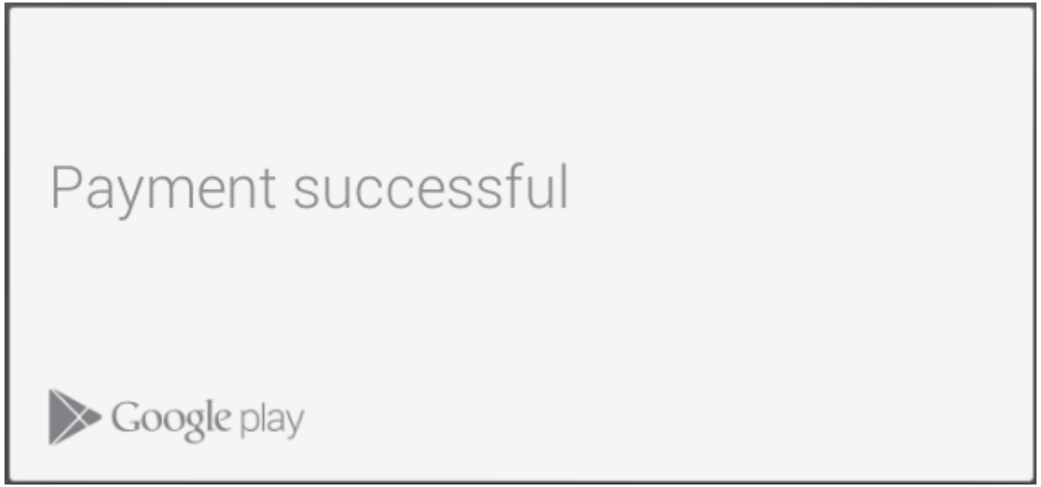
- Install signed apk on a test device. No emulators!
- Sign into device with your dev account
- Google Play version 2.3.4+ or MyApps app 5.0.12+
- Android 2.2+
- Run your app and purchase reserved product IDs
- Factory reset device for new purchase!



Sample Title

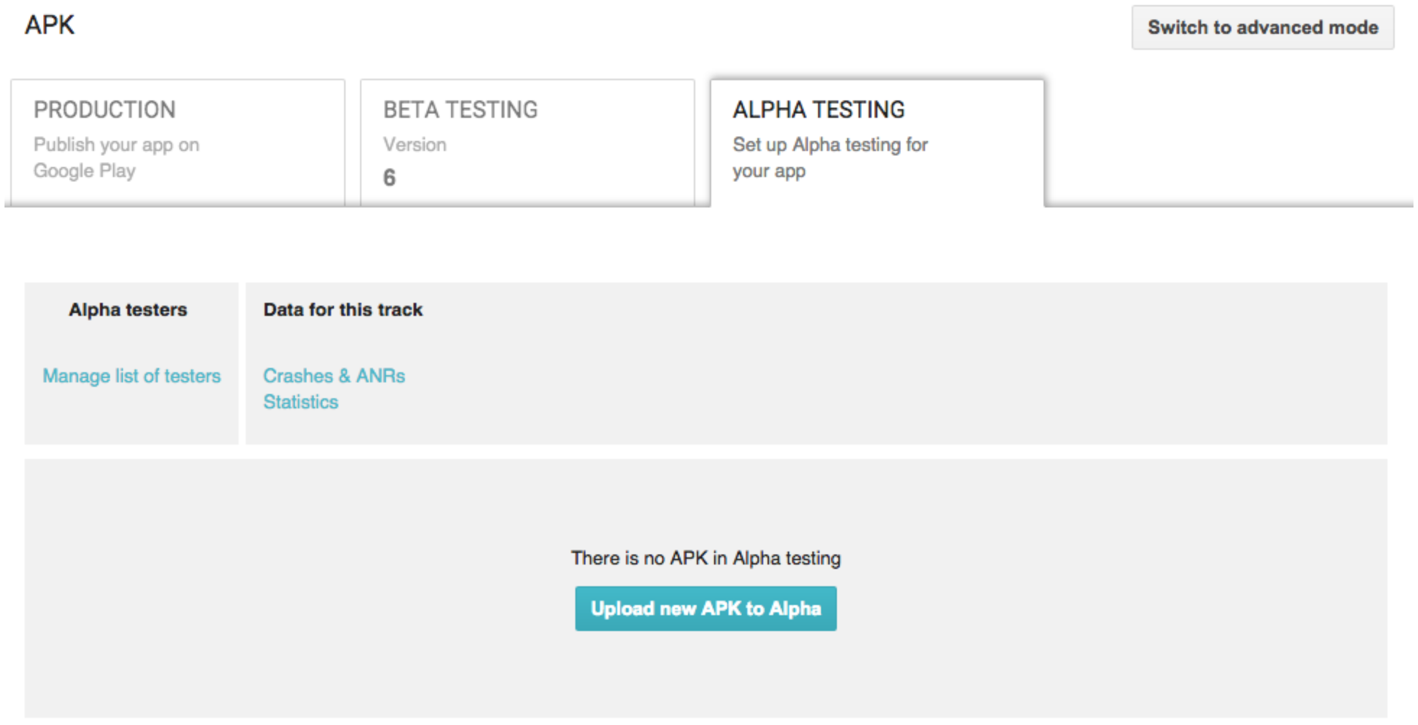
VISA xxx-FAKE

$0.99



Payment successful

Upload alpha apk. Hit Publish!



APK

PRODUCTION
Publish your app on Google Play

BETA TESTING
Version
6

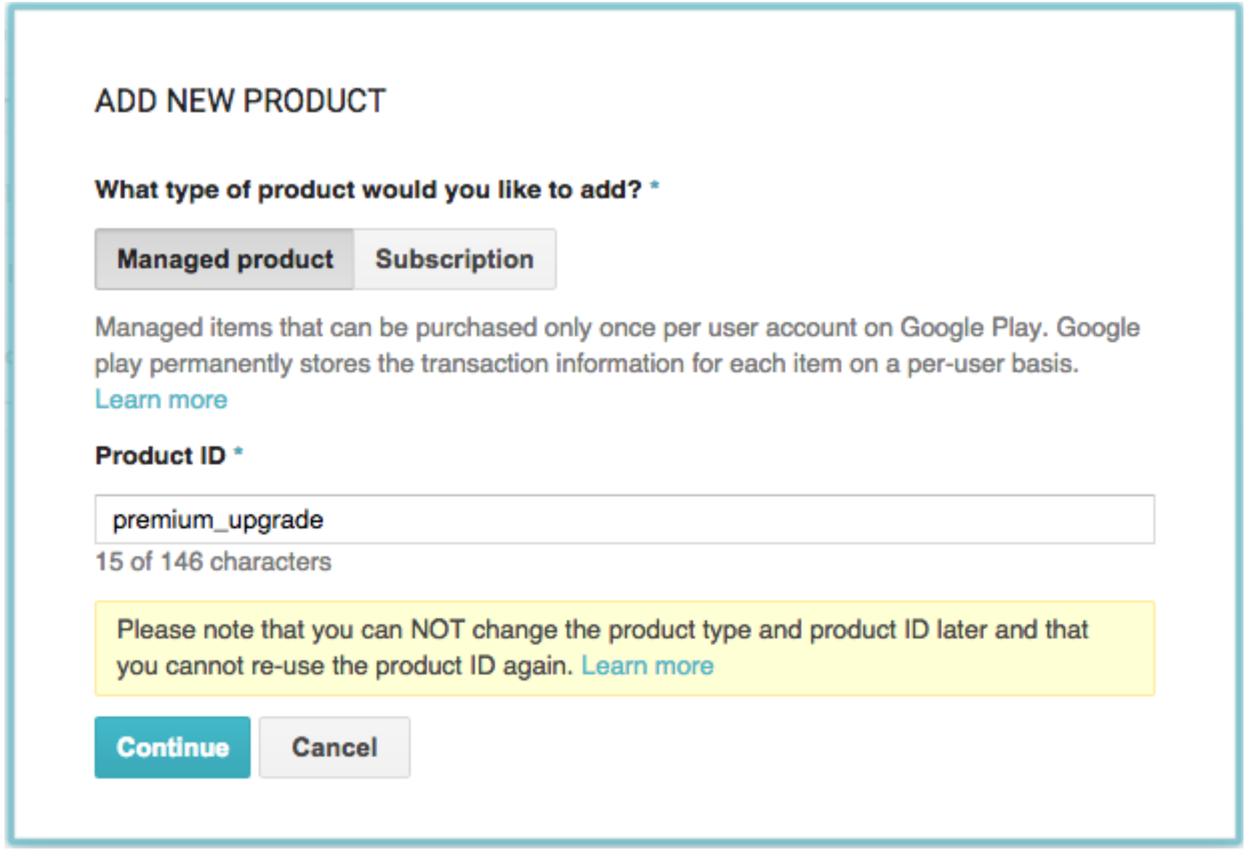
ALPHA TESTING
Set up Alpha testing for your app

Alpha testers
Manage list of testers

Data for this track
Crashes & ANRs
Statistics

There is no APK in Alpha testing

Setting up a new managed product on Developer Console



ADD NEW PRODUCT

What type of product would you like to add? *

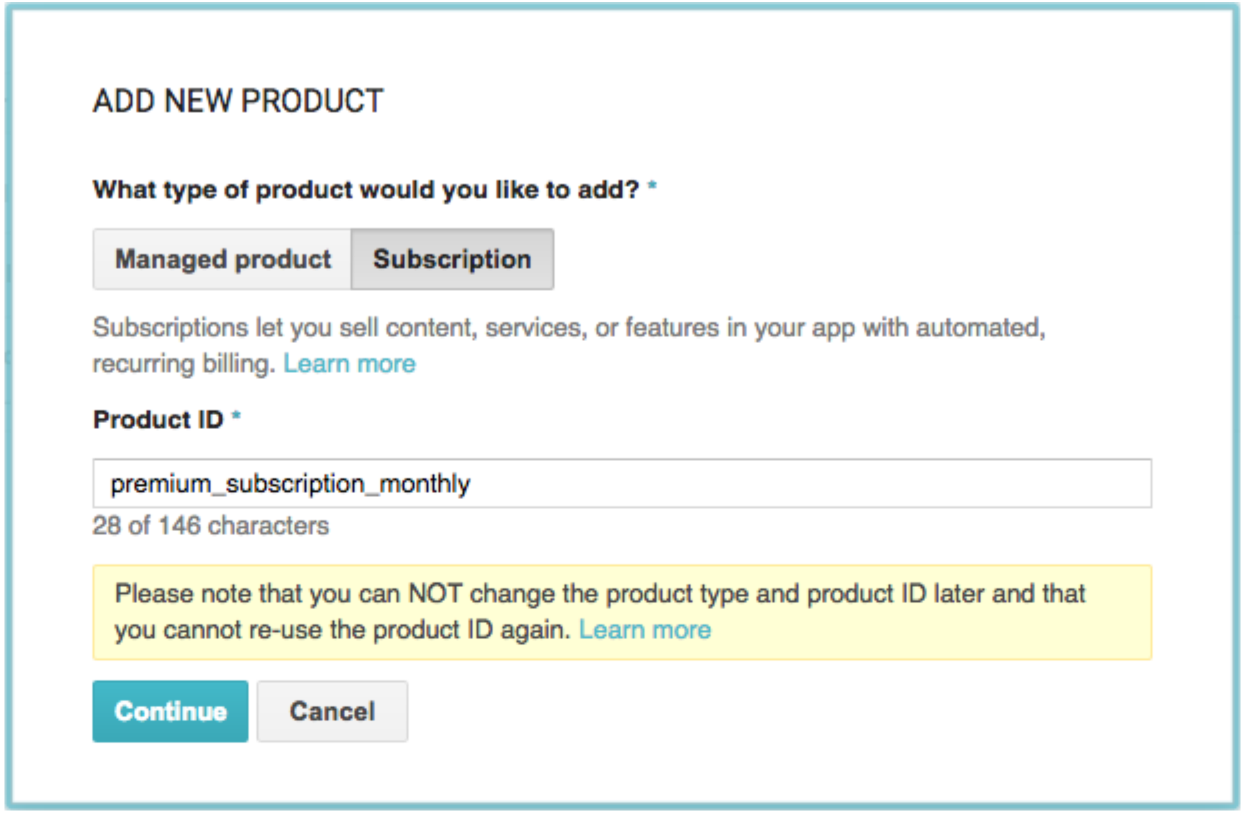
Managed items that can be purchased only once per user account on Google Play. Google play permanently stores the transaction information for each item on a per-user basis.
Learn more

Product ID *

premium_upgrade
15 of 146 characters

Please note that you can NOT change the product type and product ID later and that you cannot re-use the product ID again. Learn more

Setting up a new subscription product on Developer Console



ADD NEW PRODUCT

What type of product would you like to add? *

Subscriptions let you sell content, services, or features in your app with automated, recurring billing. Learn more

Product ID *

premium_subscription_monthly

28 of 146 characters

Please note that you can NOT change the product type and product ID later and that you cannot re-use the product ID again. Learn more

Testing with actual product IDs

- No draft apks: Upload signed apk to alpha/beta channel
- Add a real product to Developer Console
- Install signed apk on a test device. No emulators!
- Google Play version 2.3.4+ or MyApps app 5.0.12+
- Android 2.2+
- Purchase the real product with a real credit card
- Factory reset device for same product purchase!

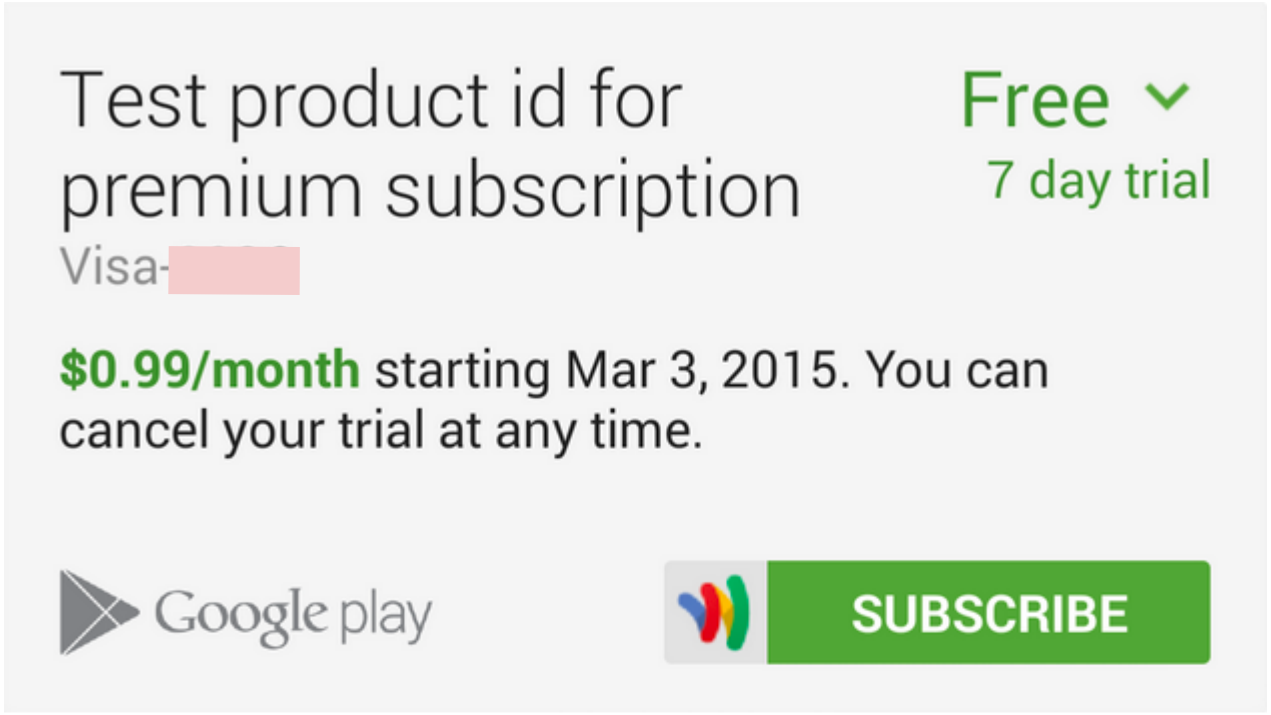
Testing subscriptions before Feb 2015

- There was no sandbox!
- Create a test $0.99 monthly product
- Purchase product with real credit card
- Refund & cancel order on the Google Wallet console
- Wait till end of subscription period to retest or factory reset device!

Testing subscriptions after Feb 2015

- Use android.test.purchased reserved product
- Buy the product
- Wait for 1 day to retest subscription

Valid subscription product purchase!



Test product id for premium subscription

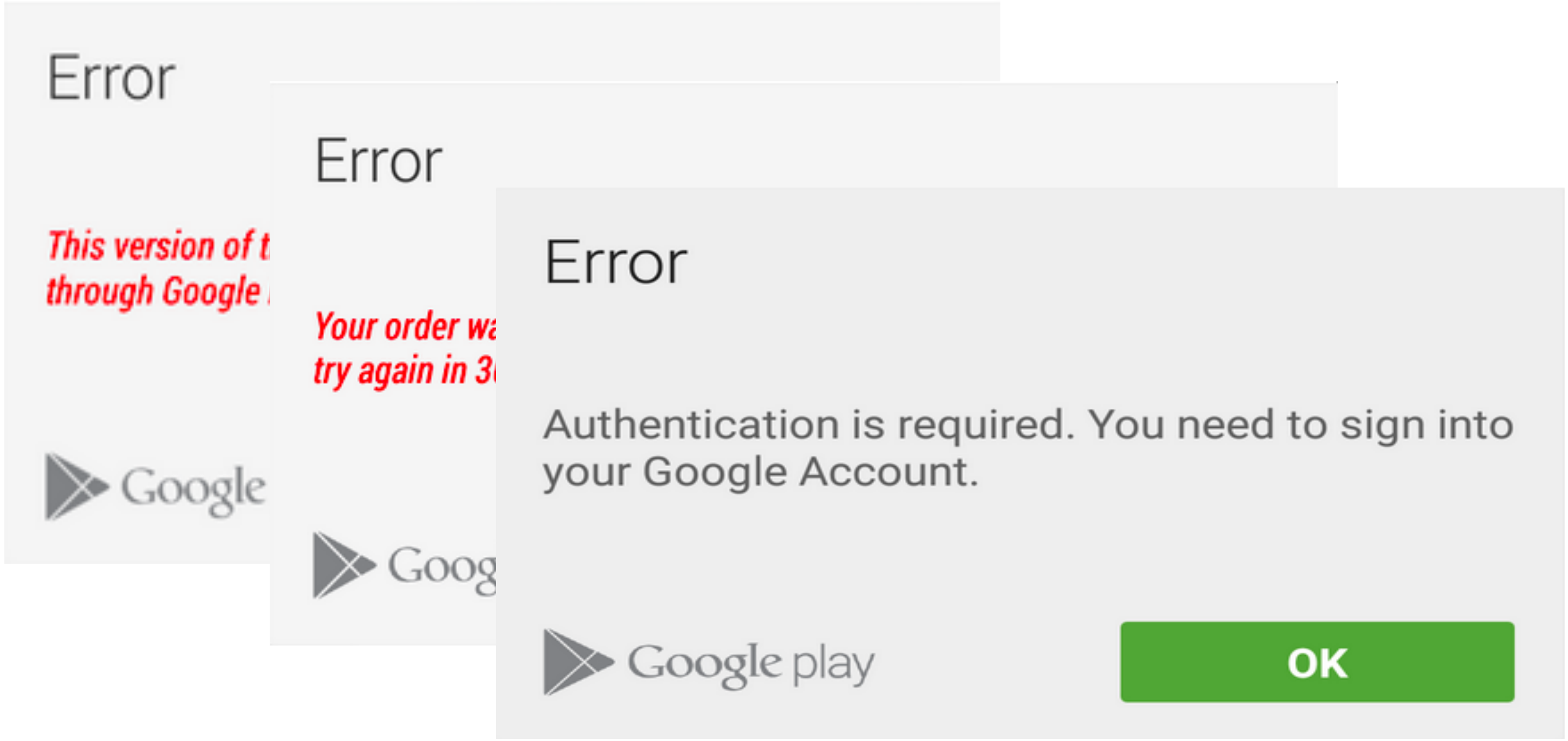
Free

7 day trial

Visa-[redacted]

$0.99/month starting Mar 3, 2015. You can cancel your trial at any time.

Common errors


Error

This version of t
through Google

Error

Your order wa
try again in 3

Error

Authentication is required. You need to sign into your Google Account.

Security

Client side workflow

Security

- Obfuscate code using Proguard

  `-keep class com.android.vending.billing.**`
- Protect unlocked/premium content
- Protect your Google Play Public Key
- Modify sample application code
- Use developer payload to uniquely identify user
- Signature verification on your server

Purchase data: developer payload

```
{ "orderId": "12999111111111111.145333335334137..0",
  "packageName": "com.sample.app",
  "productId": "test_inapp_upgrade",
  "purchaseTime": 1384834368656,
  "purchaseState": 0,
  "developerPayload": "abcdef",
  "purchaseToken": "fdgdghdjdp86dg",
  "autoRenewing":true }
```

Signature & Response code

Signature: jHCzy6MGlTNtOuF0nfYyiEyGw==

Response codes

0: Purchased

1: Canceled

2: Refunded

Signature verification

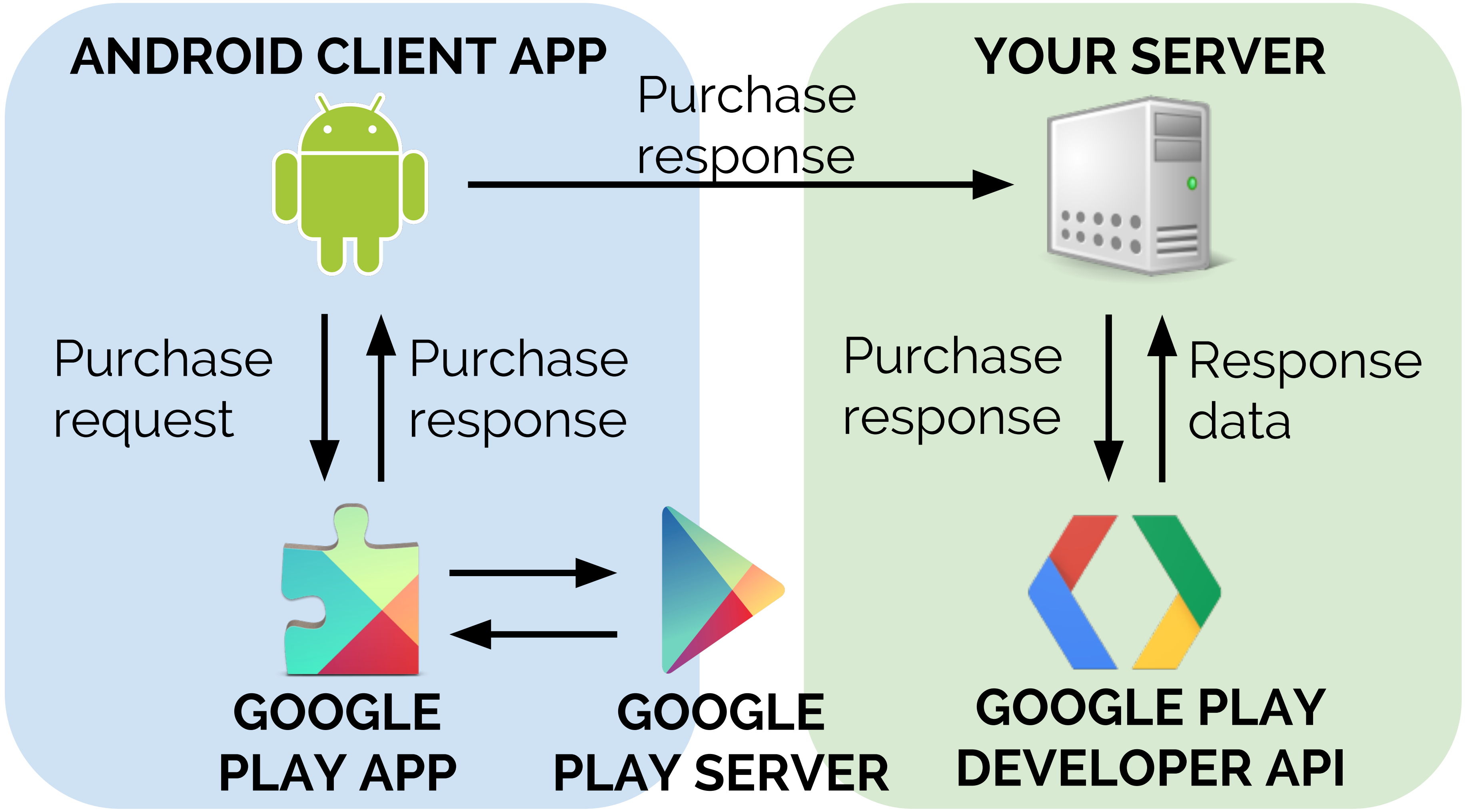
- Base64 Public Key
- Purchase data
- Signature
- Do this on client and server side!

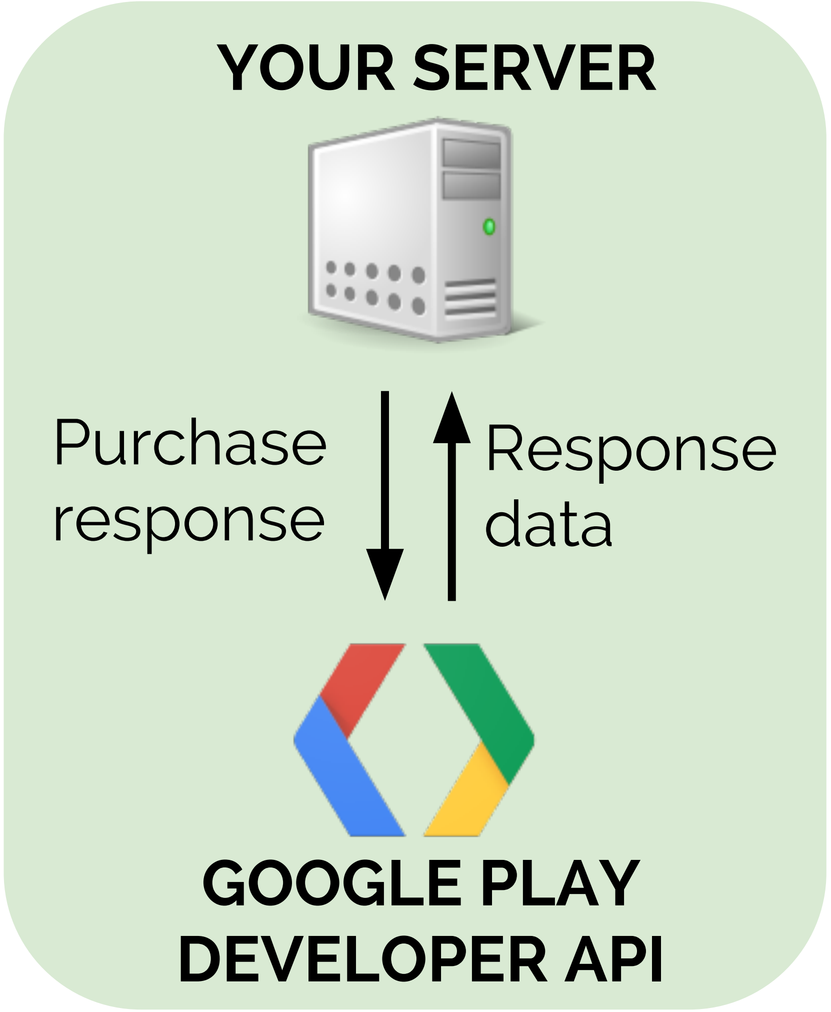
Decode base64 Public Key & verify signature

```
public static boolean verifyPurchase(String base64PublicKey,
    String purchaseData, String signature) {
        boolean verified = false;
        if (!TextUtils.isEmpty(signature)) {
            PublicKey key = Security.generatePublicKey(base64PublicKey);
            verified = Security.verify(key, purchaseData, signature);
        }
        return true;
    }
```

Google Play Developer API

Server side workflow


ANDROID CLIENT APP
YOUR SERVER
Purchase response
Purchase request
Purchase response
Purchase response
Response data
GOOGLE PLAY APP
GOOGLE PLAY SERVER
GOOGLE PLAY DEVELOPER API


YOUR SERVER
Purchase response
Response data
GOOGLE PLAY DEVELOPER API

Why use Google Play Developer API?

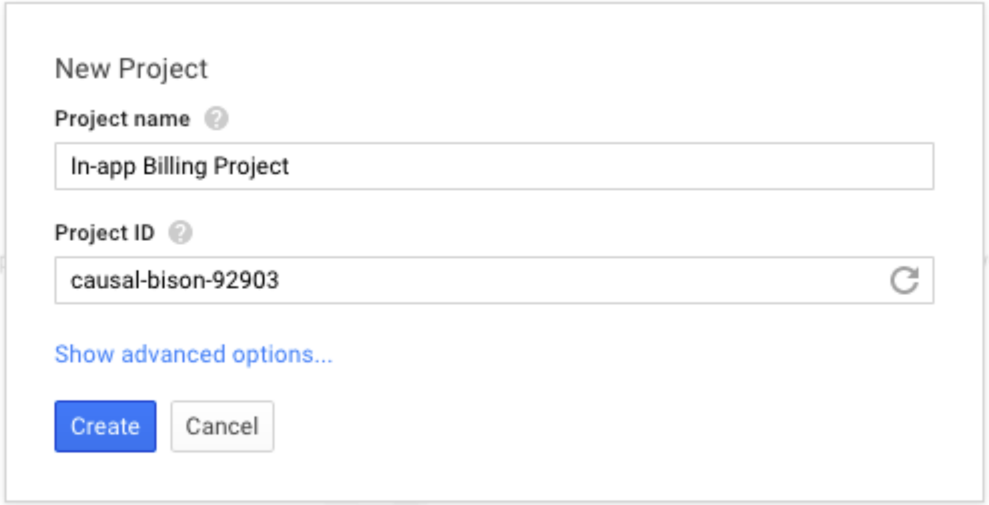
- 200,000 queries per day for free!
- Publishing API
  - Automates app distribution tasks
- Subscriptions & In-app purchases API
  - Get details about an in-app product
  - Retrieve details of a user's purchase
  - Insert/delete/update in-app products
  - List all the in-app and subscription products
  - Cancel/defer/refund/revoke subscriptions

Setting up Google Play Developer API

1. Set up an APIs Console Project
2. Create an OAuth 2.0 Client ID
3. Generate the refresh token
4. Generate an access token
5. Access the API

Step 1: Set up an APIs console project

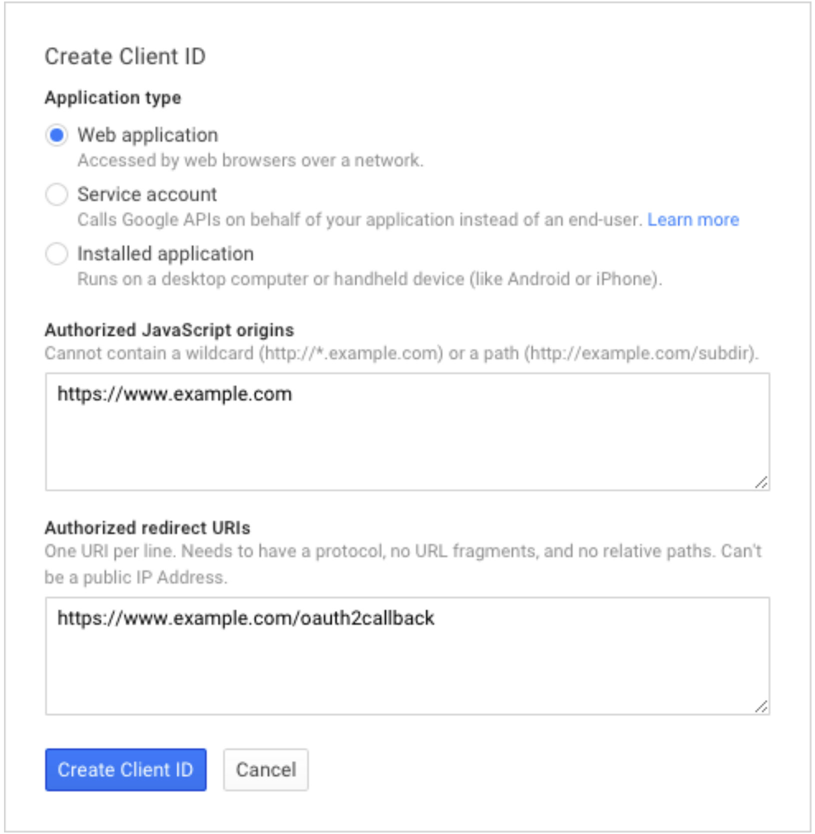
- https://console.developers.google.com/
- Set up a new project & turn on Developer API


New Project

Project name: In-app Billing Project

Project ID: causal-bison-92903

Step 2: Create an OAuth 2.0 Client ID



Create Client ID

Application type

[✓ Web application]
Accessed by web browsers over a network.

Service account
Calls Google APIs on behalf of your application instead of an end-user. Learn more

Installed application
Runs on a desktop computer or handheld device (like Android or iPhone).

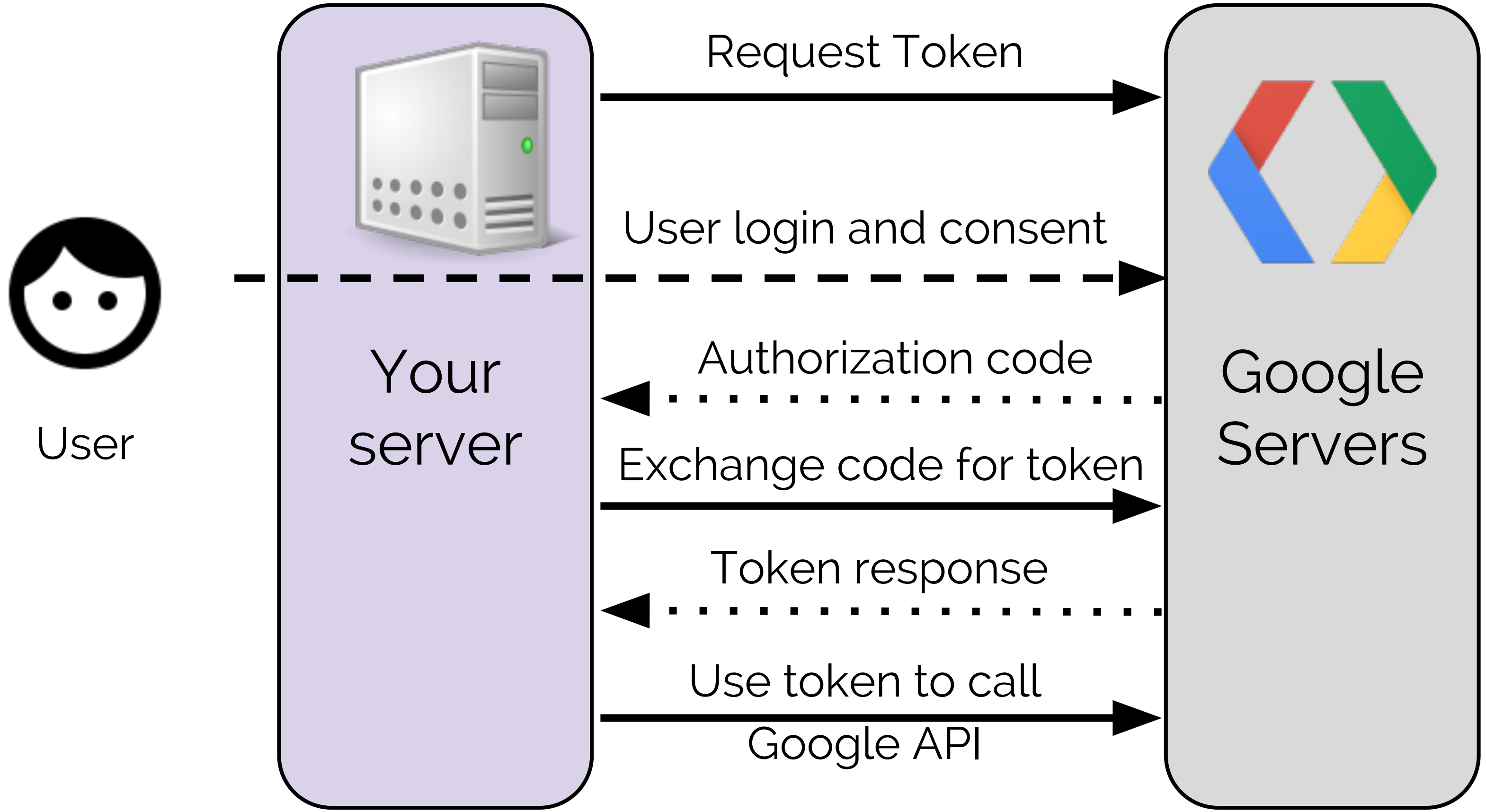
Authorized JavaScript origins
Cannot contain a wildcard (http://*.example.com) or a path (http://example.com/subdir).

https://www.example.com

Authorized redirect URIs
One URI per line. Needs to have a protocol, no URL fragments, and no relative paths. Can't be a public IP Address.

https://www.example.com/oauth2callback

Step 3: Generate refresh token with OAuth 2.0



Step 3: Generate the refresh token

1. Go to this URI while logged into the console account https://accounts.google.com/o/oauth2/auth?scope=https://www.googleapis.com/auth/androidpublisher&response_type=code&access_type=offline&redirect_uri=...&client_id=...
2. Hit Allow Access when prompted
3. The browser will be redirected to your redirect URI with a code parameter 4/eWdxD7b-YSQ5CNNb-c2KQx19.wp6198ti5Zc7dXOl0T3aRLxQmbwI

Step 3: Generate the refresh token

4. Send a POST request to https://accounts.google.com/o/oauth2/token

```
grant_type=authorization_code
code=<the code from the previous step>
client_id=<the client ID token created in the APIs Console>
client_secret=<the client secret corresponding to the client ID>
redirect_uri=<the URI registered with the client ID>
```

Step 3: Generate the refresh token

Get access token and request token in response

```
{
  "access_token" : "ya29.AHES3ZQ_Mc9TBWIbjW5ilJkXvLTeSl53ONa2",
  "token_type" : "Bearer",
  "expires_in" : 3600,
  "refresh_token" : "1/zaaHNytlC3BX7F2cfrHcqJEa3KoAHYeXES6nmho"
}
```

Step 4: Generate access token

Send a POST request to https://accounts.google.com/o/oauth2/token

grant_type=refresh_token

client_id=<the client ID token created in the APIs Console>

client_secret=<the client secret corresponding to the client ID>

refresh_token=<the refresh token from the previous step>

Step 4: Generate access token

Get access token in response

```
{
  "access_token" : "ya29.AHES3ZQ_Mc9TBWIbjW5ilJkXvLTeSl530Na2",
  "token_type" : "Bearer",
  "expires_in" : 3600,
}
```

Step 5: Access the API

```
{ "orderId": "12999111111111111.145333335334137..0",
  "packageName": "com.sample.app",
  "productId": "test_inapp_upgrade",
  "purchaseTime": 1384834368656,
  "purchaseState": 0,
  "developerPayload": "abcdef",
  "purchaseToken": "fdgdghdjdp86dg",
  "autoRenewing":true }
```

Purchases.products.get request

Checks the purchase and consumption status of an inapp item.

GET https://www.googleapis.com/androidpublisher/v2/applications/packageName/purchases/products/productId/tokens/token

Purchases.products.get response

```
{
  "kind": "androidpublisher#productPurchase",
  "purchaseTimeMillis": long,
  "purchaseState": integer,
  "consumptionState": integer,
  "developerPayload": string
}
```

Purchases.subscriptions.get request

Checks whether a user's subscription purchase is valid and returns its expiry time.

GET https://www.googleapis.com/androidpublisher/v2/applications/packageName/purchases/subscriptions/subscriptionId/tokens/token

Purchases.subscriptions.get response

```
{
  "kind": "androidpublisher#subscriptionPurchase",
  "startTimeMillis": long,
  "expiryTimeMillis": long,
  "autoRenewing": boolean
}
```

Google API Client libraries

- Easier to set up authentication and authorization
- Reduces OAuth 2.0 code
- Featured: Java, Python, .NET, PHP, Javascript
- Early-stage: Go, Dart, Ruby, Node-js, Objective-C
- https://developers.google.com/api-client-library/

Q&A

Yash Prabhu
Senior Software Engineer @dramafever

Twitter @yashvprabhu
Work at DramaFever Android Developer

References

- Google Play In-app Billing
- Training: Using Google Play to Distribute and Monetize
- Video: In-app Billing Version 3 (Google I/O 2013)
- Using OAuth 2.0 to access Google Play APIs
- Google API client libraries