BizSmartPush_Tutorial(Android)

Version 1.1.2

Issue Date: 2014/10/27

SK 텔레콤

Document Revision History

Revision	Comment	Reviser	Release Date
V1.0.0	1. First Publication	Nable IMS Client Group aomcsupport@nablecomm.com	2013-07-02
V1.0.1	1.6.2.1 GetErrorReason() 메소드 설명 추가 3 장 GCMID, AOMID 내용 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-07-26
V1.0.2	8.1.1.1 SendPushAck() 메소드 호출 예외 상황 시 응용 처리 가이드 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-08-01
V1.0.3	11.4 Setting of LOG_TAG. 가이드 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-08-05
V1.0.4	8.1.1 Push Message 수신 Push Message 수신 시 Register 수행 하도록 가이드 수정	Nable IMS Client Group aomcsupport@nablecomm.com	2013-10-02
V1.0.5	5.1.3 AOM BroadcastReceiver 등록 절차 5.1.4 GCM IntentService 등록 절차 가이드 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-10-23
V1.0.6	11.5 SMS Hooking 연동 방안 가이드 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-11-04
V1.0.7	6.1.3 SetBroadcastActionName 등록 추가 8.1.1 Push Message 수신 시 GetPushPayloadList 메소드 호출 하도록 수정 11.5 CheckAomService 를 통한 Aom 상태 조회 방법 추가 11.6 OnAomServiceStatus 를 통한 Aom 상태 변화 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-11-06
V1.0.8	5.1.5 Receiver 해제 절차 추가 11.2 3rd Party App 이 강제 종료 되었다가 Push Message 를 수신받아 깨어난 경우의 시나리오(BizSmartPush Intent 검사 기능) 추가	Nable IMS Client Group aomcsupport@nablecomm.com	2013-11-18
V1.0.9	3.1 설명 추가 5.1.2.3 예제 PackageName 변경 6.1.3.2 Sample Code 추가 7.3 인증 Registration 정보 삭제 7.4.2 Authentication error Return 항목 삭제	배종욱	2014-03-25

V1.1.0	5.1.2.1 과 5.1.2.3 절에 나와 있는 AndroidManifest 예시 에서 AOMC 연동 Permission 을 제거. **8 장 Push Message 수신 시 가이드 내용 수정** Push Message 수신 시 OnPushMessage() 리스너로 전달된 fragment 값이 true 인 경우 GetPushPayloadList() 메소드 호출 후 SendPushAck () 메소드 호출, fragment 값이 false 인 경우 바로 SendPushAck() 메소드 호출 하도록 가이드 수정. **9 장 Push Notification On/Off 설정 기능**을 제공하기 위한 PushNotificationOnOff() 메소드 가이드 추가. Table 2 와 Table 3 의 BizSmartPush Library 사용을 위해 필수 설정 항목 설명 추가	배종욱	2014-05-08
V1.1.1	3.1 BizSmartPush Library 를 위한 설정 항목 추가 및 변경 6 장 Biz Smart Push Library 사용을 위한 필수 설정 항목 추가 및 변경	이순우	2014-05-30
V1.1.2	5.1.2.1 AOM 관련 AndroidManifest 설정 변경 5.1.2.3 BizSmartPush관련 AndroidManifest 설정 변경 5.1.2.4 AndroidManifest 예제의 Permission 변경	이순우	2014-10-27

Table of Contents

Table

Figure

1 Introduction

1.1 Purpose

본 문서는 Android BizSmartPush API 을 이용하여 Android BizSmartPush Application 을 개발하는데 필요한 정보를 제공하는 것을 목적으로 한다.

본 문서에서는 Android BizSmartPush Library 을 사용하기 위한 API 를 설명하지만, 각 API 에 대한 상세한 정보를 제공하지는 않는다. 세부적인 API 사용법은 API Reference 를 참고한다.

본 문서의 내용은 SKT 의 BizSmartPush 서비스 제공 계획 및 관련 표준 규격의 갱신 등에 따라 추후 보완 및 변경될 수 있다.

규격에 구체적으로 명시되어 있지 않은 사항은 참고 문헌의 규격 사항에 준하며, 진행 중인 규격의 경우 최종 규격을 따르도록 한다.

1.2 Audience

이 문서의 대상은 외부에 비공개 및 기밀 유지를 기본으로 한다.

1.3 References

참고 문서와 본 요구사항에 상충되는 부분은 본 문서를 준수한다. 단, 본 요구사항에 언급되지 않는 사항들은 아래 문서를 준수한다.

Table 1 References Documents

기관	References
Google apis	https://code.google.com/apis/console/
Android Developers	http://developer.android.com/google/gcm/index.html/
SK Telecom	SKT-AOM-Client-개발가이드 v.1.2(20120716). pdf
JSON(JavaScript Object Notation)	http://www.json.org/json-ko.html

1.4 약어 및 용어

- SP Service Provider
- OMA Open Mobile Alliance
- TLS Transport Layer Security
- QoS Quality of Service
- BSP BizSmartPush
- GCM Google Cloud Messging
- APNs Apple Push Notification Service

2 BizSmartPush 란?

2.1 서비스 개요

BizSmartPush Platform 의 목적은 다양한 Mobile O/S, MNO, 서비스에 의존되는 기존의 Push 서비스를 통합하여 추상화 함으로서 서비스 제공 및 연동 소프트웨어 개발의 편의를 제공하고, SP 가 전송하는 모든 메시지에 대하여 100% 착신을 보장하여 서비스에 대한 신뢰성을 제공한다.

다양한 종류의 Mobile 서비스를 제공하기 위한 App(lication)이 등장하였고 이러한 App 이 필요한 정보를 신속히 얻고자 자신의 서비스 제공자로 Mobile 네트워크를 이용하여 통신을 시도하고 이는 시도호 및 트래픽 증가와 Mobile 기기의 배터리 및 자원을 점유하게 된다. 이러한 문제를 해결하고자 등장한 서비스가 Push 서비스 이다.

하지만 현재의 Push 서비스는 Mobile O/S 제공자 및 통신사에 의해 각각의 다른 프로토콜로 서비스 되고 있다. 이로 인해 동일한 Push 서비스를 제공하기 위하여 서로 다른 Push Infra 와 연동해야 하는 문제가 발생한다.
BizSmartPush Platform 은 일종의 Proxy 역할을 하여 SP 로 하여금 다수의 Push Infra 를 하나의 프로토콜로 연동할 수 있도록 돕는다.

또한, Push Infra 의 특성 상 SP 로 부터 수신한 모든 메시지에 대하여 100% 의 착신 성공을 보장할 수 없다. 그로 인해 SP 는 Smart Device 로 전송되는 모든 메시지를 자체 구축한 Database 에 저장하고 있어야 하였다.
SNS 같은 Service 는 Push Notification 을 수신할 경우 자체 SP Server 에서 미 수신된 메시지를 모두 수신하는 구조이지만, 현재 SMS 로 제공되고 있는 금융 관련 서비스는 모든 메시지에 대하여 100% 착신이 보장되어야 하므로 Push Infra 에 전적으로 의존하기엔 QoS 를 보장할 수 없다.

따라서, BizSmartPush Platform 은 SP 가 Push Notification 1 회 전송으로 100% 착신 성공률을 제공하도록 한다. 자체 Push Solution 을 구축할 수 없는 소규모 SP 또는 100% 에 미치지 못하는 Push Notification 착신 성공률로 인하여 SMS 기반 메시징 서비스를 제공할 수 밖에 없는 SP 의 서비스 유치를 촉진하도록 한다.

.

2.2 시스템 구성도

2.2.1 전체 구성도

BizSmartPush Platform 은 Push Infra 와 연동하여 Service Provider 에서 전달하는 Push 를 Relay 하기 위한 Proxy 역할을 한다.

Service Provider 입장에선 Smart Device O/S 별 또는 통신사별 Push Infra 와 각각 연동할 필요 없이 단순히 BizSmartPush Platform 과 연동하여 Smart Device 로 Push 를 전송할 수 있다.

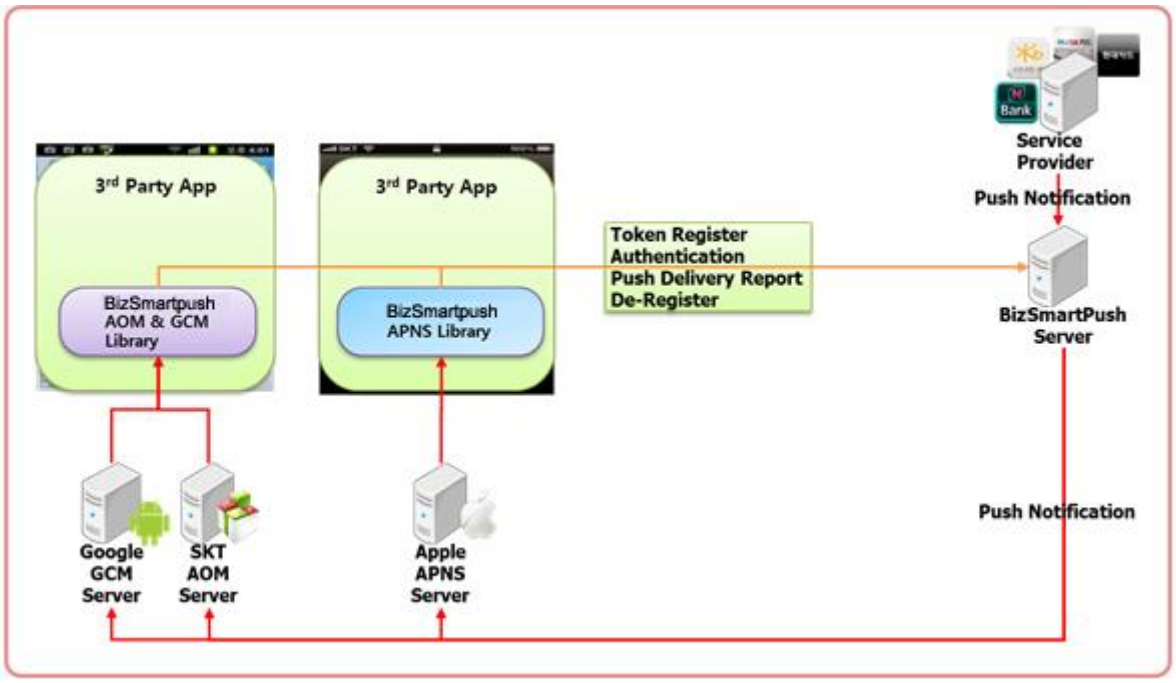


Figure 1 BizSmartPush 전체 구성도

2.2.2 클라이언트 구성도

BizSmartPush Library 를 사용하는 3rd Party 클라이언트는 GCM, APNS, AOMS 서버와 Internet 으로 연동하게 된다.

BizSmartPush Client Library 는 3rd Party App 의 라이브러리로 포함되어 연동한다.

- Internet
  - → Push Infra(GCM, APNS, AOMS) 와의 Internet 연동을 위하여 HTTPS & TLS 로 연동한다.
  - → BSP 서버 와의 Internet 연동을 위하여 HTTPS 로 연동한다.

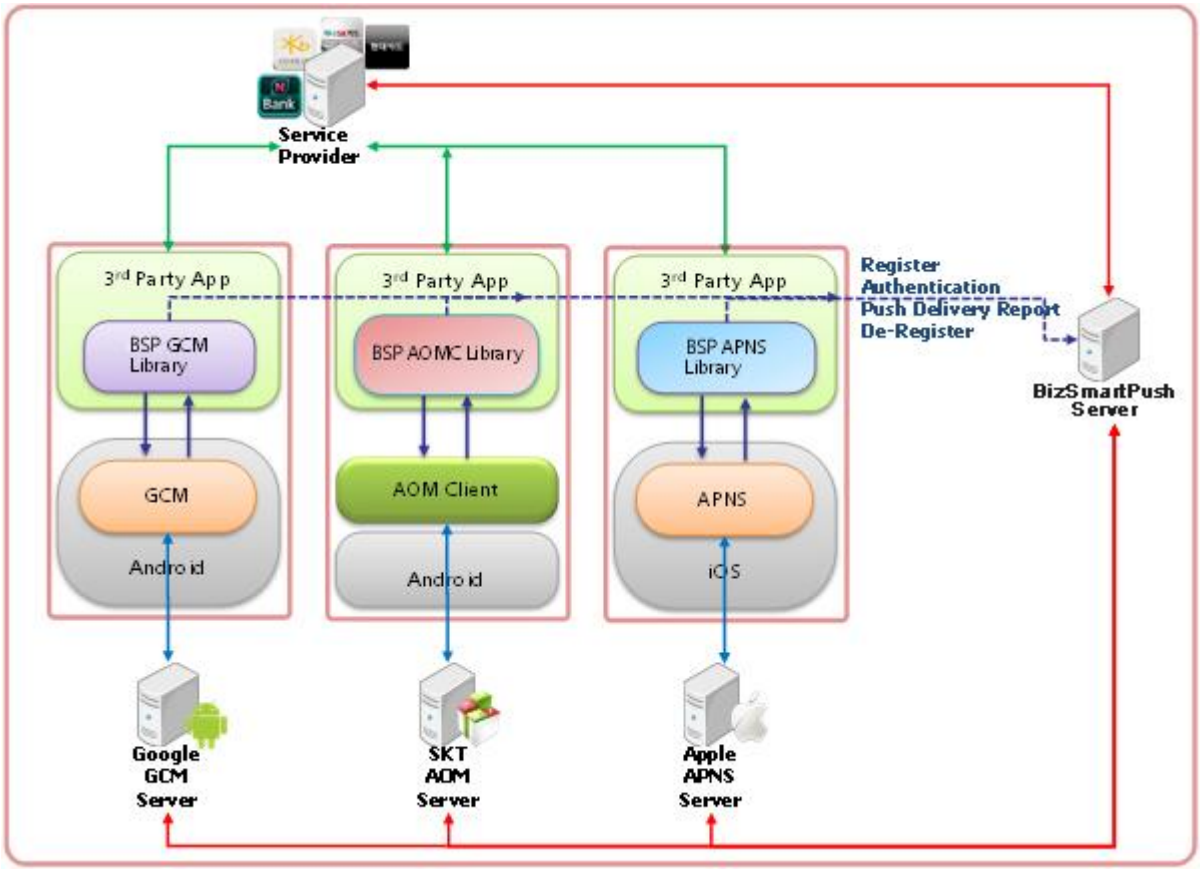


Figure 2 BizSmartPush Client 구성도

3 Preparation for using API

3.1 BizSmartPush Library 를 위한 설정

BizSmartPush Library 을 사용하기 위해서는 단말 별로 BizSmartPush 서버에 계정이 등록되어 있어야 하며, 등록된 BizSmartPush 서버 및 각 계정에 대한 정보가 준비 되어 있어야 한다.

Registration 과정 시 인증정보를 요구하는 경우가 있으며, 이를 위해 단말은 BizSmartPush server IP address 와 인증과정시 단말로 할당된 AuthCode 정보가 필요하다.

BizSmartPush Library 사용을 위해, 단말이 알아야 할 필수 정보는 아래 [표 2]과 같다.

Table 2 BizSmartPush Library 사용을 위해 3rd Party APP 이 알아야 할 정보

Section	항목	설명	예
BIZPUSH	Version	BizSmartPush Library 버전 3rd Party APP 가이드 : 3rd Party App 에서 설정할 필요 없다	Version=100
	PushType	사용할 Push Service 종류 Register 호출 시 입력 Push type (1=AOM,2=GCM,3=AUTO,4=APNS)	PushType=
	ServerInfo	Biz push server ip 주소 설정 상용, Sendbox 두가지 버전을 제공하며 DNS 정보는 아래와 같습니다. 개발시 Sandbox 에서 충분히 테스트 후 상용서버로 변경을 부탁드립니다. **Sandbox DNS** URL : clnt.sb.smartbizpush.com IP/PORT : 220.103.203.75:8443 **상용서버 DNS** URL : clnt.smartbizpush.com IP/PORT :220.103.235.93:8443 3rd Party APP 가이드: 해당 도메인을 DNS 쿼리를 거쳐 획득한, IP 와 Port 를 입력한다. 입력시 https:// 로 해주시고 Port:8443 고정값	ServerInfo=

		입니다. 예) ServerInfo=https://220.103.203.75:8443	
	requestUri	Request URI 정보 3rd Party APP 가이드 : 3rd Party App 에서 설정할 필요 없다	requestUri=/json
	AppID	T dev inside 인증서 등록시 입력한 APP ID 를 입력하시면 됩니다. **App ID(Apple APNS App ID 를 설정할 때 사용. 즉, Android 에서는 Setting 하지 않음)**	AppID=
	UserID	Push 전송 시 API 에서 사용할 servicekey 개인 식별자 (unique) 전화번호, Mac, Email, etc	UserID=
	PhoneNumber	SMS 용도로 필요한 속성으로 현재 버전에서는 제공하지 않습니다. 향후 SMS 제공되면 폰번호를 입력하시면 됩니다. 이부분은 "00000000000" 11 자리 값으로 채워서 보내주시면 됩니다.	PhoneNumber=
	OperatorName	단말이 가입된 통신사를 식별하는 값. (MCC + MNC) 사업자 번호 (SKT=45005,KT=45008,LGU=45006) 테플릿 디바이스는 통신사 정보를 확인할수 없을 경우 3 개중 1 개를 선택해서 넣어주시면됩니다. Biz Smart Push 에서는 과금을 위한 통계용도로 사용합니다.	OperatorName=
	AuthKey	T dev inside 프로젝트 Keys 상세화면에서 제공하는 Biz Smart Push Key 정보를 입력하시면됩니다. BSP 서버에서 관리하는 Unique ID 정보 예) 00000218271396343325000130391490	AuthKey=
	GcmID	T dev inside 인증서 등록시 입력한 GCM ID 를 입력하시면 됩니다.	GcmID=
	AomID	T dev inside 에서 발급된 AOM ID 를 입력해주시면 됩니다. Sandbox AOM 인증서가 우선발급되며 Sandbox 에서 정상적으로 테스트 완료 되시면 상용 AOM 인증서를 발급 받으시면됩니다.	AomID=
TIMER	TransactionTimeout	BizSmartPush 서버의 response 가 n 초까지 없는 경우 에러 처리	[TIMER] TransactionTimeout=20

		3rd Party APP 가이드: 배포되는 파일 Smartbizpush.ini 에서 TransactionTimeout 에 값을 설정 한다.	
	TokenWaitTimeout	token 에 대한 응답이 n 초까지 없는 경우 에러 처리 3rd Party APP 가이드: 배포되는 파일 Smartbizpush.ini 에서 TokenWaitTimeout 에 값을 설정 한다.	[TIMER] TokenWaitTimeout=10
CONFIG	AUTOSAVE	자동으로 Smartbizpush.ini 파일에 설정을 저장할지 여부 선택.	AUTOSAVE=0
LOGGER	Level	Log level 설정 (VERBOSE / DEBUG / INFO / WARN / ERROR)	Level=VERBOSE
	Enable	출력 로그를 사용할지 여부 결정	Enable=1
	EnableDebugWindow	디버그 창에서 로그를 출력 할지 여부 결정.	EnableDebugWindow=1
	EnableFile	외부에 로그 파일을 저장할지 여부 결정. 안드로이드의 경우, 쓰기 가능한 외장형 스토리지 (예: SD 카드)가 장착 될 때 출력 로그를 파일로 저장할 수 있다.	EnableFile=1
CREASH_REPORTER	Enable	CrashRepoter 동작 여부	Enable=1
	Recipient	Report 받을 e-mail 주소	Recipient=aomcsupport@nablecomm.com
	AttachLogcat	Logcat Log 첨부 여부	AttachLogcat=1

3.2 Listener 구현 시 주의사항

BizSmartPush Library 은 Listener 구현 부에서 BizSmartPush 메소드 호출을 금지한다.

내부적으로 PostMessage 등을 이용하여 BizSmartPush Library 동작에 이상이 없도록 한다.

금지된 구현방법은 아래와 같다.

금지사항 1. Listener 구현 부에서 BizSmartPush 를 수행하는 경우

```
private BizPushListener m_bizpushListener = new BizPushListener() {

@Override
        public void OnPushMessage(String msg) {
```

```
                    int ret = m_bizPushTester.m_bizpush.SendPushAck(msg);
            }
};
```

금지사항 2. Listener 구현 부에서 다른 작업을 하는 경우

```
private BizPushListener m_bizpushListener = new BizPushListener() {

@Override
public void OnState(BP_PUSH_TYPE type, BP_STATE state, int code) {
    if (state == Ims.State.STARTED)  {
            WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
            if (wifiManager.isWifiEnabled() == true) {
                    WifiInfo info = wifiManager.getConnectionInfo();
            }
            String ip = null;
            for (Enumeration<NetworkInterface> en = NetworkInterface.getNetworkInterfaces();
en.hasMoreElements();) {
              NetworkInterface intf = en.nextElement();
              for (Enumeration<InetAddress> enumIpAddr = intf.getInetAddresses();
enumIpAddr.hasMoreElements();) {
                  InetAddress inetAddress = enumIpAddr.nextElement();

                  if (null != inetAddress && !inetAddress.isLoopbackAddress()) {
                      return inetAddress.getHostAddress().toString();
                  }
              }
          }
    }
  }
};
```

4 Deliverable

BizSmartPush Library 는 jar, so 파일, Release Note 및 smartbizpush.ini 파일로 구성되어 압축된 형식으로 제공된다.

아래는 jar 파일과 so 의 구성이다.

1. Jar files

Jar 파일명	용도
bizpush.jar	BizSmartPush Native Library 를 사용하기 위한 Java API 를 제공.
gcm.jar	gcm Native Library 를 사용하기 위한 Java API 제공.

2. So files

so 파일명	용도
libBizPushEngine.so	BizSmartPush Native Library 의 묶음
libNAF.so	JNI wrapping 기능과 공통 Library 의 묶음

3. Ini files

ini 파일명	용도
smartbizpush.ini	BizSmartPush Library 를 사용하기 위한 설정값 저장을 제공.

. so 파일과 .jar 파일을 아래의 폴더에 복사한다. 폴더가 존재하지 않으면, 폴더를 생성한다.

```
1. so files: <Application Project Root>/project/libs/armeabi/*.so
2. Jar file: <Application Project Root>/project/jar/*.jar
```

. jar 파일은 설정의 Build path 에 추가하고, Referenced Libraries 에 등록해야 한다.

BizSmartPush API 는 ini 파일을 각 설정을 저장하기 위해 사용한다. Ini 파일은 아래의 경로에 저장해야 한다.

```
1. smartbizpush.ini file: <Application Project Root>/project/assets/ smartbizpush.ini
```

5 How to use BizSmartPush API

3rd Party APP 은 BizSmartPush API 를 이용하기 위해 아래와 같은 작업을 선행한다.

5.1 Android 구현 준비

5.1.1.1 BizSmartPush Package import 추가

Package 는 com.smartbizpush.push 이며 BizSmartPush API 를 호출하는 class 에 import 를 추가한다.

```
import com.smartbizpush.push;
```

5.1.2 AndroidManifest.xml

5.1.2.1 AOM 관련 AndroidManifest 설정

```
<!-- ############ AOM : Start  ############ -->
 <uses-permission android:name="com.skt.aom.permission.AOM_RECEIVE" />
<!-- ############ AOM : End  ############ -->
```

5.1.2.2 GCM 관련 AndroidManifest 설정

3rd Party APP 에서 GCM Push 를 받기위해서 다음과 같은 Permission 을 설정한다.

```
<permission android:name="my_app_package.permission.C2D_MESSAGE"android:protectionLevel="signature" />
<uses-permission android:name="my_app_package.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

broadcast receiver 등록하기 위해서 아래와 같이 설정한다. (GCM message 가 도착하면 해당 broadcast 로 호출 된다.)

```
<receiver
android:name="com.google.android.gcm.android:name="com.google.android.gcm.GCMBroadcastReceiver"
android:permission="com.google.android.c2dm.permission.SEND" >
 <intent-filter>
  <action android:name="com.google.android.c2dm.intent.RECEIVE" />
  <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
  <category android:name="my_app_package" />
 </intent-filter>
 </receiver>
```

IntentService 등록하기 위해서 아래와 같이 설정한다. (해당 service 는 GCM library 에 있는 GCMBroadcastReceiver 에 의해 호출된다.)

```
<service android:name=".GCMIntentService" />
```

5.1.2.3 BizSmartPush 관련 AndroidManifest 설정

BizSmartPush 의 접근 권한 설정은 3rd Party App 의 **"Package 명+.BIZPUSH_MESSAGE"**로 설정한다.
예를 들어 3rd Party App 의 Package 명이 com.nable.push.tester 이면 BizSmartPush 접근 권한은 com.nable.push.tester.BIZPUSH_MESSAGE 로 설정하여 사용한다.
또한 설정한 접근 권한을 사용하기 위해서 use-permision 을 정의한다.

```
<!-- ############ Smart Biz Push 의 접근 권한 ############### -->
<permission android:name="앱패키지명.BIZPUSH_MESSAGE" android:protectionLevel="signature" />
<uses-permission android:name="앱패키지명.BIZPUSH_MESSAGE"/>
<!-- ############ Smart Biz Push : End ################ -->
```

5.1.2.4 AndroidManifest 예제

아래는 참고용 AndroidManifest 이며 반드시 3rd Party App 의 Package 이름에 맞게 이용하여야 한다.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    package="com.nable.training.bizpush.tester"
    android:versionCode="100"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="14" />

<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.READ_LOGS" />

  <!-- ############ AOM : Start  ############ -->
 <uses-permission android:name="com.skt.aom.permission.AOM_RECEIVE" />
 <!-- ############ AOM : End  ############ -->

  <!-- ############ Smart Biz Push 의 접근 권한 ############### -->
  <!—Permission Name 은 3rd Party App 의 패키지명+.BIZPUSH_MESSAGE 으로 설정한다. -->
 <permission                android:name="                com.nable.training.bizpush.tester.BIZPUSH_MESSAGE"
android:protectionLevel="signature" />
  <uses-permission android:name=" com.nable.training.bizpush.tester.BIZPUSH_MESSAGE"/>
   <!-- ############ Smart Biz Push : End ################ -->

<application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:largeHeap="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.nable.training.bizpush.tester.BizPushTester"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="com.nable.trainning.bizpush.AOM_MESSAGE"/>
```

```
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
        <intent-filter>
            <action android:name="com.nable.trainning.bizpush.GCM_MESSAGE"/>
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>


    </activity><receiver android:name="com.nable.stub.NativeWalkReceiver" android:process=":remote" />
<receiver android:name="com.nable.training.bizpush.tester.TesterBroadcastReceiver"
android:enabled="true"><intent-filter>
            <action android:name="com.nable.trainning.bizpush.intent"/>
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
    </receiver>
    <!-- ########### GCM : Start  ########### -->
<!--
  BroadcastReceiver that will receive intents from GCM
  services and handle them to the custom IntentService.

  The com.google.android.c2dm.permission.SEND permission is necessary
  so only GCM services can send data messages for the app.
-->
<receiver
    android:name="com.google.android.gcm.GCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <!-- Receives the actual messages. -->
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <!-- Receives the registration id. -->
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />

        <category android:name="com.nable.training.bizpush.tester" /></intent-filter>
</receiver>

    <!--Application-specific subclass of GCMBaseIntentService that will
      handle received messages.

      By default, it must be named .GCMIntentService, unless the
```

```
        application uses a custom BroadcastReceiver that redefines its name.
      -->
      <service android:name=".GCMIntentService" />
    <!-- ########### GCM : End  ########### -->

    <!-- ########### AOM : End  ########### -->
    <receiver android:name=".AOMBroadcastReceiver" android:enabled="true">
                <!-- 부팅 완료 intent 수신함 -->
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.REGISTRATION" />
              <category android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.MESSAGE" />
              <category android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.SERVICE_AVAILABLE" /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.SERVICE_AVAILABILITY" /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.SESSION_CHANGED" /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.SERVICE_UNAVAILABLE" /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.STATUS_OF_SERVICE" /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
              <action android:name="com.skt.aom.intent.receive.STATUS_OF_MY_PUSH" /><category
android:name="com.nable.training.bizpush.tester" />
```

```
            </intent-filter>
            <intent-filter>
                <action                 android:name="com.skt.aom.intent.receive.RE_REGISTER"                 /><category
android:name="com.nable.training.bizpush.tester" />
            </intent-filter>
            <intent-filter>
                <action                 android:name="com.skt.aom.intent.receive.KEEP_ALIVE"                 /><category
android:name="com.nable.training.bizpush.tester" /></intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.CAMERA" />

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />

    <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
     <!-- ############ GCM : Start  ############ -->

    <!-- GCM connects to Google Services. -->
    <uses-permission android:name="android.permission.INTERNET" />

    <!-- GCM requires a Google account. -->
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />

    <!--Creates a custom permission so only this app can receive its messages.

     NOTE: the permission *must* be called PACKAGE.permission.C2D_MESSAGE,
           where PACKAGE is the application's package name.
    -->
```

```
    <permission
        android:name="com.nable.training.bizpush.tester.permission.C2D_MESSAGE"
        android:protectionLevel="signature" /><uses-permission
        android:name="com.nable.training.bizpush.tester.permission.C2D_MESSAGE" />

    <!-- This app has permission to register and receive data message. -->
    <uses-permission
        android:name="com.google.android.c2dm.permission.RECEIVE" />
     <!-- ############ GCM : End  ############ -->

</manifest>
```

5.1.3 AOM BroadcastReceiver 등록 절차

3rd Party App 에서는 반드시 AOM BroadcastReceiver 를 추가해 주어야 한다.

AOM BroadcastReceiver 를 등록해야만 AOM Client 로부터 토큰 발급이나, 응답을 받을 수 있다.

5.1.3.1 Basic Procedure

아래는 AOM BroadcastReceiver 를 등록하기 위한 예이다.

[사용예제]

```
import com.smartbizpush.push.AOMReceiverAPI;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class AOMBroadcastReceiver extends BroadcastReceiver {

 @Override
 public void onReceive(Context context, Intent intent2) {
            AOMReceiverAPI.onReceive(context, intent2);
 }
}
```

5.1.4 GCM IntentService 등록 절차

3rd Party App 에서는 반드시 GCM IntentService 를 추가해 주어야 한다.

GCM IntentService 를 등록해야만 GCM Service 로부터 응답을 받을 수 있다.

5.1.4.1 Basic Procedure

아래는 GCM IntentService 를 등록하기 위한 예이다.

[사용예제]

```
import android.content.Context;
import android.content.Intent;
import com.google.android.gcm.GCMBaseIntentService;
import com.smartbizpush.push.GCMServiceAPI;

/**
 * IntentService responsible for handling GCM messages.
 */
public class GCMIntentService extends GCMBaseIntentService {


    @Override
    protected void onRegistered(Context context, String registrationId) {
            GCMServiceAPI.onRegistered(context, registrationId);
    }

    @Override
    protected void onUnregistered(Context context, String registrationId) {
            GCMServiceAPI.onUnregistered(context, registrationId);
    }


    @Override
    protected void onMessage(Context context, Intent intent2) {
            GCMServiceAPI.onMessage(context, intent2);
    }
```

```
    @Override
    protected void onDeletedMessages(Context context, int total) {
            GCMServiceAPI.onDeletedMessages(context, total);
    }

    @Override
    public void onError(Context context, String errorId) {
            GCMServiceAPI.onError(context, errorId);
    }

    @Override
    protected boolean onRecoverableError(Context context, String errorId) {
            return GCMServiceAPI.onRecoverableError(context, errorId);

    }
}
```

5.1.5 Receiver 해제 절차

3rd Party App 에서는 App 종료를 하기 전 반드시 등록한 Receiver 해제 하고 App 종료를 해야 한다.

등록한 Receiver 를 해제 해야만 Android Intenet Receiver Leaked 현상을 방지 한다.

Receiver 를 해제 하기 위해서는 아래와 같이 BizSmartPush API 인 Destroy 를 호출한다.

[API Prototype]

```
/**
    * 응용에서 등록된 Receiver 를 해제 하는 기능을 제공
    * 3rd Party App 에서는 App 이 종료되기 전에 등록된 Receiver 를 해제해 주어야 한다.
    */
    public void Destroy()
```

5.1.5.1 Basic Procedure

아래는 Receiver 를 해제하기 위한 예이다.

[사용예제]

```
@Override
protected void onDestroy() {
		// TODO Auto-generated method stub
		Log.e(LOG_TAG, "onDestroy()");
		m_bizpush.Destory();
		super.onDestroy();
}
```

5.2 BizSmartPush Library API Overview

5.2.1 BizSmartPush State Machine

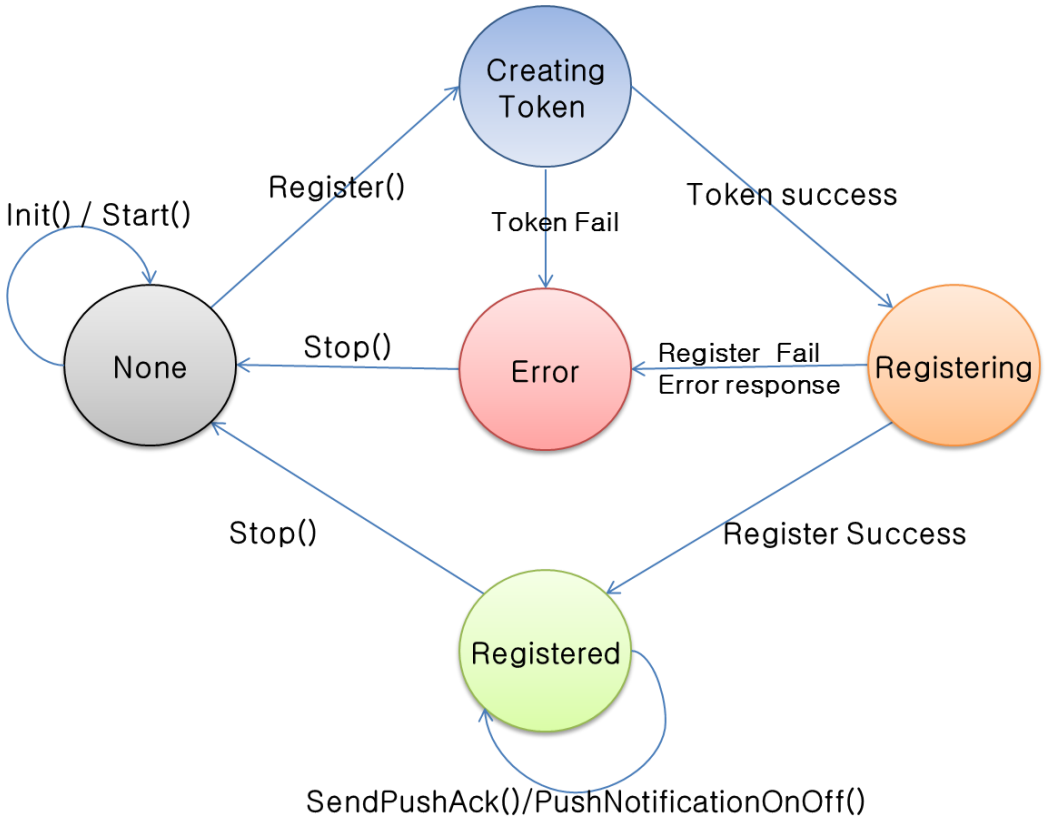


Figure 3 BizSmartPush State Diagram

BizSmartPush 는 위 그림과 같이 크게 5 개의 상태를 가지고 있다

- **None 상태 : Start 나 Init 을 하지 않는 상태이다.**
- **CreatingToken 상태 : 응용에서 Register()를 호출한 상태로 결과는 callback 으로 호출된다.**
- **Registering 상태 : token 이 정상적으로 발급이 된경우 서버에 register 를 요청한 상태이다.**
- **Registered 상태 : 서버에 정상적으로 등록이 성공된 상태이다.**
- **UnRegistered 상태 : 응용에서 UnRegiste()를 호출한 상태이다.**
- **Error 상태 : Register() 호출시 error 가 발생하거나 서버에서 error 응답이 온 상태이다.**

그외 추가 상태는 아래와 같다.

- **Standby 상태 : 응용에서 Init()을 호출한 상태이다.**

5.2.2 BizSmartPush API Life Cycle

아래 그림은 BizSmartPush API 에 사용에 대한 간략한 Life Cycle 을 나타낸다.

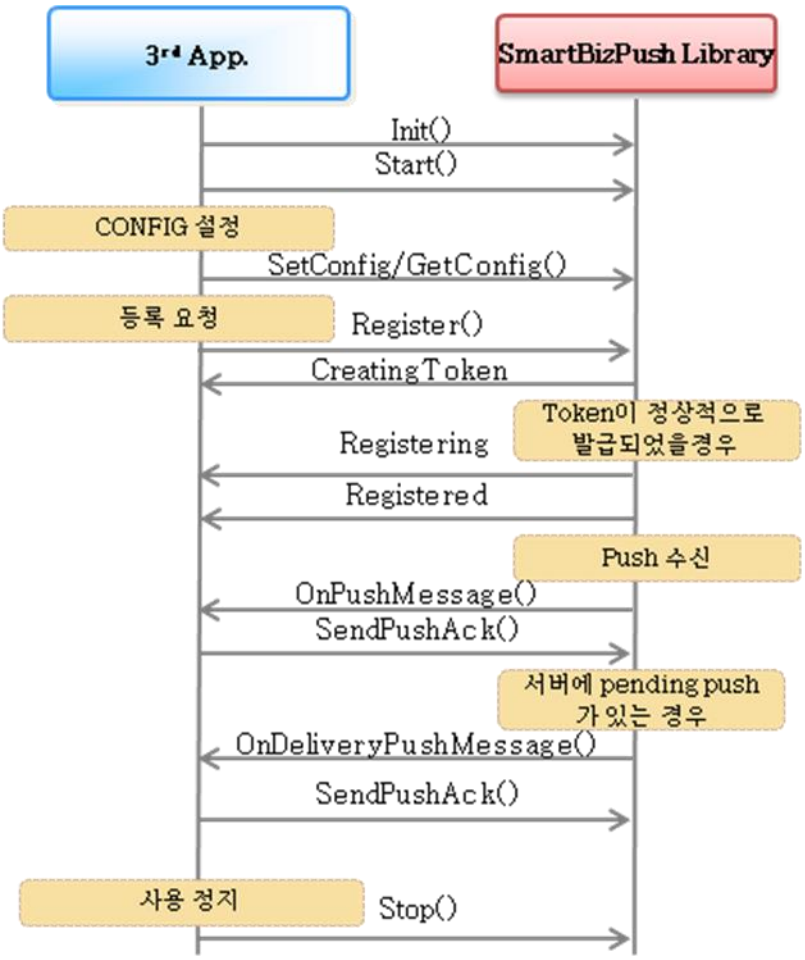


Figure 4 BizSmartPush API Life-Cycle

응용은 Register API 를 호출하기 전에 Init API 호출이후 SetConfig API 를 사용하여 라이브러리에 필요한 값을 설정한다.

설정이 끝난 후 응용은 Reigster()를 호출하여 서버에 단말 정보 등록을 요청한다. 등록이 성공하면 응용으로 Registered 상태가 Listener 를 통해 전달된다. BizSmartPush 라이브러리는 Push 를 수신하면 OnPushMessage 를 Listener 를 통해 호출하며, 응용은 SendPushAck 로 BSP 서버에게 Push 가 정상적으로 수신되었음을 알려준다. 만약 서버에 SendPushAck 가 수신되지 않은 Push 메시지가 남아 있을 경우 BizSmartPush 라이브러리는 OnDeliveryPushMessage 를 Listener 를 통해 호출한다.

사용 정지 시 Stop()을 호출하면 BizSmartPush 라이브러리는 모든 동작을 종료한다.

5.2.3 API Invoke Model

BizSmartPush API 는 서버와 메시지 교환 시 대기를 피하기 위해 비동기식으로 제공한다.

Register() 또는 Unregister() 호출 시 Return 이 Success 면 서버에 정상적으로 메시지를 보냈다는 의미이고, 최종 서버의 회신 결과는 State Listener 로 통해 전달된다.

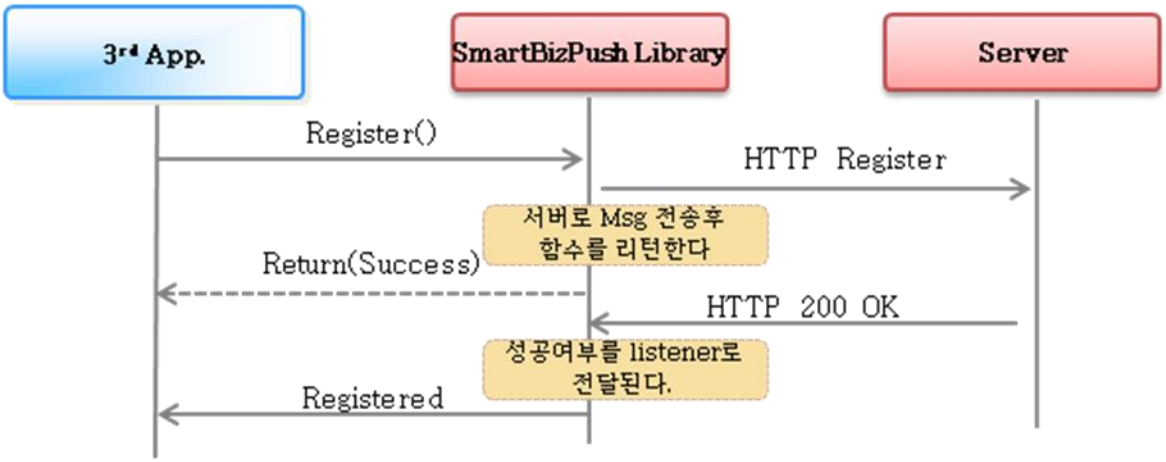


Figure 5 BizSmartPush API Invoke Model

6 Configuration 및 시작

BizSmartPush API 을 사용하기 위해서는 먼저, BizPushListener implement 구현, BizSmartPush Library 초기화 이후 BizSmartPush Engine 및 Registration 에 필요한 정보들을 설정해야 한다.

본 절에서는 이에 필요한 기본 절차 및 설정 항목, 설정 방법을 설명한다.

6.1 Configuration

6.1.1 Basic Procedure

BizSmartPush API 를 사용하기 위한 기본 절차는 다음과 같은 단계가 필요하다.

- **BizPushListener implement 구현**
- **BizSmartPush Library 초기화**
- **Configuration 설정**
- **BizSmartPush 시작(Start)**

이 기본 절차는 다른 BizSmartPush API 를 사용하기 전에 필수 수행해야한다.

6.1.2 BizSmartPush 초기화

3rd Party APP 은 BizSmartPush API 를 사용하기 위해서 BizSmartPush 객체를 생성한다. 생성이 끝나면 Init()메소드를 이용하여 BizSmartPush Library 을 초기화 한다. 인자로는 context, smartbizpush.ini 와 BizPushListener 를 등록한다.

```
// BizSmartPush 객체 생성(Singleton)
BizPush m_bizpush = BizPush.getBizPush();

// BizSmartPush Library 초기화(Init())
BP_RET ret = m_bizPushTester.m_bizpush.Init(m_bizPushTester, "smartbizpush.ini", m_bizpushListener);

// BizPushListener implement 구현
private BizPushListener m_bizpushListener = new BizPushListener() {
```

```
@Override
public void OnState(BP_PUSH_TYPE type, BP_STATE state, int code) {
		// TODO Auto-generated method stub
		Bundle data = new Bundle();
		data.putInt("type", state.getValue());
		data.putInt("state", state.getValue());
		data.putInt("code", code);

		Message msg = Message.obtain();
		msg.what = MessageID.m_nMSG_ID_ONSTATE;
		msg.setData(data);

		m_handler.sendMessage(msg);
}

@Override
public void OnReceiveResult(int responseCode, int transactionID,
				String content) {
		// TODO Auto-generated method stub

}

@Override
public void OnPushAckResult(int id, int code) {
		// TODO Auto-generated method stub

}

@Override
public void OnDeliveryPushMessage(String[] msgs) {
		// TODO Auto-generated method stub
		Bundle data = new Bundle();
		data.putString("strFullMsg", fullMsg);
		data.putInt("msgLength", m_msgLength);

		Message msg = Message.obtain();
		msg.what = MessageID.m_nMSG_ID_ONDELIVERYPUSHMESSAGE;
		msg.setData(data);
```

```
                m_handler.sendMessage(msg);
        }

        @Override
        public void OnPushMessage(String msg) {
                // TODO Auto-generated method stub
                Bundle data = new Bundle();
                data.putString("strFullMsg", fullMsg);
                data.putInt("msgLength", m_msgLength);

                Message pushMsg = Message.obtain();
                pushMsg.what = MessageID.m_nMSG_ID_ONPUSHMESSAGE;
                pushMsg.setData(data);

                m_handler.sendMessage(pushMsg);
        }

        @Override
        public void OnAuthenticationResult(int id, int code, int timeout) {
                // TODO Auto-generated method stub
        }

        @Override
        public void OnAomCheckStatus(AOM_STATUS_REASON error) {
                // TODO Auto-generated method stub
                m_bizPushTester.PrintLog("#% OnAomCheckStatus error :" + error.getValue());
        }

        @Override
        public void OnAomServiceStatus(AOM_SERVICE_STATUS status) {
                // TODO Auto-generated method stub
                m_bizPushTester.PrintLog("#% OnAomServiceStatus status :" + status.getValue());
        }
};
```

6.1.3 BizSmartPush 환경 설정

6.1.3.1 BizSmartPush Library 사용을 위해 필수 설정 항목 정리

BizSmartPush Library 를 위해 설정 할 항목은 다음과 같으며, 설정/조회가 가능하다.

Table 3 BizSmartPush Library 사용을 위해 필수 설정 항목

Section	항목	설명	예	필수
	PushType	Push type 설정 (1=AOM,2=GCM,3=AUTO,4=APNS)	PushType=	O
	ServerInfo	BizPush server ip 설정 [예시] 상용서버 DNS : clnt.smartbizpush.com Sanbox 서버 DNS : clnt.sb.smartbizpush.com BSP 서버 도메인 주소는 서버 담당자에게 문의 3rd Party APP 가이드 : DNS 쿼리를 거쳐 획득한, IP 와 Port 를 입력 (예시:ServerInfo=**https://211.115.7.234:8443**)	ServerInfo=	O
	AppID	APNS 의 App ID 설정 3rd Party APP 가이드 : iOS 에서 반드시 설정해야 할 항목	AppID=	O (APNS 사용시)
	UserID	전화번호, Mac, Email, etc 로 설정 3rd Party APP 가이드 : UserID 는 BizPush Server 에서 Client 로 PUSH 전송 시에 사용할 고유 service_key 값을 의미한다	UserID=	O
	PhoneNumber	전화번호 설정 SMS 를 수신할 단말 전화번호	PhoneNumber=	O
	OperatorName	MCC/MNC 5 자리 사업자 번호 3rd Party APP 가이드 : USIM 에서 MCC/MNC 정보를 획득하여 설정한다. (ex:SKT=45005,KT=45008,LGU=45006)	OperatorName=	O
	AuthKey	3rd party app 의 service key 설정 3rd Party APP 가이드 : AuthKey 설정은 7.3.1 인증 Registration 을 참조한다.	AuthKey=	O
	GcmID	GCM 의 App ID 설정 (Android 에서 반드시 설정해야 할 항목)	GcmID=	O (GCM 사용시)

	AomID	AOM 의 App ID 설정 (Android 에서 반드시 설정해야 할 항목)	AomID=	O (AOM 사용시)
LOGGER	Level	Log level 설정 (ex:VERBOSE / DEBUG / INFO / WARN / ERROR)	Level=VERBOSE	
	Enable	출력 로그를 사용할지 여부 결정	Enable=1	
	EnableDebugWindow	디버그 창에서 로그를 출력 할지 여부 결정.	EnableDebugWindow=1	
	EnableFile	외부에 로그 파일 저장 여부 설정 (안드로이드의 경우, 쓰기 가능한 외장형 스토리지(예 : SD 카드)가 장착 될 때 출력 로그를 파일로 저장)	EnableFile=1	
CREASH_REPORTER	Enable	CrashRepoter 동작 설정	Enable=1	
	Recipient	Report 받을 e-mail 주소 설정	Recipient=aomcsupport@nablecomm.com	
	AttachLogcat	Logcat Log 첨부 설정	AttachLogcat=1	

6.1.3.2 Configuration 항목 Set 하는 방법

[API Prototype]

```
/**
    *  설정값을 변경하는 메소드
    * @param tab              BizSmartPush tab
    * @param key              BizSmartPush key
    * @param value            BizSmartPush value
    * @return BP_RET          SetConfig 에 성공한 경우 Success 가 반환된다.
    */
    public BP_RET SetConfig(String tab, String key, String value)
```

[사용예제]

```
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "ServerInfo", " https://211.115.7.234:8443");
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "UserID"," 01012345678");
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "PhoneNumber"," 01012345678");
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "OperatorName"," 45005");
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "AuthKey"," K7AzsDatMIgwMzB3yDG111");
```

```
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "GcmID," 397613945111");
BP_RET ret = m_bizpush.SetConfig("BIZPUSH", "AomID," BPPTEST111");
```

6.1.3.3 Configuration 항목 Get 하는 방법

[API Prototype]

```
/**
    * 설정값을 받아오는 메소드
    * @param tab                BizSmartPush tab
    * @param key                BizSmartPush key
    * @return String GetConfig 에 성공한 경우 해당 설정값이 반환된다.
    */
   public String GetConfig(String tab, String key)
```

[사용예제]

```
String strConfig = "";
strConfig =m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "ServerInfo");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "UserID");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "PhoneNumber");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "OperatorName");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "authKey");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "GcmID");
strConfig = m_bizPushTester.m_bizpush.GetConfig("BIZPUSH", "AomID");
```

6.1.4 Broadcast Receiver 를 통한 Push 메시지 수신 처리

Push 메시지 Receiver 를 통한 수신은 앱이 실행되어있지 않거나 앱이 실행되어도 BizSmartPush Library 를 통해 리스너를 등록하지 않았을 경우에만 동작한다.

Broadcast Receiver 를 통해 Push 수신 및 상태 정보를 수신하기 위해서 BizSmartPush Library 에서 3rd Party App 으로 보내는 Broadcast Intent 를 받기 위한 Action Name 을 설정하여 등록해야 한다.

SetBroadcastActionName 을 설정하지 않으면 3rd Party APP 에서 Push Message 수신 시 Main Activity 가 호출 된다. 3rd Party App 에서 설정한 BroadcastReceiver 로 수신 받기를 원하는 경우에 반드시 아래와 같이 SetBroadcastActionName()메소드를 호출 한다.

[API Prototype]

```
/**
    * Intent 를 activity 가 아닌 BroadcastReceiver 를 받기를 원하는경우 설정.
    * 사용예)m_bizpush. SetAppReceiverName("TesterBroadcastReceiver");
    * @param name App 의 action string
    * @return boolean                    SetBroadcastActionName 에 성공한 경우 true 가 반환된다.
    */
   public boolean SetBroadcastActionName(String name)
```

[사용예제]

```
// AndroidManifest.xml 에서 Receiver 등록예제
<receiver android:name="com.nable.training.bizpush.tester.TesterBroadcastReceiver"  android:enabled="true">
          <intent-filter>
             <action android:name="com.nable.trainning.bizpush.tester.intent "/>
             <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
       </receiver>
```

```
// Receiver 에 등록된 Intent 명을 입력한다.
// Intent Action 명 설정은 각 APP 의 Package 명을 조합하여 생성한다.
// Intent 설정 예) Package 명이 com.nable.trainning.bizpush.tester 일 경우, com.nable.trainning.bizpush.tester.intent 로 생성한다.
// 호출 시점은 BizSmartPush Library 의 Start() 메소스 호출 이후에 설정한다.
Boolean retIntent = m_bizpush.SetBroadcastActionName("com.nable.trainning.bizpush.tester.intent");
```

```
// BroadcastReceiver 수신 처리 예제

public class TesterBroadcastReceiver extends BroadcastReceiver {

String LOG_TAG = "BizPushTester";

// Notification Bar 에 표시
private void ShowNotification(Context context, String message) {
```

```
        Notification notification = null;

        int icon = R.drawable.ic_launcher;
        long when = System.currentTimeMillis();

        NotificationManager      notificationManager      =      (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
        notification = new Notification(icon, message, when);

        String title = context.getString(R.string.app_name);
        Intent notificationIntent = new Intent(context, BizPushTester.class);

        notificationIntent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP|
Intent.FLAG_ACTIVITY_SINGLE_TOP);
        PendingIntent     intent     =     PendingIntent.getActivity(context,     0,
notificationIntent, 0);

        notification.setLatestEventInfo(context, title, message, intent);
        notification.flags = Notification.FLAG_AUTO_CANCEL;
        notification.vibrate = new long[] {500, 100, 500, 100};
        notification.sound                                                    =
Uri.parse("/system/media/audio/notifications/Beat_Box_Android.ogg");

        notificationManager.notify(0, notification);
 }

// Push Intent Payload Parsing
public String ParsingPushPayloadFromBSPIntent(Intent intent) {

        Log.d(LOG_TAG, "IN : ParsingMessageFromBSPIntent()");

        String action = intent.getAction();
        Log.d(LOG_TAG, "#@      " + action + " : ParsingMessageFromBSPIntent");

        Bundle extras= intent.getExtras();
        if (extras != null)
               Log.d(LOG_TAG,   "#@               "   +   extras.toString()   +   "   :
ParsingMessageFromBSPIntent");

        String intentCommand = intent.getStringExtra("command");
        String intentAction = intent.getStringExtra("action");

        Log.d(LOG_TAG, "Command : " +  intentCommand);
        Log.d(LOG_TAG, "Action : " +  intentAction);

        String bspMessage = "";

        if (intentCommand.equals("com.smartbizpush.push.AOM_MESSAGE")) {

               try {
                      char type = intent.getCharExtra(("type"), (char)0);
                      final byte[] message = intent.getByteArrayExtra("message");
                      boolean needAck = intent.getBooleanExtra("needAck", true);
                      int transactionId = intent.getIntExtra("transactionId", 0);

                      String aomMessage = new String(message);

                      Log.d(LOG_TAG, "AOM Push Message : [ " + aomMessage + " ]");
```

```
                        bspMessage = aomMessage;

                } catch (Exception e) {
                        Log.e(LOG_TAG,"Exception : " + e.getMessage());
                }

        } else if (intentCommand.equals("com.smartbizpush.push.GCM_MESSAGE")) {

                try {
                        String title = intent.getStringExtra("title");
                        String gcmMessage = intent.getStringExtra("msg");

                        Log.d(LOG_TAG, "GCM Push Message : [ " + gcmMessage + " ]");

                        bspMessage = gcmMessage;

                } catch (Exception e) {
                        Log.e(LOG_TAG,"Exception : " + e.getMessage());
                }
        }

        Log.d(LOG_TAG, "BSP Push Payload : [ " + bspMessage + " ]");

        return bspMessage;
 }
```

// Push 메시지 수신 처리

```
@Override
public void onReceive(Context context, Intent intent) {
        Log.d(LOG_TAG,"In onReceiver : TesterBroadcastReceiver");
        String payload = ParsingPushPayloadFromBSPIntent(intent);
        ShowNotification(context, payload);
    }
}
```

6.2 BizSmartPush 시작

BizSmartPush 초기화 이후, Start() 메소드를 명시적으로 호출하여 BizSmartPush Library 를 구동한다.

[API Prototype]

```
/**
    * BizSmartPush Library 를 구동하기 위한 Start 메소드
    * @param localIP            단말이 할당받은 Local IP 주소
    * @return BP_RET            Start 에 성공한 경우 Success 가 반환된다.
    */
   public BP_RET Start(String localIP)
```

[사용예제]

```
//할당받은 단말 Local IP 를 넣는다.
BP_RET ret = m_bizpush.Start("192.168.0.1");
```

6.3 Error

BizSmartPush API 는 BP_RET 형의 값을 반환한다. 메소드 호출 후 실패한 경우, 실패에 대한 이유를 확인한다.

6.3.1 GetErrorReason

[시나리오]

1. 3rd Party APP 에서 Register(AUTO, "")호출을 통해 Push Service 등록 시도

2. 등록 시도를 요청하면, 3rd Party APP 에서 등록한 OnState() 리스너로 토큰발급 시도 중인 CreatingToken 이 전달된다.

3. Device Token 발급이 되었다면, BizSmartPush Library 에서 발급받은 DeviceToken 을 이용하여 Register 를 시도한다.

4. 6.1.2.1 에서 명시한 필수 항목이 하나라도 빠져 있는경우, 아래 그림과 같이 3rd Party App 에서 등록한 OnState() 리스너로 Error 가 전달된다.

5. Error 가 전달된경우 3rd Party APP 에서는 명시적으로 GetErrorReason() 메소드를 호출하여 Error 의 원인을 획득한 후 예외처리를 수행한다.

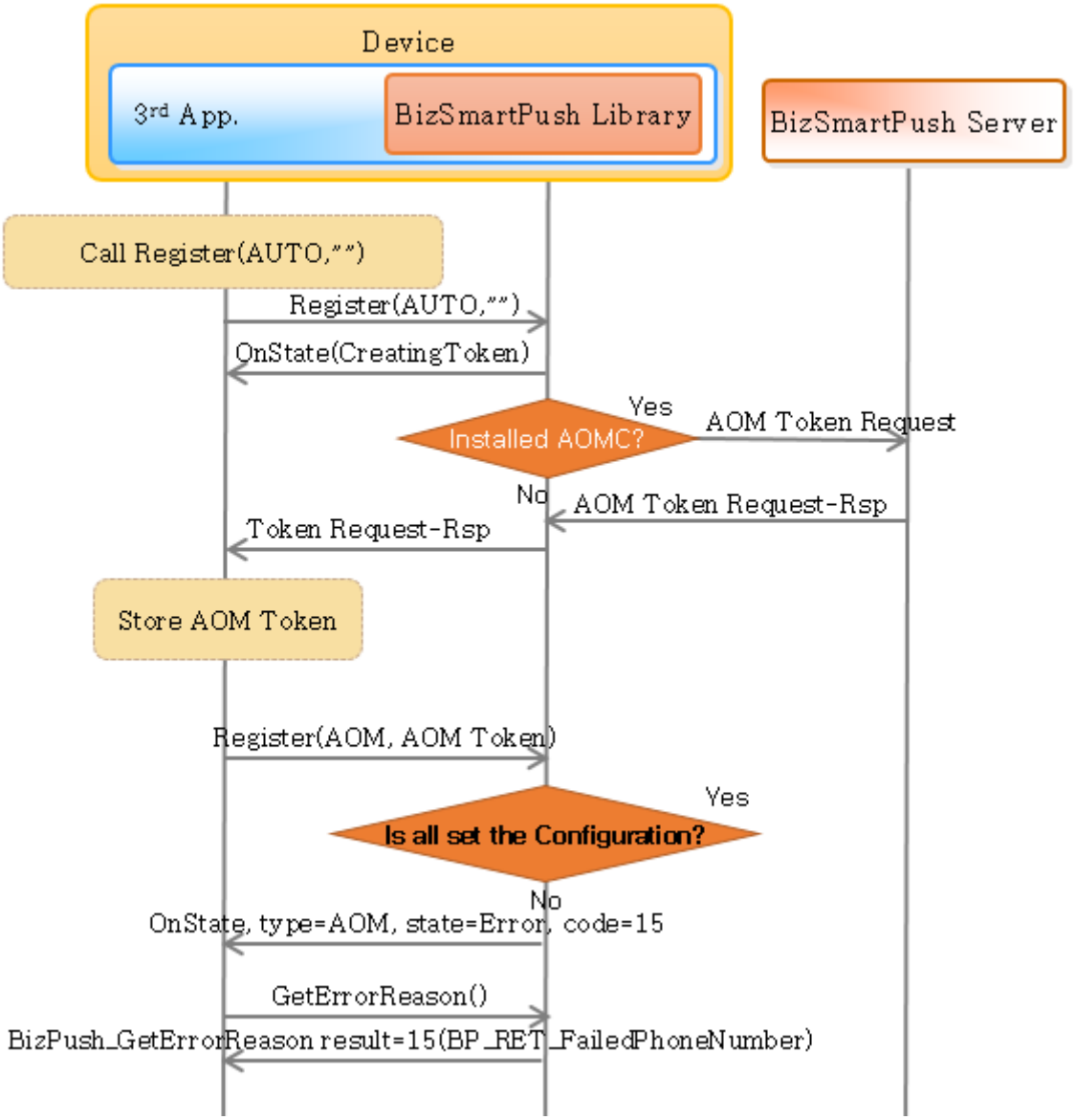


Figure 6 BizSmartPush API 호출 실패에 따른 원인 파악을 위한 절차

BizSmartPush API 호출시 수행에 실패한 경우 3rd Party APP 에서는 BizSmartPush 함수 호출 실패에 대한 이유를 알아내기 위해 명시적으로 GetErrorReason() 메소드를 호출 한다.

[API Prototype]

```
/**
    * 오류에 관련된 구체적인 원인을 확인하기 위한 메소드
    * OnState 리스너로 Error 이 전달된 경우 GetErrorReason 메소드를 이용하여 구체적인 오류 원인을 확인할 수 있다.
    * @return int                GetErrorReason 에 성공한 경우 오류 원인이 반환 된다.
    */
```

```
public int GetErrorReason()
```

[사용예제]

```
int getError = m_bizpush.GetErrorReason();
```

Table 4 BP_RET_Return

BP_RET	Description
BP_RET_Success	BizSmartPush 함수 호출 성공
BP_RET_Failed	BizSmartPush 함수 호출 실패
BP_RET_ListenerNotComplete	BizPushListener 인스턴스가 잘못 생성된 경우
BP_RET_FailedVersion	Version 설정이 안된 경우
BP_RET_FailedPushType	PushType 설정이 안된 경우
BP_RET_FailedServerInfo	ServerInfo 설정이 안된 경우
BP_RET_FailedrequestUri	requestUri 설정이 안된 경우
BP_RET_FailedAppID	AppID 설정이 안된 경우(GDMID, AOMID 가 설정되지 않으면 Fail)
BP_RET_FailedUserID	UserID 설정이 안된 경우
BP_RET_FailedOperatorName	OperatorName 설정이 안된 경우
BP_RET_FailedAuthKey	AuthKey 설정이 안된 경우
BP_RET_FailedDeviceToken	DeviceToken 설정이 안된 경우
BP_RET_FailedMsgSize	MsgSize 설정이 안된 경우
BP_RET_NotRegisted	Register 설정이 안된 경우
BP_RET_RecursiveCall	재귀 호출을 시도한 경우
BP_RET_FailedPhoneNumber	PhoneNumber 설정이 안된 경우
BP_RET_AOMCNotinstalled	AOMC 가 설치되어 있지 않은 경우
BP_RET_FailedQuery	Query 실패된 경우

7 Token 발급 및 등록

Token 발급 및 등록은 Push Message 수신이 가능하도록 BizSmartPush Server 에 등록하는 것을 의미한다. 3rd Party APP 은 Token 발급 및 등록을 시도 하기 전에 BizSmartPush Server 정보가 포함된 설정을 반드시 해야 한다. (5 장 참조)

BizSmartPush Library 는 3rd Party APP 에서 Register()메소드를 호출하면 Token 발급 시도 후, 획득한 Token 값을 이용하여 BizSmartPush Server 로 등록을 시도한다.

아래 가이드는 크게 Token 발급과 등록으로 나누었지만, 3rd Party APP 에서는 Register()메소드만 호출하면 Token 발급 및 등록을 한번에 진행 한다.

7.1 Pre-activities

Token 발급 및 등록을 수행하기 전에, 4 장 퍼미션 설정, 5 장의 필수 설정 항목을 수행해야한다.

7.2 Token 발급

Auto Token 발급, Manual Token 발급의 조건 표이다.

Table 5 BizSmartPush 의 State

Push Type	AOM CLIENT 설치	AOM CLIENT 미설치
Auto Token 발급	AOM Token 획득	GCM Token 획득
AOM Manual Token 발급	AOM Token 획득	**Token 획득 실패**
GCM Manual Token 발급	GCM Token 획득	GCM Token 획득

7.2.1 Auto Token 발급

Table 6 Auto Token 발급

Push Type	AOM CLIENT 설치	AOM CLIENT 미설치
Auto Token 발급 시도 시	AOM Token 획득	GCM Token 획득

위 표에 대한 설명이다:

1.AOM CLIENT 가 설치되어 있는 단말의 경우, Auto Token 발급 요청 시 AOM Token 획득 후 BizSmartPush Server 에 등록 시도 한다.

2.AOM CLIENT 가 설치되어 있지 않은 단말의 경우, Auto Token 발급 요청 시 GCM Token 획득 후 BizSmartPush Server 에 등록 시도 한다.

[API Prototype]

```
/**
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type              Push Type 설정 값 입력
    * @param authCode          AuthCode(6 자리) 값 입력
    * @return BP_RET           Register 에 성공한 경우 success 가 반환된다.
    */
   public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

[사용예제]

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.
String strAuthCode = "";
BP_RET ret = m_bizpush.Register(BP_PUSH_TYPE.AUTO, strAuthCode);
```

7.2.2 Auto Token 발급의 예외 사항

1. GCM 등록 후 AOM 으로 Push Sevice 가 변경되는 경우.

[시나리오]

1. AOM CLIENT 미설치

2. 3rd Party APP 에서 Register(AUTO, AuthCode)호출을 통해 GCM 으로 정상적으로 등록

3. 사용자가 T-Store 실행에 의해 AOM CLIENT 가 설치

4. 단말이 재부팅되거나, APP 강제 종료 후 APP 시작

5. Register(AUTO, Authcode)호출 하면, AOM 으로 등록.

이유: Push Type 이 AUTO 인 경우, AOM Push Service 를 우선으로 제공한다.

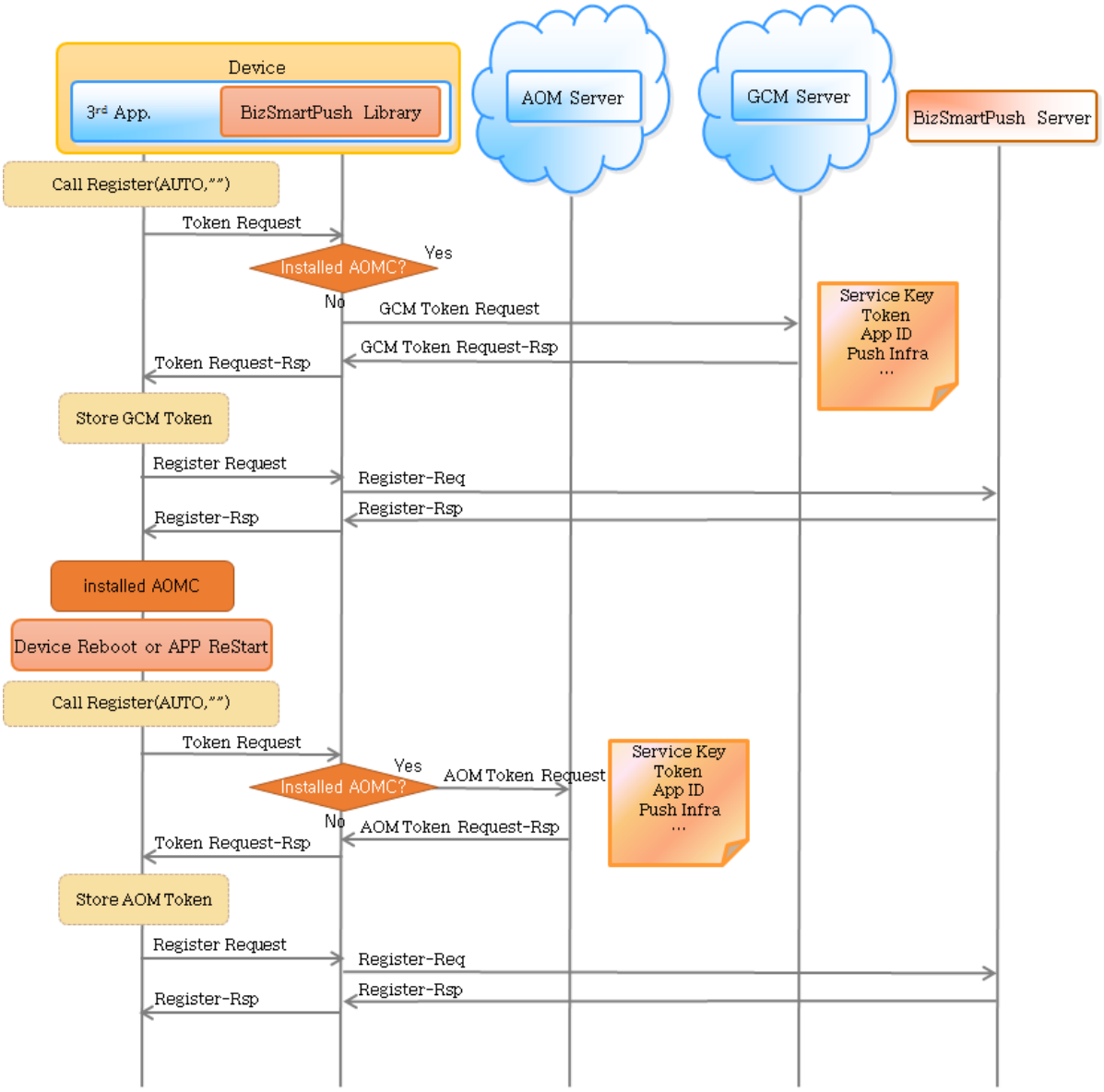


Figure 7 Auto Token 발급의 예외 사항

2. AOM 으로 Push Service 사용중 AOM Client 삭제된 경우

[시나리오]

1. AOM CLIENT 설치.

2. 3rd Party APP 에서 Register(AUTO, AuthCode)호출을 통해 AOM 으로 Push Service 등록

3. 사용자에 의해 AOM CLIENT 가 강제 삭제

4. 3rd Party APP 에서는 AOM CLIENT Remove Intent 를 수신 후, GCM 으로 Push Service 등록 시도.

5. Register(AUTO, Authcode)호출 하면, GCM 으로 등록.

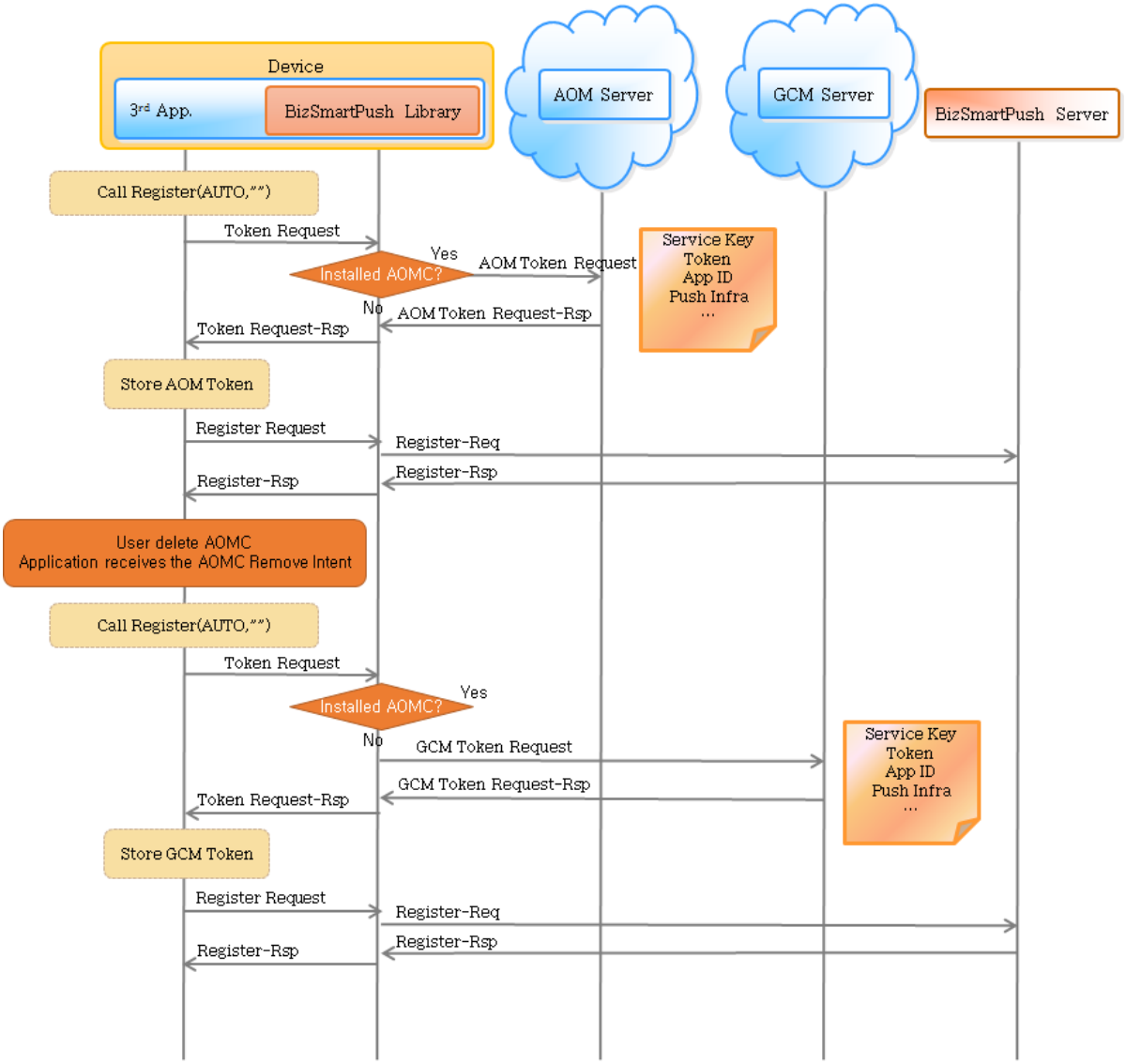


Figure 8 AOM 으로 Push Service 사용중 AOM Client 삭제 시 동작

7.2.3 Manual Token 발급

7.2.3.1 AOM Push Service 등록

[시나리오]

1. 3rd Party APP 에서 Register(AOM, "")를 호출.

2. BizSmartPush Library 는 AOM 으로 Token 발급이 성공되면, BizSmartPush Server 로 등록을 시도한다.

3. BizSmartPush Server 로부터 200 OK 를 수신받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

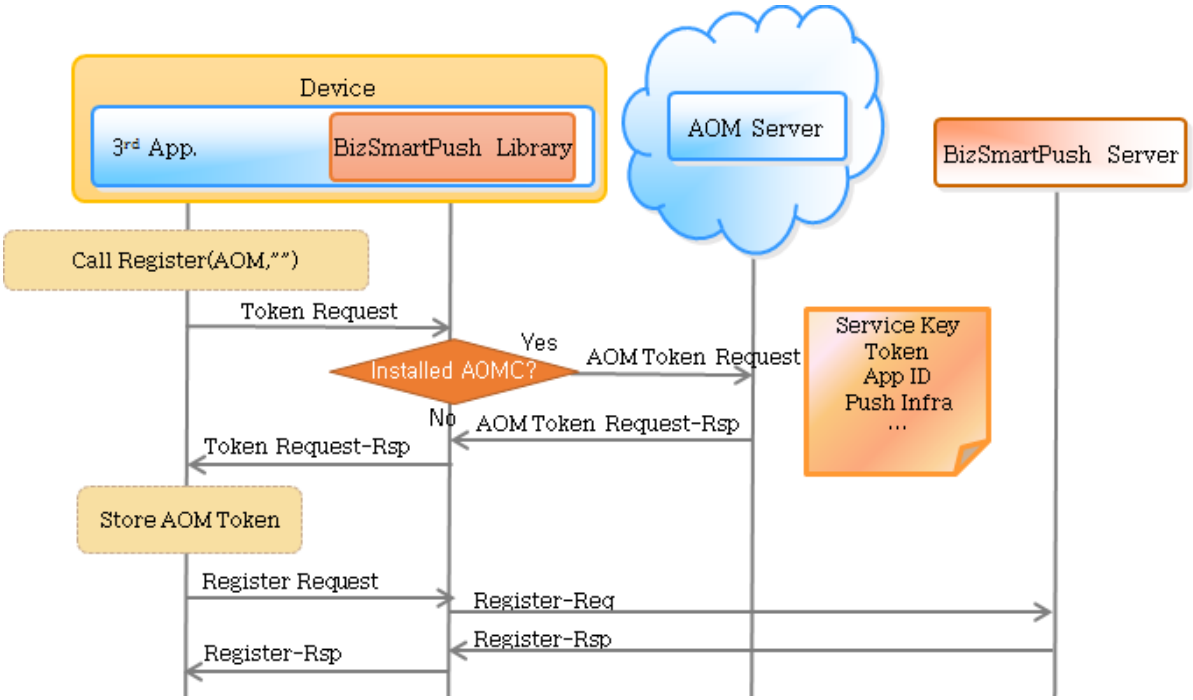


Figure 9 AOM Push Service 등록

[API Prototype]

```
/**
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type            Push Type 설정 값 입력
    * @param authCode        AuthCode(6 자리) 값 입력
    * @return BP_RET         Register 에 성공한 경우 success 가 반환된다.
    */
   public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

[사용예제]

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.
String strAuthCode = "";
BP_RET ret = m_bizpush.Register(BP_PUSH_TYPE.AOM,strAuthCode);
```

7.2.3.2 GCM Push Service 등록

[시나리오]

1. 3rd Party APP 에서 Register(GCM, "")를 호출 한다.

2. BizSmartPush Library 는 GCM 으로 Token 발급이 성공되면, BizSmartPush Server 로 등록을 시도한다.

3. BizSmartPush Server 로부터 200 OK 를 수신받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

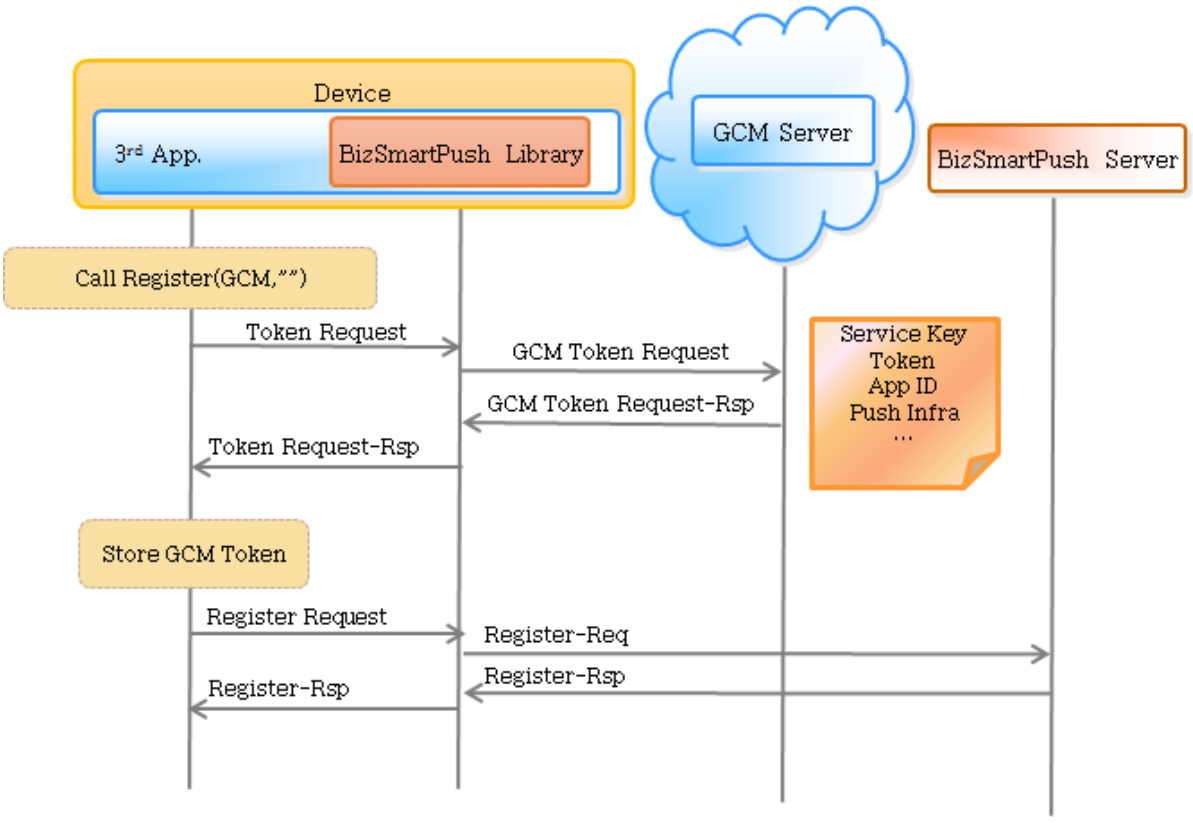


Figure 10 GCM Push Service 등록

[API Prototype]

```
/**
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type              Push Type 설정 값 입력
    * @param authCode          AuthCode(6 자리) 값 입력
    * @return BP_RET           Register 에 성공한 경우 success 가 반환된다.
    */
   public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

[사용예제]

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.
String strAuthCode = "";
BP_RET ret = m_bizpush.Register(BP_PUSH_TYPE.GCM,strAuthCode);
```

7.2.4 Manual Token 발급 예외 사항

[시나리오]

1. 단말에 AOM CLIENT 미설치.

2. 3rd Party APP 에서 Register(AOM, “”)를 호출 한다.

3. AOM CLIENT 가 설치되어 있지 않으므로, 3rd Party APP 으로 Register(AOM, “”) 메소드 호출에 대한 TokenCreateFailed 에러가 리턴된다.

4. 리턴값이 TokenCreateFailed 이면, 3rd Party APP 에서는 GCM 으로 등록하기 위해, Register(GCM,””) 또는 Register(AUTO,””)로 메소드를 호출 해야 한다.

5. BizSmartPush Server 로부터 200 OK 를 수신받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

3rd Party APP 에서 AOM CLIENT 가 없는데 AOM 으로 Token 발급 요청시에는 Token 발급에 실패할 것이므로 아래와 같이 에러가 발생한다.

TokenCreateFailed 에러 발생시 3rd Party APP 에서는 명시적으로 PushType 을 GCM 또는 AUTO 으로 등록시도를 해야 한다.

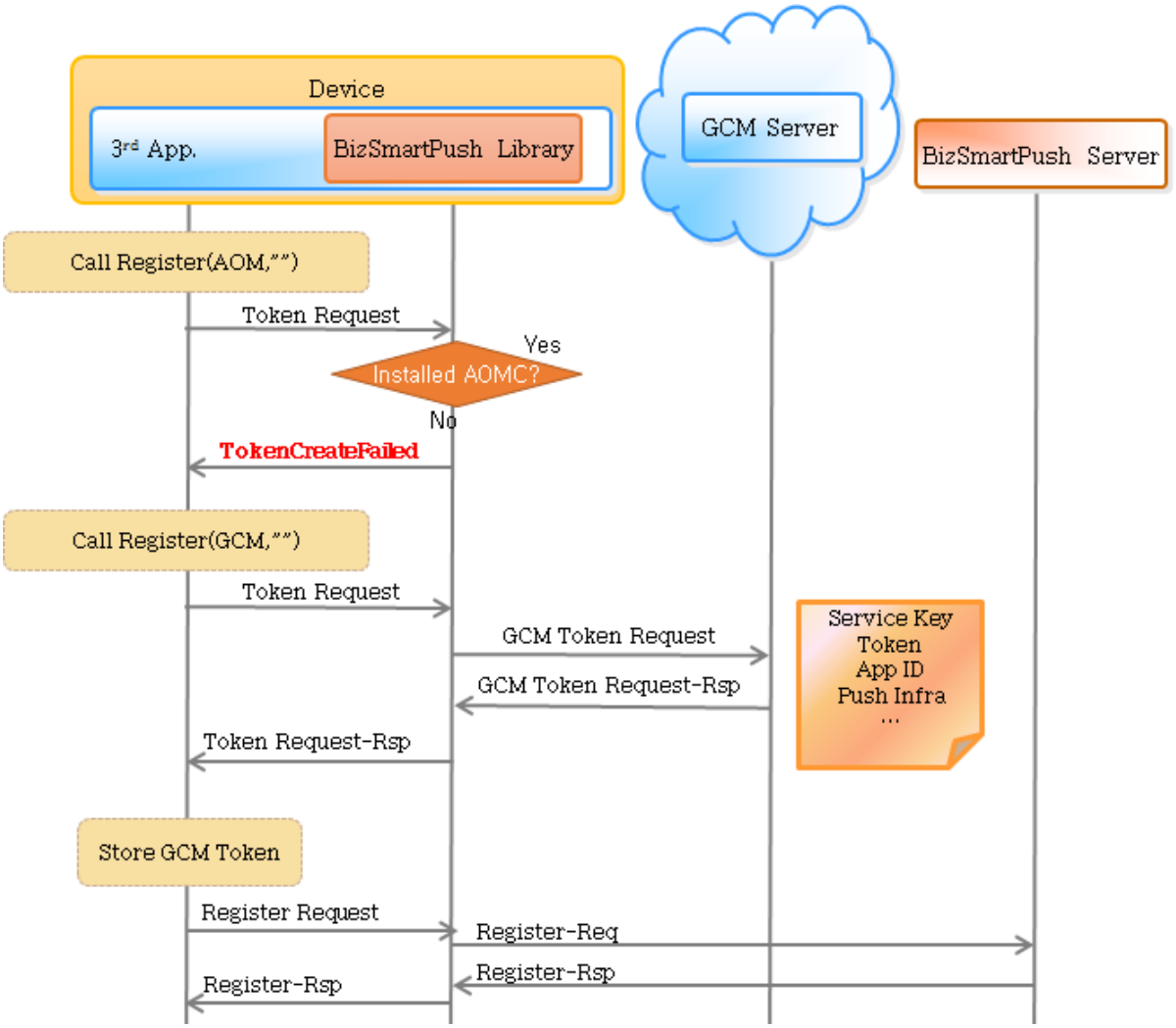


Figure 11 Manual Token 발급 예외 사항

[API Prototype]

```
/**
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type            Push Type 설정 값 입력
    * @param authCode        AuthCode(6 자리) 값 입력
    * @return BP_RET         Register 에 성공한 경우 success 가 반환된다.
    */
```

```
public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

[사용예제]

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.
String strAuthCode = "";
BP_RET ret = m_bizpush.Register(BP_PUSH_TYPE.GCM,strAuthCode);
```

7.3 Registration

Token 발급 후 BizSmartPush Server 로 등록되는 과정을 설명 한다.

등록은 인증 Registration 과 무인증 Registration 으로 구분된다.

인증등록에는 Authentication 과정이 포함되어 있으며, 각각의 자세한 사항은 해당 절을 확인하도록 한다.

7.3.1 인증 Registration

인증 Registration 은 Regi 단계에서 인증 코드 수신을 위한 단계를 추가적으로 수행 후 수신된 인증 코드(6 자리 숫자)를 이용하여 Regi 를 재시도하는 절차이다.

[시나리오]

1. 초기 Regi 시 AuthCode 를 포함하지 않은 Register(AUTO, "")를 호출 한다.

2. BizSmartPush Server 로부터 401(UnAuthorized)을 수신 받으면 3rd Party APP 으로 OnState(code=401)이 올라간다.

3. Registration 에 대한 응답으로 401 을 수신받으면, 3rd Party APP 은 AuthCode 를 획득하기 위해 RequestAuthentication() 메소드를 호출 한다.

4. RequestAuthentication 의 응답으로 200 OK 를 수신 받으면 3rd Party APP 으로 OnAuthentication(id, code=200, timeout=600)이 올라간다.

5. RequestAuthentication 의 인자로 전달된 전화번호나 Email 주소로 AuthCode(6 자리 숫자)가 수신된다.

6. 3rd Party APP 은 인증 timeout(600 초) 이내에 AuthCode 를 인자로 갖는 Register(AUTO, AuthCode(6 자리 숫자))를 호출한다.

7. BizSmartPush Server 로부터 Register()메소드 호출에 대한 응답으로 200 OK 를 수신받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

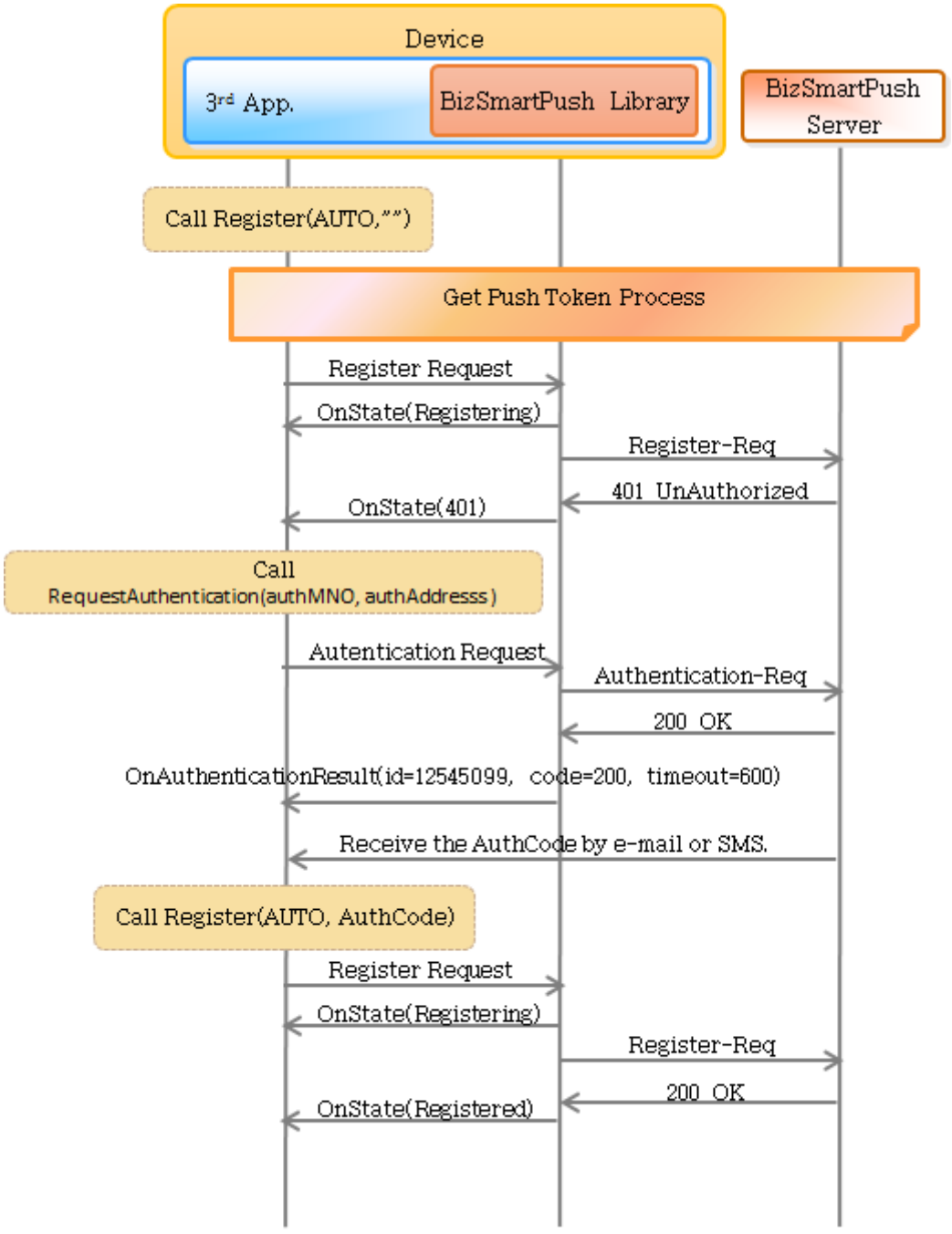


Figure 12 인증 Registration

[API Prototype]

```
/**
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type              Push Type 설정 값 입력
    * @param authCode          AuthCode(6 자리) 값 입력
    * @return BP_RET           Register 에 성공한 경우 success 가 반환된다.
    */
   public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

[사용예제]

```
// 초기 Regi 시 AuthCode 를 포함하지 않은 Register(AUTO, "")를 호출 한다.
String strAuthCode= "";
BP_RET ret = m_bizpush.Register(BP_PUSH_TYPE.AUTO, strAuthCode);
```

BizSmartPush Server 로부터 401 을 수신 받으면 아래와 같이 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 OnState(type=AOM, state=Error, code=401)를 알려준다.

200 OK 를 수신 받지 못한 경우이므로, 3rd Party APP 으로 state=Error, code=401 로 알려준다.

```
public void OnState(BP_PUSH_TYPE type, BP_STATE state, int code) {
            Bundle data = new Bundle();
            data.putInt("type", state.getValue());
            data.putInt("state", state.getValue());
            data.putInt("code", code);

            Message msg = Message.obtain();
            msg.what = MessageID.m_nMSG_ID_ONSTATE;
            msg.setData(data);

            m_handler.sendMessage(msg);
 }

 private Handler m_handler = new Handler() {
```

```
            public void handleMessage(Message msg) {

                        switch (msg.what) {
                        case MessageID.m_nMSG_ID_ONSTATE:
                                    ProcessOnState(msg);
                                    break;
                        . . .
                        }
            }
 };

private void ProcessOnState(Message msg) {
            Bundle data = msg.getData();
            int type = data.getInt("type");
            int state = data.getInt("state");
            int code = data.getInt("code");
            . . .
}
```

7.3.1.1 Authentication

3rd Party APP 은 Register() 메소드 호출에 대한 응답으로 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 OnState(type=AOM, state=Error, code=401) 수신 시, RequestAuthentication()를 호출한다.

RequestAuthentication()메소드 호출 결과에 대한 응답으로 200OK 수신 시, 3rd Party APP 에서 등록한 리스너인 OnAuthenticationResult()리스너로 호출 결과를 알려준다.

```
public void OnAuthenticationResult(int id, int code, int timeout) {
            if(Code == 200){
                        // timeout 시간 이내에 AuthCode 를 이용한 인증 Register()를 수행해야 한다.
            }
            else{
                        // 에러가 발생한 경우, 3rd Party APP 에서 예외 처리를 수행한다.
            }
};
```

3rd Party APP 은 수신받은 200OK 내의 인증 유효 시간(timeout=600 초) 이내에 Auth Code 가 포함된 Register(AUTO, authCode)를 실행 한다.

[API Prototype]

```
/**
    * AuthCode(6 자리)를 수신받기 위한 메소드.
    * Register() 메소드 호출에 대한 응답으로 OnState(type=AOM, state=Error, code=401) 수신 시, RequestAuthentication()를 호출한다
    * @param authMNO                MNO(단말이 가입된 통신사)
    * @param authAddress     AuthCode 를 수신받을 Email 또는 전화번호
    * @return int                      RequestAuthentication 에 성공한 경우 transaction_id 가 반환된다.
    */
   public int RequestAuthentication(String authMNO, String authAddress)
```

[사용예제]

```
String authMNO = "45005";
// authAddress 는 Email 또는 전화번호
String authAddresss = "bizPushAuthcode@gmail.com";
RequestAuthentication(authMNO, authAddresss);
```

RequestAuthentication()메소드의 첫번째 인자로 MNO(단말이 가입된 통신사), 두번째 인자로 authAddress(인증번호를 수신할 전화 번호 또는 E-Mail 주소)를 입력하여 호출한다.

RequestAuthentication()메소드 호출에 대한 응답으로 200 OK 수신시 authAddress 에 입력한 전화번호 또는 E-Mail 주소로 6 자리의 Auth Code 가 전달된다.

3rd Party APP 에서는 6 자리 AuthCode 를 확인하여, 아래와 같이 AuthCode 가 포함된 Register()메소드를 호출 한다.

```
String strAuth= "123456";
Register(BP_PUSH_TYPE.AUTO, strAuth);
```

BizSmartPush Server 로부터 Register 에 대해 200 OK 을 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

```
// AOM 으로 등록 성공된 경우
OnState(type=AOM, state=Registered, code=0)
// GCM 으로 등록 성공된 경우
OnState(type=GCM, state=Registered, code=0)
```

7.3.2 무인증 Registration

무인증 Registration 은 Regi 단계에서 인증 코드 수신 단계를 진행하지 않고 Regi 를 시도하는 절차이다.

[시나리오]

1. 3rd Party APP 은 두번째 인자인 AuthCode 에 공백을 입력하여 Register(AUTO, "")를 호출 한다.

2. 3rd Party APP 으로 Register 시도중인 OnState(Registering) 리스너가 올라간다.

3. BizSmartPush Server 로부터 Register() 메소드 호출응답으로 200 OK 를 수신받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

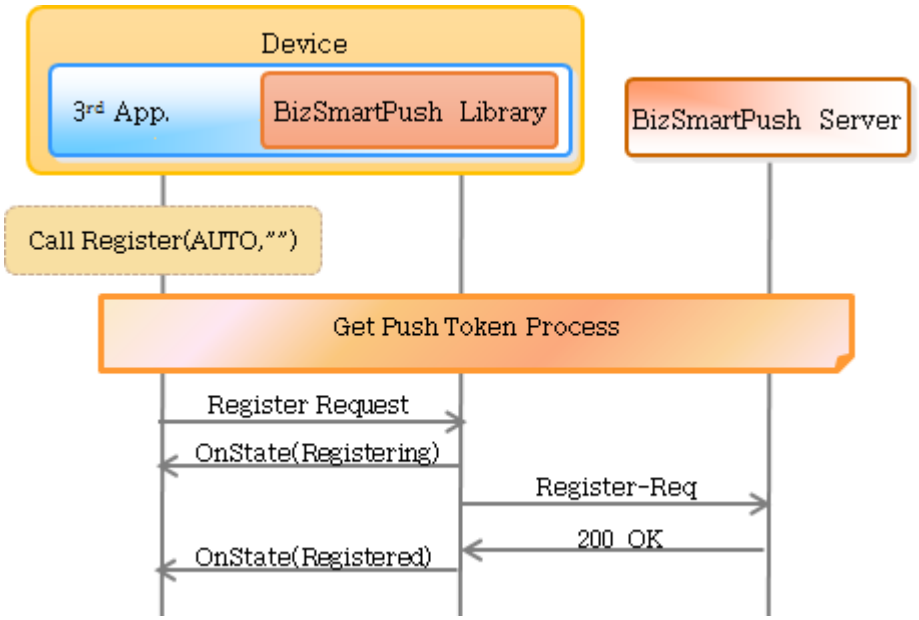


Figure 13 무인증 Registration

[API Prototype]

```
/**
```

```
    * BizSmartPush 등록 메소드
    * DeviceToken 발급 시도 후, 획득한 DeviceToken 값을 이용하여 BizSmartPush Server 로 등록을 시도하는 메소드
    * @param type              Push Type 설정 값 입력
    * @param authCode          AuthCode(6 자리) 값 입력
    * @return BP_RET           Register 에 성공한 경우 success 가 반환된다.
    */
   public BP_RET Register(BP_PUSH_TYPE type, String authCode)
```

무인증 Registration 에서는 두번째 인자인 authCode 에 공백을 입력하여 Register()메소드를 호출 한다.

1.Auto Register 의 Register() 메소드 호출 예

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.
String strAuthCode= "";
Register(BP_PUSH_TYPE.AUTO, strAuthCode);
```

2.Manual Register 의 Register() 메소드 호출 예

```
// strAuthCode 는 인증모드일때만 사용한다.
// 무인증 모드일 경우, 공백을 입력 한다.

String strAuthCode= "";
// AOM Register
Register(BP_PUSH_TYPE.AOM, strAuthCode);
// GCM Register
Register(BP_PUSH_TYPE.GCM, strAuthCode);
```

BizSmartPush Server 로부터 Register() 메소드 호출에 대한 응답으로 200 OK 을 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

```
// AOM 으로 등록 성공된 경우
OnState(type=AOM, state=Registered, code=0)
// GCM 으로 등록 성공된 경우
OnState(type=GCM, state=Registered, code=0)
```

7.3.3 Register state & Procedure

7.3.3.1 State 변경 확인

사용자 API 호출 이나 서버응답에 의해 State 가 변경되면 3rd Party APP 에서 리스너로 등록한 OnState() 로 State 변경을 알려준다.

BizSmartPush 의 State 는 아래와 같다.

Table 7 BizSmartPush 의 State

BP_STATE	Description
None	Register 시도 이전 상태
Registering	Register 시도중인 상태
Registered	Register 등록이 완료된 상태
UnRegistered	Register 해지 완료 상태
Error	Error
CreatingToken	Token 발급 시도 상태
Standby	BizSmartPush 초기화(Init) 상태

7.4 Error Reason 확인

7.4.1 BP_RET Return

BP_RET	Description
BP_RET_Success	BizSmartPush 함수 호출 성공
BP_RET_Failed	BizSmartPush 함수 호출 실패
BP_RET_ListenerNotComplete	BizPushListener 인스턴스가 잘못 생성된 경우
BP_RET_FailedVersion	Version 설정이 안된 경우
BP_RET_FailedPushType	PushType 설정이 안된 경우
BP_RET_FailedServerInfo	ServerIngo 설정이 안된 경우
BP_RET_FailedrequestUri	requestUri 설정이 안된 경우
BP_RET_FailedAppID	AppID 설정이 안된 경우 (GDMID, AOMID 가 설정되지 않으면 Fail)
BP_RET_FailedUserID	UserID 설정이 안된 경우
BP_RET_FailedOperatorName	OperatorName 설정이 안된 경우
BP_RET_FailedAuthKey	AuthKey 설정이 안된 경우
BP_RET_FailedDeviceToken	DeviceToken 설정이 안된 경우
BP_RET_FailedMsgSize	MsgSize 설정이 안된 경우

BP_RET_NotRegisted	Register 설정이 안된 경우
BP_RET_RecursiveCall	재귀 호출을 시도한 경우

7.4.2 Authentication Error Return

Table 8 Authentication Error Return

Error Return	Description	예외처리
400	Bad Request(Request Message Parameter Invaild)	Authentication 메소드 호출시 인자값이 잘못되었는지 확인
401	UnAuthorized	Auth Key 가 정상적으로 입력 되어있는지 확인
404	Not Found(AppId)	BizPush 서버에 해당 AppId(GCMID, AOMID)가 등록되어 있는지 확인 필요(서버문의)
406	Message Format Error(Json)	Request Message 가 Json 포멧 규칙에 맞게 구성되어 있는지 확인.
408	Request Timeout	Register Request 이후 BizPush Server 로부터 응답이 없는 경우 발생. 서버 무응답 이유를 확인
500	BizSmartPush Internal Server Error	서버에러 서버에러 이유를 확인
503	Service Unavailable	서버에러 서버에러 이유를 확인

7.4.3 Register Error Return

Table 9 Register Error Return

Error Return	Description	예외처리
400	Bad Request	Register 시도시 설정값이 잘못되었는지 확인.
401	UnAuthorized	Authentication 을 수행(인증 Registration 수행)
403	Forbidden(Device_Register Request AuthKey 불일치)	1. AuthKey 를 정상적으로 입력하였는지 확인 2. 3rd Party App 에서 등록한 OnAuthenticationResult 리스너로 timeout 시간이 전달된다. 사용자가 timeout 시간 이후에 AuthKey 를 포함한 Registraion 을 수행하게 되면 403 에러가 발생한다. 403 에러가 발생한 경우 인증 Registration 을 다시 시행한다.
404	Not Found(AppId)	설정한 AppId(GCMID, AOMID)가 BizPush Server 에 등록되어 있는지 확인.
406	Message Format Error(Json)	Request Message 가 Json 포멧 규칙에 맞게 구성되어 있는지 확인.

408	Request Timeout	Register Request 이후 BizPush Server 로부터 응답이 없는 경우 발생. 서버 무응답 이유를 확인
500	BizSmartPush Internal Server Error	서버에러 서버에러 이유를 확인
503	Service Unavailable	서버에러 서버에러 이유를 확인

8 Push 수신 및 Delivery Report

Delivery Report 는 3rd Party APP 이 Push Message 를 수신한 경우, 해당 Push Message 의 수신 결과를 BizSmartPush Server 로 알려주는 기능이다.

서버로부터 Push Message 수신 시 3rd Party APP 에서 등록한 OnPushMessage() 리스너로 Push Message 가 전달된다.

8 장은 3rd Party APP 이 BizSmartPush Server 로 부터 Push Message 수신 시 수신 결과를 전송하는 방법에 대해 자세히 설명 한다.

8.1 Push Message 수신

8.1.1 Push Infra(AOM/GCM)로부터 Full Payload Push Message 수신 (fragment 의 값이 false 인 경우)

[시나리오]

1. BizSmartPush Server 로부터 Push Message 수신 시 3rd Party APP 에서 등록한 OnPushMessage() 리스너로 Push Message 가 전달된다.

2. 3rd Party APP 은 전달 받은 Push Message 의 fragment 값을 Parsing 하여 값을 확인 한다. (fragment 의 값은 true 또는 false 이다.)

3. **fragment 의 값이 false** 인 경우 수신 받은 Push Message 가 **Full Payload** 이므로 SendPushAck() 메소드를 호출 하여 Push Message 수신 결과를 BizSmartPush Server 로 알려준다.

4. BizSmartPush Server 로부터 SendPushAck() 메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnPushAckResult() 리스너로 성공 여부를 알려준다.

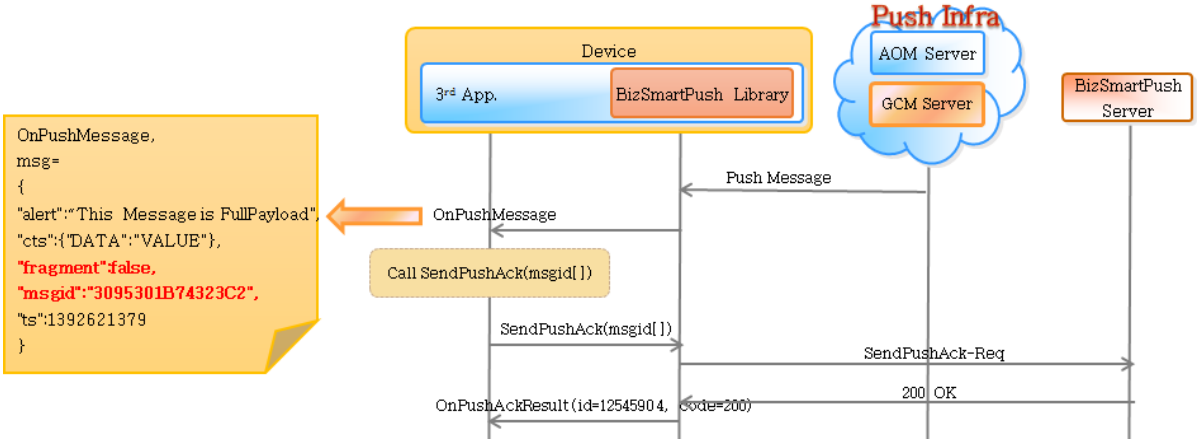


Figure 14 Push Infra(AOM/GCM)로부터 Full Payload Push Message 수신

8.1.2 Push Infra(AOM/GCM)로부터 Partial Payload Push Message 수신 (fragment 의 값이 true 인 경우)

[시나리오]

1. BizSmartPush Server 로부터 Push Message 수신 시 3rd Party APP 에서 등록한 OnPushMessage() 리스너로 Push Message 가 전달된다.

2. 3rd Party APP 은 전달 받은 Push Message 의 fragment 값을 Parsing 하여 값을 확인 한다. (fragment 의 값은 true 또는 false 이다.)

3. **fragment 의 값이 true** 인 경우 수신 받은 Push Message 가 Full Payload 가 아니므로 **GetPushPayloadList() 메소드를 호출 하여 Full Payload 를 요청**한다.

4. 3rd Party APP 이 GetPushPayloadList() 메소드 호출 이후에 BizSmartPush Server 로부터 200 OK 를 수신한 경우 3rd Party APP 에서 리스너로 등록한 OnDeliveryPushMessage() 통해 Push Message 가 전달 된다.

5. OnDeliveryPushMessage() 리스너 안에는 String msgs[] 배열 형태로 Pending Count 개수만큼 Message 의 내용이 아래와 같이 3rd Party APP 으로 전달 된다.

6. 3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해, OnDeliveryPushMessage() 리스너로 올라온 msgid 값을 Parsing 하여 수신된 Push Message 개수만큼 msgId[] 배열에 저장하여 SendPushAck(msgId[]) 메소드를 호출 한다.

7. BizSmartPush Server 로부터 SendPushAck()메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnPushAckResult() 리스너로 성공 여부를 알려준다.

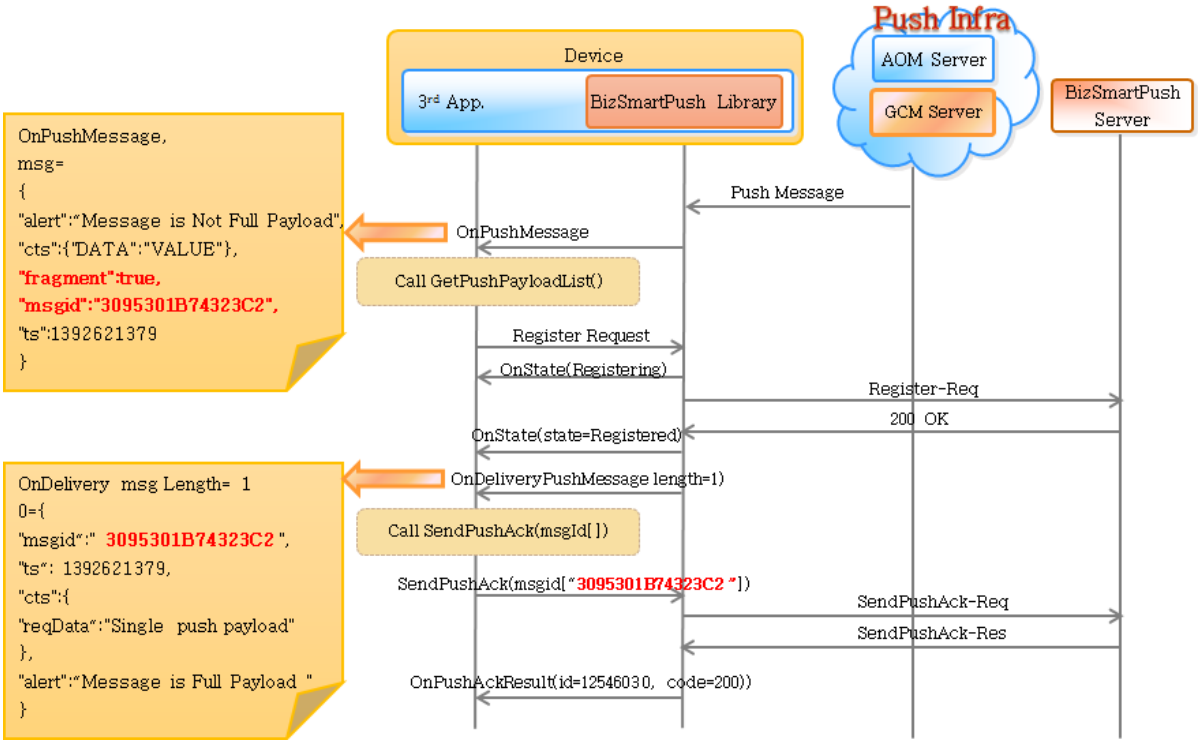


Figure 15 Push Infra(AOM/GCM)로부터 Partial Payload Push Message 수신

8.1.3 Push Message 수신 시 fragment 값 확인

단말이 Push Message 를 수신 받으면, 아래와 같이 3rd Party APP 에서 리스너로 등록한 OnPushMessage() 를 통해 Push 수신을 알려준다.

아래는 OnPushMessage() 리스너로 수신 받은 Message 가 전달 되었을 때 3rd Party APP 에서 Push Message 의 fragment 값을 획득 하기 위한 Parsing 예이다.

아래의 예시와 같이 Push Message 를 수신 받으면 Push Message 를 Parsing 하여 fragment 가 true 이면 수신 받은 Push Message 가 Full Payload 가 아니므로 GetPushPayloadList() 메소드 호출 후 SendPushAck() 메소드를 호출 한다.

반대로 fragment 가 false 이면 수신 받은 Push Message 가 Full Payload 이므로 SendPushAck() 메소드를 호출 한다.

```
@Override
public void OnPushMessage(String msg) {
        // TODO Auto-generated method stub
        String fullMsg = null;
        fullMsg = msg;
        m_msgLength = 1;
```

```
            Bundle data = new Bundle();
            data.putString("strFullMsg", fullMsg);
            data.putInt("msgLength", m_msgLength);

            Message pushMsg = Message.obtain();
            pushMsg.what = MessageID.m_nMSG_ID_ONPUSHMESSAGE;
            pushMsg.setData(data);

            m_handler.sendMessage(pushMsg);
}

 private Handler m_handler = new Handler() {
            public void handleMessage(Message msg) {

                        switch (msg.what) {
                        case MessageID.MSG_ID_ONSTATE:
                                    ProcessOnState(msg);
                                    break;

                        case MessageID.MSG_ID_ONPUSHMESSAGE:
                        {
                                    // Push 를  수신 받으면 fragment 를 Parsing 하여
                                    // fragment 가 true 이면 : GetPushPayloadList()호출
                                    // fragment 가 false 이면 : Delivery Push Ack 호출
                                    Bundle data = msg.getData();
                                    String fullMsg = data.getString("strFullMsg");

                                    String strfragment = "₩"fragment₩":";
                                    int n_findFragment = fullMsg.indexOf(strfragment, 0);
                                    int n_findComma = fullMsg.indexOf(",", n_findFragment);
                                    String retFragment = fullMsg.substring(n_findFragment + strfragment.length(),
n_findComma);

                                    if(retFragment.equals("false")) {
                                                m_bizPushTester.PrintLog("#% fragment of PushMessage is FALSE, So
Start SendPushAck()");
                                                ProcessMsgIDParsing(msg, MessageID.MSG_ID_ONPUSHMESSAGE);
                                    }
                                    else {
```

```
                                        m_bizPushTester.PrintLog("#% fragment of PushMessage is TRUE, So
Start GetPushPayloadList()");
                                        BP_RET ret = m_bizPushTester.m_bizpush.GetPushPayloadList();
                                        m_bizPushTester.PrintLog("#% BizPush GetPushPayloadList = " + ret);
                                }
                        }
                                break;

                        case MessageID.MSG_ID_ONPUSHACKRESULT:
                                AutoDeliveryReport();
                                break;

                        case MessageID.MSG_ID_ONDELIVERYPUSHMESSAGE:
                                ProcessMsgIDParsing(msg, MessageID.MSG_ID_ONDELIVERYPUSHMESSAGE);
                                break;
                        }
                }
 };
// 3rd Party APP 에서 리스너로 등록한 OnPushMessage() 를 통해 전달되는 Push Message 의 예
1) fragment 가 false 인 경우
OnPushMessage,
msg=
{
"alert":"aaabbbccc",
"cts":{"DATA":"VALUE"},
"fragment":false,
"msgid":"3095301B74323C2",
"ts":1392621379
}
2) fragment 가 true 인 경우
OnPushMessage,
msg=
{
"alert":"aaabbbccc",
"cts":{"DATA":"VALUE"},
"fragment":true,
"msgid":"3095301B74323C2",
"ts":1392621379
}
```

8.1.4 Full Payload Push Message 수신

3rd Party APP 은 리스너로 등록한 OnPushMessage 통해 Push Message 를 수신 시 수신된 Push Message 의 fragment 의 값이 true 인 경우 Full Payload 를 획득하기 위하여, GetPushPayloadList 메소드 호출을 수행한다. (GetPushPayloadList 호출을 수행하면 Push 에 대한 Full Payload Message 를 수신 한다.)

[API Prototype]

```
/**
    * OnPushMessage 전달 받은 후 Push Message 가 Full Payload 가 아닌 경우, Payload 전체를 얻어 오는 메소드
    * 3rd Party App 으로 OnPushMessage(String msg)리스너가 전달된 후 Push Message 의 fragment 의 값이 true 인 경우,
    * 3rd Party App 에서는 Full Payload 를 획득하기 위해서 GetPushPayloadList() 메소드를 호출한다.
    * @return BP_RET            성공한 경우 success 가 반환된다.
    */
   public BP_RET GetPushPayloadList()
```

Figure 11, Figure 13 과 같이 3rd Party APP 이 GetPushPayloadList() 메소드 호출 이후에 BizSmartPush Server 로부터 200 OK 를 수신한 경우 3rd Party APP 에서 리스너로 등록한 OnDeliveryPushMessage() 통해 Push Message 가 전달 된다.

OnDeliveryPushMessage() 리스너 안에는 String msgs[] 배열 형태로 Pending Count 개수만큼 Message 의 내용이 아래와 같이 3rd Party APP 으로 전달 된다.

```
    public void OnDeliveryPushMessage(String[] msgs)
    {
            m_msgLength = msgs.length;
            Log.d(LOG_TAG, "#% OnDelivery msg Length= " + m_msgLength);

            for (int index = 0; index < msgs.length; index++) {
                    Log.d(LOG_TAG, index + "=" + msgs[index]);
                    . . .
            }
```

```
}

// 수신된 Push Message 의 예
OnDelivery msg Length= 1
0={
"msgid":          "201751BEF10A002E",
"ts":    1371468042,
"cts":  {
"reqData":        "Single push payload"
},
"alert":           "Simulator Test"
}
```

3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해, OnDeliveryPushMessage() 리스너로 올라온 msgid 값을 Parsing 하여 수신된 Push Message 개수만큼 msgId[] 배열에 저장하여 SendPushAck(msgId[]) 메소드를 호출 한다.

[API ProtoType]

```
/**
    * Delivery Report 기능을 수행하는 메소드
    * Push Message 를 수신한 경우, Push Message 를 수신하였다는 수신 결과를 BizSmartPush Server 로 알리는 메소드.
    * @param msgId[]          수신된 Push Message[] 내용
    * @return int             SendPushAck 에 성공한 경우 transaction_id 가 반환된다.
    */
   public  int SendPushAck(String msgId[])
```

[사용예제]

```
String m_MessageID = new String[msgLength];
// Parsing 된 결과를 인자에 넣어 SendPushAck 함수를 호출 한다.
int ret = m_bizpush.SendPushAck(m_MessageID);
```

SendPushAck()메소드 호출 결과로 3rd Party APP 에서 등록한 리스너인 OnPushAckResult() 리스너가 3rd Party APP 으로 올라간다.

```
public void OnPushAckResult(int id, int code)
{
        if(code == 200){
                // SendPushAck()메소드 호출 성공
        }
        else{
                // SendPushAck() 메소드 호출 실패, 3rd party App 에서 에러 처리
        }
}
```

8.1.5 미수신 Push Message 처리

단말이 Push Message 를 BizSmartPush Server 로부터 수신 받지 못한 경우(단말이 꺼져 있거나 또는 네트워크 상태가 불량) 전달 하지 못한 Push Message 는 BizSmartPush Server 에 저장되어 있다.

단말이 Register() 혹은 SendPushAck() 메소드 호출 하게 되면 BizSmartPush Server 의 각 API 호출에 대한 응답으로 200 OK 전달하며 미수신된 Push 정보를 알려주는 Pending Count 를 포함하여 단말로 Push 를 전달한다.

미수신 Push Message 가 없을 경우에는 Pending Count 는 0 개 이다.

1. Register 를 시도 후 200 OK 응답에 미수신 Push 수신

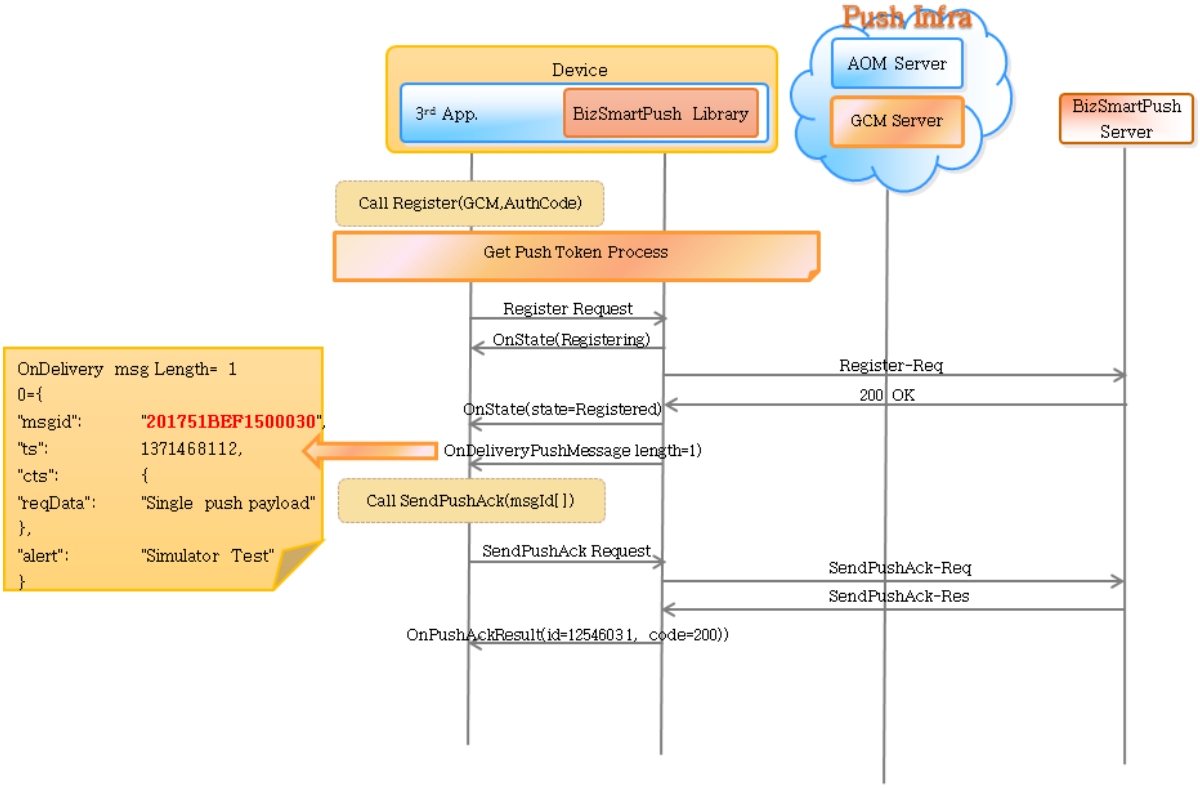


Figure 16 Register 를 시도 후 200 OK 응답에 미수신 Push 수신

위의 그림과 같이 3rd Party APP 이 Register()메소드 호출 이후에 BizSmartPush Server 로부터 수신한 200 OK 에 미수신 Push 를 수신한 경우 3rd Party APP 에서 리스너로 등록한 OnDeliveryPushMessage()통해 미수신 Push Message 가 전달 된다.

OnDeliveryPushMessage()리스너 안에는 String msgs[] 배열 형태로 Pending Count 개수만큼 Message 의 내용이 아래와 같이 3rd Party APP 으로 전달 된다.

```
public void OnDeliveryPushMessage(String[] msgs)
{
        m_msgLength = msgs.length;
        Log.d(LOG_TAG, "#% OnDelivery msg Length= " + m_msgLength);

        for (int index = 0; index < msgs.length; index++) {
                Log.d(LOG_TAG, index + "=" + msgs[index]);
                . . .
        }
}

// 수신된 Push Message 의 예
OnDelivery msg Length= 1
```

```
0={
"msgid":        "201751BEF10A002E",
"ts":   1371468042,
"cts":  {
"reqData":      "Single push payload"
},
"alert":        "Simulator Test"
}
```

3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해, OnDeliveryPushMessage() 리스너로 올라온 msgid 값을 Parsing 하여 수신된 Push Message 개수만큼 msgId[] 배열에 저장하여 SendPushAck(msgId[])메소드를 호출 한다.

[API ProtoType]

```
/**
    * Delivery Report 기능을 수행하는 메소드
    * Push Message 를 수신한 경우, Push Message 를 수신하였다는 수신 결과를 BizSmartPush Server 로 알리는 메소드.
    * @param msgId[]          수신된 Push Message[] 내용
    * @return int             SendPushAck 에 성공한 경우 transaction_id 가 반환된다.
    */
   public  int SendPushAck(String msgId[])
```

[사용예제]

```
String m_MessageID = new String[msgLength];
// Parsing 된 결과를 인자에 넣어 SendPushAck 함수를 호출 한다.
int ret = m_bizpush.SendPushAck(m_MessageID);
```

SendPushAck()메소드 호출 결과로 3rd Party APP 에서 등록한 리스너인 OnPushAckResult()리스너가 3rd Party APP 으로 올라간다.

```
public void OnPushAckResult(int id, int code)
{
        if(code == 200){// SendPushAck()메소드 호출 성공
```

```
}
else{// SendPushAck() 메소드 호출 실패, 3rd party App 에서 에러 처리
}
}
```

2. SendPushAck()를 시도후 200 OK 응답에 미수신 Push 수신

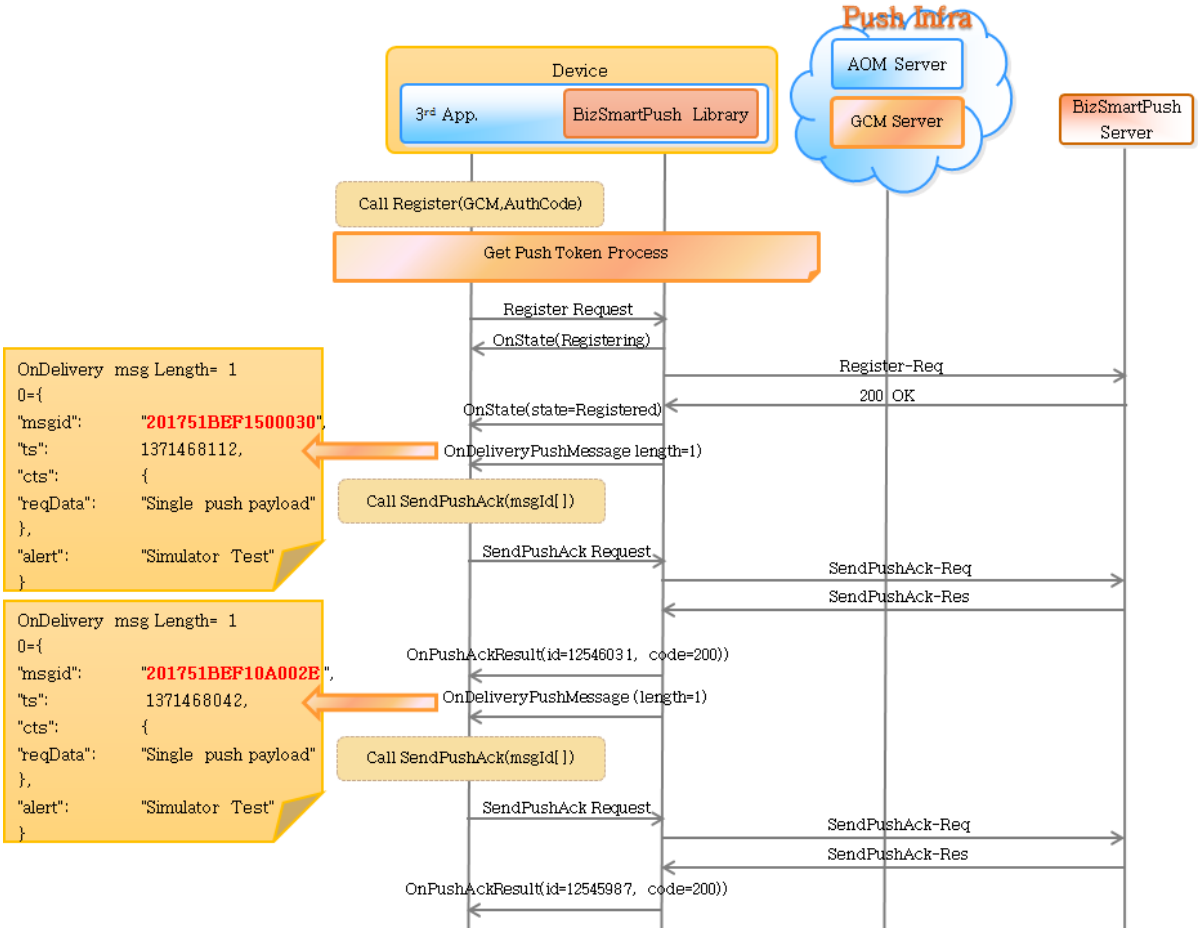


Figure 17 SendPushAck()를 시도 후 200 OK 응답에 미수신 Push 수신

위의 그림과 같이 3rd Party APP 이 Register()메소드 호출 이후에 BizSmartPush 서버로부터 수신한 200OK 내에 Pending Count 가 존재 하는 경우, 3rd Party APP 에서 리스너로 등록한 OnDeliveryPushMessage() 를 통해 Push 수신을 알려준다.

OnDeliveryPushMessage()리스너 안에는 String msgs[] 배열 형태로 Pending Count 개수만큼 Message 의 내용이 아래와 같이 3rd Party APP 으로 전달 된다.

```
OnDelivery msg Length= 1
0={
```

```
"msgid":  "201751BEF1500030",
"ts":         1371468112,
"cts":        {
"reqData": "Single push payload"
},
"alert":      "Simulator Test"
}
```

3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해, OnDeliveryPushMessage 리스너로 올라온 msgid 값을 Parsing 하여 SendPushAck(msgId[])메소드를 호출 한다.

[API ProtoType]

```
/**
    * Delivery Report 기능을 수행하는 메소드
    * Push Message 를 수신한 경우, Push Message 를 수신하였다는 수신 결과를 BizSmartPush Server 로 알리는 메소드.
    * @param msgId[]          수신된 Push Message[] 내용
    * @return int             SendPushAck 에 성공한 경우 transaction_id 가 반환된다.
    */
   public  int SendPushAck(String msgId[])
```

SendPushAck()메소드 호출 결과로 3rd Party APP 에서 리스너로 등록한 OnPushAckResult() 를 통해 호출 결과를 알려준다.

```
public void OnPushAckResult(int id, int code)
{
            if(code == 200){// SendPushAck()메소드 호출 성공
 }
 else{// SendPushAck() 메소드 호출 실패, 3rd party App 에서 에러 처리
 }
}
```

OnPushAckResult()리스너가 올라왔지만, 또 OnDeliveryPushMessage()리스너가 3rd Party APP 으로 올라온 경우, 3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해,

OnDeliveryPushMessage 리스너로 올라온 msgid 값을 Parsing 하여 SendPushAck(msgId[])메소드를 호출 한다.

```
    /**
   * SendPushAck
   * @param msgId
   * @return int
   */
  public  int SendPushAck(String msgId[])
```

SendPushAck()메소드 호출 결과로 3rd Party APP 에서 리스너로 등록한 OnPushAckResult() 를 통해 호출 결과를 알려준다.

```
public void OnPushAckResult(int id, int code)
{
            if(code == 200){// SendPushAck()메소드 호출 성공
 }
 else{// SendPushAck() 메소드 호출 실패, 3rd party App 에서 에러 처리
 }
}
```

9 Push Notification 설정 기능

Push Notification 설정 기능은 3rd Party APP 에서 일시적으로 Push Message 수신을 받고 싶지 않을 경우 제공하는 기능이다. Push Notification 설정 기능은 PushNotificationOnOff(fasle) 메소드를 사용하여 설정하며, PushNotificationOnOff 설정에 따라 BizSmartPush Server 로 부터 Push Message 를 단말로 전송 여부를 설정 할 수 있다.

9.1 Push Notification 설정 기능

9.1.1 Pre-activities

Push Notification 설정 기능을 수행하려면 3rd Party APP 은 반드시 Registered 상태여야 한다.

9.1.2 Push Notification On / Off

본 절은 Registration 후 BizSmartPush Server 로부터 Push 수신 기능을 On / Off 과정을 설명 한다.

Push Notification 설정은 Push Notification On 과 Push Notification Off 로 구분된다.

아래는 PushNotificationOnOff() 메소드의 원형이다.

[API Prototype]

```
/**
    * BizSmartPush Push 수신 On/Off 기능을 수행 하는 메소드
    * @return BP_RET PushNotificationOnOff 에 성공한 경우 Success 가 반환된다.
    */
   public BP_RET PushNotificationOnOff(boolean pushStatus)
```

[사용예제]

```
// Push Message 수신 On
```

```
BP_RET ret = m_bizPushTester.m_bizpush.PushNotificationOnOff(true);

// Push Message 수신 Off
BP_RET ret = m_bizPushTester.m_bizpush.PushNotificationOnOff(false);
```

BizSmartPush Server 로부터 Push Notificaition 설정에 대한 응답으로 200 또는 에러코드를 수신 받으면 아래와 같이 3rd Party APP 에서 등록한 OnPushNotificaionResult(pushStatus, code) 리스너를 통해 호출 결과를 알려준다.

```
@Override
public void OnPushNotificaionResult(boolean pushStatus, int code) {
                        ...
};
```

아래는 OnPushNotificaionResult(boolean pushStatus, int code) 리스너의 인자에 대한 설명이다.

인자 값	Description
boolean pushStatus	PushNotificationOnOff() 메소드 호출 시 인자로 넣은 pushStatus 의 값
int code	서버 응답 결과 코드

아래는 PushNotificaionOnOff() 메소드 호출 결과 성공인 경우 OnPushNotificationResult() 리스너로 전달되는 예이다.

```
// Push Notificaion On 성공된 경우
OnPushNotificaionResult (true, 200)

// Push Notificaion Off 성공된 경우
OnPushNotificaionResult (false, 200)
```

아래는 PushNotificaionOnOff() 메소드 호출 결과 실패된 경우 OnPushNotificationResult() 리스너로 전달되는 예이다.

```
// Push Notificaion On 실패된 경우
OnPushNotificaionResult (true, Error Code)
```

```
// Push Notificaion Off 실패된 경우
OnPushNotificaionResult (false, Error Code)
```

9.1.2.1 Push Notification Off

Push Notification Off 은 3rd Party APP 에서 PushNotificationOnOff() 메소드를 사용하여 Push 수신 기능을 Off 상태로 설정하는 절차이다.

[시나리오]

1. 3rd Party APP 는 BizSmartPush Server 에 등록하기 위해서 Register()를 호출 한다.

2. BizSmartPush Server 로부터 Register() 메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

3. 3rd Party APP 는 Push 수신 기능을 Off 시키기 위해서 PushNotificationOnOff(**false**) 메소드를 호출한다.

4. BizSmartPush Server 로부터 PushNotificaionOnOff(**false**) 메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnPushNotificationResult() 리스너로 등록 성공 여부를 알려준다.

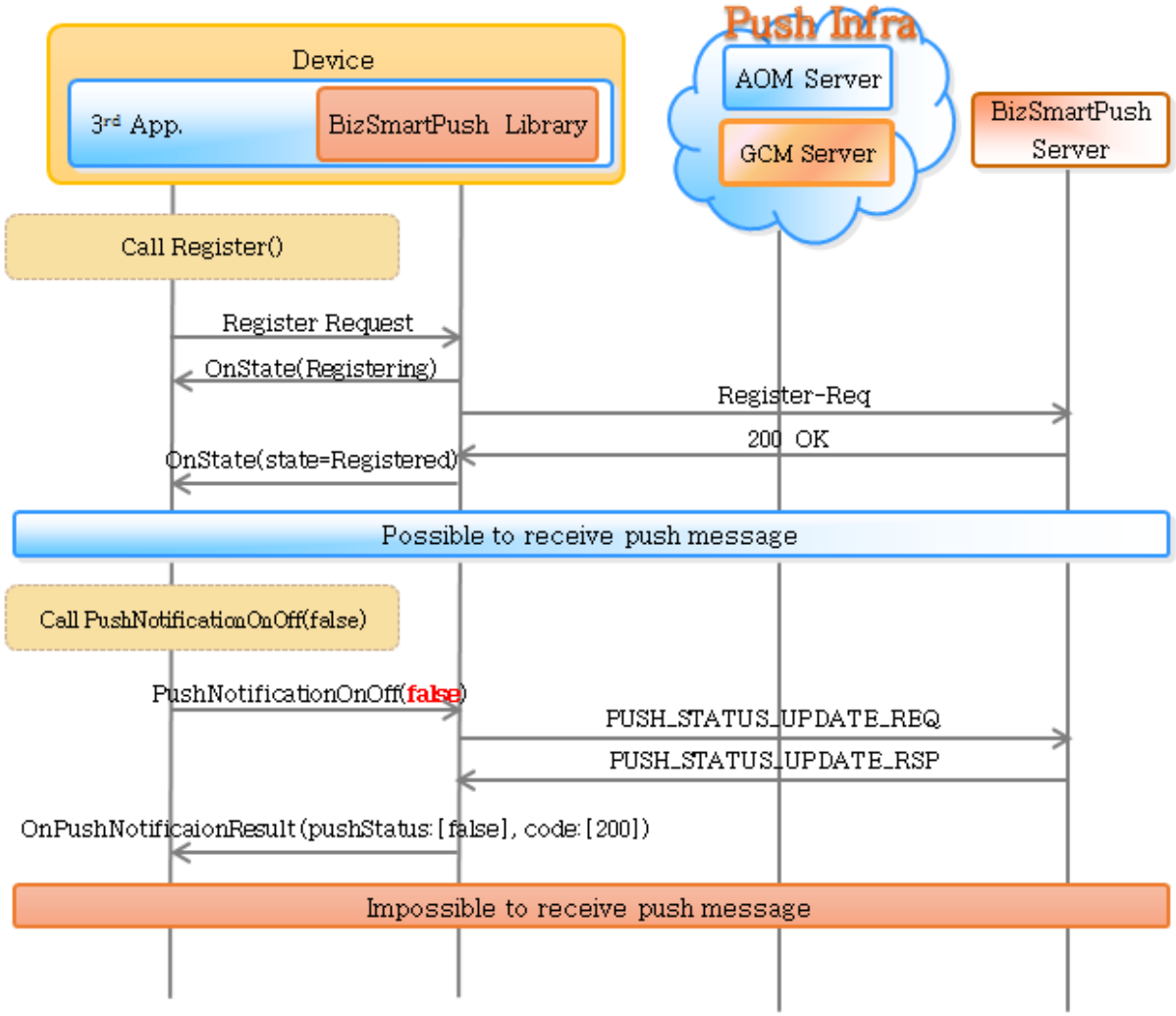


Figure 18 Push Notification Off 설정

9.1.2.2 Push Notification On

Push Notification On 은 3rd Party APP 에서 PushNotificationOnOff() 메소드를 사용하여 Push 수신 기능을 On 상태로 설정하는 절차이다.

[시나리오]

1. 3rd Party APP 는 BizSmartPush Server 에 등록하기 위해서 Register()를 호출 한다.

2. BizSmartPush Server 로부터 Register()메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnState()리스너로 등록 성공 여부를 알려준다.

3. 3rd Party APP 는 Push 수신 기능을 Off 시키기 위해서 PushNotificationOnOff(**false**) 메소드를 호출한다.

4. BizSmartPush Server 로부터 PushNotificaionOnOff(**false**) 메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnPushNotificationResult() 리스너로 등록 성공 여부를 알려준다.

5. 3rd Party APP 는 현재 Off 되어 있는 Push 수신 기능을 On 으로 바꾸기 위해서 PushNotificationOnOff(**true**) 메소드를 호출한다.

6. BizSmartPush Server 로부터 PushNotificaionOnOff(**true**) 메소드 호출에 대한 응답으로 200 OK 를 수신 받으면 3rd Party APP 에서 등록한 리스너인 OnPushNotificationResult() 리스너로 등록 성공 여부를 알려준다.

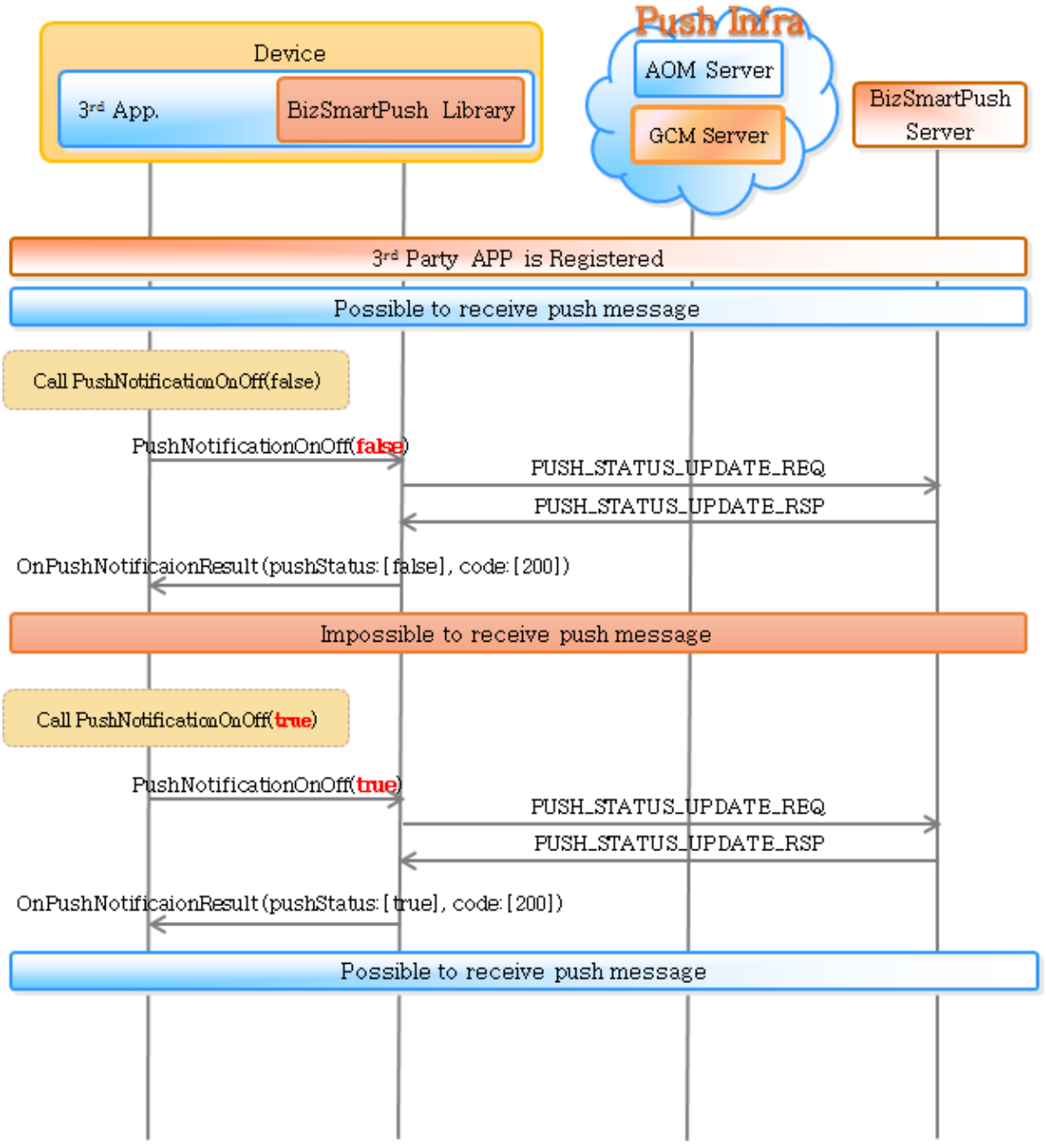


Figure 19 Push Notification On 설정

9.2 Push Notification 설정 시 예외 사항

Case 1. Push Notification Off 상태에서 Smart BizPush Server 로부터 Push Message 를 수신

[시나리오]

1. 3rd Party APP 에서 Register() 호출을 통해 정상적으로 등록
2. 3rd Party APP 에서 PushNotificationOnOff(**false**) 호출을 통해 Push 수신 기능 OFF 상태로 전이
3. Push Notification Off 상태에서 Smart BizPush Server 로 Push Message 수신

[응용 가이드]

3rd Party APP 에서 SendPushAck()를 수행 하지 말고, 해당 Push Message 를 무시 한다.

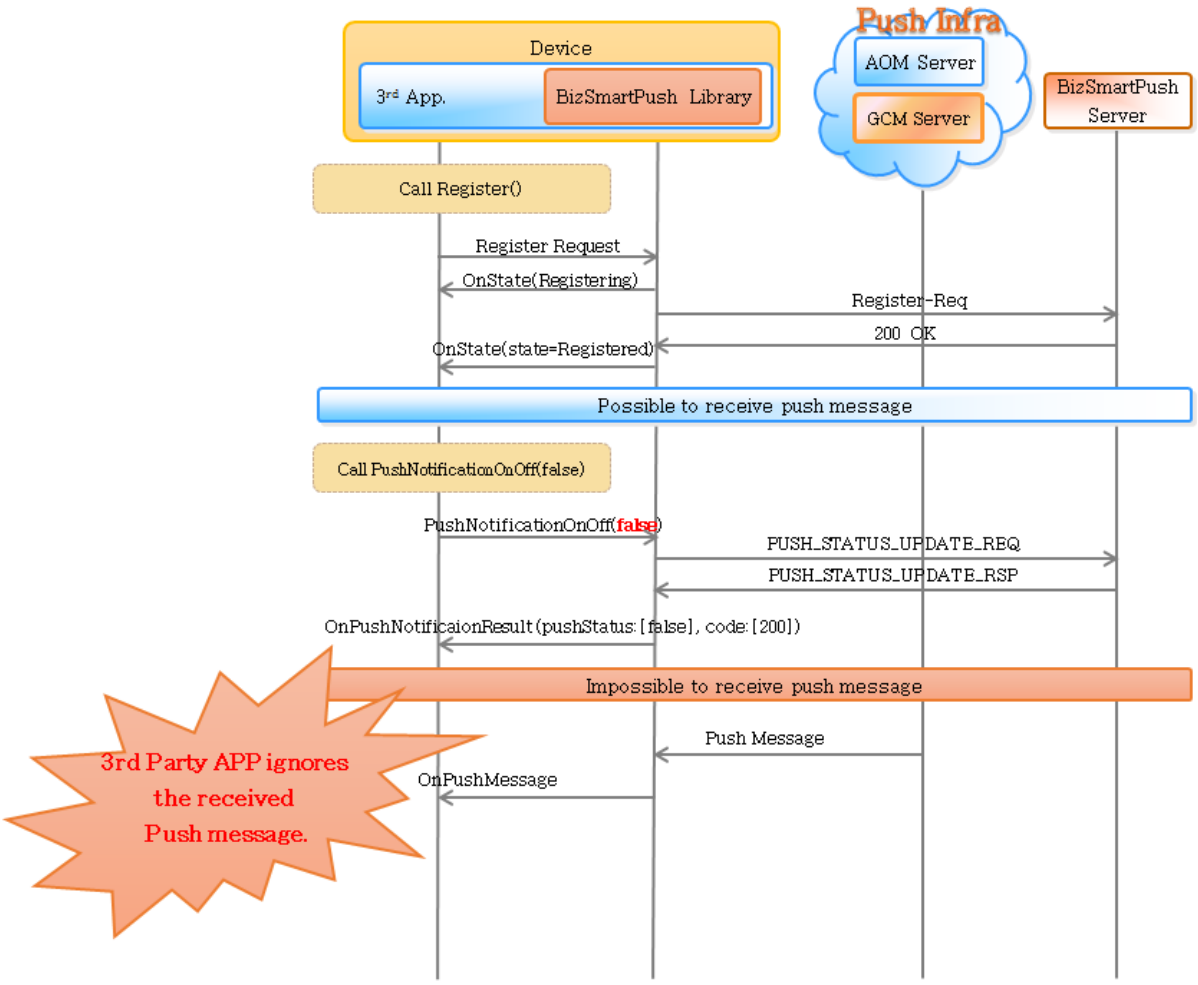


Figure 20 Push Notification Off 상태에서 Push Message 수신 처리

Case 2. PushNotificationOnOff(false) 메소드 호출 후 Smart BizPush Server 로부터 Error Code 를 수신

[시나리오]

1. 3rd Party APP 에서 Register() 호출을 통해 정상적으로 등록
2. 3rd Party APP 에서 PushNotificationOnOff(**false**) 호출
3. Smart BizPush Server 로 Error Code 수신

[응용 가이드]

3rd Party APP 은 Push 수신 상태 관리를 On 으로 관리 한다.

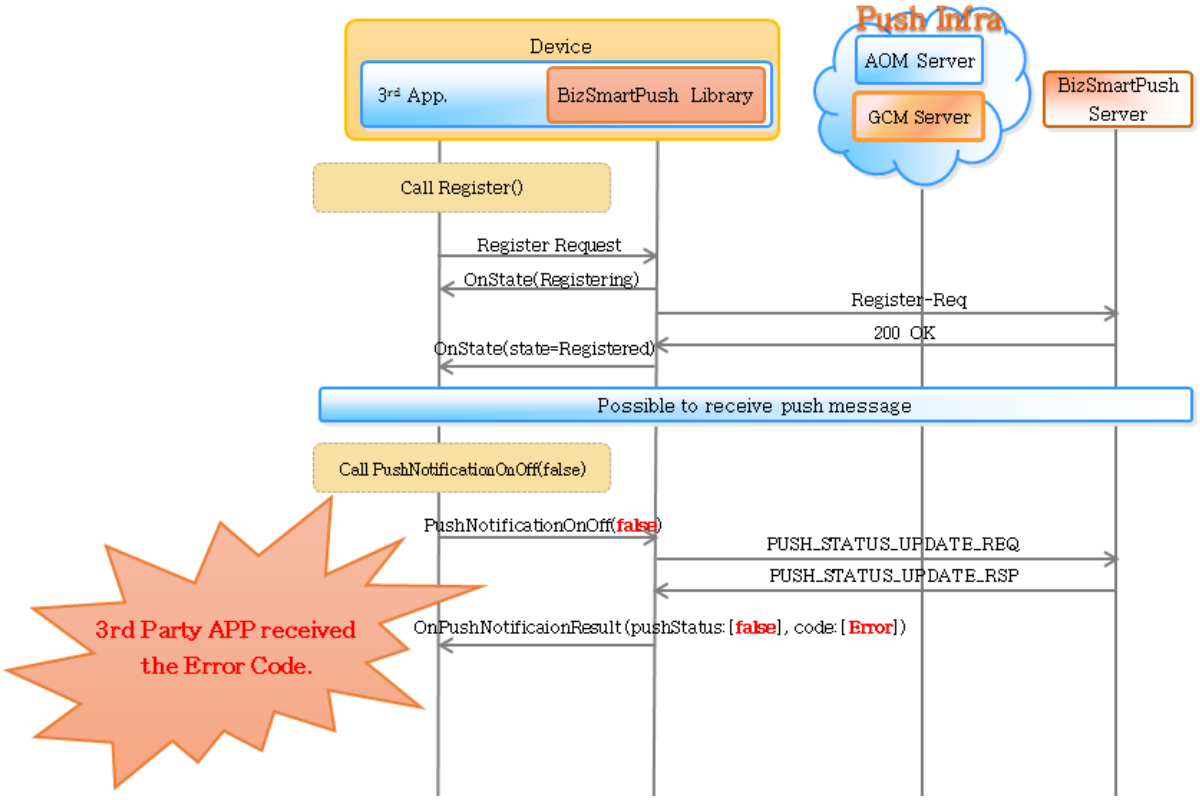


Figure 21 Push Notification Off 호출에 대해 Error Code 수신

Case 3. PushNotificationOnOff(true) 메소드 호출 후 Smart BizPush Server 로부터 Error Code 를 수신

[시나리오]

1. 3rd Party APP 에서 Register() 호출을 통해 정상적으로 등록
2. 3rd Party APP 에서 PushNotificationOnOff(**false**) 호출을 통해 Push 수신 기능 OFF 상태로 전이
3. 3rd Party APP 에서 PushNotificationOnOff(**true**) 호출

4. Smart BizPush Server 로 Error Code 수신

[응용 가이드]

3rd Party APP 은 Push 수신 상태 관리를 Off 으로 관리 한다.

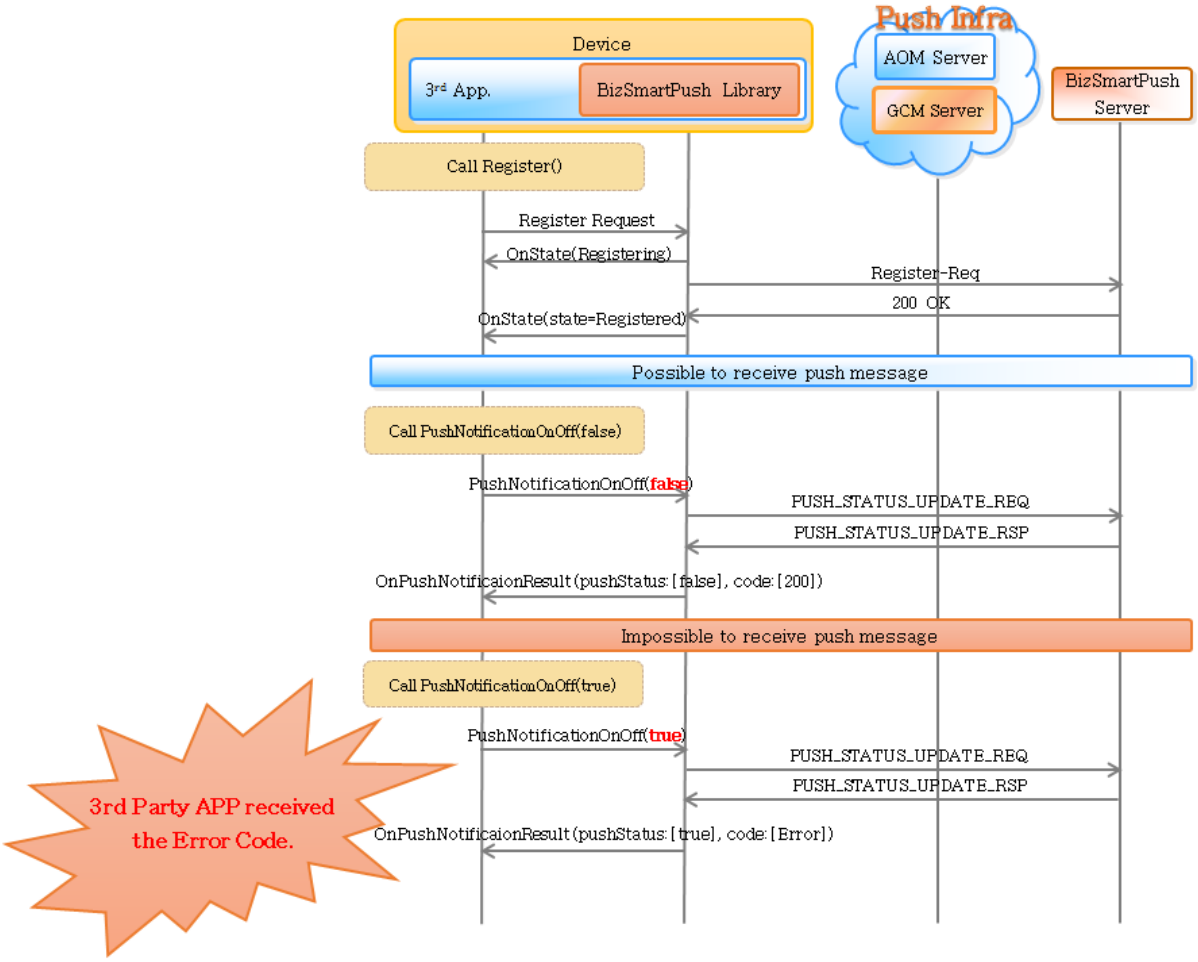


Figure 22 Push Notification On 호출에 대해 Error Code 수신

9.3 Push Notification 설정 결과 Code

Table 10 Push Notification Result Code

Error Return	Description	예외처리
200	OK	메소드 호출 결과 정상 처리.
400	Bad Request(Request Message Parameter Invaild)	Authentication 메소드 호출시 인자값이 잘못되었는지 확인
401	UnAuthorized	Auth Key 가 정상적으로 입력 되어있는지 확인
404	Not Found(AppId)	BizPush 서버에 해당 AppId(GCMID, AOMID)가 등록되어 있는지 확인 필요(서버문의)

406	Message Format Error(Json)	Request Message 가 Json 포멧 규칙에 맞게 구성되어 있는지 확인.
410	**Gone(Not Found ServiceKey)**	ServiceKey 가 정상적으로 설정되어 있는지 확인.
500	BizSmartPush Internal Server Error	서버에러 서버에러 이유를 확인
503	Service Unavailable	서버에러 서버에러 이유를 확인
504	**Request Timeout**	Register Request 이후 BizPush Server 로부터 응답이 없는 경우 발생. 서버 무응답 이유를 확인

10 BizSmartPush 종료

BizPush Library 를 더 이상 사용하지 않을 경우, 다음 API 를 사용하여 종료한다.

10.1 BizSmartPush 종료

3rd Party APP 종료 시 명시적으로 Stop () 메소드를 호출 한다.

[API Prototype]

```
/**
    * BizSmartPush Library 를 정지하기 위한 Stop 메소드
    * @return BP_RET            Stop 에 성공한 경우 Success 가 반환된다.
    */
   public BP_RET Stop()
```

[사용예제]

```
BP_RET ret = m_bizpush.Stop();
er();
```

11 Additional Feature

11.1 Crash Reporter

Crash Reporter 는 단말이 Exception 이 발생하여 죽은경우, Exception 이 발생한 Log 를 E-mail 을 이용하여 전달 할 수 있는 기능이다.

Exception 이 발생하면, 단말에서 바로 Email 을 보낼수 있도록 화면이 전환된다.

Crash Reporter 사용유무를 설정하는 방법은 아래와 같이 SetConfig()메소드를 이용한다.

[사용예제]

```
SetConfig("CREASH_REPORTER", "Enable", "1");
SetConfig("CREASH_REPORTER", "Recipient", aomcsupport@nablecomm.com);
SetConfig("CREASH_REPORTER", "AttachLogcat", "1");
```

11.1.1 SearchForException

단말이 Exception 이 나서 죽었으나, Email 을 통하여 Exception Report 를 못한 경우, 해당 3rd Party APP 에서 SearchForException()를 호출 하여 Exception Report 를 한다.

3rd Party APP 에서 SearchForException()메소드를 호출하면, 단말에서 바로 Email 을 보낼수 있도록 화면이 전환 된다.

[API Prototype]

```
/**
    * Exception 이 발생하면, 단말에서 바로 Exception Report 를 할 수 있도록 Email 을 보내는 기능.
    * @return boolean                    SearchForException 에 성공한 경우 true 가 반환된다.
    */
   public boolean SearchForException()
```

[사용예제]

```
boolean ret = m_bizpush.SearchForException();
```

11.2 Actvity 호출로 실행 시 BSP Push 호출 여부 검사(CheckIntent)

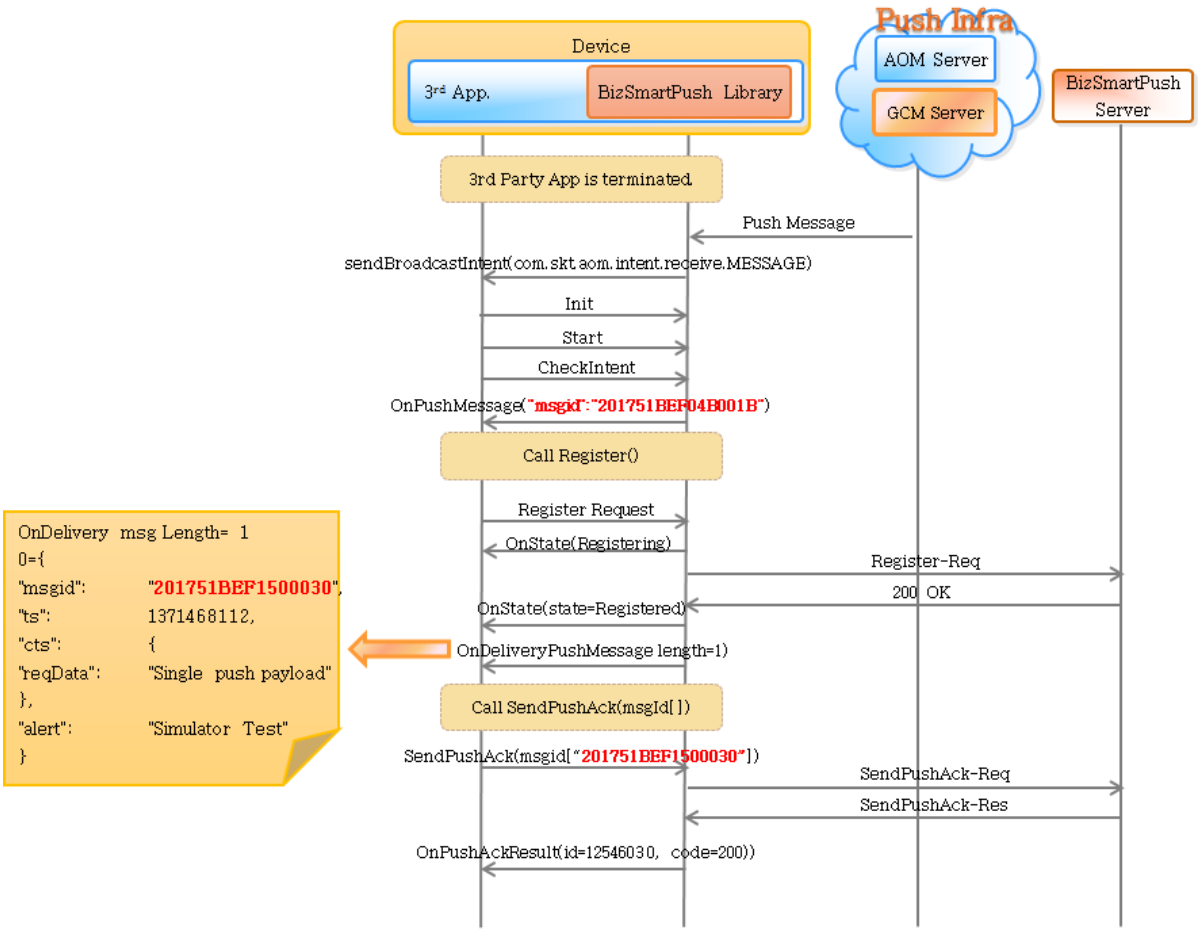


Figure 23 App 강제 후 Push Message 를 수신 받아 깨어난 경우 (CheckIntent)

CheckIntent() 메소드 정의: 3rd Party App 에서 수신 받은 Intent 가 BizSmartPush Intent 인지 확인하는 API.

용도: 3rd Party APP 이 사용자에 의해 강제 종료된 경우 Push Message 수신시 Main Activity 를 통해서 Intent 가 전달된다. 전달된 Intent 가 BizSmartPush Intent 인지 확인 용도로 사용 된다.

주의사항:

1. **6.1.3 절의 SetBroadcastActionName() API 를 사용하여 BrocastReceiver 를 등록한 경우는 명시적으로 응용에서 등록한 BroadcastReceiver 쪽으로 Intent 가 전달되므로 응용은 CheckIntent() API 를 호출 하지 않는다. 즉, 응용은 SetBroadcastActionName() 을 설정하지 않고**

3rd Party APP 에서 Push Message 수신 시 Main Activity 가 호출 될 때 CheckIntent()을 호출한다.

2. **3rd Party App 이 강제종료 되었다가 Intent 를 전달받아 깨어난 경우 이므로, 반드시 응용은 BizPush Init()과 Start() API 를 명시적으로 호출한 다음 CheckIntent() API 를 호출해야 한다.**

[API Prototype]

```
/**
    * 3rd Party App 에서 수신 받은 Intent 가 BizSmartPush Intent 인지 확인 하는 메소드
    * 3rd Party APP 이 사용자에 의해 강제 종료된 경우 Push Message 수신시 Intent 를 통해 3rd Party App 이 깨어 난다.
    * 3rd Party App 이 깨어 날 때 수신된 Intent 가 BizSmartPush Intent 인지 확인 용도로 사용 된다.
    * @param intent
    * @return boolean          CheckIntent 에 성공한 경우 true 가 반환된다.
    */
   public boolean CheckIntent(Intent intent)
```

[사용예제]

CheckIntent() API 의 사용예제는 아래와 같다.

```
boolean ret = m_bizpush.CheckIntent(m_bizPushTester.m_recvIntent);
```

```
//3rd Party App 에서는 Intent 를 받아 App 이 실행된 경우 BizPush Library 의 Init()과 Start()메소드 호출 이후에
//아래의 예제처럼 CheckIntent 를 호출 하면 된다.

Intent m_recvIntent = null;
@Override
 protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);Intent intent = getIntent();
            if (intent != null) {Log.d(LOG_TAG, "intent action = " + intent.getAction());
                        m_recvIntent = intent;
                        boolean ret = m_bizpush.CheckIntent(m_bizPushTester.m_recvIntent);
            }
}
```

CheckIntent() API 의 호출 결과 True 가 3rd Party App 으로 전달된 경우, 수신 받은 Intent 가 BizSmartPush Intent 이므로 3rd Party App 은 BizSmartPush Server 에 등록하기 위해 Register() API 를 호출 한다.

Register() API 호출 결과 BizSmartPush Server 로부터 200 OK 를 수신한 경우 3rd Party APP 에서 리스너로 등록한 OnDeliveryPushMessage() 통해 Push Message 가 전달 된다.

OnDeliveryPushMessage() 리스너 안에는 String msgs[] 배열 형태로 Pending Count 개수만큼 Message 의 내용이 아래와 같이 3rd Party APP 으로 전달 된다.

```
public void OnDeliveryPushMessage(String[] msgs)
{
        m_msgLength = msgs.length;
        Log.d(LOG_TAG, "#% OnDelivery msg Length= " + m_msgLength);

        for (int index = 0; index < msgs.length; index++) {
                Log.d(LOG_TAG, index + "=" + msgs[index]);
                . . .
        }
}

// 수신된 Push Message 의 예
OnDelivery msg Length= 1
0={
"msgid":        "201751BEF10A002E",
"ts":   1371468042,
"cts":  {
"reqData":      "Single push payload"
},
"alert":        "Simulator Test"
}
```

3rd Party APP 은 Message 수신 결과를 BizSmartPush Server 에게 알리기 위해, OnDeliveryPushMessage() 리스너로 올라온 msgid 값을 Parsing 하여 수신된 Push Message 개수만큼 msgId[] 배열에 저장하여 SendPushAck(msgId[])메소드를 호출 한다.

[API ProtoType]

```
/**
```

```
    * Delivery Report 기능을 수행하는 메소드
    * Push Message 를 수신한 경우, Push Message 를 수신하였다는 수신 결과를 BizSmartPush Server 로 알리는 메소드.
    * @param msgId[]          수신된 Push Message[] 내용
    * @return int             SendPushAck 에 성공한 경우 transaction_id 가 반환된다.
    */
   public  int SendPushAck(String msgId[])
```

[사용예제]

```
String m_MessageID = new String[msgLength];
// Parsing 된 결과를 인자에 넣어 SendPushAck 함수를 호출 한다.
int ret = m_bizpush.SendPushAck(m_MessageID);
```

SendPushAck()메소드 호출 결과로 3rd Party APP 에서 등록한 리스너인 OnPushAckResult()리스너가 3rd Party APP 으로 올라간다.

```
public void OnPushAckResult(int id, int code)
{
            if(code == 200){// SendPushAck()메소드 호출 성공
 }
 else{// SendPushAck() 메소드 호출 실패, 3rd party App 에서 에러 처리
 }
}
```

11.3 LOG_TAG 설정

LOG_TAG 설정 이유: BizSmartPush 는 Library 형태로 제공된다. 따라서 BizSmartPush Library 를 사용하여 개발된 3rd Party APP 이 여러 개 생성될수 있다.

BizSmartPush Library 를 사용한 여러 개의 3rd Party APP 이 한 개의 단말에 설치될 경우 ADB LOG 상에 나오는 LOG 가 서로 섞여 Debug 가 어려운 문제 발생.

위와 같은 문제를 해결하기 위해 아래와 같은 기능 제공.

1. ADB Log 의 LogTag 를 사용자가 변경할 수 있도록 기능제공.

2. Android 단말에 저장되는 FileLog 의 prefix 를 사용자가 변경할 수 있도록 기능제공

[사용예제]

3rd Party App 으로 제공되는 smartbizpush.ini 파일에 아래와 같이 설정.

```
// [LOGGER] Section 에서 LogTag 를 원하는 값으로 설정(예: BizPushTest)
[LOGGER]
;Use Log Tag
LogTag=BizPushTest
```

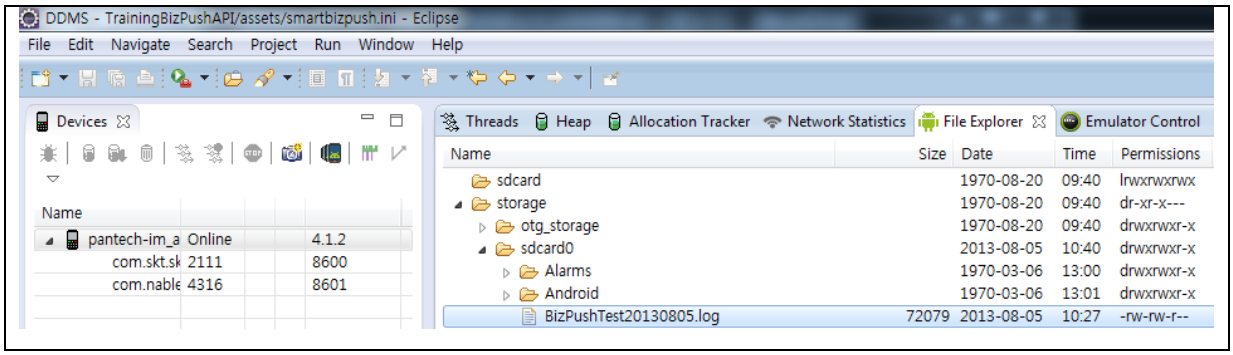
1.ADB Log 확인

위 예시와 같이 [LOGGER] Section 에서 LogTag 를 BizPushTest 로 설정 하면 ADB Log 에 아래와 같이 출력됨.

```
BizPushTest     102738|[DEBUG] #@ IN BizPush_Init(smartbizpush.ini,0x67cc28fc)
BizPushTest     102738|[DEBUG] #@ RET BizPush_Init result=0(BP_RET_Success)
BizPushTest     #! OUT OnState, type=NONE, state=Standby, code=0
. . .
```

2. FileLog 확인

위 예시와 같이 [LOGGER] Section 에서 LogTag 를 BizPushTest 로 설정 하면 Android File Explorer 에 BizPushTest20130805.log 와 같이 FileLog 가 생성됨을 확인 할 수 있다.



11.4 AOM 서비스 상태 조회 및 처리 방안

정의: 3rd Party App 에서 Aom Client 의 상태를 확인 할 수 있는 메소드

용도: CheckAomService 메소드를 사용하여 3rd Party App 에서는 Aom Client 의 현재상태를 확인 한다.

[API Prototype]

```
/**
    * 응용에서 Aom Client 의 상태를 확인할 수 있는 기능을 제공
    * @return BP_RET          Query 에 성공한 경우 Success(0)가 반환된다.
    *                         CheckAomService 를 중복 호출한 경우 RecursiveCall(14)가 반환 된다.
    *                         단말에 Aom Client 가 설치되어 있지 않은 경우 AOMCNotinstalled(16)가 반환 된다.
    *                         Query 에 실패한 경우 Failed(1)가 반환된다.
    */
   public BP_RET CheckAomService()
```

[호출 결과 정리 및 응용 가이드]

CheckAomService()의 호출 결과는 아래와 같이 CheckAomService() 메소드의 BP_RET 값 또는 3rd Party App 에서 리스너로 등록한 OnAomCheckStatus(AOM_STATUS_REASON error)리스너로 전달된다.

3rd Party App 에서는 CheckAomService()메소드를 주기적(Ex. 10 분, 20 분, etc)으로 호출 하여 Aom 의 상태를 체크하여 Registered 상태를 유지하도록 한다.

또한, Android AlarmManger 를 활용하여 단말이 Sleep 상태에 있더라도 CheckAomService() 메소드 호출에 이상이 없도록 예외처리를 한다.

아래는 CheckAomService() 메소드 호출 결과와 그에 따른 응용가이드를 정리한 표이다.

Table 11 CheckAomService 메소드의 BP_RET Return

BP_RET	Description	응용 가이드
Success	Aom Service 가 정상적으로 등록	호출 성공
RecursiveCall	CheckAomService() 메소드 중복 호출	중복 호출 에러
FailedQuery	CheckAomService() 메소드 호출 후 Query 에 실패	AOM Query 에 실패한 경우, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
AOMCNotinstalled	단말에 Aom Client 가 설치되어 있지 않음	Aom 이 설치되어 있지 않은 단말 이므로 AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행

아래는 3rd Party App 에서 등록한 OnAomCheckStatus(AOM_STATUS_REASON error)리스너의 AOM_STATUS_REASON 을 정리한 표이다.

Table 12 AOM_STATUS_REASON

AOM_STATUS_REASON	Description	응용 가이드
Success	PushType 을 Aom 또는 Auto 로 Register 시도 가능 상태	AOM Type 으로 Register 상태이면 현상태 유지, PushType 이 GCM 이면 Auto 또는 AOM 으로 설정 후 Register 수행
NotSupportedDevice	AOM 서비스를 지원하지 않는 단말	AOM 서비스를 지원하지 않은 단말, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
OutOfService	Aom Client 가 현재 Aom Server 에 등록되지 않은 경우	Aom 이 등록 되어 있지 않음, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
MyPushOff	Aom Client 에 3rd Party App 이 비활성화 되어 있는 경우	3rd Party App 이 비활성화, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
Timeout	CheckAomService() 메소드 호출 후 Aom Client 로부터 응답이 없어 Timeout 이 발생한 경우	Aom Client 로부터 응답이 없어 Timeout 이 발생, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
NotRegistered	3rd Party App 이 미등록 상태인 경우	Aom 에 3rd Party App 이 비활성 상태, AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행

11.5 AOM 서비스 상태 알림 수신 및 처리 방안

Aom 의 상태가 변경되었을 때 3rd Party App 에서 등록한 OnAomServiceStatus(AOM_SERVICE_STATUS status) 리스너로 AOM 의 서비스 상태가 전달된다.

[Listener Prototype]

```
private BizPushListener m_bizpushListener = new BizPushListener() {
        @Override
        public void OnAomServiceStatus(AOM_SERVICE_STATUS status) {
                // TODO Auto-generated method stub
                m_bizPushTester.PrintLog("#% OnAomServiceStatus status :" + status.getValue());
        }
```

```
}
```

[Listener 결과 및 응용 가이드]

AOM Client 의 상태변화가 발생하면 3rd Party App 에서 등록한 OnAomServiceStatus () 리스너로 아래와 같이 상태변화에 대한 결과가 전달된다.

아래는 Aom 상태변화와 그에 따른 응용가이드를 정리한 표이다.

Table 13 AOM_SERVICE_STATUS

AOM_SERVICE_STATUS	Description	응용 가이드
UNAVAILABLE	AOM Client 와 AOM SERVER 간 서비스 불가 상태.	AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 GCM 으로 설정 후 Register 재수행
AVAILABLE	AOM Client 와 AOM SERVER 간 서비스 정상 상태.	AOM Type 으로 Register 상태이면 현상태 유지, PushType 이 GCM 이면 Auto 또는 AOM 으로 설정 후 Register 수행
RE_REGISTER	AOMC 가 삭제 된 후 재설 치, 혹은 환경설정에서 AOMC 의 정보를 삭제하면 AOMC 는 3rd Party Application 에 재등록 요청 후 성공시 RE_REGISTER 를 3rd Party App 에 전달.	AUTO 모드 시 AUTO 모드로 Register 재수행, AOM Type 으로 연동 시 AOM 으로 설정 후 Register 재수행

11.6 Register state & Procedure

11.6.1.1 State 변경 확인

사용자 API 호출 이나 서버응답에 의해 State 가 변경되면 3rd Party APP 에서 리스너로 등록한 OnState() 로 State 변경을 알려준다.

BizSmartPush 의 State 는 아래와 같다

Table 14 BizSmartPush 의 State

BP_STATE	Description
None	Register 시도 이전 상태
Registering	Register 시도중인 상태
Registered	Register 등록이 완료된 상태
UnRegistered	Register 해지 완료 상태

Error	Error
CreatingToken	Token 발급 시도 상태
Standby	BizSmartPush 초기화(Init) 상태

11.7 SMS Hooking 연동 방안

안드로이드는 SMS 수신 시 SMS 도착 여부를 모니터링하는 방안을 제공한다. App. 개발 시 아래와 같이 Manifest 에 SMS Received Intent Filter 를 정의하고, 해당 Broadcast Receiver 에서 SMS 수신 시 자신의 SMS 인지 여부를 판단하는 액션을 기술한다.

SMS 수신 시 비교에 필요한 SMS 속성에 대해서는 아래의 사이트를 참조한다.

[참고 사이트] http://developer.android.com/reference/android/telephony/gsm/SmsMessage.html

[Manifest.xml]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.nable.training.bizpush.tester"
    android:versionCode="100"
    android:versionName="1.0" >

<receiver android:name=".SystemBroadcastReceiver" android:enabled="true">
            ...
            <!-- SMS Hooking  -->
           <intent-filter android:priority="2147483647">
               <action android:name="android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
            ...
</receiver>
```

[사용예제]

```
//아래의 예제처럼 Broadcast Reveiver 에서 SMS 수신을 체크하여 각 3rd Party App 에 맞게 동작하면 된다.
static final String INTENT_SMS_RECEIVED = "android.provider.Telephony.SMS_RECEIVED";

@Override
 public void onReceive(Context context, Intent intent) {
              AOMLog.d(LOGTAG, "IN : onReceive() : SystemBroadcastReceiver");String action = intent.getAction();
```

```
try {
Log.d(LOGTAG, "#@ [IN] " + intent.toString() + " : onReceive : SystemBroadcastReceiver");

Bundle extras= intent.getExtras();
if (extras != null)
        Log.d(LOGTAG, "#@       " + extras.toString() + " : onReceive : SystemBroadcastReceiver");

// SMS Hooking
if (action.equals(INTENT_SMS_RECEIVED)) {
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        if (bundle != null) {
                StringBuffer buffer = new StringBuffer();
                Object[] pdus = (Object[]) bundle.get("pdus");
                msgs = new SmsMessage[pdus.length];
                for (int i=0; i<msgs.length; i++){
                        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);
                        buffer.append(msgs[i].getMessageBody().toString());
                        Log.e(logTag,      "DisplayOriginatingAddress      :      "            +      msgs
[i].getDisplayOriginatingAddress());
                        Log.e(logTag,         "DisplayMessageBody         :         "         +         msgs
[i].getDisplayMessageBody());
                    Log.e(logTag, "EmailBody : " + msgs[i].getEmailBody());
                    Log.e(logTag, "EmailFrom : " + msgs[i].getEmailFrom());
                    Log.e(logTag, "OriginatingAddress : " + msgs[i].getOriginatingAddress());
                    Log.e(logTag, "MessageBody : " + msgs[i].getMessageBody());
                    Log.e(logTag, "ServiceCenterAddress : " + msgs[i].getServiceCenterAddress());
                    Log.e(logTag, "TimestampMillis : " + msgs[i].getTimestampMillis());
                }
        buffer.append("₩n");

        String data = buffer.toString();

        // SMS Push String 일치 여부 검사
        if (data.indexOf(SMS_PUSH_FILTER) != -1)
        {
                // 각 3rd Party App. 의 서비스 시나리오 동작

                // SMS Broadcast 중지 요청
```

```
                this.abortBroadcast();
            }

        return;
    }
```

12 Appendix – Android GCM 연동 방법 가이드

12.1 Introduction

12.2 Preface

본 문서는 GCM 라이브러리를 이용하여 응용 개발하는데 필요한 정보를 제공한다.

12.3 Terms

아래 표는 GCM 과 관련된 주요 용어와 컨셉을 요약한 것이며, 두 개의 카테고리로 분리.

● Components – 단말기(3rd-Party APP), 3rd-Party APP Server, GCM Server

● Credentials – 각종 ID 와 토큰.

Components	
Mobile Device	GCM 을 사용하는 APP 이 동작하는 디바이스. 디바이스는 반드시 Google Play Store 가 설치되어 있는 OS 2.2(Froyo) 버전 이상이어야 하며, 4.04 보다 낮은 버전의 OS 를 사용하는 디바이스인 경우에는 적어도 한 개의 계정이 구글 계정에 로그인 되어 있어야 한다.
3rd-party APP Server	3rd-party APP Server 는 GCM Server 를 통해 안드로이드 디바이스의 3rd-party APP 으로 데이터를 전달
GCM Server	3rd-party APP Server 에서 메시지를 디바이스로 보내는 것을 처리하는 구글 Server.
Credentials	
Application ID	메시지 수신이 설정되어 있는 APP. APP 은 AndroidManifest 파일의 패키지명에 의해 식별된다. 메시지가 정확하게 APP 으로 전송되도록 한다.
Project Number	API Console Site 에서 얻은 project Number. (API Console Site 에 접속 하여 정보등록 후 값을 얻어 오는 설명은 아래 2.5.1 에 기술되어 있음) Project Number 는 단말이 GCM Server 로 Push 수신 등록 과정에 사용 한다.
Registration ID	Push 메시지 수신 허용된 APP 에게 GCM Server 가 발급한

	Registration ID. APP 이 Registration ID 를 받으면 ID 를 3rd-party APP Server 로 보내야 한다. Registration ID 는 APP 에 대한 메시지 수신이 허용된 디바이스 각각을 구분하기 위해 사용 된다. 즉, Registration ID 는 특정 디바이스에서 동작되는 특정 APP 과 연결 한다.
Google User Account	만약 디바이스가 안드로이드 4.04 버전보다 낮은 경우 GCM 이 올바르게 작동 되기 위해서는 적어도 한 개의 Google 계정이 해당 디바이스에 설정되어 있어야 한다.
Sender Auth Token	3rd-Party APP Server 에 저장되는 API 키는 구글 서비스의 접근이 허용된 APP Server 에게 주어지는 키 이다. 이 API 키는 메시지를 보낼 POST 요청의 헤더에 포함되어 있다.
C2DM	Cloud to device message 의 줄임말 안드로이드 프로요 이상에서만 가능하며 C2DM Server 는 google 에서 device 간에 Message 를 전달하기 위해서 만든 서버.

12.4 References

본 요구사항에 언급되지 않은 사항은 아래를 준수 한다.

기관	Specifications & Standards
Google apis	https://code.google.com/apis/console
Android Developers	http://developer.android.com/google/gcm/index.html

12.5 How to use GCM

12.6 GCM 이란?

- GCM 이란 Google Cloud Messaging for Android 로써, Android 를 지원하는 Cloud Service 의 형태를 가진 Google 의 푸쉬 알림 서비스 이다.
- App 이 종료되어 있을때에도 Android 에서 해당 메시지를 모니터링 하고있기 때문에, 알림 메시지가 수신되면 해당 app 으로 알림 메시지를 전달한다.
- APP 은 해당 Push 메시지를 받기 위해 실행 중일 필요가 없다. APP 이 적절한 브로드캐스트 리시버와 퍼미션으로 구성되어 있으면 시스템이 인텐트 브로드캐스트를 통해서 해당 APP 을 깨워서 정상적으로 Push 메시지 수신을 함.

참고:

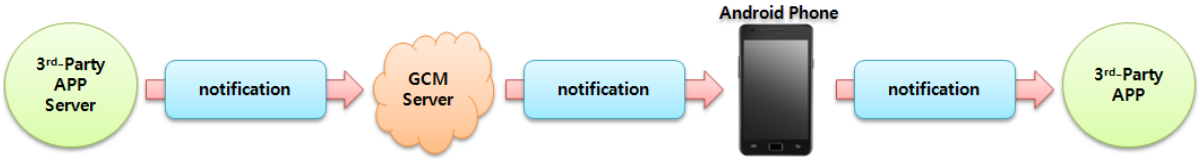
C2DM(Android Cloud to Device Messaging)의 베타 버전을 대체

C2DM 은 사용 중단, 더 이상 신규 C2DM 가입 불가

C2DM 과 GCM 의 차이

조건 및 기능	C2DM	GCM
App 을 재설치할 때마다 Push 에 사용되는 고유값	APP 재설치 마다 변경	디바이스에 같은 APP 이라면 동일한 고유값 고정
인증	Auth 인증 절차 필요	Auth 인증 절차 없음. Simple API Key 만을 사용
Multicast messages 지원 여부	지원 안함	지원

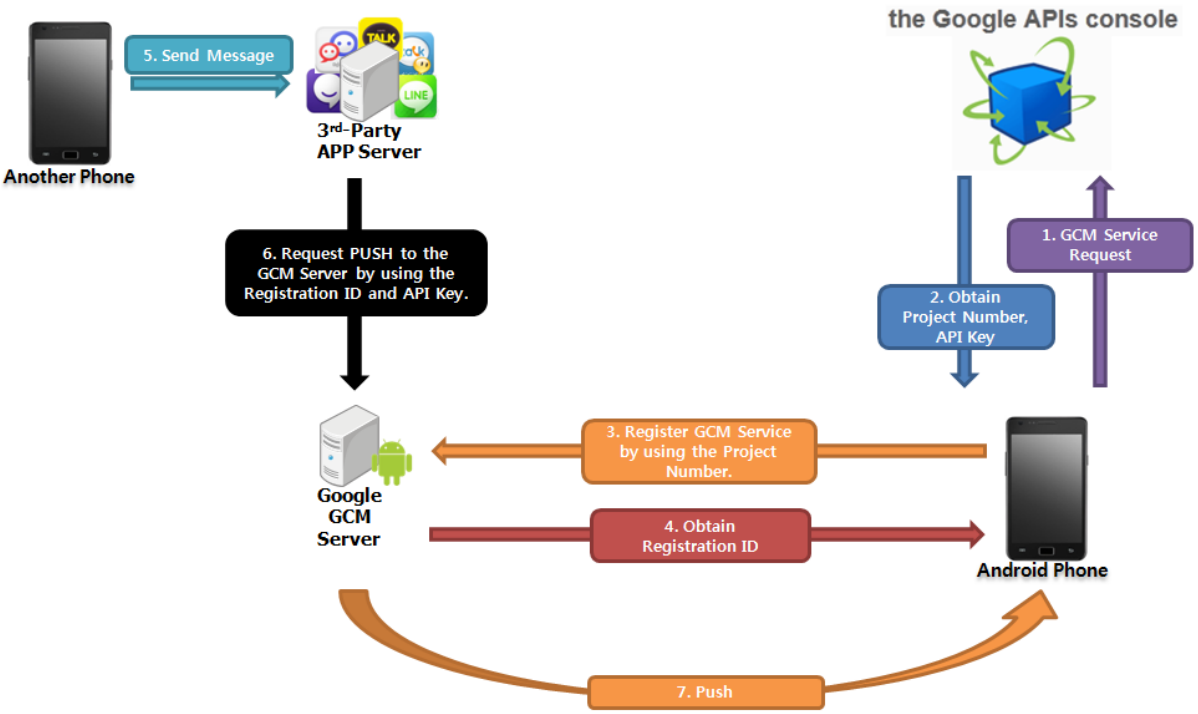
12.7 서비스 구성



서비스의 구성은 위의 그림과 같이 3rd-Party APP Server 에서 GCM Server 로 Push 요청을 하면, GCM Server 가 3rd-Party APP 이 설치된 Android Phone 으로 Push Message 를 보냄.

1. **3rd-Party APP Server** - 3rd Party APP 의 서비스를 제공 하는 Server 로 Android 단말에 설치된 3rd Party APP 으로 Message 를 수, 발신 하는 역할
2. **GCM Server -** Android Phone 으로 Push Message 를 보내는 Cloud Server.
3. **3rd-Party APP** – GCM Server 에게 Push 를 받을수 있도록 등록, GCM Server 로부터 Push 를 수신 받는 개발자가 만든 Application.

12.8 GCM 연동 방법



1. Google APIs Console 페이지로 이동하여 GCM 서비스 신청
2. **Project Number**(단말에서 GCM 등록을 위한 값), **API Key**(서버에서 단말에 푸시를 보낼때 사용하는 값) 획득
3. Push 를 받을 단말에서, Project Number 를 이용하여 GCM 서비스 등록
4. GCM Server 로부터 Registration ID 획득.
5. 다른 단말이 Message 전송
6. API Key 와 Registration ID 를 이용하여, GCM 에게 Push 전송 요청
7. GCM 서버가 단말에게 Push 전송

12.9 GCM 사용에 필요한 준비

1. Google API Project site 에서 계정 등록
2. 소스 코드 작성시 필요한 GCM 라이브러리(gcm.jar, gcm-server.jar) 다운로드
3. 계정 등록 결과로 Registraion ID(단말에서 GCM 등록을 위한 값), API Key(서버에서 단말에 푸시를 보낼때 사용하는 값) 획득
4. AndroidManifest 에 GCM 사용 관련 permission 과 receiver, service 를 등록. (2.6.3 절 참조)

12.10 GCM 연동 방법

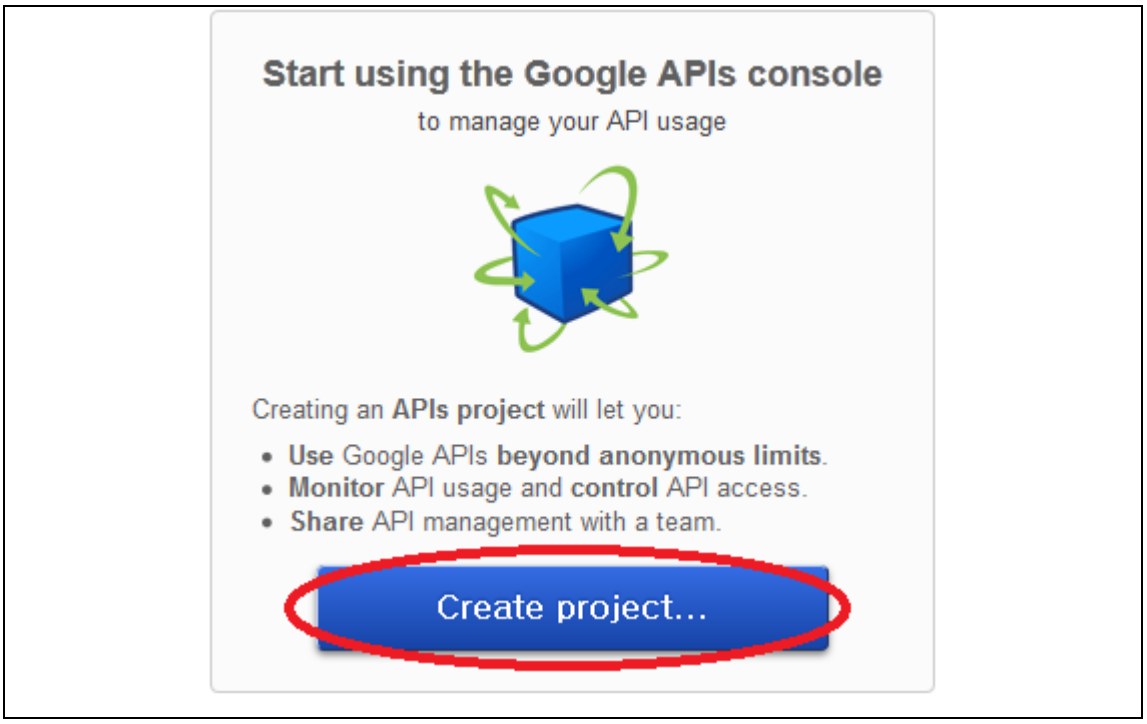
12.10.1 GCM 서비스 신청 하기

1.Project ID 와 API Key 를 획득하기 위해서 아래 주소로 이동

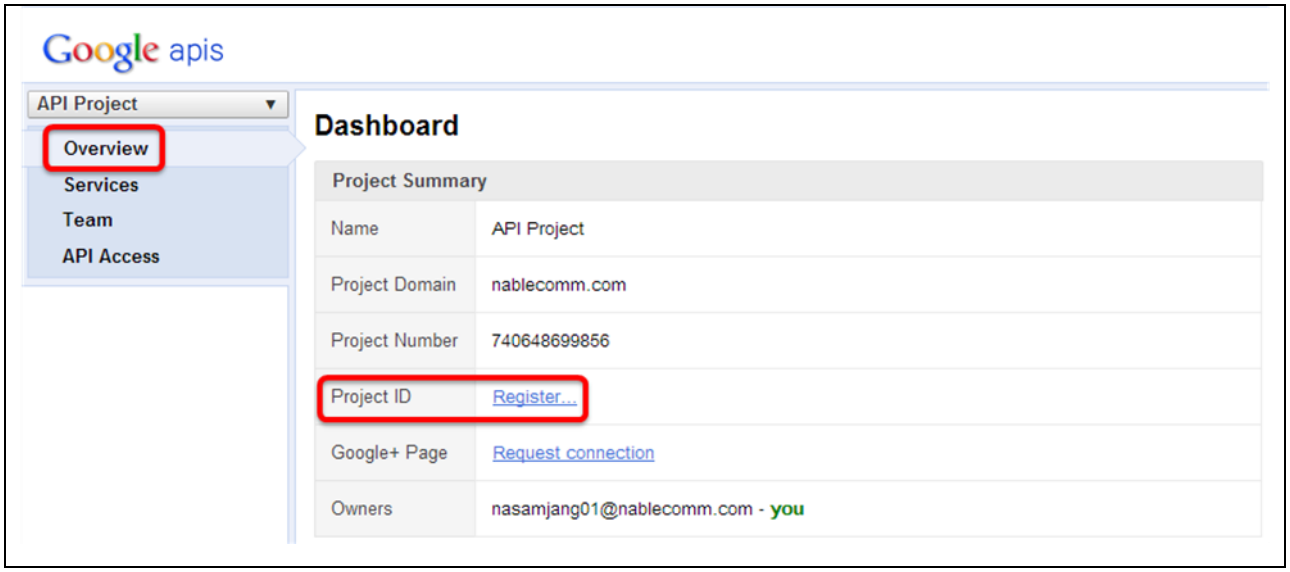
https://code.google.com/apis/console/

2.Google API 를 한번도 사용한 적이 없으면, 아래와 같은 화면이 나온다.

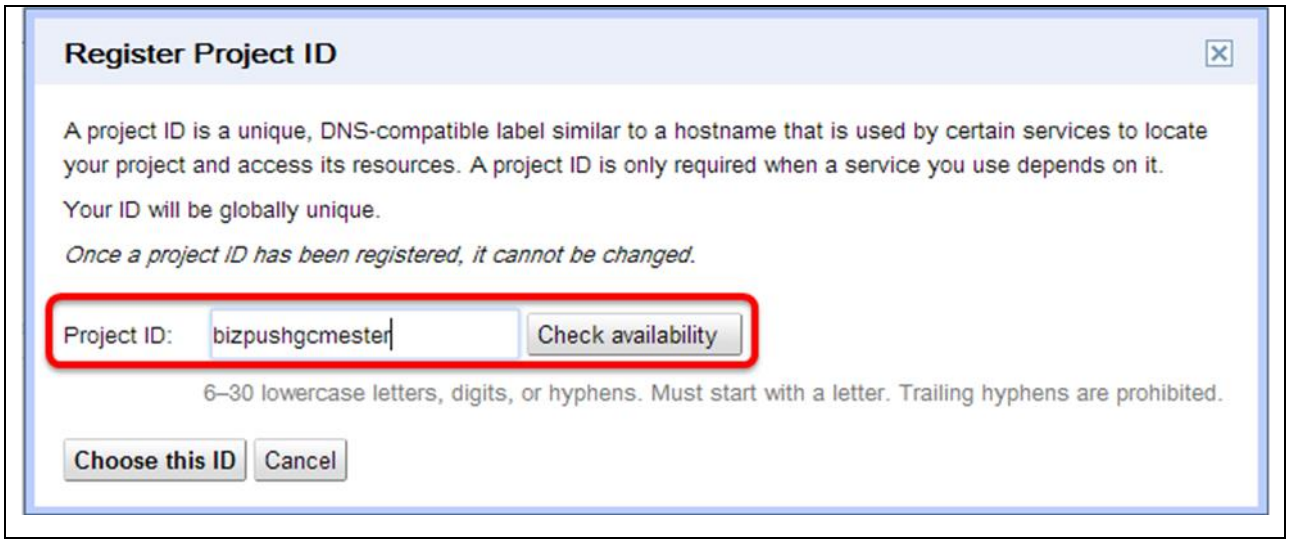
'Create project……'버튼을 눌러 등록을 시작한다.



3. 프로젝트 생성 후 아래와 같이 'Overview' 로 이동한다. 그 후 'Project ID'를 등록 한다.

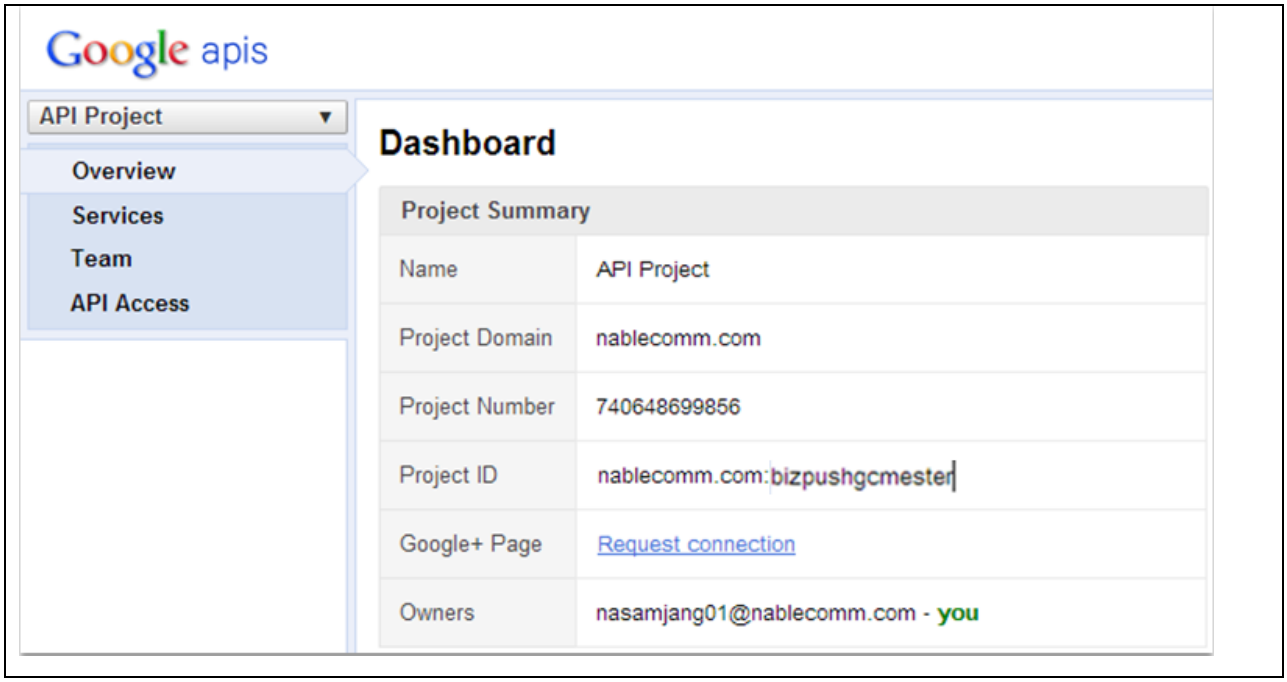


4. 등록 하기 원하는 ID 명을 'Project ID'란에 등록 한다.

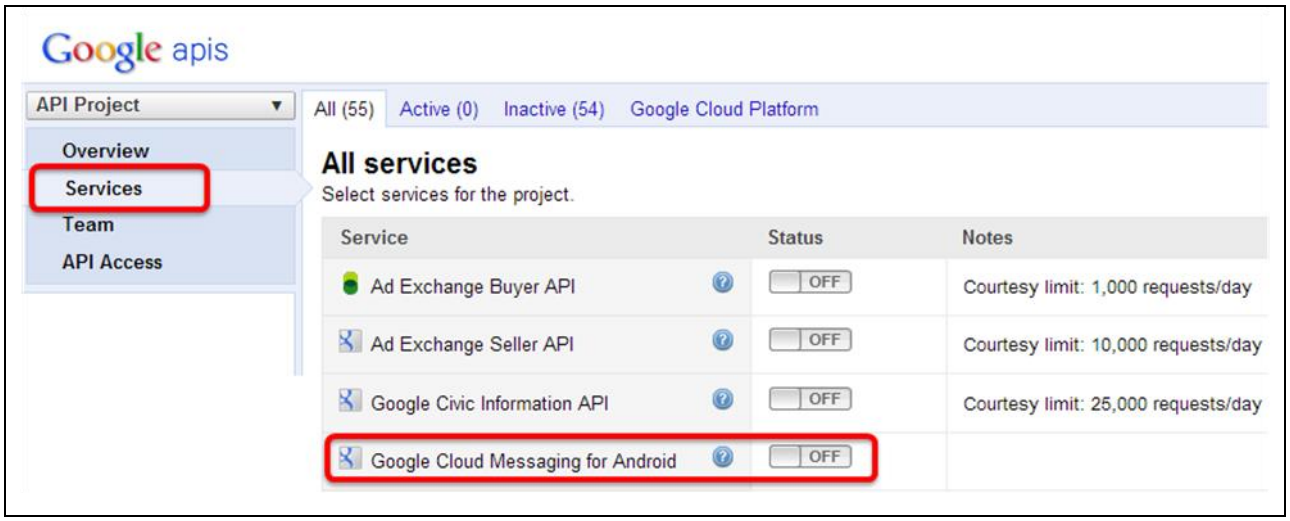


5.'Project ID'등록이 완료 되면, 아래와 같이 'Project Number'가 생성된다.

GCM 등록할 때 필요한 값이 므로 잘 보관해 놓는다.

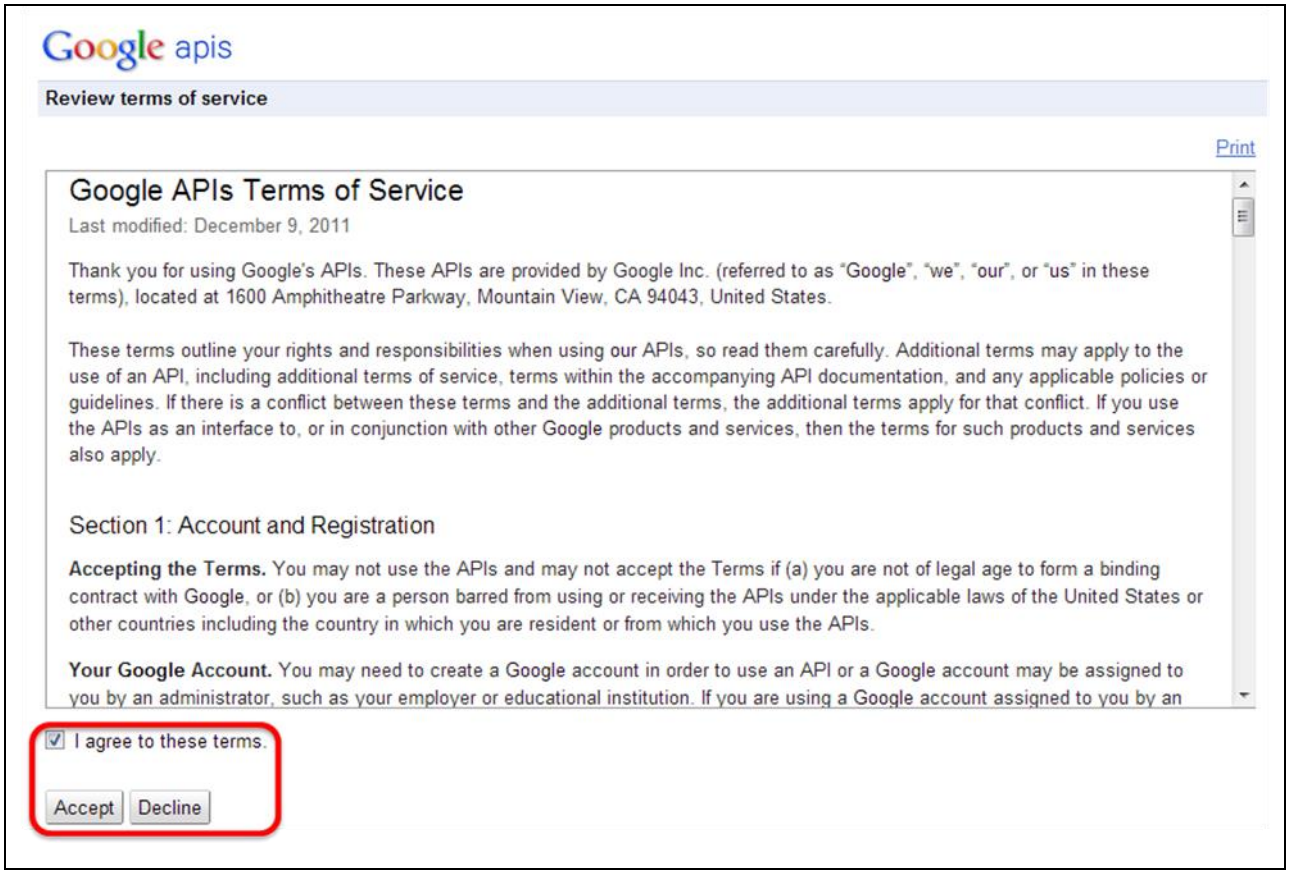


6. 왼쪽 메뉴 중 Servies 메뉴로 이동 후 GCM 서비스를 활성화 시킨다.



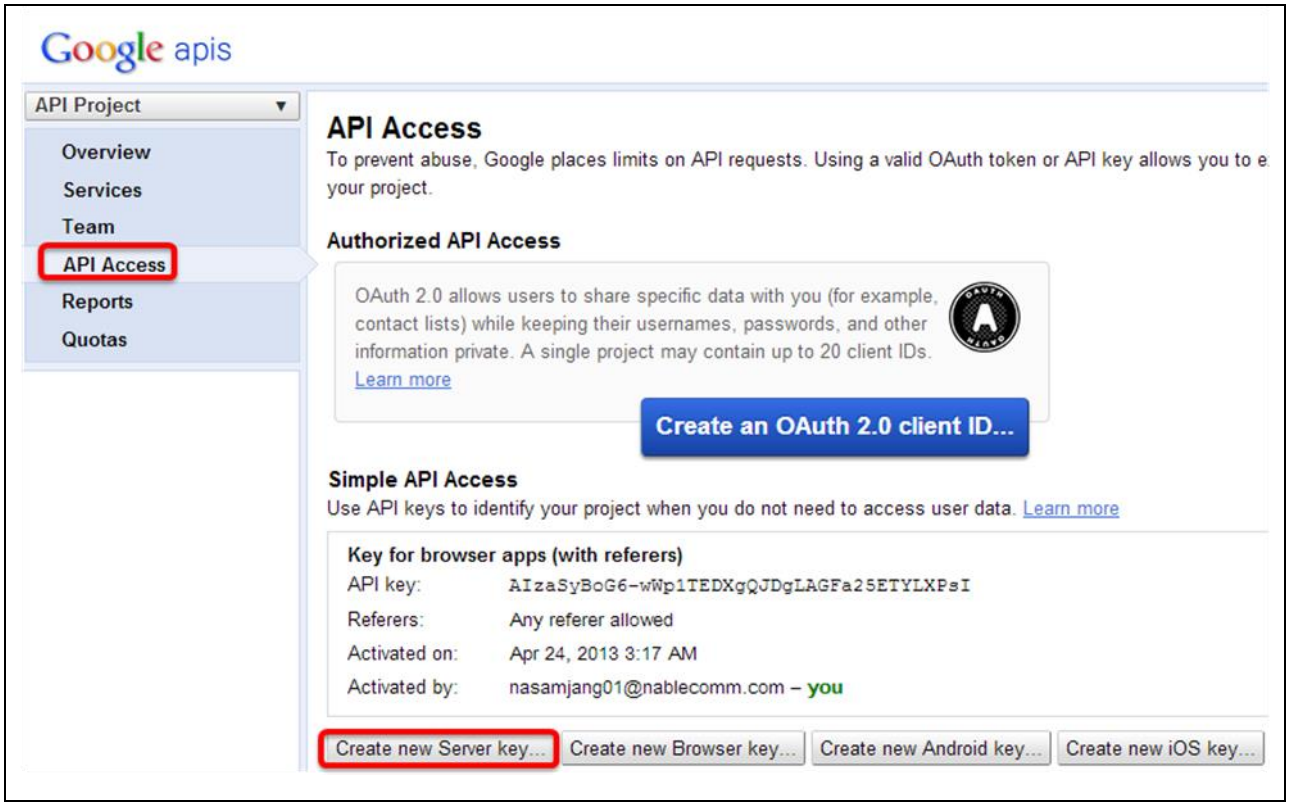
7. GCM 서비스를 활성화 시키면, Google 서비스 API 를 이용 약관이 나온다.

API 를 사용하기 위해서, 아래와 같이 동의를 한다.



8. GCM 서비스를 활성화 시켰으면 왼쪽 메뉴 중 'API Access' 메뉴로 이동한다.

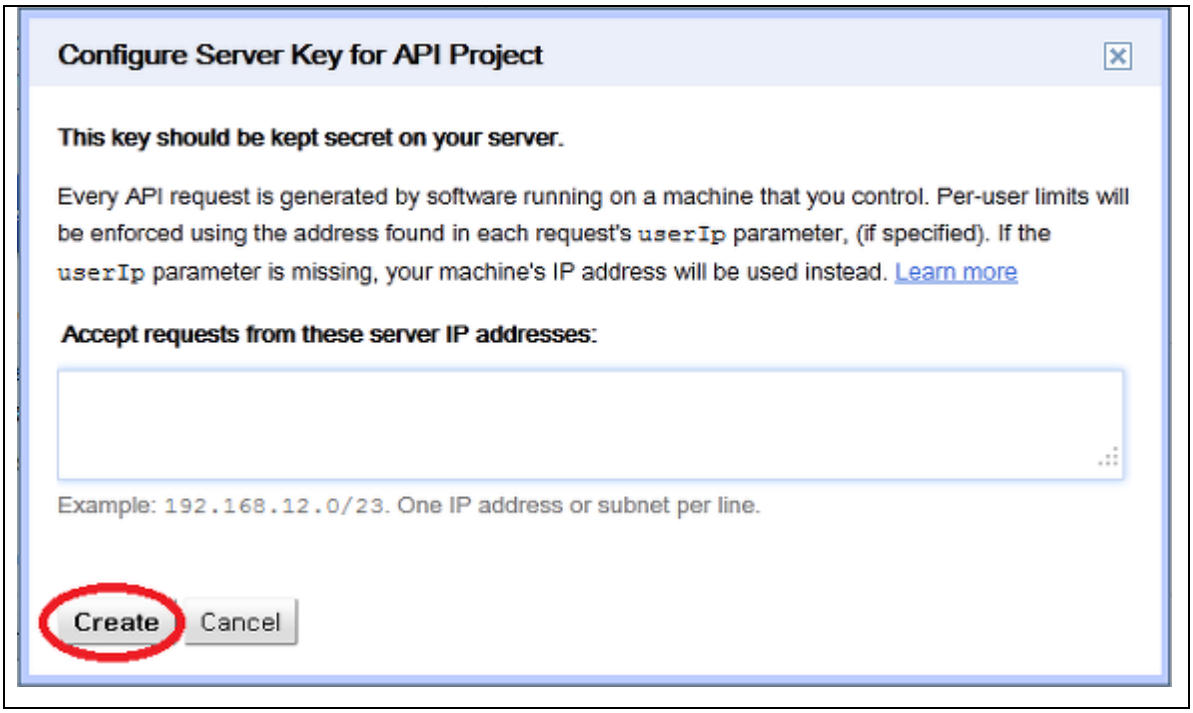
그리고 'Create new Server key' 버튼을 누른다.



9. 버튼을 누르면 조그만 창이 나온다. 그러면 create 버튼을 누른다.

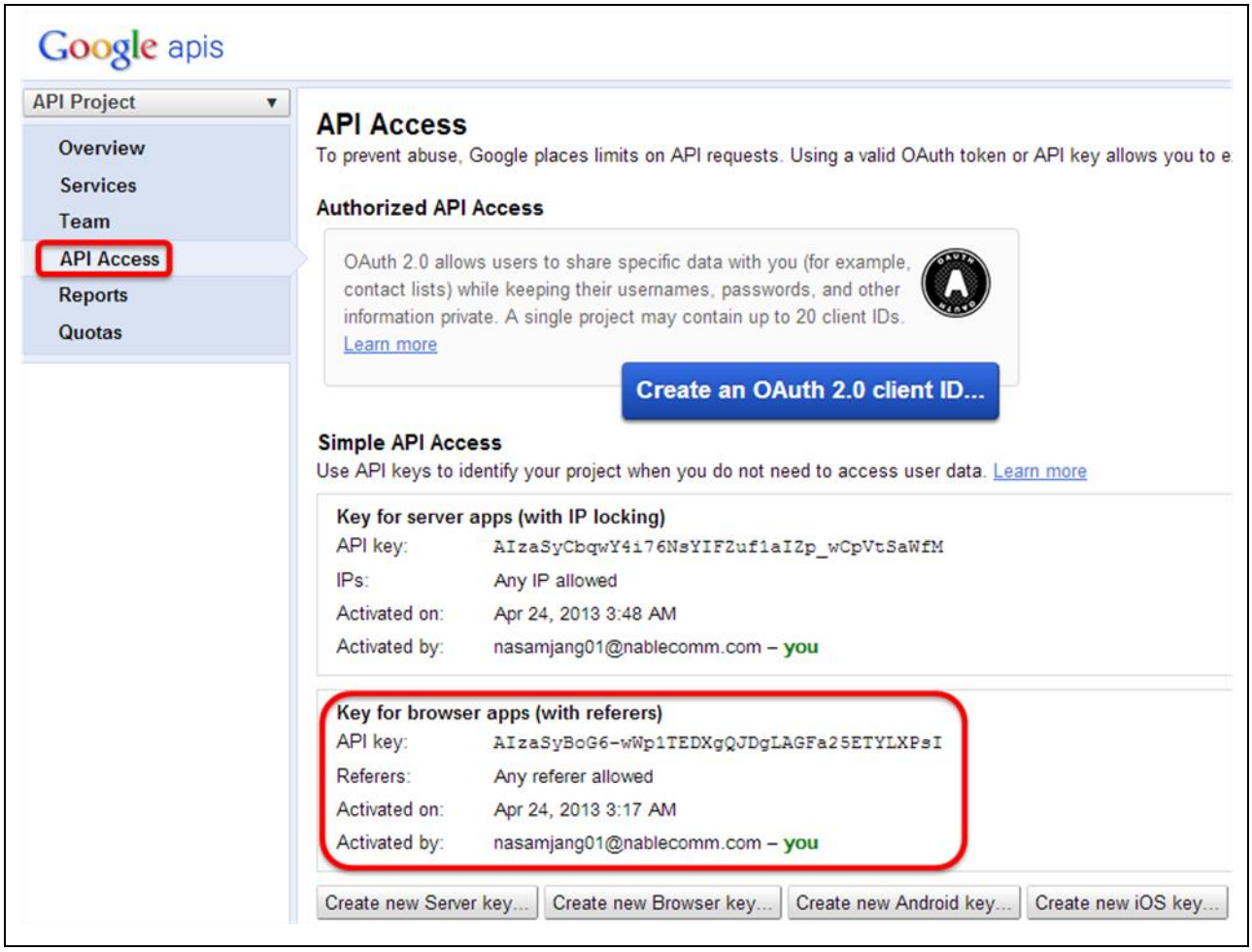
*만약 특정 서버만 허용을 할 것이면 버튼 위에 있는 'Accpet request from these server IP addresses:' 공간에 서버 IP 를 적으면 해당 서버만 허용한다.

*아무것도 적지 않으면 모든 ip 서버를 허용한다.



10. 서버 키가 생성되면 해당 키를 잘 적어놓는다. 서버에서 단말로 푸시를 보낼때 사용되는 값이다.

생성된 키는 언제든지 지우거나 새로 생성할 수 있다. 아래와 같은 화면이 나오면 서비스 신청은 끝났다.



결론: 신청 후 중요한 것(2 가지)

1. **PROJECT_NUMBER** (단말에서 GCM 등록을 위한 값)

- 프로젝트 ID 를 등록후 생성된 'Project Number'

2. **SERVER_KEY** (서버에서 GCM Server 로 푸시 요청을 보낼때 사용하는 값)

- API Access 에서 Create Server key 를 누르면 생성되는 'API key' 값

12.11 3rd-Party APP 구현하기

전제조건:

GCM 은 구글계정을 필요로 하기 때문에 API 8 (프로요 2.2)이상.

목표:

1. gcm.jar 를 이용하여 3rd-Party APP 프로젝트 작성.

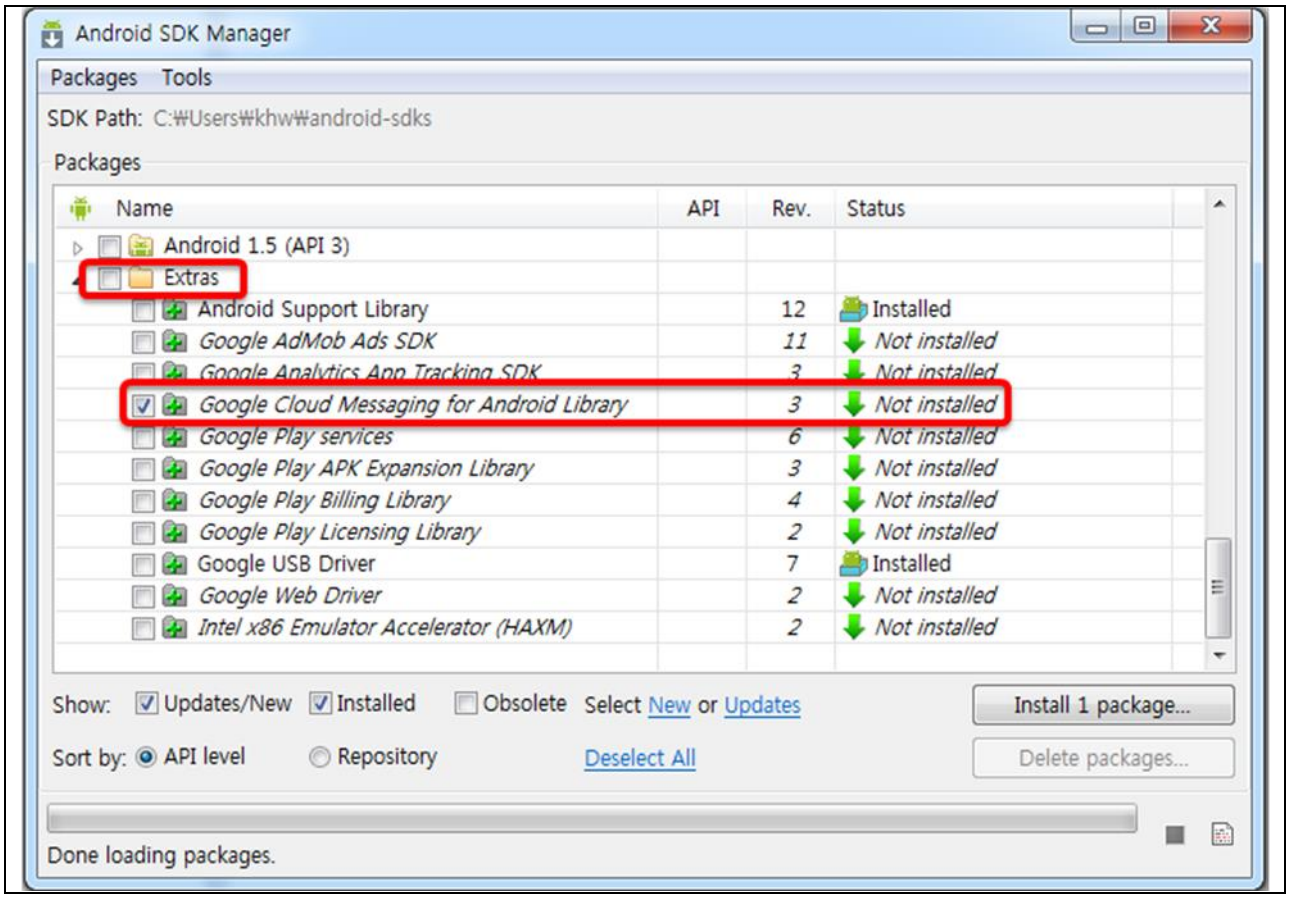
2. gcm-server.jar 를 이용하여 3rd-Party APP Server 프로젝트 생성.

3.3rd-Party APP Server 를 이용하여 3rd-Party APP 으로 Push 보내기.

4.3rd-Party APP 에서 수신된 Push 를 사용자에게 알리는 기능 구현.

12.11.1 GCM 라이브러리 다운로드 받기

SDK 매니저를 실행하여 Extras 의 Google Cloud Messaging for Android Library 를 다운 받는다.

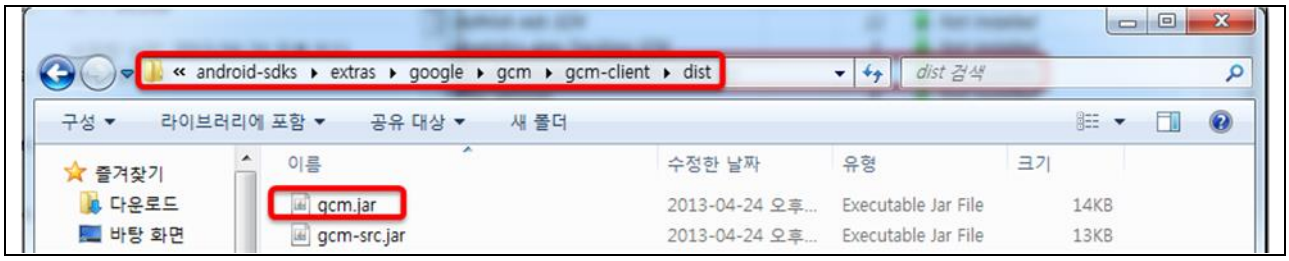


12.11.2 프로젝트에 GCM 라이브러리 추가하기

다운로드 받고 안드로이드 sdk 설치 폴더(기본은 C:₩android-sdk-window)에 들어가 보면

extras 폴더 -> google -> gcm -> gcm-client -> dist 에 2 개의 jar 파일이 있다.

그 중 gcm.jar 파일을 자신의 프로젝트에 추가.



12.11.3 AndroidManifest 에 GCM 사용 관련 permission 과 receiver, service 를 등록

1.GCM 관련 퍼미션 등록

```
<permission android:name="com.example.mainactivity.permission.C2D_MESSAGE"
android:protectionLevel="signature" />
<uses-permission android:name="com.example.mainactivity.permission.C2D_MESSAGE" />
```

2.GCM Receiver 퍼미션 등록

```
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
```

3.GCM 을 받으려면 구글 계정 퍼미션 등록

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
```

4. 메시지 받을 때 wake up 하기 위해 퍼미션 등록

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

5. 네트워크 접속 권한 퍼미션 등록

```
<uses-permission android:name="android.permission.INTERNET" />
```

12.11.4 GCM IntentService(Push 메시지 수신시 리스너) 구현하기

구현의 목적: GCMBroadcastReceiver 에서 전달 받은 푸시를 정해진 리스너로 결과를 받기 위해 구현.

아래와 같이 GCMIntentService 는 GCMBaseIntentService 를 상속받아야 한다.

```
public class GCMIntentService extends GCMBaseIntentService {

    /** 푸시로 받은 메시지 */
    @Override
    protected void onMessage(Context context, Intent intent) {
            String msg = intent.getStringExtra("msg");
            Log.d(tag, "onMessage. msg : "+msg);
    }

    /**에러 발생시*/
    @Override
    protected void on_error(Context context, String errorId) {
        Log.d(tag, "on_error. errorId : "+errorId);
    }

    /**단말에서 GCM 서비스 등록 했을 때 등록 id 를 받는다*/
    @Override
    protected void onRegistered(Context context, String regId) {
        Log.d(tag, "onRegistered. regId : "+regId);
    }

    /**단말에서 GCM 서비스 등록 해지를 하면 해지된 등록 id 를 받는다*/
    @Override
    protected void onUnregistered(Context context, String regId) {
        Log.d(tag, "onUnregistered. regId : "+regId);
    }
}
```

12.11.5 3rd-Party APP Server 에 서비스 등록 코드 구현하기

여기까지 소스코드를 구현 하면, 3rd-Party APP 쪽에서는 GCM 을 이용한 푸시를 받을 준비가 되었다.

```
public class MainActivity extends Activity {
	private final static String PROJECT_NUMBER = "Google APIs 에서 발급받은 Regi ID 를 입력";

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.activity_main);
		// 단말이 GCM 을 지원하는지 검사
		GCMRegistrar.checkDevice(this);
		// 응용프로그램의 AndroidManifest 가 제대로 구성되어 있는 지 확인.
		GCMRegistrar.checkManifest(this);

		// GCM 서비스의 응용프로그램에 대한 현재 등록 ID 를 가져옴.
		final String regId = GCMRegistrar.getRegistrationId(this);
		PrintLog("Registration id = "+regId);

		// 현재 GMC Regi 가 안되어 있다면, register 시작
		if (!GCMRegistrar.isRegistered(this)) {
			// 단말 등록 호출
			GCMRegistrar.register(this, PROJECT_NUMBER);
		} else {
			Log.e("reg_id = ", regId);
		}
}
```

3rd-Party APP 을 실행하면 GCM 서비스를 등록한 후 등록 id 가 출력 된다.

이 등록 id 는 2.7 절 3rd-Party APP Server 구현시 테스트에 필요하므로 기억해 둔다.

참고:

1. 등록 id 는 앱을 삭제 후 다시 설치하고 다시 gcm 서비스를 등록해도 같은 등록 id 가 나온다.

2. 실제 앱 개발시 등록 id 가 발급되면 Provider 서버에 등록 id 를 전송 해야 한다.

왜냐하면, 3rd-Party APP Server 에서는 등록 id 를 이용하여 푸시 데이터를 GCM Server 에 전송한다.

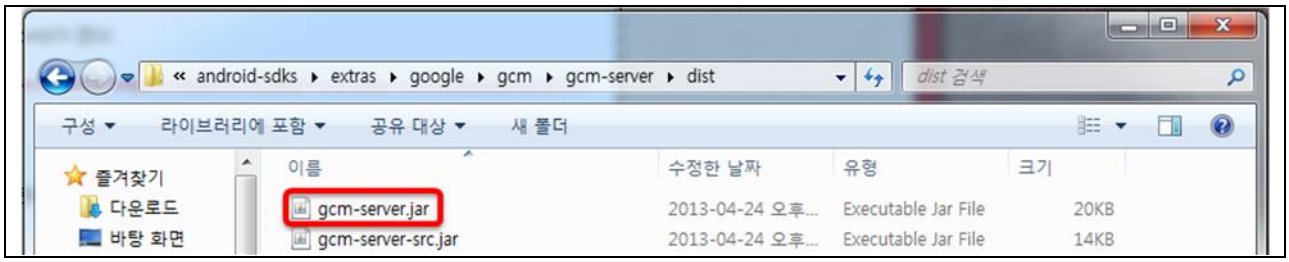
12.12 3rd-Party APP Server 구현하기

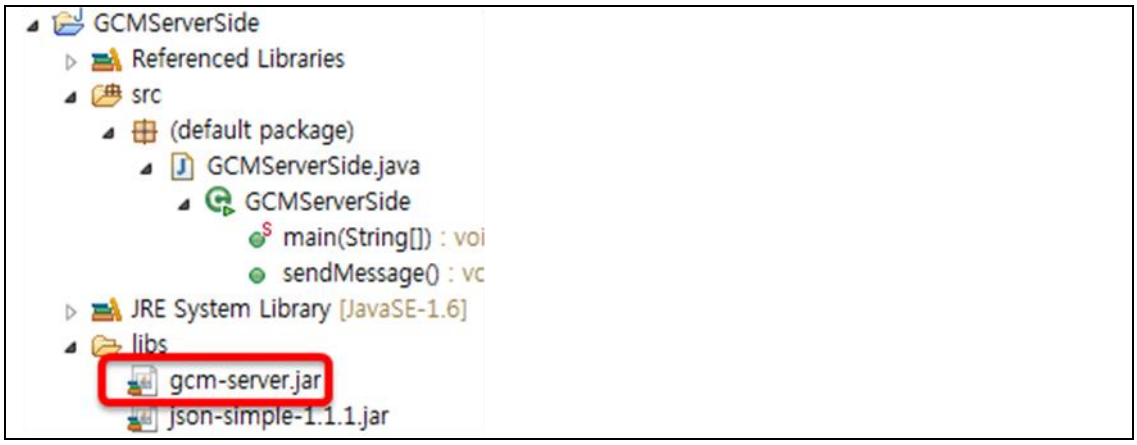
12.12.1 3rd-Party APP Server 프로젝트에 라이브러리 추가하기

안드로이드 sdk 설치 폴더(기본은 C:₩android-sdk-window)에 들어가 보면

extras 폴더 -> google -> gcm -> gcm-server -> dist 에 2 개의 jar 파일이 있다.

그 중 gcm-server.jar 파일을 자신의 프로젝트에 추가.





12.12.2 Java Project 를 이용하여, 3rd-Party APP Server 소스코드 구현하기

다운로드 받으면 안드로이드 sdk 설치 폴더(기본은 C:₩android-sdk-window)에 들어가 보면

extras 폴더 -> google -> gcm -> gcm-server -> dist 에 2 개의 jar 파일이 있다.

그 중 gcm-server.jar 파일을 자신의 프로젝트에 추가.

```
public class GCMServerSide {

public static void main(String[] args) throws Exception {

        String myApiKey = "Google APIs 의 API key 입력";
        String regId = "단말이 등록한 Regi ID 입력";
```

```
        // GOOGLE_API_KEY 를 이용해 Sender 초기화
        Sender sender = new Sender(myApiKey);

        // 메시지 이름을 "msg"로 해서 보냈으니 Client 의 onMessage()리스너 에서 수신 가능 하다.
        Message message = new Message.Builder().addData("msg", "push notify").build();

        List<String> list = new ArrayList<String>();
        list.add(regId); MulticastResult multiResult = sender.send(message, list, 5);

        if (multiResult != null) {
                List<Result> resultList = multiResult.getResults();
                for (Result result : resultList) {
                        System.out.println(result.toString());
                }
        }
}
```

12.13 결과 화면

12.13.1 3rd-Party APP 으로 GCM Server 에 Register

위 그림과 같이 GCM Register 버튼을 눌러 GCM Server 에 등록을 시작한다.

4. 아래 adb Log 를 보면 GCMIntentService 클래스의 onRegistered 리스너로 등록된 키값이 찍히는 것을 확인 할 수 있다.

```
protected void onRegistered(Context context, String reg_id)
{
        StartIntentActivity("Device Registered = ₩n"+reg_id);
}
```

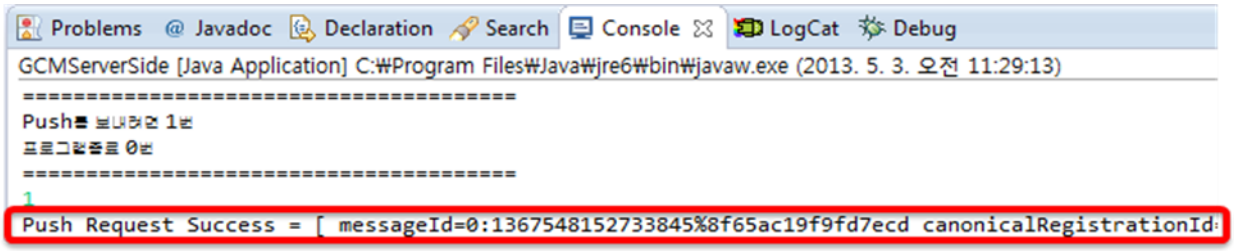
```
NableTester                 Registration id =
GCMRegistrar                resetting backoff for com.example.mainactivity
GCMRegistrar                Registering app com.example.mainactivity of senders 740648699856
NableTester                 Start GCM = 740648699856
GCMBroadcastReceiver        onReceive: com.google.android.c2dm.intent.REGISTRATION
GCMBroadcastReceiver        GCM IntentService class: com.example.mainactivity.GCMIntentService
GCMBaseIntentService        Acquiring wakelock
GCMBaseIntentService        Intent service name: GCMIntentService-DynamicSenderIds-1
GCMRegistrar                internal error: retry receiver class not set yet
GCMRegistrar                Registering receiver
GCMBaseIntentService        handleRegistration: registrationId = APA91bH_IaWyB-iu0hZ-P4CDO9diRlG8Qi4mIh1tP
                            UQ8bxuI0nYQUSrJDTe36km_7BVdWHE7q3OgTQPPJvhDXZ9ROHtZYBqjeMBaHY4VGm---qqf7z8PF_T
                            TC7RzdvDFIYvpork_sz2xPpaFKdd7cVGySkxbozjeZg, error = null, unregistered = null
GCMRegistrar                resetting backoff for com.example.mainactivity
GCMRegistrar                Saving regId on app version 1
NableTester                 Device Registered =
NableTester                 APA91bH_IaWyB-iu0hZ-P4CDO9diRlG8Qi4mIh1tPUQ8bxuI0nYQUSrJDTe36km_7BVdWHE7q3OgTQ
                            PPJvhDXZ9ROHtZYBqjeMBaHY4VGm---qqf7z8PF_TTC7RzdvDFIYvpork_sz2xPpaFKdd7cVGySkxb
                            ozjeZg
```

위는 'Project Number' 등록이 완료 된후 3rd-Party APP 에 찍히는 로그 이다.

12.13.2 3rd-Party APP Server 에서 3rd-Party APP 으로 Push 전송 예

5. 이제 3rd-Party APP 이 GCM Server 로 등록이 되었으니, 3rd-Party APP Server 를 이용하여 Push 를 보내 보도록 하겠다.

   아래 그림과 같이 Java Project 를 이용하여, 3rd-Party APP 으로 Push 를 보낸다.



   위의 빨간 박스와 같이 message Id 와 CanonicalRegistration Id 가 console 로그창에 찍히면 정상적으로 Push 전송이 완료된 것이다.

6. 아래 adb Log 를 보면 GCMIntentService 클래스의 onMessage 리스너로 올라온 Push Message 가 찍히는 것을 확인 할 수 있다.

```
protected void onMessage(Context context, Intent intent)
{
        String msg = intent.getStringExtra("msg");
        GenerateNotification(context, "GCM push");
        StartIntentActivity("You've got Message = "+msg);
}
```

```
GCMBroadcastReceiver        onReceive: com.google.android.c2dm.intent.RECEIVE
GCMBroadcastReceiver        GCM IntentService class: com.example.mainactivity.GCMIntentService
GCMBaseIntentService        Acquiring wakelock
GCMBaseIntentService        Intent service name: GCMIntentService-DynamicSenderIds-1
NableTester                 You've got Message = push notify
GCMBaseIntentService        Releasing wakelock
```

   아래 그림은 3rd-Party APP 에서 Push 를 받았을때의 화면이다.

12.13.3 3rd-Party APP 으로 GCM Server 에 DeRegister

GCM Server 에 Register 되어 있는 3rd-Party APP 을 아래와 같이 GCM DeRegister 버튼을 누르면 등록 해지 과정을 실행 한다.

아래 adb Log 를 보면 GCMIntentService 클래스의 onUnregistered 리스너로 등록 해지 로그를 확인 할 수 있다.

```
protected void onUnregistered(Context arg0, String arg1)
{
        StartIntentActivity("Device DeRegistered = ₩n"+arg1);
}
```

```
GCMRegistrar                resetting backoff for com.example.mainactivity
GCMRegistrar                Unregistering app com.example.mainactivity
NableTester                 Stop GCM
GCMBroadcastReceiver        onReceive: com.google.android.c2dm.intent.REGISTRATION
GCMBroadcastReceiver        GCM IntentService class: com.example.mainactivity.GCMIntentService
GCMBaseIntentService        Acquiring wakelock
GCMBaseIntentService        Intent service name: GCMIntentService-DynamicSenderIds-2
GCMRegistrar                internal error: retry receiver class not set yet
GCMRegistrar                Registering receiver
GCMBaseIntentService        handleRegistration: registrationId = null, error = null, unregistered = com.ex □
                            ample.mainactivity
GCMRegistrar                resetting backoff for com.example.mainactivity
GCMRegistrar                Saving regId on app version 1
NableTester                 Device DeRegistered =
NableTester                 APA91bH_IaWyB-iu0hZ-P4CDO9diRlG8Qi4mIh1tPUQ8bxuI0nYQUSrJDTe36km_7BVdWHE7q3OgTQ □
                            PPJvhDXZ9ROHtZYBqjeMBaHY4VGm---qqf7z8PF_TTC7RzdvDFIYvpork_sz2xPpaFKdd7cVGySkxb □
                            ozjeZg
GCMBaseIntentService        Releasing wakelock
```

위와 같이 등록 해지에 성공하면, 3rd-Party APP 으로 Notification 을 확인 할 수 있다.