In-app Billing Version3 ---

역자 .. 김경미 (nolleh7707@gmail.com / http://nolleh.tistory.com)

Product Type

Managed In-app Products

구글에게 관리되는 상품. 유저별로 기반하여 각각의 구매정보를 저장한다. 나중에 질의할수 있게함. 유저가 어플리케이션을 바꾸거나 장치를 바꿨을때도 보존된다.

만약 3 api를 사용한다면 관리되는 아이템을 어플리케이션 내부에서 소비할수 도 있다. 일반적으로 구매한 아이템을 여러회 사용할수 있게 구현할 것이며 (캐쉬, 연료, 마법스펠등), 따라서 한번 구매된 관리되는 아이템은 아이템을 소비할때까지 구매할수 없게 된다. 인앱상품의 소비에대해서는 다음 사이트를 참조. – Consuming Items

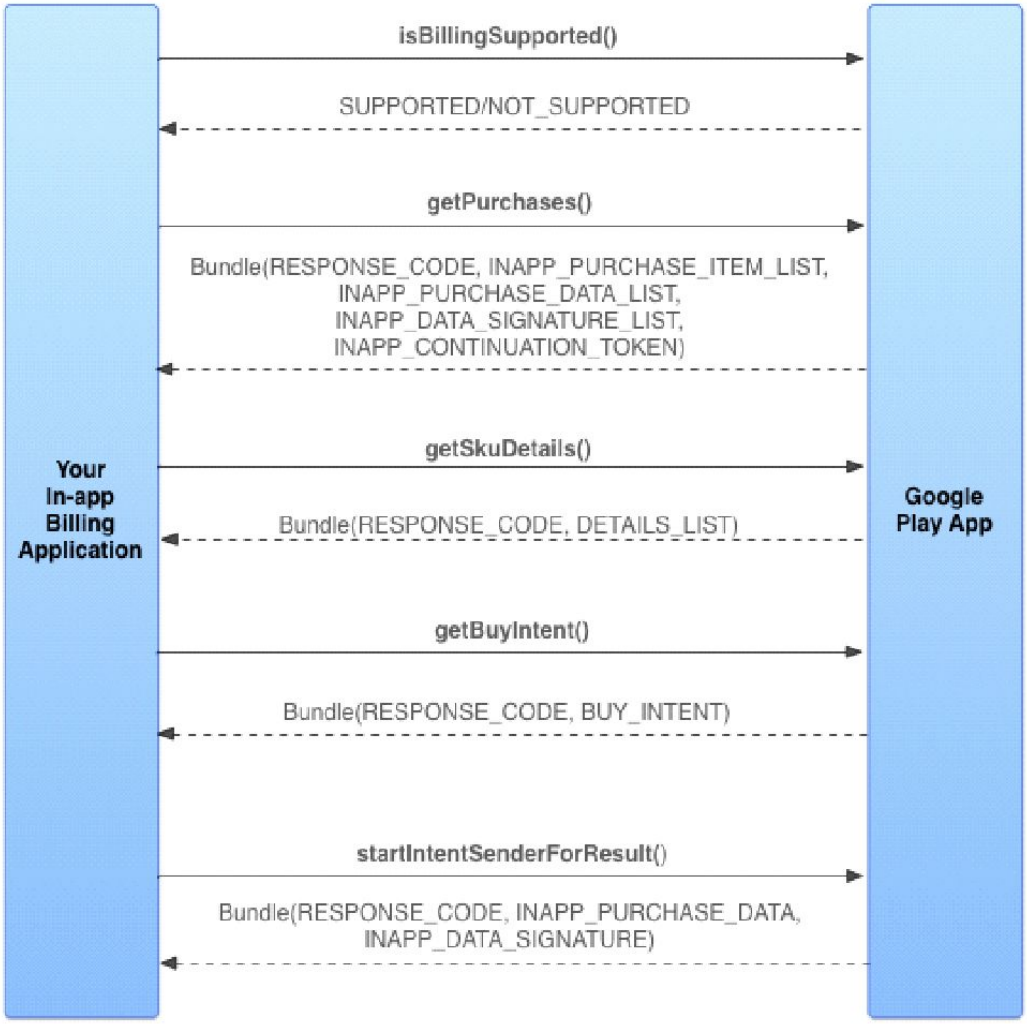
Subscriptons

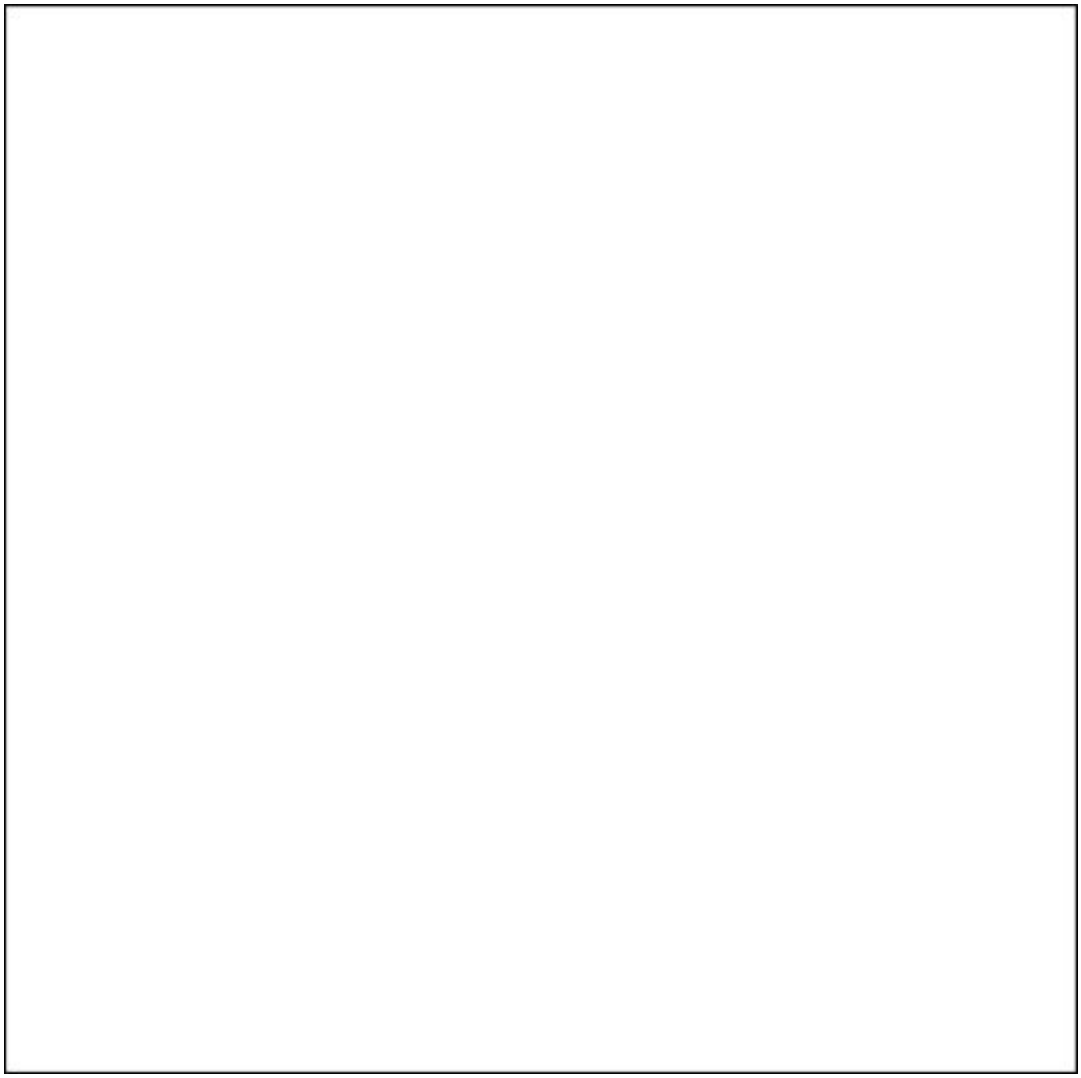
정기구독은 당신의 컨텐츠, 서비스, 기타 당신의 앱에 있는 특징들이 월단위 혹은 년단위로 재 구매 발생한다. 앱이나 게임의 디지털 컨텐츠 대부분을 정기구독의 형태로 판매할수 있다. 더 이해하기 위해 다음 사이트 참조 – Subscriptions.

ver3 api에서 정기구독을 구매하는것과 보존하는 것이 보통 인앱상품과 동일한 플로우로 구현할 수 있다. 코드 샘플을 보기위해 다음 사이트 참조 – Implementing Subscriptions.

- Important : 정기구독은 소비 될 수 없다.

Purchasing Items


Your In-app Billing Application
Google Play App
isBillingSupported()
SUPPORTED/NOT_SUPPORTED
getPurchases()
Bundle(RESPONSE_CODE, INAPP_PURCHASE_ITEM_LIST, INAPP_PURCHASE_DATA_LIST, INAPP_DATA_SIGNATURE_LIST, INAPP_CONTINUATION_TOKEN)
getSkuDetails()
Bundle(RESPONSE_CODE, DETAILS_LIST)
getBuyIntent()
Bundle(RESPONSE_CODE, BUY_INTENT)
startIntentSenderForResult()
Bundle(RESPONSE_CODE, INAPP_PURCHASE_DATA, INAPP_DATA_SIGNATURE)



- 가능한지 체크.
- 실행되거나 로그인 되었을때 어떤 아이템들을 갖고 있는지 질의.
- 어떤 아이템이 구매가능한지 리스팅해주고 싶을수 있다. 그때 getSkuDetails()를 호출. 이때 productID가 지정되어있어야한다.
- 사용자가 구매하지 않았다면 구매할 수 있도록 초기화해줄수 있다.
    - pendingIntent를 돌려준다. (getBuyIntent)
    - startIntentSenderForResult를 통해 실행.
    - 결제가 종료되면 onActivityResult로 결과를 알려줌. purchaseToken이 함께 전달되는에 이로 트랜젝션을 구분할수 있다. 프라이빗개발자 키를 이용한 Signature 또한 포함하고있다.

Api Call과 서버응답을 더 알아보기 위해 다음 참조- In-app Billing Reference.

Consuming In-app Products

유저의 인앱상품에 대한 소유권을 추적하기 위해 사용할수 있는 메커니즘.
Version3에서 모든 인앱상품은 managed이다. 이것은 유저의 인앱상품에 대한 소유권이 구글플레이에서 관리됨을 의미하며, 당신의 어플리케이션은 구매 정보를 필요할때 구글에 질의할수 있다. 유저가 구매를 마치면 구글플레이에 구매가 기록된다. 한번 구매가 완료되면 "소유"상태로 간주된다. '소유'상태의 인앱상품은 구글플레이에서 구매될수 없다. 소비 요청을 보내야 구글플레이에서 다시 구매가능한 상태로 만든다. 소비는 다시 '비소유'상태로 돌려놓을 것이며 이전 구매 데이터를 버리게한다.

유저가 소유한 리스트를 가져오기 위해 getPurchases를 구글플레이로 보낸다.
그후 consumePurchase를 부를 수 있으며, 이때 구매시에 얻었던 purchaseToken을 함께 넘겨주어야한다. 소비가 성공적으로 기록되면 상태 코드를 리턴한다.

Non-consumable and Consumable In-app Products

Non-consumable Items
일반적으로 consumption 을 구현하지 않아 단 한번만 구매되고 영구적인 효과를 얻을수 있다. 한번 구매되면 유저의 구글 계정에 영원히 관여된다.

Consumable items
소비를 이용해서 여러번 구매하게 제공할수 있다. 일반적으로 이 아이템들은 어떤 특별한 효과를 일시적으로 주고싶을때 사용한다. 이 효과나 이점을 제공하는것을 인앱상품을 프로비져닝한다라고 이야기한다. 유저에게 어떻게 프로비젼될지는 개발자가 제어하고 추적한다.

- Important : 어플리케이션에 컨슈머블아이템을 프로비저닝하기전에, 구글플레이에 소비(consumption) 요청을 보내 성공 응답을 먼저 받아야한다.

Managing consumable purchases in your application

컨슈머블 상품을 구매하는 기본 플로우

- getBuyIntent call
- 리턴된 번들이 구매가 성공적으로 되었음을 나타냄.
- 2라면 consumePurchase를 호출하여 소비.
- 소비가 성공적으로 되었음을 나타냄.
- 소비가 성공적으로 되었으면 효력지급(프로비전) 하라.

Local Caching

구글플레이 클라이언트가 인앱빌링 정보를 디바이스 로컬에 캐싱하고 있으므로 getPurchases 같은 질의를3api 를 더 자주 할 수 있다. 이전버전하고 다르게 구글플레이의 네트워크 커넥션대신에 캐쉬를 보므로 응답시간이 더 빠르다.

Implementing In-app Billing (IAB version 3)

직관적이고 간단한 인터페이스를 이용하여 빌링요청과 트랜젝션 관리를 할수 있다. 인앱서비스에 어떻게 콜을 보내는 기초등이 기술되어있다.

완벽한 구현과 어떻게 테스트하는지를 배우기위해서는 다음 사이트 참조 – Selling In-app Products

커넥션등을 세팅하는 키 테스크, 백그라운드 스레드 관리하는 메인액티비티를 포함하여 완벽한 샘플이 구성되어있다.

시작하기 전에 이전 Overview 를 읽어 개념을 잡아 구현이 더 쉬워지게 하자.

어플리케이션에 인앱빌링을 구현하기위해 다음 절차를 따른다.

- 프로젝트에 라이브러리를 추가한다.
- AndroidManifest.xml 파일을 업데이트한다.
- ServiceConnection 을 만들고 IInAppBillingService에 바인드 한다.
- 어플리케이션에서 IInAppBillingService로 요청을 보낸다.
- 구글플레이로부터 받은 응답을 처리한다.

Adding the AIDL file to your project

TriviaDriva 샘플은 구글 인앱빌링 서비스의 인터페이스 정의를 위해 Android Interface Definition Language (AIDL) 파일을 포함하고 있다. 이 파일을 프로젝트에 추가하면 안드로이드 빌드환경이 인터페이스 파일을 만들고 이 인터페이스를 IPC 메서드 콜에 사용할수 있다.

라이브러리를 프로젝트에 추가하기 위해

- IInAppBillingService.aidl 파일을 안드로이드 프로젝트에 복사.
    - 이클립스를 사용하면 : IInAppBillingService.aidl 파일을 /src 디렉토리에 import. 자동으로 인터페이스 파일을 생성할 것이다.
    - 이클립스가 아닌 환경이면 : /src/com/android/vending/billing 폴더를 생성하고 IInAppBillingService.aidl 파일을 여기에 복사한다. Ant tool 이 프로젝트를 빌드하여 java파일이 생성되게 한다.
- 어플리케이션을 빌드하라. /gen 폴더에 IInAppBillingService.java 파일이 위치해있어야한다.

Updating Your Application’s Manifest

인앱빌링은 구글플레이 서버와 모든 통신을 담당하는 구글플레이 어플리케이션에 의존한다. 이 구글플레이 어플리케이션을 이용하기 위해 어플리케이션은 반드시 알맞는 권한을 사용해야한다. Com.android.vending.BILLING 권한을 추가하여 그리할 수 있다. 만약 인앱빌링 권한을 선언하지 않은채로 빌링 요청을 보낸다면 구글플레이는 이 요청을 거절하고 에러와 함께 응답할 것이다.

Creating a ServiceConnection

어플리케이션과 구글플레이 사이에서 메시징을 용이하게 하기 위해 서비스커넥션을 갖고 있어야한다. 최소한 어플리케이션은 다음을 수행해야한다.

- Bind to IInAppBillingService
- Send billing Requests (as IPC calls)
- 동기 응답을 처리한다.

Binding to IInAppBillingService

구글플레이 서비스에 인앱빌링 연결을 구축하기위해 ServiceConnection 을 구현하여 IInAppBillingService 에 바인드한다.

-Important : 액티비티에서의 작업이 모두 끝나면 unbind를 호출하는 것을 잊지 말라. 만약 unbind 를 하지 않는다면 커넥션을 여는것이 단말기의 성능을 저해할 수 있다.

서비스 바인드에 관해 완벽한 구현을 보고 싶다면 다음을 참조하라.
Selling In-app Products

Making In-app Billing Requests

일단 구글플레이에 어플리케이션이 연결되면 구매요청을 초기화할수 있다. 구글은 유저가 구매 수단으로 들어갈수 있도록 체크아웃 인터페이스를 제공하며 따라서 개발자는 구매 트랜젝션을 직접적으로 관리할 필요가 없다. 아이템이 구매가 되면 , 구글플레이는 유저가 해당 아이템에 대해 소유권을 갖고 있음을 인지하고 같은 상품 아이디를 가진 아이템이 다시 구매가 되는것을 소비가 되기 전까지 막는다. 소비가 되는것은 어플리케이션에서 구글플레이로 알려줌으로써 제어할수있다. 또한 구글플레이에 구매 목록을 가져오도록 질의할수 있다. 유저가 앱을 launch 했을때 유저의 구매목록을 불러오려할때 유용하다.

Querying for Items Available for Purchase

어플리케이션에서 아이템의 상세 정보를 질의 할수 있다. ITEM_ID_LIST를 키로 가진 스트링 어레이 리스트를 포함한 번들을 생성하여 요청을 보낸다.

```
ArrayList skuList = new ArrayList();
skuList.add("premiumUpgrade");
skuList.add("gas");
Bundle querySkus = new Bundle();
querySkus.putStringArrayList(“ITEM_ID_LIST”, skuList);
```

구글플레이로부터 이 정보를 보존하기 위해 api version을 3으로 하여getSkuDetails 메서드를 패키지 네임과 구매타입을 함께 번들에 담아 호출한다.

```
Bundle skuDetails = mService.getSkuDetails(3,
   getPackageName(), "inapp", querySkus);
```

만약 요청이 성공적이라면 반환되는 bundle 의 응답코드가 BILLING_RESPONSE_RESULT_OK 일 것이다.

- Warning : getSkuDetails 메서드를 메인스레드에서 호출하지 말 것. 네트워크 통신을 하므로 메인스레드를 블락할 것이므로 분리된 스레드에서 호출하기.

가능한 모든 응답코드 확인을 위해 다음 사이트 참조 - In-app Billing Reference.

질의 결과는 String ArrayList 로 DETAILS_LIST 키에 json format 으로 저장되어있다. 리턴된 상품의 상세정보 타입들을 보려면 다음 사이트참조 – In-app Billing Reference.

이 예제에서 이전 코드조각에서 반환된 skuDetails 의 Bundle 로부터 상품의 가격을 보존한다.

```
int response = skuDetails.getInt("RESPONSE_CODE");
if (response == 0) {
   ArrayList responseList
      = skuDetails.getStringArrayList("DETAILS_LIST");

   for (String thisResponse : responseList) {
      JSONObject object = new JSONObject(thisResponse);
      String sku = object.getString("productId");
      String price = object.getString("price");
      if (sku.equals("premiumUpgrade")) mPremiumUpgradePrice = price;
      else if (sku.equals("gas")) mGasPrice = price;
   }
}
```

Purchasing an Item

구매요청을 시작하기위해 getBuyIntent 메서드를 인앱빌링 서비스로 호출한다. 빌링 에이피아이 버전은 3 으로 세팅하고 패키지 네임과 구매할 상품의 아이디, 그리고 구매형태 (inapp / subs) 와 payload 를 함께 포함한다. Payload는 구글플레이가 다시 돌려줄 어떤 추가 정보도 가능하다.

```
Bundle buyIntentBundle = mService.getBuyIntent(3, getPackageName(),
   sku, "inapp", "bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ");
```

만약 요청이 성공적이라면, 반환되는 번들은 BILLING_RESPONSE_RESULT_OK (0) 일거고 pendingIntent 가 포함되어돌아올것 이다. 모든 가능한 응답코드를 보려면 다음 사이트참조 –
In-app Billing Reference.

pendingIntent는 BUY_INTENT를 이용해 추출한다.

```
PendingIntent pendingIntent = buyIntentBundle.getParcelable("BUY_INTENT");
```

구매 트렌젝션을 완수하기위해 startIntentSenderForResult 메서드를 pendingIntent를 넣어 실행. 임의의 수 1001이 요청코드로 예시로 들어가있다.

```
startIntentSenderForResult(pendingIntent.getIntentSender(),
   1001, new Intent(), Integer.valueOf(0), Integer.valueOf(0),
   Integer.valueOf(0));
```

구글플레이는 onActivityResult로 어플리케이션에 응답을 준다. RESULT_OK이거나 RESULT_CANCELD 이다.

제이슨형태의 INAPP_PURCHASE_DATA 키에 응답인텐트에 포함이 된 데이터는 예를 들자면 다음과 같은 형태다.

```
'{
   "orderId":"12999763169054705758.1371079406387615",
   "packageName":"com.example.app",
   "productId":"exampleSku",
   "purchaseTime":1345678900000,
   "purchaseState":0,
   "developerPayload":"bGoa+V7g/yqDXvKRqq+JTFn4uQZbPiQJo4pf9RzJ",
   "purchaseToken":"rojeslcdyyiapnqcynkjyyjh"
 }'
```

이전 예제에서 계속 진행하자면 응답코드 , 구매데이터, 시그니처 응답을 인텐트로부터 얻을수 있다.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
   if (requestCode == 1001) {
      int responseCode = data.getIntExtra("RESPONSE_CODE", 0);
      String purchaseData = data.getStringExtra("INAPP_PURCHASE_DATA");
      String dataSignature = data.getStringExtra("INAPP_DATA_SIGNATURE");

      if (resultCode == RESULT_OK) {
         try {
            JSONObject jo = new JSONObject(purchaseData);
            String sku = jo.getString("productId");
```

```
            alert("You have bought the " + sku + ". Excellent choice,
                adventurer!");
        }
        catch (JSONException e) {
            alert("Failed to parse purchase data.");
            e.printStackTrace();
        }
    }
  }
}
```

보안 추천사항 : 페이로드에 유니크하게 아이덴티파이 할수 있는 토큰 정보를 포함하는 편이 좋다.
랜덤하게 스트링을 만들어도 좋다. 응답을 받았을때 시그니쳐 체크와 오더아이디, 페이로드체크를 하는걸 놓치지 마라. 이전에 처리하지 않은 오더아이디 임을 확인하고 페이로드가 매칭되는지 확인하고.

Querying for Purchased Items

getPurchases 를 불러 성공적으로 처리가 되면 응답코드가 0이다.
상품아이디의 정보, 주문 상세, 시그니쳐도 각 포함되어있다.

성능을 향상시키기위해 인앱빌링 서비스는 700가지까지만 getPurchases 가 처음 호출되었을때 리턴된다. 더 많은 수의 상품을 소유하고 있을때 INAPP_CONTINUATION_TOKEN 키를 갖게 되며 그러면 연속된 getPurchases를 이 토큰을 포함하여 다시 콜하여 더 정보를 얻을수있다.

```
int response = ownedItems.getInt("RESPONSE_CODE");
if (response == 0) {
   ArrayList ownedSkus =
      ownedItems.getStringArrayList("INAPP_PURCHASE_ITEM_LIST");
   ArrayList purchaseDataList =
      ownedItems.getStringArrayList("INAPP_PURCHASE_DATA_LIST");
   ArrayList signatureList =
      ownedItems.getStringArrayList("INAPP_DATA_SIGNATURE");
   String continuationToken =
      ownedItems.getString("INAPP_CONTINUATION_TOKEN");

   for (int i = 0; i < purchaseDataList.size(); ++i) {
      String purchaseData = purchaseDataList.get(i);
      String signature = signatureList.get(i);
      String sku = ownedSkus.get(i);

      // do something with this purchase information
      // e.g. display the updated list of products owned by user
```

```
    }

    // if continuationToken != null, call getPurchases again
    // and pass in the token to retrieve more items
}
```

Consuming a Purchase

어떻게 소비 정책을 가져갈지는 당신에게 달렸습니다.
일반적으로 일시적인 효력을 유저에게 지급하여 유저가 여러번 사게 하고 싶을수 있다.
또 소비하지 않게하여 영원한 효과를 지급하고 싶을때도 있다.
Important: Managed in-app products are consumable, but subscriptions are not.

구매 소비를 기록하기 위해 consumePurchase 메서드를 purchaseToken 을 포함하여 요청한다. (구매할때 전달받은)

Warning: Do not call the `consumePurchase` method on the main thread. Calling this method triggers a network request which could block your main thread. Instead, create a separate thread and call the`consumePurchase` method from inside that thread.

효력지급을 하는것은 당신의 몫이다. 게임 캐쉬를 산다면 유저의 인벤토리에 캐쉬의 양을 업데이트하는 것같은 것 ㄱ말이다.

Security Recommendation: You must send a consumption request before provisioning the benefit of the consumable in-app purchase to the user. Make sure that you have received a successful consumption response from Google Play before you provision the item.

Implementing Subscriptions

일반 상품 구매와 비슷하다. 예외는 상품 타입이 subs 가 된다는 것.
onActivityResult내용도 뭐 비슷하다.

```
Bundle bundle = mService.getBuyIntent(3, "com.example.myapp",
    MY_SKU, "subs", developerPayload);

PendingIntent pendingIntent = bundle.getParcelable(RESPONSE_BUY_INTENT);
if (bundle.getInt(RESPONSE_CODE) == BILLING_RESPONSE_RESULT_OK) {
    // Start purchase flow (this brings up the Google Play UI).
    // Result will be delivered through onActivityResult().
    startIntentSenderForResult(pendingIntent, RC_BUY, new Intent(),
            Integer.valueOf(0), Integer.valueOf(0), Integer.valueOf(0));
}
```

액티브한 정기구독을 질의하기 위해 getPurchases 를 호출하며타입을 subs 로 하여 한다.

```
Bundle activeSubs = mService.getPurchases(3, "com.example.myapp",
                    "subs", continueToken);
```

갱신없이 만료가 되면 번들에 더이상 포함이 되지 않는다.

http://developer.android.com/google/play/billing/billing_reference.html