|  |
| --- |
| Samsung In App Purchase v2.0  Library Programming Guide |

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description | Author | Date |
| 1.0 | Created |  | 13.10.31 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

목차

[1. Samsung In App Purchase v2.0 4](#_Toc363047621)

[1.1 IAP 2.0 제약 사항 4](#_Toc363047622)

[1.2 상품 구매 프로세스 4](#_Toc363047623)

[1.3 상품 타입 6](#_Toc363047624)

[2. IAP 연동 구현 7](#_Toc363047625)

[2.1 IAP 라이브러리 추가 7](#_Toc363047626)

[2.2 Permission 추가 8](#_Toc363047627)

[2.3 IAP 설치와 Samsung Account 인증 8](#_Toc363047628)

[2.4 ServiceConnection 생성과 IAPConnector 에 bind 11](#_Toc363047629)

[2.5 IAPConnector로 IAP 요청 및 응답 처리 12](#_Toc363047630)

[개발 모드 12](#_Toc363047631)

[① init() method 13](#_Toc363047632)

[② getItemList() method 15](#_Toc363047633)

[③ getItemsInbox() method 16](#_Toc363047634)

[④ 결제 17](#_Toc363047635)

[3. IAP Reference 25](#_Toc363047636)

[3.1 API Reference 25](#_Toc363047637)

[① init method 25](#_Toc363047638)

[② getItemList method 26](#_Toc363047639)

[③ getItemsInbox method 28](#_Toc363047640)

[④ PaymentMethodListActivity 30](#_Toc363047641)

[3.2. Response Code 32](#_Toc363047642)

# 1. IAP 2 Library

Samsung In App Purchase(이하 IAP라 함) Version 2는 3rd Party Application 내에서 삼성 결제 서비스를 연동할 수 있도록 지원하고 있습니다. Version 1 에서는 jar 형태의 라이브러리로 연동하는 방식이었으나 Version 2 에서는 AIDL(Android Interface Definition Language) 방식으로 IAP 프로세스와 3rd Party Application이 통신하는 방식을 사용하고 있기 때문에 이전 버전보다 좀 더 쉬운 연동방법을 제공합니다. IAP에 대해서 좀더 자세한 내용을 알고 싶다면 다음 문서를 참고하시기 바랍니다.

1. Samsung In App Purchase 개발 가이드 (\*링크제공)

그러나 서비스 연동, 삼성 어카운트 연동, 패키지 유효성 검사 등의 다양한 처리가 요구되기 때문에 개발자들이 IAP와 연동하는 것이 번거로울 뿐만 아니라 기술적으로 어려움에 부딪치는 경우가 많았던 것 또한 사실입니다.

그래서 좀더 쉽게 개발자들이 IAP와 연동할 수 있도록 지원하기 위해서 별도의 라이브러리 프로젝트를 제공하게 되었습니다.

IAP Library를 개발함에 있어서 최종 목표가 5분 안에 연동할 수 있는 결제시스템일 정도로 간편화 하기 위해서 많은 노력을 기울였습니다.

## 1.1 IAP 2.0 제약 사항

IAP는 안드로이드 API 10 부터 지원합니다. 그 이하의 버전에서는 정상적으로 동작하지 않습니다. IAP는 삼성전자 단말기에서만 동작합니다. 그 이외의 제조사 단말기에서는 동작하지 않습다.

그 이외의 제약 사항은 없습니다.

## 1.3 상품 타입

IAP에서는 아래와 같은 상품 타입을 지원합니다.

|  |  |
| --- | --- |
| **타입** | **설명** |
| Consumable  (소모성 상품) | 구매한 상품을 사용하면 상품이 소진되는 경우로 재 구매가 가능한 상품 입니다.  ※ 예 : 게임에서 총알과 같은 소모성 상품 |
| NonConsumable  (비소모성 상품) | 구매한 상품에 대해서 영구적으로 재사용이 가능한 상품으로 재 구매가 불가능한 상품입니다.  ※ 예 : 단행본 책과 같이 한번 구입하면 재 구매할 필요가 없는 비 소모성 상품 |
| Subscription  (기간제 상품) | 구매한 상품에 대해서 일정기간이 지나면 재 구매가 가능한 상품  ※ 예 : 월간 잡지나 기간제 이용권과 같이 구매한 후에 일정 기간이 지나면 재 구매가 가능한 상품. |
| All(전체) | 위 세가지 타입을 모두 포함한다. |

# 2. IAP Library Programming

Application에서 IAP의 결제 서비스를 사용하기 위해서는 아래와 같은 과정으로 진행되어야 합니다.

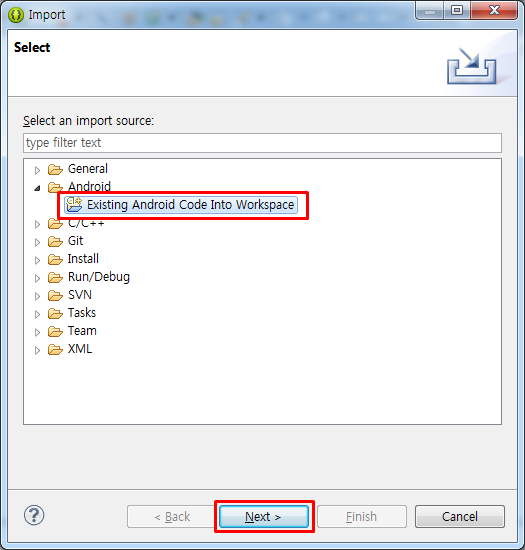
1. IAP Library 프로젝트 생성
2. 새 안드로이드 프로젝트 생성
3. 프로젝트에 IAP Library 프로젝트 추가
4. 매니페스트 수정
5. IAP 연동 개발

본 문서에서 사용된 개발 환경은 ADT(Android Developer Tools, Build : v22.2.1-833290 )입니다. 본 가이드 문서의 모든 내용은 이 개발환경을 기준으로 합니다.

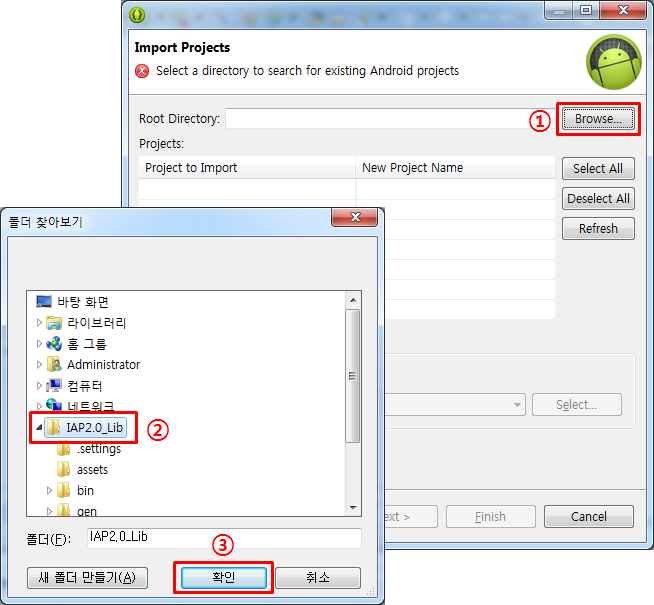
## 2.1 IAP Library 프로젝트 생성

IAP Library를 활용하기 위해서는 우선 <http://developer.samsung.com> 에서 제공하는 IAP Library 소스 코드를 이용해서 Android Library 프로젝트를 생성해야 합니다.

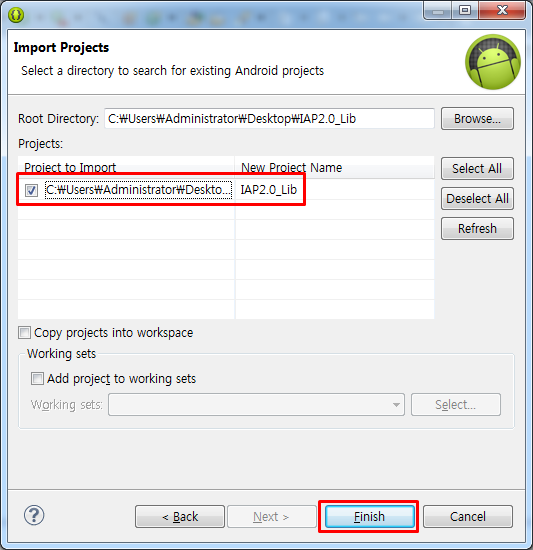
프로젝트를 생성하는 방법은 eclipse File -> Import 메뉴를 선택하면 다음 화면이 나타납니다.



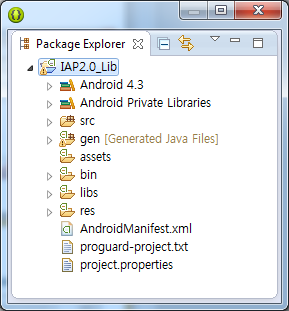
위 Import 화면에서 Android -> Existing Android Code Into Workspace를 선택하고 Next 버튼을 클릭하면 다음과 같이 Import Projects 화면이 나타납니다.

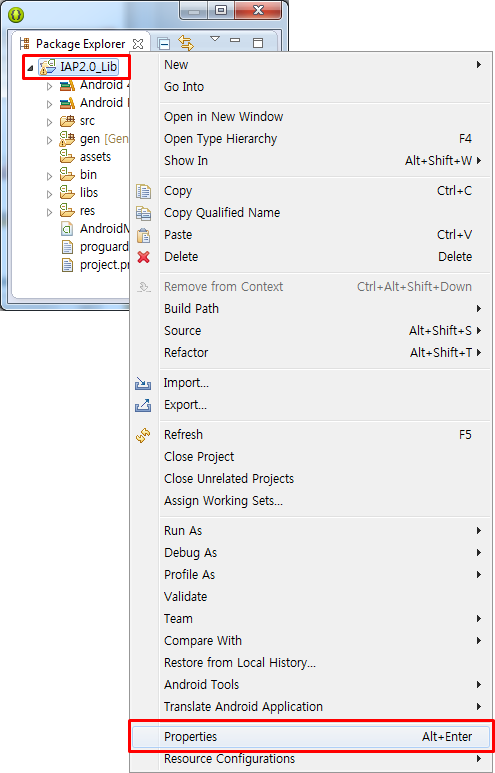


Import 화면에서 ① Browse… 버튼을 클릭하면 폴더 찾아보기 화면이 나타나고 ② IAP Library 프로젝트의 루트 디렉토리를 선택하고 ③ 확인 버튼을 클릭하면 Import 화면에 다음과 같이 선택된 디렉토리에 포함된 프로젝트 목록이 나타납니다.

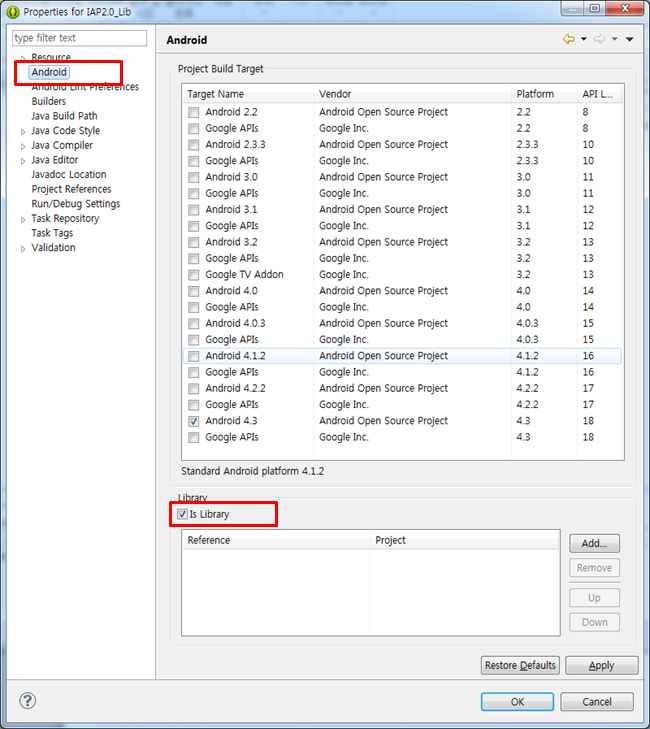


프로젝트 목록 왼쪽 체크박스를 선택하고 Finish 버튼을 클릭하면 워크스페이스에 IAP Library 프로젝트가 추가됩니다. 다음 화면과 같이 eclipse의 Package Explore에서 IAP20\_Lib 프로젝트가 추가되었음을 확인할 수 있습니다.





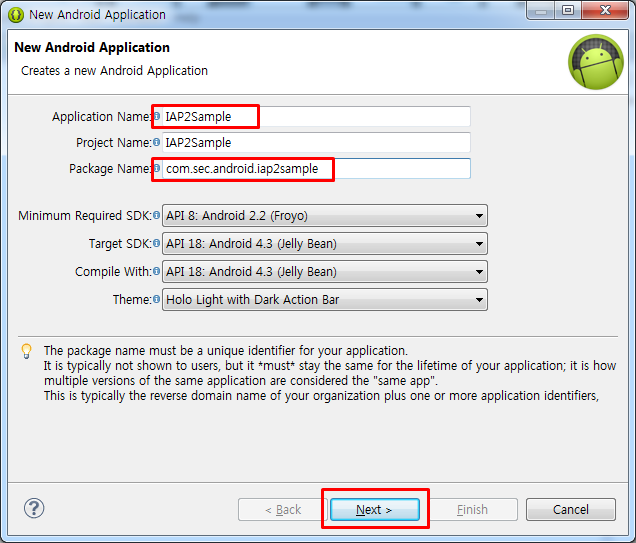
위 화면에서 추가된 IAP Library 프로젝트를 선택하고 마우스 오른쪽 버튼 클릭해서 나오는 메뉴하단의 Properties를 선택해서 프로젝트 Properties 화면의 Android 탭을 선택하면 다음 그림과 같이 Is Library에 체크되어 있는 것을 확인할 수 있습니다. 이것은 Android Library임을 나타내는 것입니다.



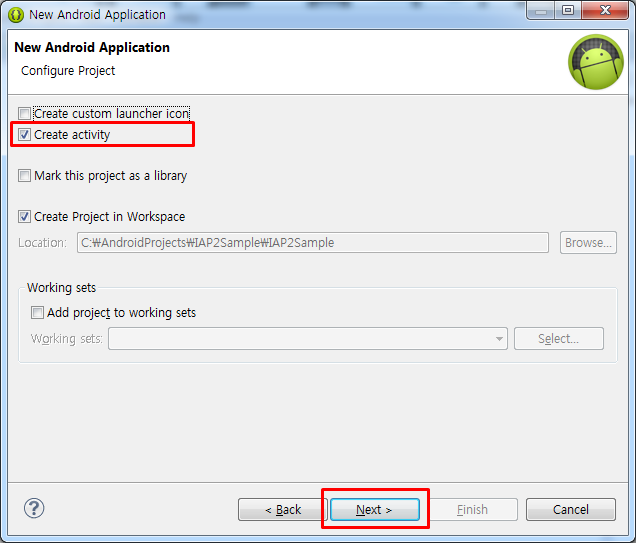
이제 IAP Library 프로젝트의 생성이 완료되었습니다. 이제 새로운 프로젝트를 생성하고 라이브러리를 활용할 수 있도록 설정하는 방법에 대해서 알아보도록 하겠습니다.

## 2.2 새 안드로이드 프로젝트 생성

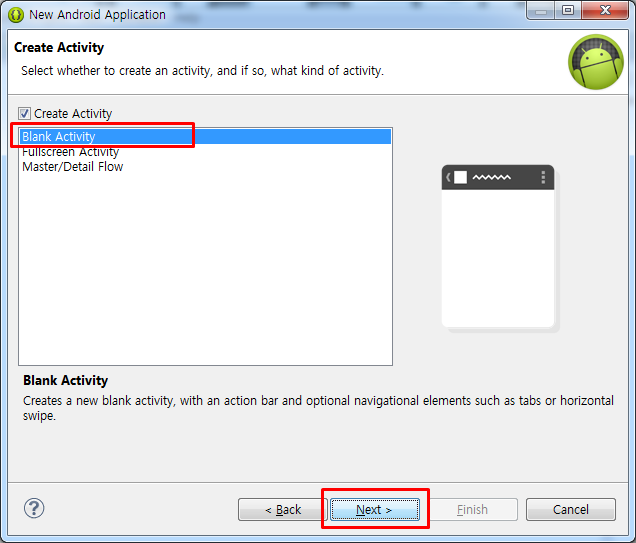
Eclipse의 File -> New -> Android Application Project 메뉴를 선택하면 다음과 같이 새로운 안드로이드 어플리케이션을 생성하는 화면이 나타납니다.



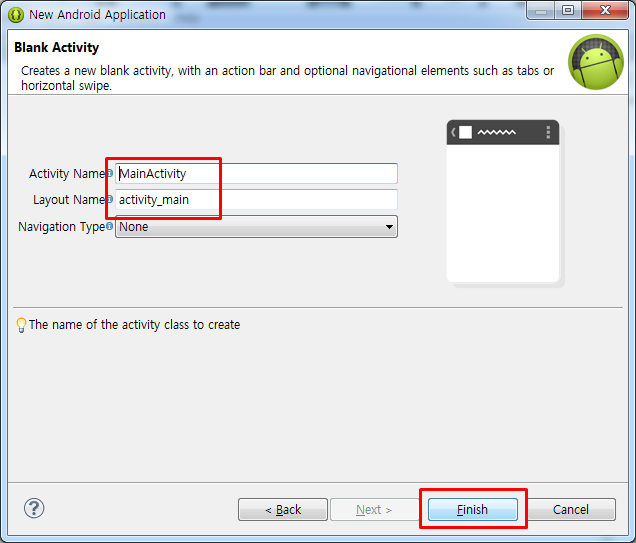
여기에서 Application Name은 IAP2Sample로 Package Name은 com.sec.android.iap2sample로 정했습니다. 각자 원하는 이름으로 입력하시면 됩니다. 이제 Next 버튼을 클릭하면 다음과 같이 생성할 프로젝트의 몇 가지 속성을 정의할 수 있는 화면이 나타납니다.



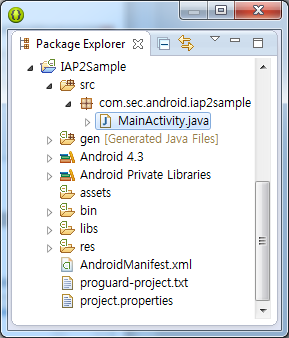
여기서는 기본 제공하는 아이콘을 사용할 것이기 때문에 체크를 해제하고 Create activity 만 체크하고 Next 버튼을 클릭합니다.



이전 화면에서 Create Custom Launcher Icon을 선택했다면 아이콘 만드는 화면이 나왔겠지만 우리는 선택하지 않았기 때문에 Activity를 생성하는 화면으로 넘어왔습니다. 여기서 Blank Activity를 선택하고 Next 버튼을 클릭합니다.



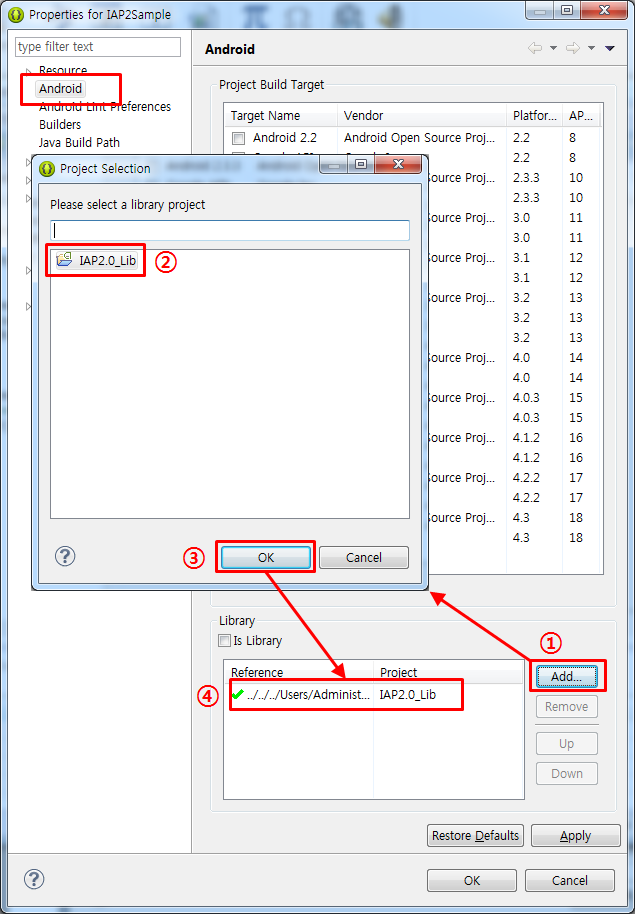
Activity 이름과 레이아웃 이름은 기본 값을 사용하고 Finish 버튼을 클릭하면 새로운 프로젝트가 생성됩니다. 다음 화면은 새로 생성된 IAP2Sample 프로젝트를 eclipse의 Package Explorer를 통해서 확인하는 것입니다.



## 2.2 IAP 라이브러리 추가

앞에서 새로 생성한 IAP2Sample 프로젝트에 IAP Library 프로젝트를 추가해야 합니다.

Package Explorer에서 IAP2Sampe 프로젝트를 선택하고 마우스 오른쪽 버튼을 클릭해서 Properties… 메뉴를 선택하거나 Alt-Enter 키로 Properties 화면으로 이동합니다.



위 화면 하단의 Library 영역의 ① Add 버튼을 클릭하면 현재 워크 스페이스에 포함된 라이브러리 프로젝트 리스트를 보여주고 필요한 라이브러리를 선택할 수 있게 해주는 화면이 나타납니다. 여기서 우리가 추가해야 하는 ② IAP2.0\_Lib 프로젝트를 선택하고 ③ Ok 버튼을 클릭하면 ④ 와 같이 라이브러리 프로젝트가 추가된 것을 확인할 수 있습니다.

## 2.3 Manifest.xml 수정

**1) 권한 추가**

IAP 연동을 위해서는 “com.sec.android.iap.permission.BILLING”권한이 필요합니다.

AndroidManifest.xml에 아래와 같이 권한을 추가합니다.

|  |
| --- |
| <uses-permission android:name=*"com.sec.android.iap.permission.BILLING"*/> |

**2) IAP Library에 포함된 Activity 추가**

IAP Library는 투명한 Activity를 통해서 3rd Party 어플리케이션과 통신을 하게 됩니다. 그래서 다음의 세가지 Activity를 manifest 파일에 추가해야 합니다.

|  |  |
| --- | --- |
| **Activity name** | **설명** |
| *com.sec.android.iap.lib.activity.InboxActivity* | 구매 내역을 요청하기 위해서 필요한 Activity |
| *com.sec.android.iap.lib.activity.PaymentActivity* | 구매요청을 위해서 필요한 Activity |
| *com.sec.android.iap.lib.activity.ItemActivity* | 상품 목록을 요청하기 위해서 필요한 Activity |

|  |
| --- |
| <activity  android:name=*"com.sec.android.iap.lib.activity.InboxActivity"*  android:theme=*"@style/Theme.Empty"*  android:configChanges=*"orientation|screenSize"*/>    <activity  android:name=*"com.sec.android.iap.lib.activity.PaymentActivity"*  android:theme=*"@style/Theme.Empty"*  android:configChanges=*"orientation|screenSize"*/>    <activity  android:name=*"com.sec.android.iap.lib.activity.ItemActivity"*  android:theme=*"@style/Theme.Empty"*  android:configChanges=*"orientation|screenSize"*/> |

※ 주의 : 위 3개의 Activity는 투명해야 하기 때문에 모두 android:theme=”@style/Theme.Empty”로 설정하고 있습니다. 만약 투명한 테마를 지정하지 않는다면 이 Activity 들이 여러분의 Application을 가리는 현상이 발생하게 됩니다.

## 2.4 IAP 연동

**1) 구매하기 구현**

다음과 같이 activity\_main.xml 레이아웃을 수정해서 구매 버튼을 추가하고 구매 버튼이 클릭되었을 때 doPurchase 메소드가 호출되도록 했습니다.

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"utf-8"*?>  <LinearLayout  xmlns:android=*"http://schemas.android.com/apk/res/android"*  android:layout\_width=*"match\_parent"*  android:layout\_height=*"match\_parent"*  android:orientation=*"vertical"*  android:padding=*"10dp"* >    <!-- Purchase Button -->  <Button  android:id=*"@+id/btn\_purchase\_one\_item"*  android:layout\_width=*"fill\_parent"*  android:layout\_height=*"wrap\_content"*  android:text=*"@string/purchase\_one\_item"*  android:onClick=*"doPurchase"*/>  </LinearLayout> |

이제 doPurchase 메소드에서 IAP에 결제를 요청하는 코드를 아래와 같이 작성했습니다. 어플리케이션을 등록할 때

|  |
| --- |
| **public** **class** MainActivity **extends** Activity **implements** OnPaymentListener  {  **private** **static** **final** **int** *MODE* = SamsungIapHelper.*IAP\_MODE\_TEST\_SUCCESS*;  **private** **static** **final** String *GROUP\_ID* = "100000100010";  **private** **static** **final** String *ITEM\_ID* = "000001000018";    @Override  **protected** **void** onCreate( Bundle savedInstanceState )  {  **super**.onCreate(savedInstanceState);  setContentView( R.layout.*activity\_main* );  }    **public** **void** doPurchase( View \_view )  {  SamsungIapHelper helper = **new** SamsungIapHelper( **this**, *MODE* );  helper.startPayment( *GROUP\_ID*, *ITEM\_ID*, **true**, **this** );  }    @Override  **public** **void** onPayment( ErrorVo \_errorVo, PurchaseVo \_purchaseVo )  {  // 여기에서 결제 성공/실패 처리를 한다.  }  } |

위 소스코드에서 IAP에 구매를 요청하는 부분은 doPurchase 메소드 내부의 아래 단 두줄 입니다.

SamsungIapHelper helper = **new** SamsungIapHelper( **this**, *MODE* );

helper.startPayment( *GROUP\_ID*, *ITEM\_ID*, **true**, **this** );

마지막 인자는OnPaymentListener로 최종 결제된 결과를 받는 인터페이스 입니다. 위 Activity는 OnPaymentListener를 onPayment 인터페이스를 implements 하고 있기 때문에 this가 인자로 들어갔습니다.

기존에 복잡했던 초기화 과정은 모두 IAP Library 내부에서 처리되고 그 구매 결과만 onPayment 인터페이스로 들어오게 됩니다.

이 예제에서는 테스트용으로 미리 만들어진 Group Id와 Item Id를 사용하고 있습니다. 실제 출시할 때에는 삼성앱스를 통해서 발급받은 정보를 사용해야 합니다. 이와 관련된 자세한 정보는 다음 문서를 참고하시기 바랍니다.

startPayment 메소드에 대한 좀 더 자세한 정보는 레퍼런스를 참고하시기 바랍니다.

**2) 구매 목록 가져오기**

**3) 상품 목록 가져오기**

1. IAP를 사용할 Application Project의 src 폴더 아래에 **'com.sec.android.iap'**라는 package를 생성합니다.
2. **'IAPConnector.aidl'**과 **'IAPServiceCallBack.aidl'** 파일을 생성한 package 인 **'com.sec.android.iap'** 아래에 복사합니다.

**Eclipse 를 사용하지 않는 다면**

1. **'src/com/sec/android/iap'** 구조의Directory를 생성합니다.
2. 생성한 Directory에 **'IAPConnector.aidl'**과 **'IAPServiceCallBack.aidl'** 파일을복사합니다.
3. Ant tool을 사용하여 Project를 build 하여, '**IAPConnector.java**'와 '**IAPServiceCallback.java**' 파일을 생성합니다.

위 과정을 통해서 Application Project 의 '/gen' Directory에 생성된 '**IAPConnector.java**'과 '**IAPServiceCallback.java**' 파일이 생성되도록 합니다.

## 2.2 Permission 추가

Application에서 IAP를 연동하기 위해서는 AndroidManifest.xml 파일에 "com.sec.android.iap.permission.BILLING" 권한을 등록해야 합니다.

AndroidManifest.xml에 아래와 같이 권한을 추가합니다.

<uses-permission android:name=*"com.sec.android.iap.permission.BILLING"*/>

! 만일 해당 퍼미션을 등록하지 않으면 다음 과정인 Samsung Account 인증 과정에서 무조건 실패 처리 됩니다.

## 2.3 IAP 설치와 Samsung Account 인증

**IAP 설치 체크**

**public** **boolean** isInstalledIapPackage( Context \_context )

{

PackageManager pm = \_context.getPackageManager();

**try**

{

pm.getApplicationInfo( **"com.sec.android.iap"**,

PackageManager.*GET\_META\_DATA* );

**return** **true**;

}

**catch**( NameNotFoundException e )

{

e.printStackTrace();

**return** **false**;

}

}

위 코드에서 flag 값이 true 이면 IAP가 이미 설치되어 있다는 것이고, false 이면 IAP가 설치되어 있지 않다는 의미입니다.

**IAP가 설치되어 있다면,**

IAP 패키지의 signature 해시코드 값을 비교해서 설치된 IAP의 유효성 여부를 검사합니다.

**private** **static** **final** **int** *IAP\_SIGNATURE\_HASHCODE* = **0x7a7eaf4b**;

**public** **boolean** isValidIAPPackage( Context \_context )

{

**boolean** result = **true**;

**try**

{

Signature[] signs = getPackageManager().getPackageInfo(

**"com.sec.android.iap"**

PackageManager.*GET\_SIGNATURES* ).signatures;

**if**( sigs[0].hashCode() != *IAP\_SIGNATURE\_HASHCODE* )

{

result = **false**;

}

}

**catch**( Exception e )

{

e.printStackTrace();

result = **false**;

}

**return** result;

}

설치된 IAP의 유효성 검사결과 유효하지 않은 경우는 다음과 같이 메시지 다이얼로그를 보여주고 결제과정이 더 이상 진행되지 않도록 해야만 합니다.

// 설치된 IAP 패키지가 유효하지 않다면

// ================================================================

**if**( **false** == *isValidIAPPackage*( \_activity ) )

{

**new** AlertDialog.Builder( \_activity )

.setTitle( R.string.*in\_app\_purchase* )

.setMessage( R.string.*invalid\_iap\_package* )

.setPositiveButton( android.R.string.*ok*,

**new** DialogInterface.OnClickListener()

{

@Override

**public** **void** onClick( DialogInterface dialog, **int** which )

{

dialog.dismiss();

\_activity.finish();

}

} ).show();

}

// ================================================================

반대로 설치된 IAP가 유효한 경우에는 다음과 같이 Samsung Account 인증 절차를 진행합니다. 인증이 성공되어야 다음 단계를 진행할 수 있습니다.

// 설치된 패키지가 유효한 패키지라면

// ================================================================

**else**

{

ComponentName com = **new** ComponentName( "com.sec.android.iap",

"com.sec.android.iap.activity.AccountActivity" );

Intent intent = **new** Intent();

intent.setComponent( com );

startActivityForResult( intent, **1001** );

}

// ================================================================

Samsung Account 인증을 하기 위해서 IAP의 AccountActivity를 호출합니다. AccountActivity는 Samsung Account 인증 및 결제 진행 시 비밀번호 재확인을 위해서 사용됩니다.

IAP의 AccountActivity 호출 시 Appplication에서는 startActivityForResult method를 이용하여 호출하고, IAP는 'Samsung Account 인증' 결과에 대해서 Application의 onActivityResult method에 전달합니다.

**onActivityResult method**

@Override

protected void onActivityResult(int \_requestCode, int \_resultCode, Intent \_intent)

{

if( \_requestCode == **1001** )

{

if( \_resultCode == RESULT\_OK )

{

/\* 로그인 성공 \*/

}

}

}

**IAP 가 설치되어 있지 않다면,**

IAP 설치를 위해서 삼성앱스의 IAP 상세 페이지로 이동합니다.

static final int FLAG\_INCLUDE\_STOPPED\_PACKAGES = 32;

Intent intent = new Intent();

intent.setData( Uri.parse( "samsungapps://ProductDetail/com.sec.android.iap" ) );

// 허니콤 MR1 이상인 경우 FLAG\_INCLUDE\_STOPPED\_PACKAGES 플래그를 추가한다.

if( Build.VERSION.SDK\_INT >= 12 )

{

intent.addFlags( Intent.FLAG\_ACTIVITY\_NEW\_TASK |

Intent.FLAG\_ACTIVITY\_CLEAR\_TOP |

FLAG\_INCLUDE\_STOPPED\_PACKAGES );

}

else

{

intent.addFlags( Intent.FLAG\_ACTIVITY\_NEW\_TASK |

Intent.FLAG\_ACTIVITY\_CLEAR\_TOP );

}

startActivity( intent );

## 2.4 ServiceConnection 생성과 IAPConnector 에 bind

Application에서 IAP와 통신을 하기 위해서는 ServiceConnection을 구현해야 합니다. 절차는 아래와 같습니다.

1. IAPConnector bind 하기
2. Application 종료 시 unbind 하기
   1. **IAPConnector bind 하기**

Application에서 IAP에 connection하기 위해서는 Activity에서 ServiceConnection을 implement하여 IAPConnector를 bind합니다. ServiceConnection을 생성한 후에 Override 된 onServiceDisconnected와 onServiceConnected 메서드를 이용하여 IAPConnector Instance를 참조할 수 있습니다.

IAPConnector mIAPConnector;

ServiceConnection mServiceConn = **new** ServiceConnection()

{

@Override

**public** **void** onServiceDisconnected( ComponentName name )

{

mIAPConnector = **null**;

}

@Override

**public** **void** onServiceConnected( ComponentName name, IBinder service )

{

mIAPConnector = IAPConnector.Stub.asInterface( service );

}

};

IAP 서비스 명을 참조하고 있는 Intent와 생성한 ServiceConnection 객체를 파라미터로 가지는 bindService 메서드를 이용하여 바인딩을 합니다.

Intent serviceIntent = **new** Intent( "com.sec.android.iap.service.iapService" );

bindService( serviceIntent, mServiceConn, Context.BIND\_AUTO\_CREATE );

위 과정을 완료하면 IAP와 통신할 수 있는 mIAPConnector를 사용할 수 있습니다.

1. **Application 종료 시 unbind 하기**

Application이 종료될 때에는 IAP를 unbind해야 합니다.

IAP의 사용이 더 이상 필요 없는 경우 위에서 생성한 mServiceConn라는 Service Connection 객체에 대해서 반드시 unbind를 수행해야 합니다.

unbindService( mServiceConn );

## 2.5 IAPConnector로 IAP 요청 및 응답 처리

IAP에서는 아래와 같은 인터페이스를 제공합니다.

|  |  |  |
| --- | --- | --- |
| **타입** | **이름** | **설명** |
| Method | **Init** | 결제 서비스를 이용하기 위한 정보 설정 및 초기화 |
| Method | **getItemList** | 상품 목록 |
| Method | **getItemsInbox** | 구매한 상품 목록 |
| Activity | **PaymentMethodListActivity** | 결제 방법을 선택하고 결제를 요청하는 Activity |

각 인터페이스에 대해서 상세히 살펴보도록 하겠습니다.

### 개발 모드

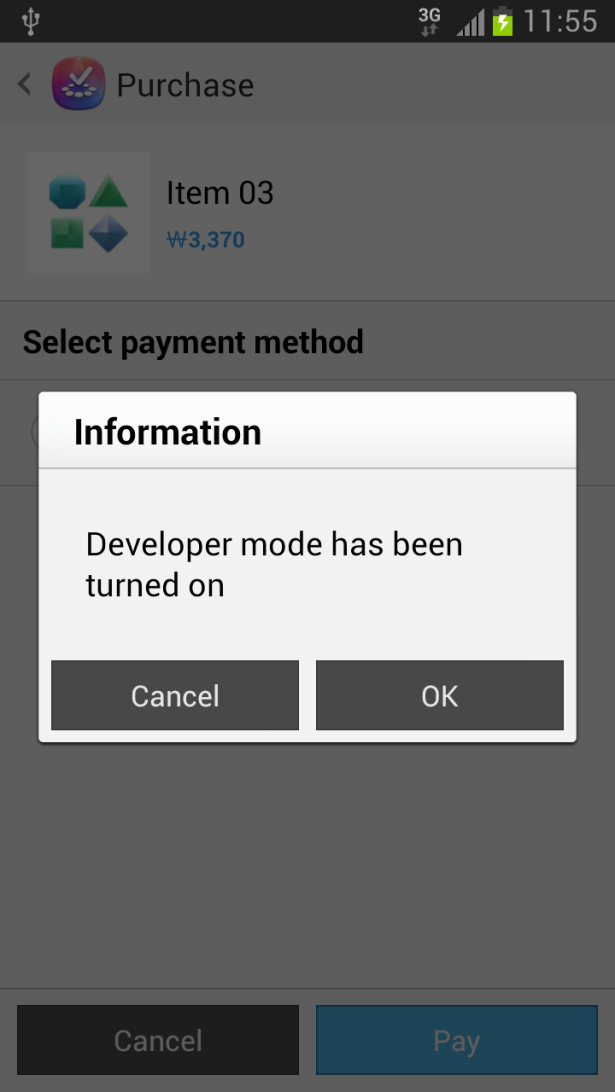
과금 때문에 결제 서비스는 테스트가 쉽지 않다는 문제가 있습니다. 그래서 IAP는 다양한 상황에 대해서 테스트할 수 있도록 개발자 모드를 지원하고 있습니다.

IAP에서 제공하는 각 인터페이스에는 integer 형의 mode 인자를 가지고 있습니다. 이 값을 바꿔줌으로써 개발자 모드를 설정할 수 있습니다. IAP가 지원하는 모드는 다음과 같습니다.

|  |  |  |
| --- | --- | --- |
| **Mode** | **값** | **설명** |
| 운영모드 | **0** | 운영모드 입니다. **Release 할 때는 반드시 운영모드로 설정해야 합니다. 운영모드에서만 실제 결제가 발생합니다.** |
| 테스트모드(성공) | **1** | 항상 성공 결과를 반환하는 테스트 모드입니다. |
| 테스트모드(실패) | **-1** | 항상 실패 결과를 반환하는 테스트 모드입니다. |

개발 시에는 항상 성공 결과를 넘겨주는 1이나 항상 실패 결과를 넘겨주는 -1 로 설정하여 다양한 상황에 대해서 충분히 테스트를 하고 **Release 할 때는 반드시 운영모드로 수정해야 합니다. 테스트 모드로 Release 했을 경우에는 실제 결제가 발생하지 않습니다.**

간혹 개발자 모드로 Release 하는 경우가 있어서 IAP는 다음과 같이 개발자 모드로 설정되어 있음을 알리는 팝업을 보여줍니다. 이 팝업을 보셨다면 반드시 개발모드를 확인하시기 바랍니다.



이제 각 인터페이스에 대해서 상세하게 살펴보도록 하겠습니다.

### init() method

IAP의 '구매한 상품 목록', '결제' 등의 서비스를 사용하기 전에 필수적으로 수행되어야 하는 인터페이스입니다.

Samsung Account 인증이 성공적으로 완료된 후에 init() 인터페이스를 호출하여 결제를 진행할 때 필요한 초기 정보를 설정합니다.

init() 인터페이스에서는 IAP 업그레이드 체크와 서버URL, 국가코드, shop아이디, 통화(화폐단위)정보, 사용자아이디, email 등의 기본정보들을 IAP에서 사용할 수 있도록 설정하는 역할을 합니다.

**※ 주의 : nit() method는 Main Thread에서 호출하면 ANR (Application Not Responding) 이 발생할 수 있습니다. 별도의 Thread를 생성하여 사용합니다.**

Bundle bundle = mIAPConnector.init( mMode );

if ( null != bundle )

{

int statusCode = bundle.getInt( "STATUS\_CODE*"* );

String errorString = bundle.getString( "ERROR\_STRING" );

// statusCode가 -1001인 경우 IAP\_UPGRADE\_URL 키워드로 이동할 페이지 주소를 전달합니다.

String extraString = bundle.getString( "IAP\_UPGRADE\_URL" );

if ( statusCode == **0** )

{

/\* 초기화 성공한 경우 처리 \*/

}

else if ( statusCode == **-1001** )

{

/\* IAP 패키지 업그레이드가 필요한 경우 처리 \*/

// 업그레이드 안내 팝업창을 보여주고 사용자가 확인하면 삼성앱스 IAP 페이지로 이동

**new** AlertDialog.Builder( **this** )

.setTitle("IAP Upgrade*"* )

.setMessage( errorString )

.setPositiveButton( android.R.string.*ok*,

**new** DialogInterface.OnClickListener()

{

@Override

**public** **void** onClick( DialogInterface dialog, **int** which )

{

Intent intent = **new** Intent();

intent.setData( Uri.*parse*( extraString ) );

intent.addFlags( Intent.*FLAG\_ACTIVITY\_NEW\_TASK* );

**try**

{

startActivity( intent );

}

**catch**( ActivityNotFoundException e )

{

e.printStackTrace();

}

}

} ).show();

}

}

'STATUS\_CODE'가 '0'이면, 초기 설정은 성공적으로 완료된 것으로 '구매한 상품 목록', '결제' 등의 서비스를 사용할 수 있게 됩니다.

**※ 주의 : 위 샘플 코드에서 statusCode 가 -1001이면 IAP 패키지 업데이트가 필요한 경우 입니다. 이때 IAP 패키지가 업데이트되지 않으면 IAP를 사용할 수 없기 때문에 반드시 삼성앱스의 IAP 페이지로 이동하여 업데이트를 유도하도록 처리를 추가해야 합니다.**

### getItemList() method

구매 가능한 '상품 목록'을 Bundle 형태로 반환합니다.

**※ 주의 : getItemList() method는 Main Thread에서 호출하면 ANR (Application Not Responding) 이 발생할 수 있습니다. 별도의 Thread를 생성하여 사용합니다.**

Bundle bundle = mIAPConnector.getItemList( mMode,

getPackageName(),

\_itemGroupId,

\_startNum,

\_endNum,

\_itemType );

요청의 성공여부는 반환된 Bundle의 키 값 'STATUS\_CODE'에 저장되어 있습니다. (성공일 경우 '0')

특히, STAUS\_CODE가 -1001인 경우는 IAP 패키지의 업데이트가 필요한 경우이기 때문에 앞 init() 메소드 샘플코드의 결과 처리 부분을 참고하시기 바랍니다.

**Bundle안의 RESULT\_LIST 사용하기**

int statusCode = bundle.getInt( "STATUS\_CODE" );

if ( statusCode == 0 )

{

ArrayList<String> arrayList = bundle.getStringArrayList( "RESULT\_LIST" );

for( String itemString : arrayList )

{

JSONObject jObject = new JSONObject( itemString );

String itemId = jObject.getString( "mItemId" );

String itemName = jObject.getString( "mItemName" );

String itemPrice = jObject.getString( "mItemPrice" );

String currencyUnit = jObject.getString( "mCurrencyUnit" );

String itemDesc = jObject.getString( " mItemDesc" );

String itemImageUrl = jObject.getString( "mItemImageUrl" );

String itemDownloadUrl = jObject.getString( "mItemDownloadUrl" );

String reserved1 = jObject.getString( "mReserved1" );

String reserved2 = jObject.getString( "mReserved2" );

String type = jObject.getString( "mType" );

}

}

### getItemsInbox() method

'구매한 상품 목록'을 Bundle 형태로 반환합니다.

**※ 주의 : getItemsInbox() method는 Main Thread에서 호출하면 ANR (Application Not Responding) 이 발생할 수 있습니다. 별도의 Thread를 생성하여 사용합니다.**

Bundle itemsInboxList = mIAPConnector.getItemsInbox( mContext.getPackageName(),

\_itemGroupId,

\_startNum,

\_endNum,

\_startDate,

\_endDate );

요청의 성공여부는 반환된 Bundle의 키 값 'STATUS\_CODE'에 저장되어 있습니다. (성공일 경우 '0')

**Bundle안의 RESULT\_LIST 사용하기**

int statusCode = bundle.getInt( "STATUS\_CODE" );

if ( statusCode == 0 )

{

ArrayList<String> arrayList = bundle.getStringArrayList( " RESULT\_LIST" );

for ( String itemString : arrayList)

{

JSONObject jObject = new JSONObject( itemString );

String itemId = jObject.getString( "mItemId" );

String itemName = jObject.getString( "mItemName" );

String itemDesc = jObject.getString( " mItemDesc " );

String itemPrice = jObject.getString( "mItemPrice" );

String currencyUnit = jObject.getString( " mCurrencyUnit " );

String itemImageUrl = jObject.getString( " mItemImageUrl " );

String itemDownloadUrl = jObject.getString( " mItemDownloadUrl " );

String purchaseDate = jObject.getString( " mPurchaseDate " );

**long** timeInMillis = Long.*parseLong*( jObject.getString( "mPurchaseDate" ) );

String paymentId = jObject.getString( "mPaymentId" );

String reserved1 = jObject.getString( "mReserved1" );

String reserved2 = jObject.getString( "mReserved2" );

String type = jObject.getString( "mType" );

String itemPriceString = jObject.getString( "mItemPriceString" );

}

}

### 결제

구매하고자 하는 상품에 대해서 결제를 진행할 수 있는 IAP의 '결제' Activity로 이동할 수 있게 합니다.

Bundle bundle = **new** Bundle();

bundle.putString( "THIRD\_PARTY\_NAME", getPackageName() );

bundle.putString( "ITEM\_GROUP\_ID", \_itemGroupId );

bundle.putString( "ITEM\_ID", \_itemId );

ComponentName com = **new** ComponentName( "com.sec.android.iap",

"com.sec.android.iap.activity.PaymentMethodListActivity" );

Intent intent = **new** Intent( Intent.ACTION\_MAIN );

intent.addCategory( Intent.CATEGORY\_LAUNCHER );

intent.setComponent( com );

intent.putExtras( bundle );

startActivityForResult( intent, 1000 );

IAP의 '결제상세' Activity 호출 시 Appplication에서는 startActivityForResult method를 이용하여 호출하고, IAP는 결제 결과에 대해서 Application의 onActivityResult method에 전달합니다.

**onActivityResult method**

결과에 대한 Intent의 'RESULT\_OBJECT' key의 데이터는 JSON String 형태로 아래와 같이 되어있습니다.

{

"mItemDesc":"Learn about the books of the Bible##46;",

"mCurrencyUnit":"￡",

"mItemImageUrl":"http:\/\/img.samsungapps.com\/product\/2013\/0510\/000000524646\/IconImage\_20130510095823098\_NEW\_WEB\_ICON.png",

"mItemDownloadUrl":"",

**"mPaymentId":"ZPMTID20130603GBI0000376",**

"mItemPrice":1,

**"mVerifyUrl":"https:\/\/iap.samsungapps.com\/iap\/appsItemVerifyIAPReceipt.as?protocolVersion=2.0",**

"mItemName":"Books",

"mPurchaseDate":1370253642115,

**"mPurchaseId":"524822074a3a8671102cbe35bf033f59bcb6f9ec79a950ab5d83de67b3ff92f8",**

"mReserved1":"",

"mReserved2":"",

"mItemId":"000000057508",

"mItemPriceString":"￡1.00"

}

아래와 같이 JSON 객체를 생성하여 쉽게 원하는 값을 얻을 수가 있습니다. 그리고 STATUS\_CODE, THIRD\_PARTY\_NAME, ERROR\_STRING, ITEM\_ID와 같은 부가적인 값들도 Bundle에 포함되어 있습니다.

// 1. IAP 결제 결과 처리

// treat result of IAPService

// ============================================================================

if( \_requestCode == **1** )

{

if( null == \_intent )

{

return;

}

Bundle extras = \_intent.getExtras();

String itemId = "";

String thirdPartyName = "";

// payment success : 0

// payment cancelled : 1

// ========================================================================

int statusCode = 1;

// ========================================================================

String errorString = "";

String purchaseData = "";

// 1) IAP 에서 전달된 Bundle 정보가 존재할 경우

// ------------------------------------------------------------------------

if( null != extras )

{

thirdPartyName = extras.getString( "THIRD\_PARTY\_NAME" );

statusCode = extras.getInt( "STATUS\_CODE" );

errorString = extras.getString( "ERROR\_STRING" );

itemId = extras.getString( "ITEM\_ID" );

purchaseData = extras.getString( "RESULT\_OBJECT" );

}

// ------------------------------------------------------------------------

// 2) IAP 에서 전달된 Bundle 정보가 존재하지 않는 경우

// ------------------------------------------------------------------------

else

{

showResultDialog(

getString( R.string.*dlg\_title\_payment\_error* ),

getString( R.string.*msg\_payment\_was\_not\_processed\_successfully* ) );

}

// ------------------------------------------------------------------------

// 3) 결제가 취소되지 않은 경우

// If payment was not cancelled

// ------------------------------------------------------------------------

if( *RESULT\_OK* == \_resultCode )

{

try

{

JSONObject jObject = new JSONObject( purchaseData );

**String paymentId = jObject.getString( "mPaymentId" );**

String itemDesc = jObject.getString( "mItemDesc" );

String itemPrice = jObject.getString( "mItemPrice" );

String currencyUnit = jObject.getString( "mCurrencyUnit" );

String itemImageUrl = jObject.getString( "mItemImageUrl" );

String itemName = jObject.getString( "mItemName" );

String purchaseDate = jObject.getString( "mPurchaseDate" );

long timeInMillis = Long.*parseLong*( jObject.getString("mPurchaseDate") );

**String purchaseId = jObject.getString( "mPurchaseId" );**

String reserved1 = jObject.getString( "mReserved1" );

String reserved2 = jObject.getString( "mReserved2" );

String itemDownloadUrl = jObject.getString( "mItemDownloadUrl" );

itemId = jObject.getString( "mItemId" );

String itemPriceString = jObject.getString( "mItemPriceString" );

**String verifyUrl = jObject.getString( "mVerifyUrl" );**

// a. 결제 결과가 성공일 경우

// ----------------------------------------------------------------

if( statusCode == 0 )

{

String serverUrl = verifyUrl + "&purchaseID=" + purchaseId;

new VerifyClientToServer( serverUrl,

purchaseId,

paymentId ).execute();

}

// ----------------------------------------------------------------

// b. 결제 결과가 실패일 경우

// ----------------------------------------------------------------

else

{

showResultDialog(

getString( R.string.*dlg\_title\_payment\_error* ),

"-itemId : " + itemId +

"\n-thirdPartyName : " + thirdPartyName +

"\n-statusCode : " + statusCode +

"\n-errorString : " + errorString );

}

// ----------------------------------------------------------------

}

catch( JSONException e )

{

e.printStackTrace();

}

}

// ------------------------------------------------------------------------

// 4) 결제가 취소된 경우

// If payment was cancelled

// ------------------------------------------------------------------------

else if( *RESULT\_CANCELED* == \_resultCode )

{

showResultDialog(

getString( R.string.*dlg\_title\_payment\_cancelled* ),

"-itemId : " + itemId +

"\n-thirdPartyName : " + thirdPartyName +

"\n-statusCode : " + statusCode );

}

// ------------------------------------------------------------------------

}

// ============================================================================

스니핑 앱에 등에 의해서 발생할 수 있는 문제로부터 안전한 구매 결과 처리를 위해서는 결과로 넘어온 Bundle 값에서 mVerifyUrl과 mPurchaseID값을 이용해서 IAP 서버에 유효한 구매인지 반드시 확인해야 합니다. 이를 위해서 위 예제에서는 VerifyClientToServer AsyncTask를 실행하고 있습니다. 다음은 VerifyClientToServer AsyncTask 소스코드 입니다.

private class VerifyClientToServer extends AsyncTask<Void, Void, Boolean>

{

String mServerUrl = null;

String mPurchaseId = null;

String mPaymentId = null;

VerificationVO mVerificationVO = null;

public VerifyClientToServer

(

String \_strUrl,

String \_purchaseId,

String \_paymentId

)

{

mServerUrl = \_strUrl;

mPurchaseId = \_purchaseId;

mPaymentId = \_paymentId;

}

@Override

protected void onPreExecute()

{

super.onPreExecute();

// 서버 URL, mPurchasedId, mPaymentId 가 비어있다면 태스크를 취소한다.

// ================================================================

if( true == TextUtils.*isEmpty*( mServerUrl ) ||

true == TextUtils.*isEmpty*( mPurchaseId ) ||

true == TextUtils.*isEmpty*( mPaymentId) )

{

this.cancel( true );

}

// ================================================================

mProgressDialog = showProgressDialog( ItemList.this );

}

@Override

protected void onCancelled()

{

dismissProgressDialog( mProgressDialog );

super.onCancelled();

}

@Override

protected Boolean doInBackground( Void... params )

{

try

{

int retryCount = 0;

String strResponse = null;

// 신뢰성을 높이기 위해서 오류가 발생했을 때 3회까지 재시도 하도록 한다.

// =============================================================

do

{

strResponse = getHttpGetData( mServerUrl,

10000,

10000 );

retryCount++;

}

while( retryCount < 3 &&

true == TextUtils.*isEmpty*( strResponse ) );

// =============================================================

if( strResponse == null || TextUtils.*isEmpty*( strResponse ) )

{

return false;

}

else

{

mVerificationVO = new VerificationVO( strResponse );

// Status Code 가 "true" 이고 mVerificationVO 의 PaymentId 와

// mPaymentId가 일치할 경우에만 성공으로 판정한다.

// ==========================================================

if( mVerificationVO != null &&

true == "true".equals( mVerificationVO.getmStatus() ) &&

true == mPaymentId.equals(

mVerificationVO.getmPaymentId() ) )

{

return true;

}

// ==========================================================

else

{

return false;

}

}

}

catch( Exception e )

{

e.printStackTrace();

return false;

}

}

@Override

protected void onPostExecute( Boolean result )

{

dismissProgressDialog( mProgressDialog );

if( true == result )

{

showResultDialog(

getString( R.string.*dlg\_title\_payment\_success* ),

"-itemId : " + mVerificationVO.getmItemId() +

"\n-paymentId : " + mVerificationVO.getmPaymentId() );

}

else

{

showResultDialog(

getString( R.string.*dlg\_title\_payment\_error* ),

getString( R.string.*msg\_invalid\_purchase* ) );

}

}

private String getHttpGetData

(

final String \_strUrl,

final int \_connTimeout,

final int \_readTimeout

)

{

String strResult = null;

URLConnection con = null;

HttpURLConnection httpConnection = null;

BufferedInputStream bis = null;

ByteArrayOutputStream buffer = null;

try

{

URL url = new URL( \_strUrl );

con = url.openConnection();

con.setConnectTimeout(10000);

con.setReadTimeout(10000);

httpConnection = (HttpURLConnection)con;

httpConnection.setRequestMethod("GET");

httpConnection.connect();

int responseCode = httpConnection.getResponseCode();

if( responseCode == 200 )

{

bis = new BufferedInputStream(

httpConnection.getInputStream(),

4096 );

buffer = new ByteArrayOutputStream( 4096 );

byte [] bData = new byte[ 4096 ];

int nRead;

while( ( nRead = bis.read( bData, 0, 4096 ) ) != -1 )

{

buffer.write( bData, 0, nRead );

}

buffer.flush();

strResult = buffer.toString();

}

}

catch( Exception e )

{

e.printStackTrace();

}

finally

{

if( bis != null )

{

try { bis.close(); } catch (Exception e) {}

}

if( buffer != null )

{

try { buffer.close(); } catch (IOException e) {}

}

con = null;

httpConnection = null;

}

return strResult;

}

}

위 VerifyClientToServer 예제에서 strResponse는 아래와 같은 JSON 데이터 형태로 구성되어 있습니다.

{

"itemId":"000000057507",

"itemName":"Additional Game Modes",

"itemDesc":"Unlock all game modes. Best value!",

"purchaseDate":"2013-06-03 21:04:50",

"paymentId":"ZPMTID20130603GBI0000384",

"paymentAmount":"1.500",

"status":"true"

}

서버로부터 strResponse를 받아서 아래의 Verification class의 생성자와 같이 원하는 값을 추출하여 사용할 수 있습니다. 아래 예제는 VerifyClientToServer AsyncTask에서 태스크 결과를 저장하는 객체로 활용되는 VerificationVO 소스의 일부분입니다.

**public** **class** VerificationVO

{

**private** String mItemId;

**private** String mItemName;

**private** String mItemDesc;

**private** String mPurchaseDate;

**private** String mPaymentId;

**private** String mPaymentAmount;

**private** String mStatus;

**public** VerificationVO( String strJson )

{

**try**

{

JSONObject jObject = **new** JSONObject( strJson );

mItemId = jObject.getString( "itemId" );

mItemName = jObject.getString( "itemName" );

mItemDesc = jObject.getString( "itemDesc" );

mPurchaseDate = jObject.getString( "purchaseDate" );

mPaymentId = jObject.getString( "paymentId" );

mPaymentAmount = jObject.getString( "paymentAmount" );

mStatus = jObject.getString( "status" );

}

**catch**( JSONException e )

{

e.printStackTrace();

}

}

중략…

}

좀더 안전한 거래 처리를 위해서 위 샘플코드의 VerifyClientToServer 와 같은 작업을 3rd Party 서버측에서 처리할 것을 권장합니다.

# 3. IAP 2 Library Reference

IAP Library에서는 구매, 구매목록, 상품목록을 위한 총 3개의 메소드를 제공하고 있습니다. 이 메소드들은 모두 SamsungIapHelper 내부에 존재합니다. SamsungIapHeler 생성자.

helper.startPayment( *GROUP\_ID*, *ITEM\_ID*, **true**, **this** );

## 3.1 SamsungIapHelper() Constructor

SamsungIapHelper( Context \_context, int \_mode)

이 메소드는 SamsungIapHelper를 생성합니다.

**- parameter 명세**

|  |  |  |
| --- | --- | --- |
| **파라미터** | **타입** | **설명** |
| \_context | Context |  |
| \_mode | int | IAP 사용모드에 대한 값  0 : 운영모드  1 : 개발모드 (반환 결과는 무조건 성공)  -1 : 개발모드 (반환 결과는 무조건 실패) |

## 3.2 startPayment Method

### getItemList method

**getItemList( String packageName, String itemGroupId, int startNum, int endNum, String itemType )**

이 method는 구매 가능한 '상품 목록'을 Bundle 형태로 반환합니다.

**- parameter 명세**

|  |  |  |
| --- | --- | --- |
| **파라미터** | **타입** | **설명** |
| mode | int | IAP 사용모드에 대한 값  0 : 운영모드  1: 개발모드 (반환 결과는 무조건 성공)  -1: 개발모드 (반환 결과는 무조건 실패) |
| packageName | String | Application의 패키지명 |
| itemGroupId | String | 상품의 그룹아이디 |
| startNum | int | 보여질 목록의 시작 번호 |
| endNum | int | 보여질 목록의 끝 번호 |
| itemType | String | 아이템 타입  Consumable(소모성) : 00  NonConsumable(비소모성) : 01  Subscription(기간제) : 02  All(전체) : 10 |

**- 반환된 Bundle에 대한 Key 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| STATUS\_CODE | 결과코드 (0이 아니면 에러 또는 실패) |
| ERROR\_STRING | 에러 또는 실패일 경우 메시지 |
| IAP\_UPGRADE\_URL | IAP 패키지 업그레이드가 필요한 경우(STATUS\_CODE == -1001) 삼성앱스의 해당 페이지로 이동하기 위한 URI 정보 |
| RESULT\_LIST | JSON 형태의 상품 목록 String |

**- 결과 Bundle에 대한 STATUS\_CODE 종류**

|  |  |  |
| --- | --- | --- |
| **응답 코드** | **코드 값** | **설명** |
| IAP\_ERROR\_NONE | 0 | 성공 |
| IAP\_ERROR\_COMMON | -1002 | IAP 실행 중 오류 |
| IAP\_ERROR\_PRODUCT\_DOES\_NOT\_EXIST | -1005 | 요청한 상품 목록이 없는 경우 |
| IAP\_ERROR\_ITEM\_GROUP\_DOES\_NOT\_EXIST | -1007 | 아이템 그룹 아이디가 없을 경우 |
| IAP\_ERROR\_NETWORK\_NOT\_AVAILABLE | -1008 | 네트워크를 사용할 수 없는 상태 |
| IAP\_ERROR\_IOEXCEPTION\_ERROR | -1009 | IOException |
| IAP\_ERROR\_SOCKET\_TIMEOUT | -1010 | SocketTimeoutException |
| IAP\_ERROR\_CONNECT\_TIMEOUT | -1011 | ConnectTimeoutException |

**- Bundle 'RESULT\_LIST'에 대한 JSON 필드 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| mItemId | 상품아이디 |
| mItemName | 상품명 |
| mItemPrice | 상품가격 |
| mCurrencyUnit | 통화 |
| mItemDesc | 상품 설명 |
| mItemImageUrl | 상품 이미지 URL |
| mItemDownloadUrl | 상품 다운로드 URL |
| mItemPriceString | 통화 + 상품가격 스트링 |
| mReserved1 | 예약1 |
| mReserved2 | 예약2 |
| mType | 상품 타입 |
| mSubscriptionDurationUnit | 상품 타입이 Subscription인 경우 상품의 유효기간 단위로 가능한 단위는**YEAR**, **MONTH**, **WEEK**, **DAY** 모두 4가지로 대문자 값이다. |
| mSubscriptionDurationMultiplier | 상품 타입이 Subscription인 경우 상품의 유효기간이다.  mSubscriptionDurationUnit과 조합하여 1MONTH와 같이 표현한다. |

### getItemsInbox method

**getItemsInbox( String packageName, String itemGroupId, int startNum, int endNum, String startDate, String endDate ) method**

이 method는 '구매한 상품 목록'을 Bundle 형태로 반환합니다.

**- parameter 명세**

|  |  |  |
| --- | --- | --- |
| **파라미터** | **타입** | **설명** |
| packageName | String | Application의 패키지명 |
| itemGroupId | String | 상품의 그룹아이디 |
| startNum | int | 보여질 목록의 시작 번호 |
| endNum | int | 보여질 목록의 끝 번호 |
| startDate | String | 구매 기간 시작 일자 (ex. 20130422) |
| endDate | String | 구매 기간 종료 일자 (ex. 20130430) |

**- 반환된 Bundle에 대한 Key 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| STATUS\_CODE | 결과코드 (0이 아니면 에러 또는 실패) |
| ERROR\_STRING | 에러 또는 실패일 경우 메시지 |
| RESULT\_LIST | JSON 형태의 구매한 상품 목록 String |

**- 결과 Bundle에 대한 STATUS\_CODE 종류**

|  |  |  |
| --- | --- | --- |
| **응답 코드** | **코드 값** | **설명** |
| IAP\_ERROR\_NONE | 0 | 성공 |
| IAP\_ERROR\_COMMON | -1002 | IAP 실행 중 오류 |
| IAP\_ERROR\_ITEM\_GROUP\_DOES\_NOT\_EXIST | -1007 | 아이템 그룹 아이디가 없을 경우 |
| IAP\_ERROR\_NETWORK\_NOT\_AVAILABLE | -1008 | 네트워크를 사용할 수 없는 상태 |
| IAP\_ERROR\_IOEXCEPTION\_ERROR | -1009 | IOException |
| IAP\_ERROR\_SOCKET\_TIMEOUT | -1010 | SocketTimeoutException |
| IAP\_ERROR\_CONNECT\_TIMEOUT | -1011 | ConnectTimeoutException |

**- Bundle 'RESULT\_LIST'에 대한 JSON 필드 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| mItemId | 상품아이디 |
| mItemName | 상품명 |
| mItemPrice | 상품가격 |
| mItemPriceString | 통화 + 상품가격 |
| mItemDesc | 상품 설명 |
| mCurrencyUnit | 통화 |
| mPurchaseDate | 구매 일시 (millisecond 형태) |
| mPaymentId | 결제아이디 |
| mPurchaseId | 구매아이디 |
| mItemImageUrl | 상품 이미지 URL |
| mItemDownloadUrl | 상품 다운로드 URL |
| mReserved1 | 예약1 |
| mReserved2 | 예약2 |
| mType | 상품 타입 |
| mSubscriptionEndDate | 상품 타입이 SubScription인 경우 유효기간 만료일. long 타입으로 리턴된다. |

### PaymentMethodListActivity

IAP 결제 기능 사용시 전달 되는 필수 Bundle 정보와 결과에 대해 설명합니다.

**- 요청 시 필수 Bundle 정보**

|  |  |  |
| --- | --- | --- |
| **파라미터** | **타입** | **설명** |
| packageName | String | Application의 패키지명 |
| itemGroupId | String | 상품의 그룹아이디 |
| itemId | String | 상품아이디 |

**- 반환된 Bundle에 대한 Key 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| STATUS\_CODE | 결과코드 (0이 아니면 에러 또는 실패) |
| ERROR\_STRING | 에러 또는 실패일 경우 메시지 |
| RESULT\_OBJECT | JSON 형태의 결제 결과 String |
| ITEM\_ID | 구매하고자 하는 상품아이디 |

**- 결과 Bundle에 대한 STATUS\_CODE 종류**

|  |  |  |
| --- | --- | --- |
| **응답 코드** | **코드 값** | **설명** |
| IAP\_ERROR\_NONE | 0 | 성공 |
| IAP\_PAYMENT\_IS\_CANCELED | 1 | 결제 취소 |
| IAP\_ERROR\_COMMON | -1002 | IAP 실행 중 오류 |
| IAP\_ERROR\_ALREADY\_PURCHASED | -1003 | 비소모성 상품에 대해서 재 구매일 경우 |
| IAP\_ERROR\_WHILE\_RUNNING | -1004 | Bundle 정보 없이 결제 상세를 호출할 경우 |
| IAP\_ERROR\_PRODUCT\_DOES\_NOT\_EXIST | -1005 | 요청한 상품이 없는 경우 |
| IAP\_ERROR\_CONFIRM\_INBOX | -1006 | 결제 결과가 성공은 아니지만 구매되었을 수 있기 때문에 구매한 상품 목록 확인이 필요할 경우 |
| IAP\_ERROR\_ITEM\_GROUP\_DOES\_NOT\_EXIST | -1007 | 아이템 그룹 아이디가 없을 경우 |
| IAP\_ERROR\_NETWORK\_NOT\_AVAILABLE | -1008 | 네트워크를 사용할 수 없는 상태 |
| IAP\_ERROR\_IOEXCEPTION\_ERROR | -1009 | IOException |
| IAP\_ERROR\_SOCKET\_TIMEOUT | -1010 | SocketTimeoutException |
| IAP\_ERROR\_CONNECT\_TIMEOUT | -1011 | ConnectTimeoutException |

**- Bundle 'RESULT\_OBJECT'에 대한 JSON 필드 설명**

|  |  |
| --- | --- |
| **파라미터** | **설명** |
| mItemId | 상품아이디 |
| mItemName | 상품명 |
| mItemDesc | 상품 설명 |
| mItemPrice | 상품 가격 |
| mItemPriceString | 통화가 포함된 상품 가격 |
| mCurrencyUnit | 통화 |
| mPaymentId | 결제아이디 |
| mPurchaseDate | 구매 일시 (millisecond 형태) |
| mPurchaseId | 구매아이디 |
| mReserved1 | 예약1 |
| mReserved2 | 예약2 |
| mItemImageUrl | 상품이미지URL |
| mItemDownloadUrl | 상품다운로드URL |
| mVerifyUrl | mPurchaseId값을 이용해서 IAP 서버에 유효한 구매인지 확인하기 위한 IAP Server URL |

## 3.2. Response Code

IAP에서 전달된 Bundle의 키 값 'STATUS\_CODE'에 대한 코드 목록입니다.

|  |  |  |
| --- | --- | --- |
| **응답 코드** | **코드 값** | **설명** |
| IAP\_ERROR\_NONE | 0 | 성공 |
| IAP\_PAYMENT\_IS\_CANCELED | 1 | 결제 취소 |
| IAP\_ERROR\_INITIALIZATION | -1000 | IAP 초기화 중 실패 |
| IAP\_ERROR\_NEED\_APP\_UPGRADE | -1001 | IAP 업그레이드가 필요 |
| IAP\_ERROR\_COMMON | -1002 | IAP 실행 중 오류 |
| IAP\_ERROR\_ALREADY\_PURCHASED | -1003 | Non-Consumable 상품 재 구매 또는 Subscription 상품을 유효기간 내에 재 구매하는 경우 |
| IAP\_ERROR\_WHILE\_RUNNING | -1004 | Bundle 정보 없이 결제상세(PaymentMethodListActivity)를 호출한 경우 |
| IAP\_ERROR\_PRODUCT\_DOES\_NOT\_EXIST | -1005 | 요청한 상품 목록이 없는 경우 |
| IAP\_ERROR\_CONFIRM\_INBOX | -1006 | 서버에 결제 요청을 한 이후에 그 결과를 받지 못했다면 결제 결과가 성공은 아니지만 결제가 성공했을 가능성이 있기 때문에 구매한 상품 목록 확인이 필요한 경우. |
| IAP\_ERROR\_ITEM\_GROUP\_DOES\_NOT\_EXIST | -1007 | 아이템 그룹 아이디가 없을 경우 |
| IAP\_ERROR\_NETWORK\_NOT\_AVAILABLE | -1008 | 네트워크를 사용할 수 없는 상태 |
| IAP\_ERROR\_IOEXCEPTION\_ERROR | -1009 | IOException |
| IAP\_ERROR\_SOCKET\_TIMEOUT | -1010 | SocketTimeoutException |
| IAP\_ERROR\_CONNECT\_TIMEOUT | -1011 | ConnectTimeoutException |