

# Android IAB, 구현해보자!

GENG, NOLLEH

Dev Fest W Korea

# 소개

- 안드로이드 개발 경력 3년차
- 벅스 For Tablet , Equalizer Plugin..
- 현재는 피망플러스 SDK 개발

# 소개

## □ 개인개발\_ 터치터치사다리

### Android 애플리케이션



터치 터치 사다리 ( ...

GENG

★★★★★ (177)

설치됨



Hola! Dice!

GENG

★★★★★ (9)

설치됨



어플 관리 ( MyApp...

GENG

★★★★★ (12)

설치됨



날 사랑하긴 하는...

GENG

★★★★★ (4)

설치됨



모니터 크래셔 (Mo...

GENG

★★★★★ (5)

설치됨



# 여는 말

- 이걸 어디에 쓸까 ?
  - 구글 플레이 스토어의 부분 유료화 수단은 구글이어야 한다.
- 30% 수수료 ..

# 목차

- Read Me, For IAB
- Version 2
- Version 2 안전하게 구현하려면 ?
- Version 3 (에서 달라진 것)
- Version 3 안전하게 구현하려면 ?
- 이걸 좀..



Read Me, For IAB

# Read Me, For IAB

- 상품의 종류
  - Managed ( 영구 상품 )
  - UnManaged ( 소멸성 상품 )

# Read Me, For IAB

- 'Interprocess Communication'
  - 다른 프로세스의 메모리에 일반적으로 접근 불가
  - 접근 가능한 인터페이스. `aidl` 파일이 프로젝트에 위치해 있어야 한다.



# Read Me, For IAB

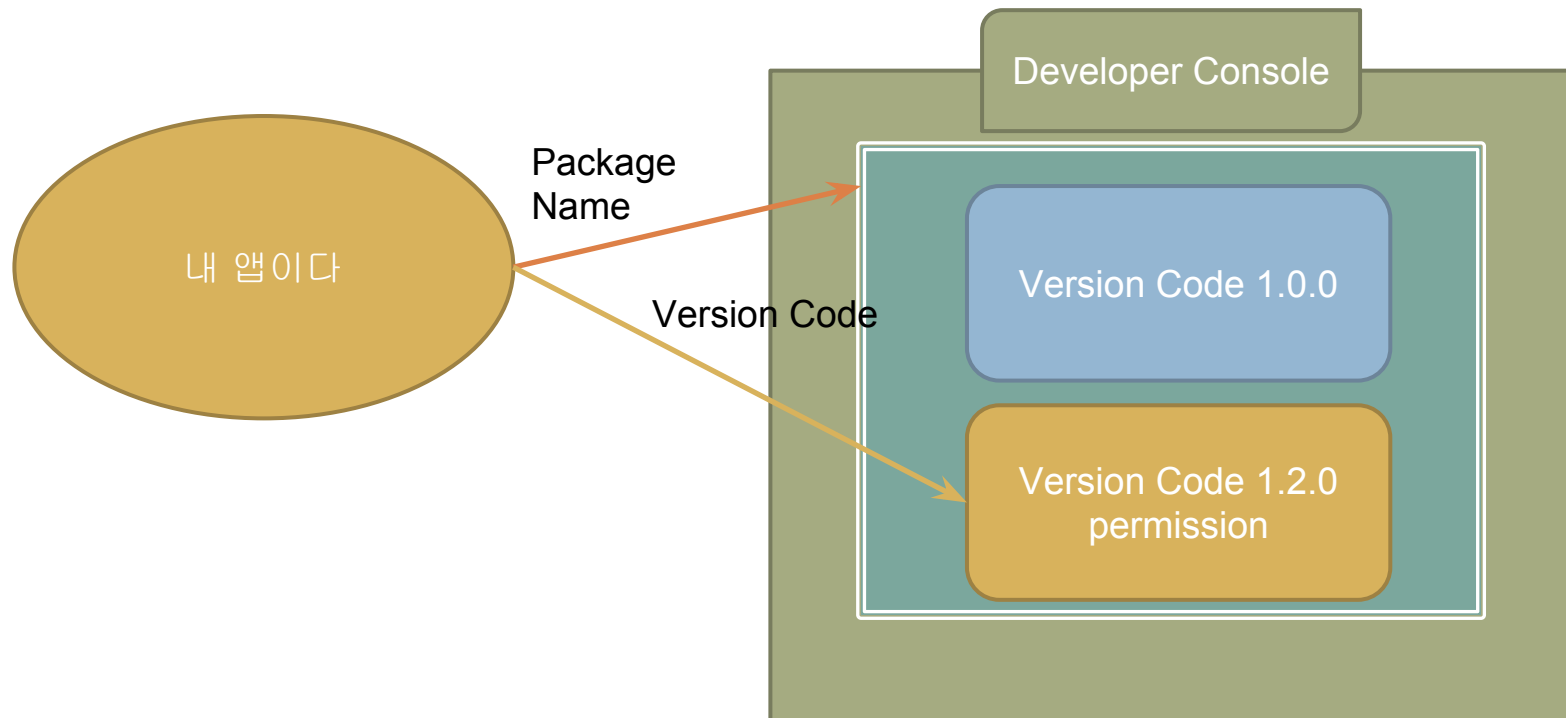
- GOOGLE\_DEVELOPER\_ERROR
  - 개발자 콘솔에 APK Upload
    - Permission 넣어서

```
<!-- VERY IMPORTANT! Don't forget this permission, or in-app billing won't work. -->  
<uses-permission android:name="com.android.vending.BILLING" />
```

- 테스트용과 동일한 키로 Signing 해서
- versionCode 맞춰서

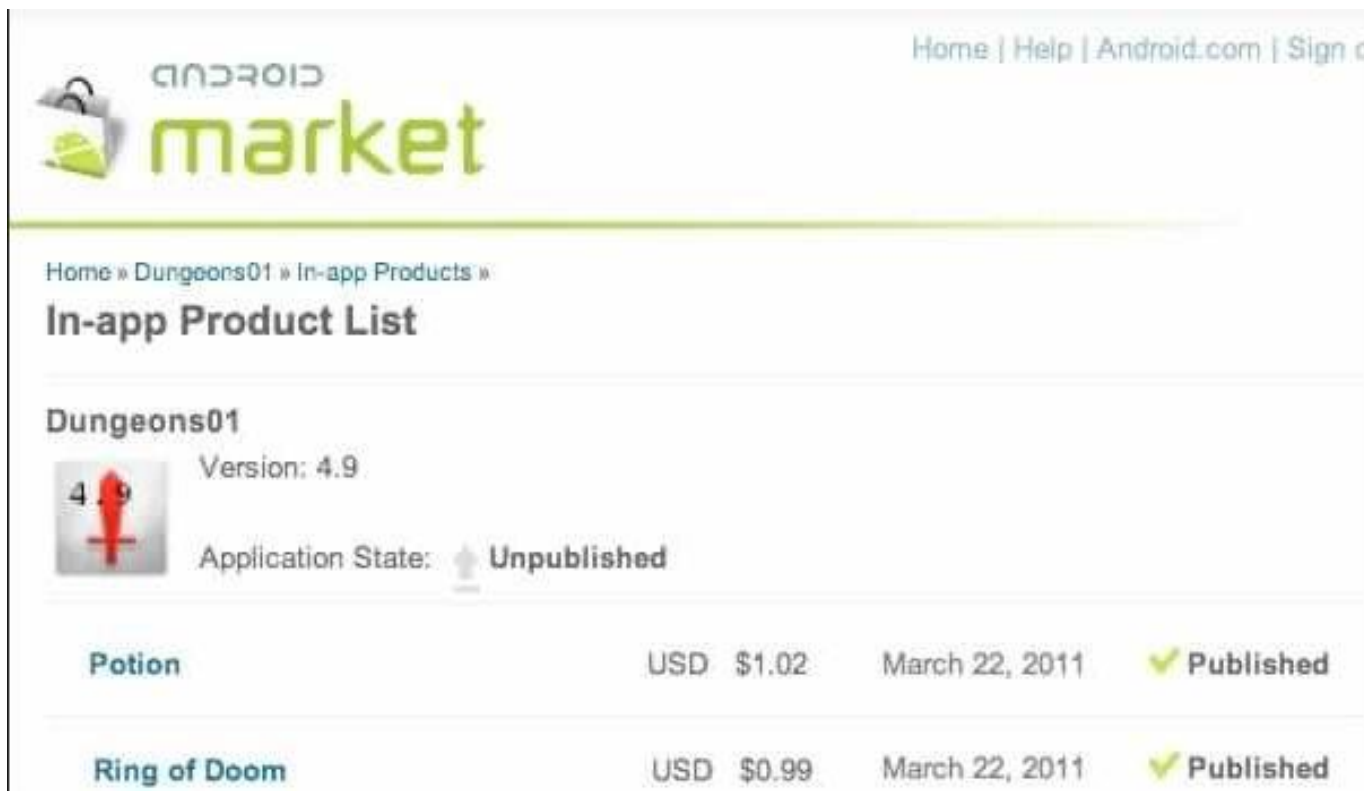
# Read Me, For IAB

- GOOGLE\_DEVELOPER\_ERROR
  - Permission, signing, versionCode



# Read Me, For IAB

- 상품은 게시

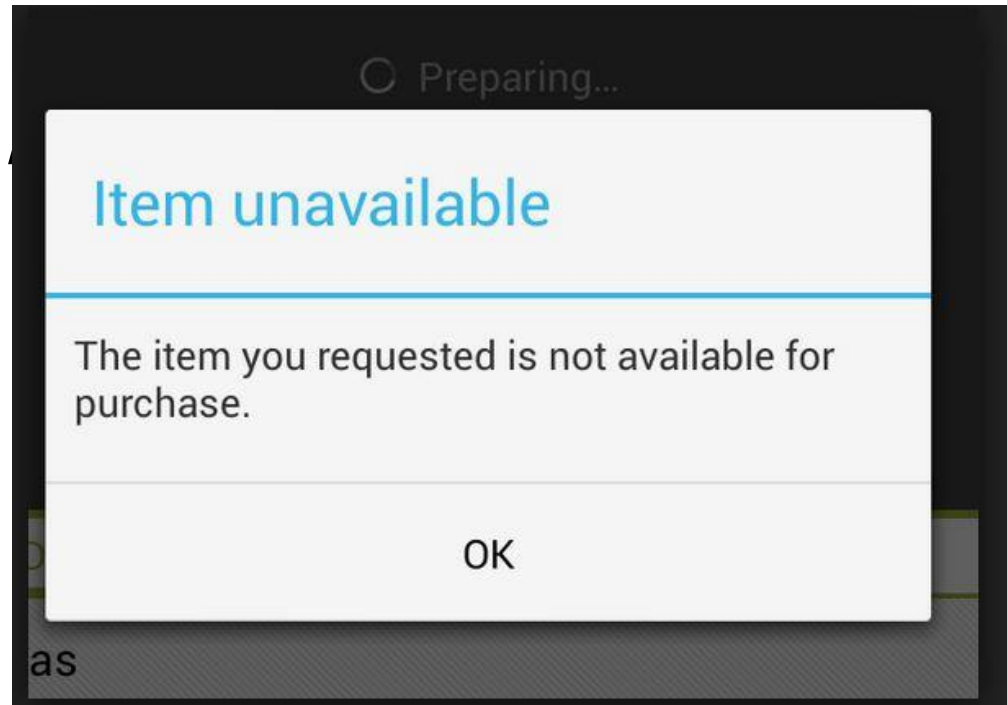


The screenshot displays the Android Market interface for the app 'Dungeons01'. The page title is 'In-app Product List'. The app's version is 4.9, and its application state is 'Unpublished'. Below this, a table lists two in-app products: 'Potion' and 'Ring of Doom'. Both products are priced in USD and were published on March 22, 2011. The 'Potion' is priced at \$1.02 and the 'Ring of Doom' at \$0.99. Both have a green checkmark and the word 'Published' next to them.

Product Name	Price	Release Date	Status
Potion	USD \$1.02	March 22, 2011	Published
Ring of Doom	USD \$0.99	March 22, 2011	Published

# Read Me, For IAB

- GOOGLE\_ITEM\_UNAVAILABLE
  - Test Account \_ publish 계정이 아니어야 함.
  - As Primary Account ( 기본계정 )
- 실제로 과금됨



# Read Me, For IAB

- 새 디자인부터 \_ App 별로 Public Key 관리
  - 계속 verify 실패시 확인

APK

스토어 목록

가격 및 배포

인앱 제품

서비스 및 API

애플리케이션의 GCM 통계에 액세스하려면 GCM API 키를 입력하여 이 애플리케이션이 사용하는 GCM 발신자 ID와 연결해야 합니다.

앱이 게시되면 통계 페이지에서 사용자 애플리케이션에 대한 GCM 통계에 액세스할 수 있습니다.

발신자 ID 연결

라이선스 및 인앱 결제

라이선스 키를 사용하여 앱의 무단 배포를 방지할 수 있습니다. 이 키를 사용하여 이 애플리케이션용 라이선스 키

바이너리에 포함하려는 Base64 인코딩 RSA 공개 키입니다. 공백을 삭제해 주세요.

~~-----BEGIN PUBLIC KEY-----~~  
~~MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA...~~  
~~-----END PUBLIC KEY-----~~

비밀



Version 2

# 특이사항

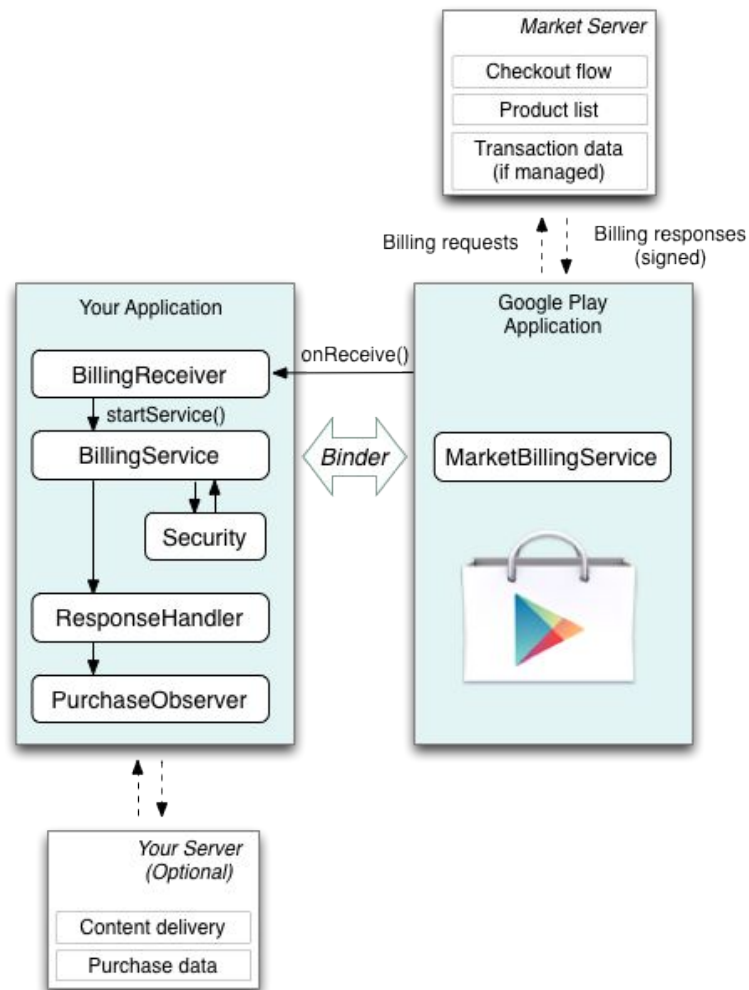
- 결제 후 비동기로 대기 ( In App Notify )
  - 다른 스토어의 경우 동기
  - 환불시에도 들어온다.
- 효력지급에 실패했더라도 괜찮아.
  - Confirm 을 보내지 않으면 계속, 온다.
  - 효력지급이 실패하더라도 다음에 메시지가 들어오므로 다음에 또 시도가능.

# 특이사항

- 기구매 managed 상품 구매시도시
  - ErrorCode가 따로 없이, GOOGLE\_ERROR
  - 왜 에러가 났는지 정확하게 알 수가 없다.



# Version 2

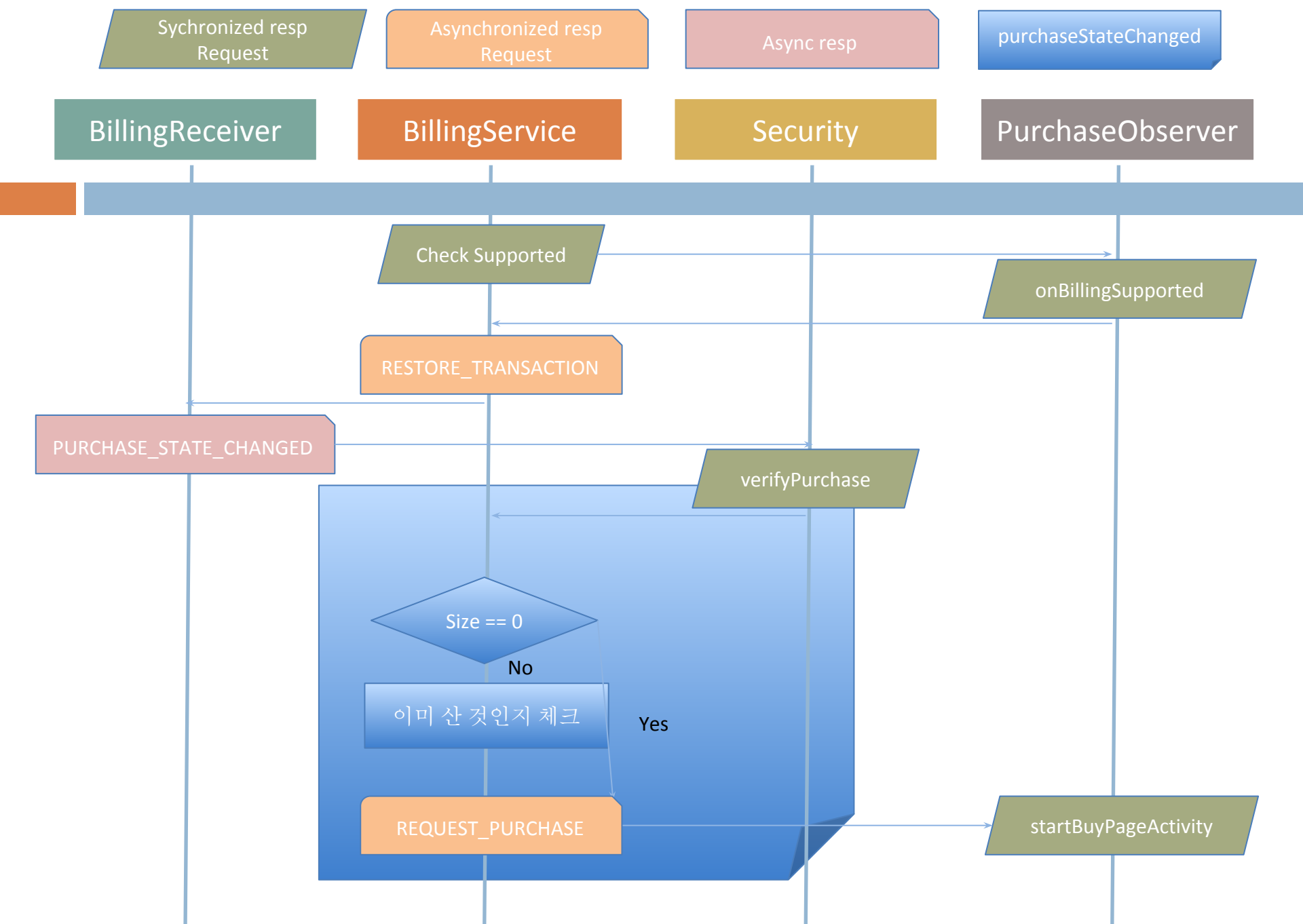


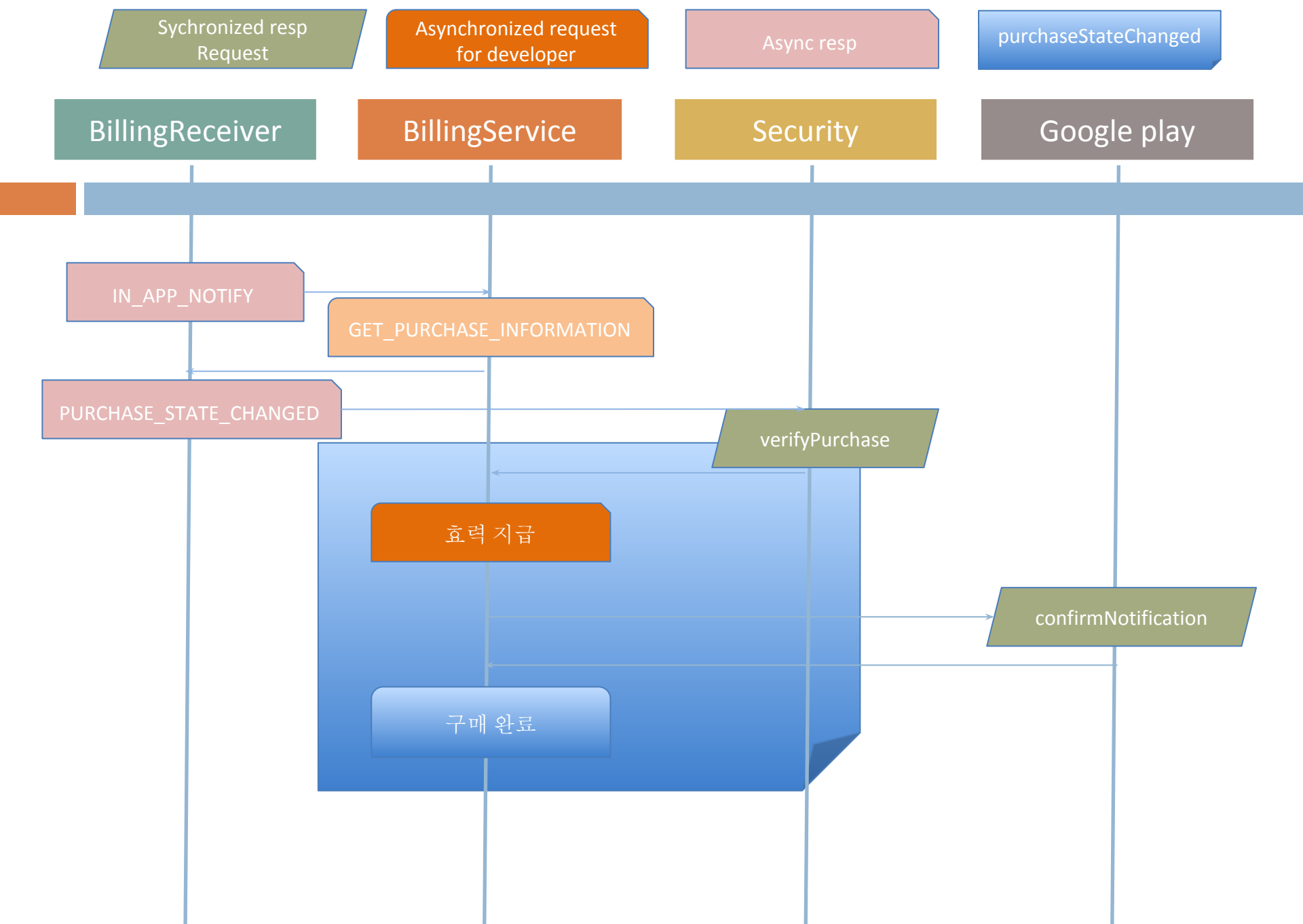
# Version 2\_REQUEST

- CHECK\_BILLING\_SUPPORTED
- RESTORE\_TRANSACTION
  - 구매한 Managed 상품 리스트
- REQUEST\_PURCHASE
  - 구매요청
- GET\_PURCHASE\_INFORMATION
  - 상품 구매 상세 정보 요청
- CONFIRM\_NOTIFICATION

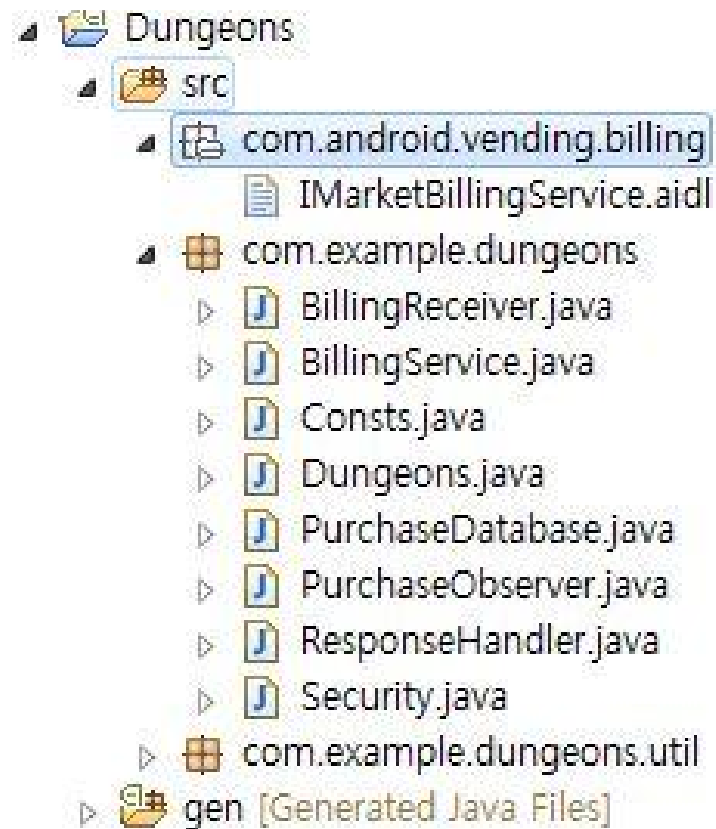
# Version 2\_RESPONSE

- IN\_APP\_NOTIFY
  - 상태 변경됨
- PURCHASE\_STATE\_CHANGED
  - 상태 변경 상세정보 받음
- RESPONSE\_CODE
  - 요청 응답





# Version 2



# Version 2

## □ AndroidManifest.xml

```
19         <service android:name="BillingService" />
20
21     <receiver android:name="BillingReceiver">
22         <intent-filter>
23             <action android:name="com.android.vending.billing.IN_APP_NOTIFY" />
24             <action android:name="com.android.vending.billing.RESPONSE_CODE" />
25             <action android:name="com.android.vending.billing.PURCHASE_STATE_CHANGED" />
26         </intent-filter>
27     </receiver>
```

- IN\_APP\_NOTIFY : 상태변경시 notifyId string Extra 와 함께. (service에서 곧바로 GetPurchaseInformation을 보낸다.)
- RESPONSE\_CODE : 구글로 보내는 어떤 메시지든지 비동기로 response가 필수적으로. (요청 에러등은 여기로 온다.)
- PURCHASE\_STATE\_CHANGED : GetPurchaseInformation 의 Response

# Version 2

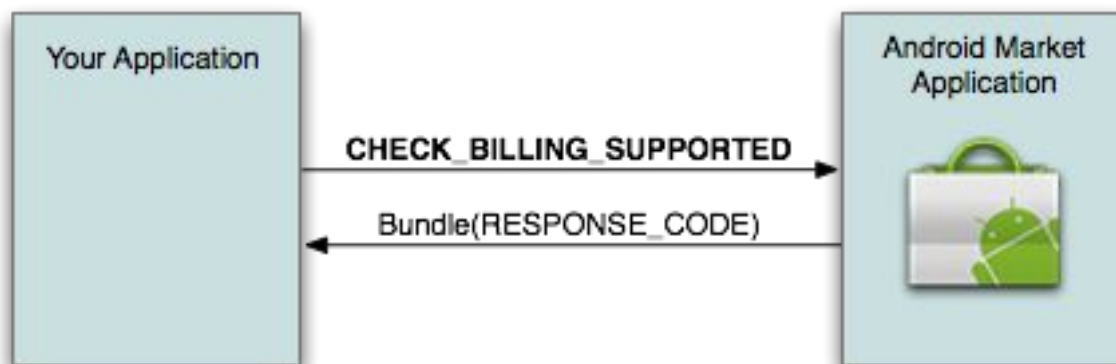
## □ Aidl

```
--  
17 package com.android.vending.billing;  
18  
19 import android.os.Bundle;  
20  
21 interface IMarketBillingService {  
22     /** Given the arguments in bundle form, returns a bundle for results. */  
23     Bundle sendBillingRequest(in Bundle bundle);  
24 }  
25
```



# Flow

## □ 환경 체크 ( ckBillingService )



```
if (!mBillingService.checkBillingSupported(Consts.ITEM_TYPE_SUBSCRIPTION)) {  
    showDialog(DIALOG_SUBSCRIPTIONS_NOT_SUPPORTED_ID);  
}  
  
/**  
 * Checks if in-app billing is supported.  
 * @param itemType Either Consts.ITEM_TYPE_INAPP or Consts.ITEM_TYPE_SUBSCRIPTION, indicating  
 *                  type of item support is being checked for.  
 * @return true if supported; false otherwise  
 */  
public boolean checkBillingSupported(String itemType) {  
    return new CheckBillingSupported(itemType).runRequest();  
}
```

# Flow

## □ 환경 체크

```
@Override
protected long run() throws RemoteException {
    Bundle request = makeRequestBundle("CHECK_BILLING_SUPPORTED");
    if (mProductType != null) {
        request.putString(Consts.BILLING_REQUEST_ITEM_TYPE, mProductType);
    }
    Bundle response = mService.sendBillingRequest(request);
    int responseCode = response.getInt(Consts.BILLING_RESPONSE_RESPONSE_CODE);
    if (Consts.DEBUG) {
        Log.i(TAG, "CheckBillingSupported response code: " +
            ResponseCode.valueOf(responseCode));
    }
    boolean billingSupported = (responseCode == ResponseCode.RESULT_OK.ordinal());
    ResponseHandler.checkBillingSupportedResponse(billingSupported, mProductType);
    return Consts.BILLING_RESPONSE_INVALID_REQUEST_ID;
}
```

# Flow

## □ 구매요청 > 구매 PendingIntent 리턴

```
@Override
protected long run() throws RemoteException {
    Bundle request = makeRequestBundle("REQUEST_PURCHASE");
    request.putString(Consts.BILLING_REQUEST_ITEM_ID, mProductId);
    request.putString(Consts.BILLING_REQUEST_ITEM_TYPE, mProductType);
    // Note that the developer payload is optional.
    if (mDeveloperPayload != null) {
        request.putString(Consts.BILLING_REQUEST_DEVELOPER_PAYLOAD, mDeveloperPayload);
    }
    Bundle response = mService.sendBillingRequest(request);
    PendingIntent pendingIntent
        = response.getParcelable(Consts.BILLING_RESPONSE_PURCHASE_INTENT);
    if (pendingIntent == null) {
        Log.e(TAG, "Error with requestPurchase");
        return Consts.BILLING_RESPONSE_INVALID_REQUEST_ID;
    }

    Intent intent = new Intent();
    ResponseHandler.buyPageIntentResponse(pendingIntent, intent);
    return response.getLong(Consts.BILLING_RESPONSE_REQUEST_ID,
        Consts.BILLING_RESPONSE_INVALID_REQUEST_ID);
}
```

# Flow

## □ 유저 결제



# Flow

- IN\_APP\_NOTIFY
  - Notify Id 를 갖고 옴.
  - 이 아이디로 BillingService에서  
GET\_PURCHASE\_INFORMATION 실행.

# Flow

- GET\_PURCHASE\_INFORMATION
  - 해당 Notify Id에 해당하는 구매 상세정보 요청

```
@Override
protected long run() throws RemoteException {
    mNonce = Security.generateNonce();

    Bundle request = makeRequestBundle("GET_PURCHASE_INFORMATION");
    request.putLong(Consts.BILLING_REQUEST_NONCE, mNonce);
    request.putStringArray(Consts.BILLING_REQUEST_NOTIFY_IDS, mNotifyIds);
    Bundle response = mService.sendBillingRequest(request);
    logResponseCode("getPurchaseInformation", response);
    return response.getLong(Consts.BILLING_RESPONSE_REQUEST_ID,
        Consts.BILLING_RESPONSE_INVALID_REQUEST_ID);
}
```

# Flow


- PURCHASE\_STATE\_CHANGED
  - GET\_PURCHASE\_INFORMATION 의 비동기 응답

```
{ "nonce" : 1836535032137741465,  
  "orders" :  
    [{ "notificationId" : "android.test.purchased",  
      "orderId" : "transactionId.android.test.purchased",  
      "packageName" : "com.example.dungeons",  
      "productId" : "android.test.purchased",  
      "developerPayload" :  
        "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ",  
      "purchaseTime" : 1290114783411,  
      "purchaseState" : 0,  
      "purchaseToken" : "rojeslcdyyiapnqcynkjyyjh" }]  
}
```

# Flow

- Server Side 검증, 효력지급
  - App은 변조가 가능
- CONFIRM\_NOTIFICATION





Version 2 안전하게 구현하려  
면?

# 효력지급

- 당연한 이야기
  - CONFIRM\_NOTIFICATION
    - 반드시 효력지급이 완료가 된 후에 !
    - 누락시 계속해서 비동기 응답은 점점 .. 느려진다.
      - Few sec , min , days, week.....
  - BillingService가 언제든지 구동될 수 있다.
    - 앱이 실행중이 아닐때도 처리할수 있게.

# 효력지급

- 당연하지 않은 이야기
  - IN\_APP\_NOTIFY 를 저장해 두기.

# 보안

- Developer Payload
  - Identify 할 수 있는 token
- Random nonce
  - RESTORE\_TRANSACTION /  
GET\_PURCHASE\_INFORMATION
- Server Side 검증
- 이미 처리된 order Id 는 아닌지

# 문제점

- PURCHASE\_STATE\_CHANGED 의 중복
  - IN\_APP\_NOTIFY 는 각 밀려있는 데이터마다 들어옴
  - 들어올 때마다 GET\_PURCHASE\_INFORMATION 호출
  - GET~은 리스트로 넘어옴. ( 따라서 4개의 데이터가 밀려있으면 4번옴 )
  - 버그는 아니지만 Buggy한 상황



Version 3

# Version 3

- 쉽게
  - Service 날렸다
- 빠르게
  - Local caching
- 편하게
  - 기존에 없던 상품 정보 질의 (ID는 알고 있어야.)

# Version 3

- 개발자가 신경 쓸 필요 없어진 것
  - Service, Receiver, Observer !
    - 왜 ? 어려우니까 !!
    - labHelper 에서 바로 구글플레이에 붙는다.
      - `mContext.bindService(new Intent("com.android.vending.billing.InAppBillingService.BIND"),  
mServiceConn, Context.BIND_AUTO_CREATE);`



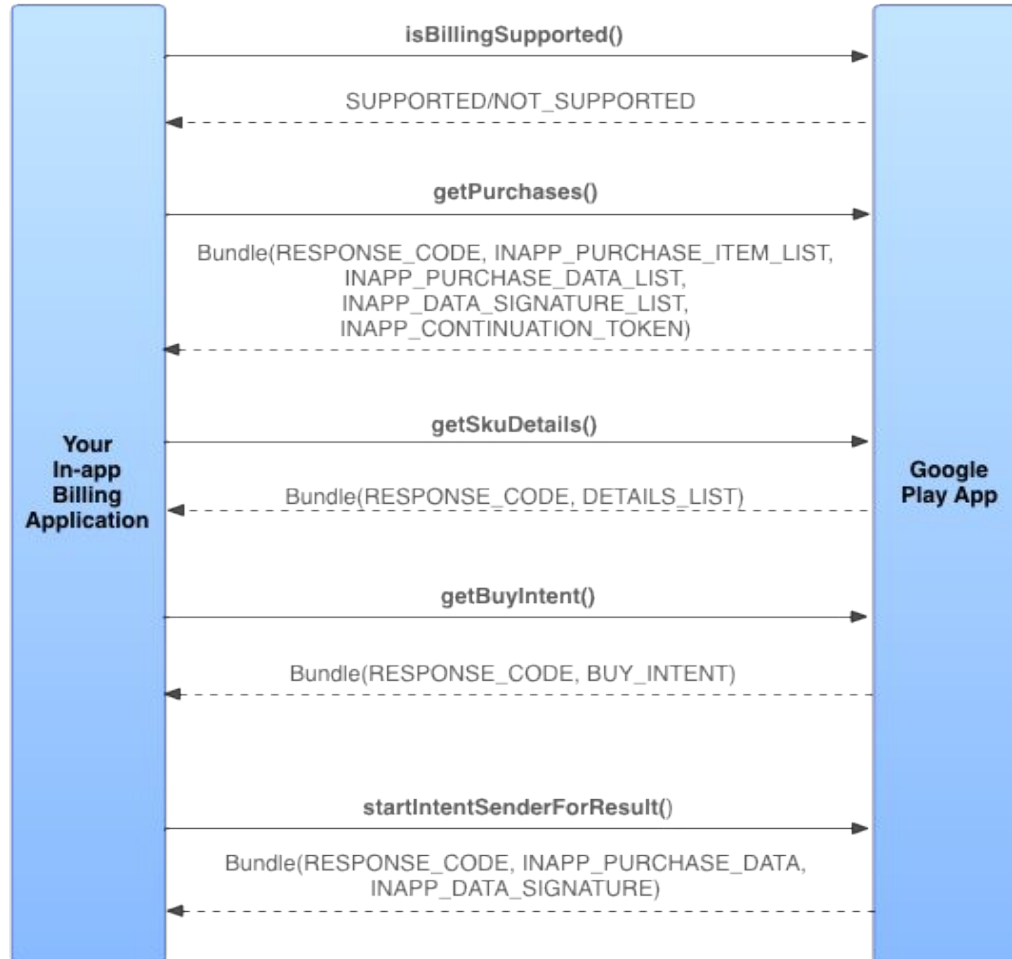
# Version 3

- 동기적 체크
  - 대신 필연적으로 1회만 온다. (onActivityResult)
  - 구글의 서버 부하가 줄었겠네요.
  
- 기구매 체크 제거
  - `RESULT_ITEM_ALREADY_OWNED = 7` - 에러코드 추가

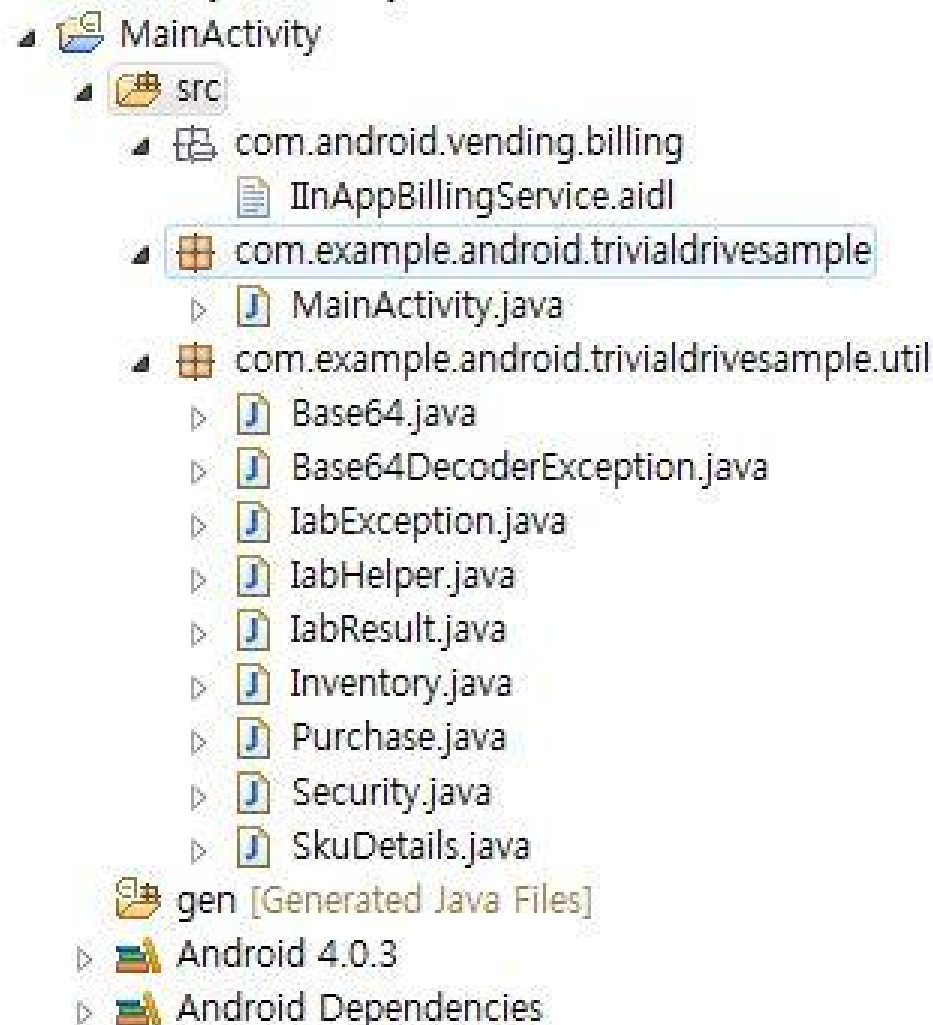
# Version 3

- 모두 managed
  - User별로 무슨 상품을 샀는지 모두 기억해 두고 있음.
  - Consume 시 제거
- 영구상품은 Consume 을 호출하지 않으면 됨.

# Version 3



# Version 3



# Version 3

- AndroidManifest.xml

```
<!-- VERY IMPORTANT! Don't forget this permission, or in-app billing won't work. -->  
<uses-permission android:name="com.android.vending.BILLING" />
```

# Version 3

## □ Aidl

```
interface IInAppBillingService {  
    int isBillingSupported(int apiVersion, String packageName, String type);  
    Bundle getSkuDetails(int apiVersion, String packageName, String type, in Bundle skusBundle);  
    Bundle getBuyIntent(int apiVersion, String packageName, String sku, String type,  
        String developerPayload);  
    Bundle getPurchases(int apiVersion, String packageName, String type, String continuationToken);  
    int consumePurchase(int apiVersion, String packageName, String purchaseToken);  
}
```

# IAB Helper?

- 구글플레이 서비스와의 통신 담당
- `purchaseObserver` 등의 통으로 된 응답방식 대신 각 메서드의 파라미터로 `listener`를 받는 형태
  - 이해가 더 쉬워짐

# Flow

- IAB Helper Start SetUp
  - Service Connection
  - Check Billing Supported

```
@Override
public void onServiceConnected(ComponentName name, IBinder service) {
    logDebug("Billing service connected.");
    mService = IInAppBillingService.Stub.asInterface(service);
    String packageName = mContext.getPackageName();
    try {
        logDebug("Checking for in-app billing 3 support.");

        // check for in-app billing v3 support
        int response = mService.isBillingSupported(3, packageName, ITEM_TYPE_INAPP);
        if (response != BILLING_RESPONSE_RESULT_OK) {
            if (listener != null) listener.onIabSetupFinished(new IabResult(response,
                "Error checking for billing v3 support."));
        }
    }
}
```



# Flow

## □ queryInventory

```
public Inventory queryInventory(boolean querySkuDetails, List<String> moreItemSkus,  
                               List<String> moreSubsSkus) throws IabException {  
    checkSetupDone("queryInventory");  
    try {  
        Inventory inv = new Inventory();  
        int r = queryPurchases(inv, ITEM_TYPE_INAPP);  
        if (r != BILLING_RESPONSE_RESULT_OK) {  
            throw new IabException(r, "Error refreshing inventory (querying owned items).");  
        }  
  
        if (querySkuDetails) {  
            r = querySkuDetails(ITEM_TYPE_INAPP, inv, moreItemSkus);  
            if (r != BILLING_RESPONSE_RESULT_OK) {  
                throw new IabException(r, "Error refreshing inventory (querying prices of items).");  
            }  
        }  
    }  
}
```

# Flow

## □ queryPurchases

```
do {  
    logDebug("Calling getPurchases with continuation token: " + continueToken);  
    Bundle ownedItems = mService.getPurchases(3, mContext.getPackageName(),  
        itemType, continueToken);  
  
    int response = getResponseCodeFromBundle(ownedItems);  
    logDebug("Owned items response: " + String.valueOf(response));  
    if (response != BILLING_RESPONSE_RESULT_OK) {
```

# Flow

- LaunchPurchaseFlow
  - PendingIntent 얻을 때 response 한번 체크
  - startIntentSenderForResult

# Flow

## □ 유저 결제



# Flow

- HandleActivityResult

- **if (resultCode == Activity.RESULT\_OK && *responseCode* ==**

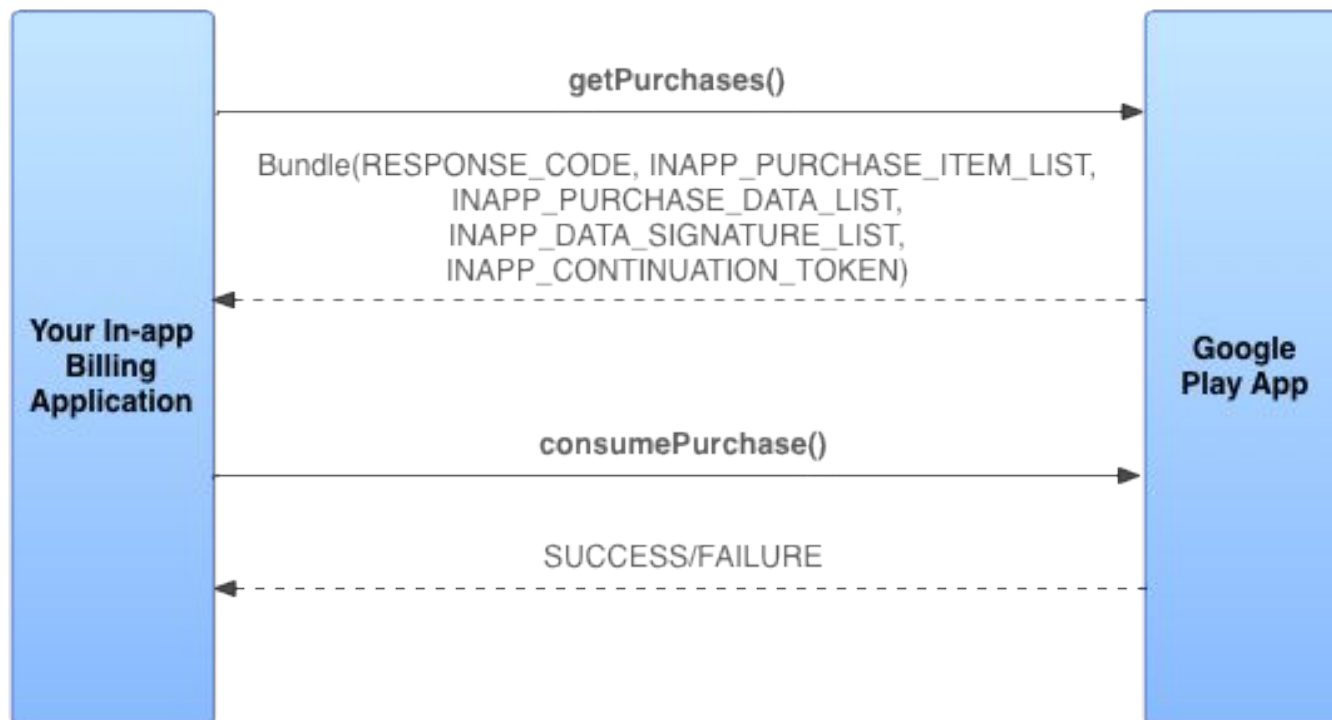
```
Purchase purchase = null;
try {
    purchase = new Purchase(mPurchasingItemType, purchaseData, dataSignature);
    String sku = purchase.getSku();

    // Verify signature
    if (!Security.verifyPurchase(mSignatureBase64, purchaseData, dataSignature)) {
```

- ***Else If (resultCode == Activity.RESULT\_OK)***
- ***Else if (resultCode == Activity.RESULT\_CANCELED)***

# Flow

## □ Consume




# Flow

## □ Consume

```
Log.d(TAG, "Purchase successful.");

if (purchase.getSku().equals(SKU_GAS)) {
    // bought 1/4 tank of gas. So consume it.
    Log.d(TAG, "Purchase is gas. Starting gas consumption.");
    mHelper.consumeAsync(purchase, mConsumeFinishedListener);
}
else if (purchase.getSku().equals(SKU_PREMIUM)) {
    // bought the premium upgrade!
    Log.d(TAG, "Purchase is premium upgrade. Congratulating user.");
    alert("Thank you for upgrading to premium!");
    mIsPremium = true;
    updateUi();
    setWaitScreen(false);
}
```



Version 3 안전하게 구현하려  
면?



# 효력지급

- Consume의 success가 확인된 후에 효력지급
  - 효력만 사용하고 아이템은 계속 남아있을 수 있다.

# 보안

- Developer Payload
  - Identify 할 수 있는 token
- Server Side 검증
- 이미 처리된 order Id 는 아닌지

# 이건 좀..

- 실제 과금
- 서버 검증 api 의 부재



고맙습니다