결제 서비스 질의 사항

인앱결제는 두 가지 방식으로 구현할 수 있습니다.

첫 번째는 원스토어에서 제공하는 인앱결제 서비스의 인터페이스를 직접 구현하는 방식이고,
두 번째는 인앱결제 서비스 인터페이스 구현체를 래핑(wrapping)한 인앱결제 SDK를 사용하는 방식입니다.

우리는 2번째 SDK방식을 사용.

1) IappManager.cpp -> sendPaymentRequest에서

```
CCLog("IapManager::sendPaymentRequest 2");
JniMethodInfo t;
if (JniHelper::getMethodInfo(t, IAP_JAVA_OBJECT, "sendPaymentRequest", "(ZLja
    CCLog("IapManager::sendPaymentRequest 3");
```

```
#define IAP_JAVA_OBJECT "com/numixent/inApp/IapManager"
```

com/numixent/inApp/IapManager 클래스의 sendpaymentrequest 메소드가 존재할시 아이템 정보 등을 설정

```
//application id 는 알고 있고 , in-app productid , productName , tid//개발자 생성값
public void sendPaymentRequest(boolean delvopmode , String productid , String productName , String productTid) {
    mTid = productTid;
    println("sendPaymentRequest productid = " + productid + "productName = " + productName + "productTid" + productTid );
    //String parameter = makeRequest(productid , productName , productTid);
    Builder payBuilder = new PaymentParams.Builder(IAP_APP_ID, productid);
    payBuilder.addTid(productTid);
    PaymentParams params = payBuilder.build();
    final String fProductId = productid;
    String requestId = mPlugin.sendPaymentRequest(
            new IapPlugin.RequestCallback() {
                @Override
                public void onResponse(IapResponse data) {
                    println("onrespoinse");
                    if (data == null || data.getContentLength() <= 0) {
                        // TODO Unusual error
                        println("onResponse() response data is null");
                        OnIapResult(false , "response data is null", "" , "" , "");
                        return;
                    }
                    // 1. JSON 데이터를 통한 객체 변환 사용 예
                    Response response = ConverterFactory.getConverter().fromJson(data.getContentToString());

                    if (response == null) {
                        // TODO invalid response data
                        println("onResponse() invalid response data");
                        OnIapResult(false , "invalid response data" , "" , "" , "");
                        return;
                    }

                    println("onrespoinse 1");
```

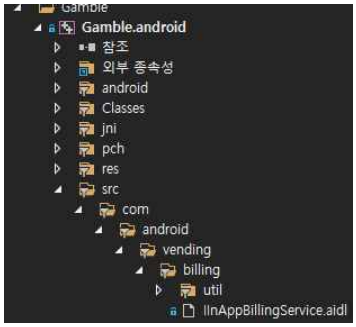
인앱결제 라이브러리 추가하기

원스토어 인앱결제 라이브러리 V6(SDK V19) 부터 AIDL(Android Interface Definition Language)은 지원하지 않습니다.

원스토어 인앱결제 SDK를 추가하는 방법은 다음을 참고하세요.

- 인앱결제 SDK 추가
  - 인앱결제 SDK 라이브러리 파일을 다운로드한 후 해당 파일을 개발사 앱 프로젝트 내 'libs' 폴더 안에 복사합니다.
  - 원스토어 인앱결제 API V6 (SDK V19) 라이브러리 다운로드
  - global-appstores.json 파일을 개발사 앱 프로젝트 내 'assets' 폴더 안에 복사합니다.

==>현재 프로젝트 내에서
IInAppBillingService.aidl 파일 존재 확인
IabHelper에서 원스토어 SDK 클래스 확인. IInAppBillingService를 활용하여 원스토어 SDK 활용함.
==> AIDL을 사용하였으므로 인앱결제 라이브러리 V6를 사용불가 -> V5버전을 사용해야함.




```
}

void LobbyShopPanel::recvWPayReserveRes(HttpMessage* msg)
{
    if (msg->hasCode()) {
        return;
    }

    W_PayReserveRes res;
    res.ParseFromArray(msg->getData(), msg->getSize());
    NXASSERT(res.IsInitialized());

    if (res.success()) {
        CCLog("W_PayReserveRes sucess");

#if (CC_TARGET_PLATFORM == CC_PLATFORM_ANDROID)
        if (IapManager::getSingleton().getPaymentType() == IapManager::ONE_STORE)
            IapManager::getSingleton().sendPaymentRequest(res.productId(), "", res.tid());
        else {
            mGoogleTid = res.productId();
            IapManager::getSingleton().launchPurchaseFlow(res.productId(), "", res.tid());
        }
```

원스토어에서 결제 정보를 IapManager로 보낼 때, launchPurchaseflow가 아니라 sendPaymentRequest이다. ~~(IabHelper에 존재하는 launchPurchaseflow와는 다르다??!)~~

- **구매 요청하기**
  - 구매를 수행하기 위해서 launchPurchaseFlowAsync 메서드를 호출합니다. 메서드 호출 시 구매하려는 인앱상품 ID, 상품명, 상품타입과 개발사가 임의로 입력한 developerPayload(최대 100byte)를 넣는데, 이 값은 결제 후에 데이터의 정합성과 부가 데이터를 확인하기 위해서 사용합니다. 그리고 파라미터로 전달하는 requestCode값은 후에 onActivityResult로 전달되는 데이터를 확인하기 위한 용도로 활용됩니다. 구매 성공시 onSuccess 리스너로 응답을 받게되며, SDK의 PurchaseData 규격으로 전달됩니다. 개발사에서는 전달받은 응답을 토대로 developerPayload 를 통하여 데이터 정합성 및 부가 데이터 확인을 진행하며, Signature 정보를 가지고 서명 검증을 진행합니다. 최종적으로 관리형 상품의 경우 상품소비를 함으로써 사용자에게 상품을 지급하도록 합니다. 원스토어는 사용자에게 할인 쿠폰, 캐시백 등의 다양한 혜택 프로모션을 진행하고 있습니다. 개발사는 구매 요청 시에 gameUserId, promotionApplicable 파라미터를 이용하여 어플리케이션을 사용하는 유저의 프로모션 참여를 제한하거나 허용 할 수 있습니다. 개발사는 해당 앱의 고유한 유저 식별 번호 및 프로모션 참여 여부를 선택하여 전달하고, 원스토어는 해당 값을 토대로 사용자의 프로모션 혜택을 적용하게 됩니다.

    > ⓘ gameUserId, promotionApplicable 파라미터는 Optional 값으로 원스토어 사업부서 담당자와 프로모션에 대해 사전협의가 된 상태에서만 사용하여야 하며, 일반적인 경우에는 값을 보내지 않아도 됩니다. 또한, 사전협의가 되어 값을 보낼 경우에도 개인정보보호 이슈가 없도록 gameUserId는 hash된 unique한 값으로 전송하여야 합니다.

```
/*
 * PurchaseClient의 LaunchPurchaseFlowAsync API (구매) 콜백 리스너
 */
PurchaseClient.PurchaseFlowListener mPurchaseFlowListener = new PurchaseClient.PurchaseFl
    @Override
    public void onSuccess(PurchaseData purchaseData) {
  Log.d(TAG, "launchPurchaseFlowAsync onSuccess, " + purchaseData.toString());
        // 구매완료 후 developer payload 검증을 수행한다.
```

IabHelper에 다음과 같이 구매 요청 변수들이 설정됨

```
// The request code used to launch purchase flow
int mRequestCode;

// The item type of the current purchase flow
String mPurchasingItemType;
```

```
 */
public void launchPurchaseFlow(Activity act, String sku, String itemType, List<String> oldSkus
        int requestCode, OnIabPurchaseFinishedListener listener, String extraData)
    throws IabAsyncInProgressException {
    checkNotDisposed();
    checkSetupDone("launchPurchaseFlow");
    flagStartAsync("launchPurchaseFlow");
    IabResult result;

    if (itemType.equals(ITEM_TYPE_SUBS) && !mSubscriptionsSupported) {
        IabResult r = new IabResult(IABHELPER_SUBSCRIPTIONS_NOT_AVAILABLE,
                "Subscriptions are not available.");
        flagEndAsync();
        if (listener != null) listener.onIabPurchaseFinished(r, null);
        return;
    }

    try {
        logDebug("Constructing buy intent for " + sku + ", item type: " + itemType);
        Bundle buyIntentBundle;
        if (oldSkus == null || oldSkus.isEmpty()) {
            // Purchasing a new item or subscription re-signup
            buyIntentBundle = mService.getBuyIntent(3, mContext.getPackageName(), sku, itemTyp
                    extraData);
        } else {
            // Subscription upgrade/downgrade
            if (!mSubscriptionUpdateSupported) {
                IabResult r = new IabResult(IABHELPER_SUBSCRIPTION_UPDATE_NOT_AVAILABLE,
```

purchaseflow 관련 코드 (IabHelper에 존재)

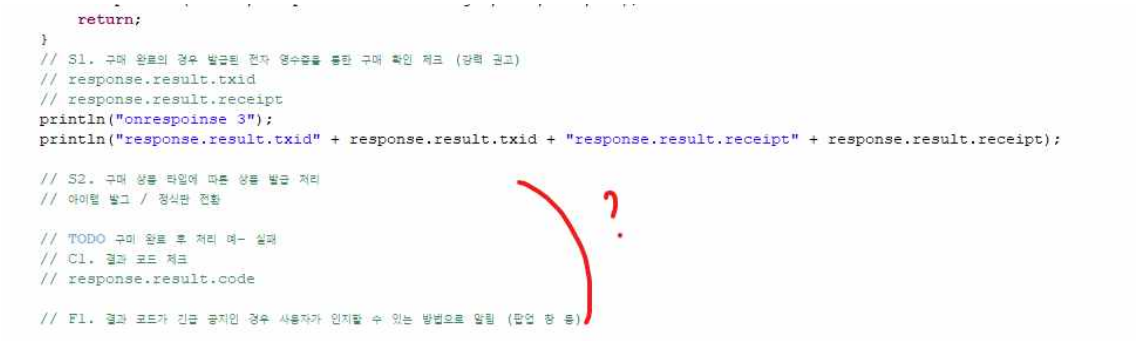
위의 코드는 예외 처리하는 것으로 보임.

실질적인 구매가 이루어지는 코드는 아래 (구매요청)

```
        PendingIntent pendingIntent = buyIntentBundle.getParcelable(RESPONSE_BUY_INTENT);
        logDebug("Launching buy intent for " + sku + ". Request code: " + requestCode);
        mRequestCode = requestCode;
        mPurchaseListener = listener;
        mPurchasingItemType = itemType;
        act.startIntentSenderForResult(pendingIntent.getIntentSender(),
                requestCode, new Intent(),
                Integer.valueOf(0), Integer.valueOf(0),
                Integer.valueOf(0));
    }
```

```
PurchaseClient.PurchaseFlowListener mPurchaseFlowListener = new PurchaseClient.PurchaseFl
    @Override
    public void onSuccess(PurchaseData purchaseData) {
  Log.d(TAG, "launchPurchaseFlowAsync onSuccess, " + purchaseData.toString());
        // 구매완료 후 developer payload 검증을 수행한다.
        if (!isValidPayload(purchaseData.getDeveloperPayload())) {
            Log.d(TAG, "launchPurchaseFlowAsync onSuccess, Payload is not valid.");
            return;
        }

        // 구매완료 후 signature 검증을 수행한다.
        boolean validPurchase = AppSecurity.isValidPurchase(purchaseData.getPurchaseData(
        if (validPurchase) {
            if (product5000.equals(purchaseData.getProductId())) {{
                // 관리형상품(inapp)은 구매 완료 후 소비를 수행한다.
                consumeItem(purchaseData);
            }
        } else {
            Log.d(TAG, "launchPurchaseFlowAsync onSuccess, Signature is not valid.");
            return;
        }
    }
```



```
    return;
}
// S1. 구매 완료의 경우 발급된 전자 영수증을 통한 구매 확인 체크 (강력 권고)
// response.result.txid
// response.result.receipt
println("onrespoinse 3");
println("response.result.txid" + response.result.txid + "response.result.receipt" + response.result.receipt);

// S2. 구매 상품 타입에 따른 상품 발급 처리
// 아이템 발급 / 정식판 전환

// TODO 구매 완료 후 처리 예- 실패
// C1. 결과 코드 체크
// response.result.code

// F1. 결과 코드가 긴급 공지인 경우 사용자가 인지할 수 있는 방법으로 알림 (팝업 창 등)
```

IapManager.**java** 파일의 구매상품 발급 처리가 비어있음
구매완료 후 처리 예가 비어있음.
Q. 기능이 필요가 없어서 구현을 하지 않은 것인지 궁금합니다.

==========2 인앱 결제 라이브러리 추가방법 V5==========

인앱결제 라이브러리 추가

- 인앱결제 라이브러리는 AIDL(Android Interface Definition Language) 파일을 추가하는 방법과 원스토어 인앱결제 SDK를 추가하는 방법이 있습니다.
  - AIDL 파일 추가
    - 'IInAppPurchaseService.aidl'을 개발사 앱 프로젝트 내 '/src/main/aidl/com/onestoe/extern/iap' 위치에 복사해 넣습니다.
    - Github에서 원스토어 인앱결제 AIDL 파일 다운로드
  - 인앱결제 SDK 추가
    - 인앱결제 API V5 (SDK V17)를 사용할 경우, 이미 'IInAppPurchaseService.aidl' 파일이 SDK 내에 포함되어 있으므로 위 '인앱결제 서비스용 AIDL 추가' 과정이 필요하지 않습니다.
      인앱결제 SDK 라이브러리 파일을 다운로드한 후 해당 파일을 개발사 앱 프로젝트 내 'libs' 디렉터리 안에 복사합니다.
    - Github에서 원스토어 인앱결제 SDK 라이브러리 다운로드

Q. 인앱 결제 라이브러리 AIDL 파일이 이미 존재하는데, 새로 추가해야하는지 궁금합니다.

Android Manifest 파일 설정

- API V5 이용을 위해 필요한 AndroidManifest.xml 파일 설정은 다음과 같습니다.
  - 필수
    API 버전 추가 : API V5를 명시적으로 선언해야 합니다.

    Example

    ```
    <application ..... >
      <meta-data
              android:name="iap:api_version"
              android:value="5" >
    ```

  - 선택
    API V5에서는 인앱상품 결제 시 팝업 형태의 결제화면도 지원합니다(이전 버전까지는 전면 결제화면만 제공).
    팝업 형태의 결제화면을 원하실 경우 meta-data를 추가하여 "iap:view_option"에 "popup" 값을 설정하시면 됩니다.
    아무런 값을 세팅하지 않을 경우 디폴트로 전체 창(android:value="full")과 같은 효과가 됩니다.

    Example

    ```
    <meta-data
        android:name="iap:view_option"
        android:value="popup | full" >
    ```

새로운 keystore를 생성하는데, 말씀해 주신바로 keystore 생성시 생긴 manifest파일을 이용하는 것으로 이해했는데, 위의 수정사항을 새로 생성된 manifest파일에 적용하는 것인지 혹은 기존에 프로젝트에 존재하는 android manifest 파일에 추가해야하는 것인지 궁금합니다.