

iPNS

# FCM Push Library 연동 가이드

버전	1.7.3
작성일자	2019.06.12
작성자	신동현

**Copyright © 2012 All Rights Reserved.**

사전 승인 없이 본 내용의 전부 또는 일부에 대한  
복사, 전재, 배포, 사용을 금합니다.

## Document History/Version Control

개정번호	개정장	개정내역	개정자	시행일자
1.0	전체	최초 작성	신동현	2015.06.18
1.1	3.5	Activity에서의 Data 수신 부분 수정	신동현	2015.06.19
1.2	4	iOS APNS Library 부분 추가	신동현	2015.06.22
1.4	3.6, 3.7, 4.6	Push설정값, Vibration 추가	신동현	2015.07.20
1.5	3.8	Sound 추가	신동현	2015.07.21
1.6	3.3	Android에서 Notification연동 부분 수정 (Receiver항목에 API추가함)	신동현	2015.08.04
1.6.1	4.2	iOS Project 설정 시 Linker설정부분 추가	신동현	2015.09.03
1.6.2	-	미사용 API 제거	신동현	2015.10.12
1.6.3	3.6, 4.6	Message 읽음 확인 기능 추가	신동현	2015.10.14
1.6.4	3.4	AppRegist에 PushYN파라미터 추가	신동현	2016.02.02
1.6.5	전체	FCM 추가	신동현	2018.01.01
1.6.6	3.8	Message History 규격 변경	신동현	2018.01.24
1.6.7	3.8	Message History 조회 시에 Title, RecvYn, ReadYn	신동현	2018.03.12
1.7.0	전체	FCM Library API 전체 변경	신동현	2018.08.10
1.7.1	3.5	E2E 관련 내용 추가	신동현	2019.01.17
1.7.2	일부	FCM Project 관련 내용 수정	신동현	2019.04.15
1.7.3	-	미세한 수정	신동현	2019.06.12

## 목차

1.연동 시스템 정의 .....	4
2.연동 인터페이스 정의 .....	4
3.ANDROID용 LIBRARY .....	4
3.1.FCM Push Library 구성 .....	4
3.2.Firebase Console 에서 Project 등록 .....	6
3.3.FCM Library 를 App Project 에 적용 .....	13
3.4.Permission추가 및 Receiver 등록 .....	14
3.5.App 등록 .....	16
3.6.Push 수신여부 등 설정 전달 및 수신 .....	19
3.7.Message 읽음 확인 보고 .....	24
3.8.Message History 조회 .....	27

## 1. 연동 시스템 정의

본 문서는 iPNS서비스를 제공하기 위해 iPNS FCM Library와 3rd Party App과의 기능 연동 시스템을 정의합니다.

\* 기준 FCM LIBRARY 버전 : 3.1 RELEASE

## 2. 연동 인터페이스 정의

Android 의 경우 Library(aar 파일) 파일이 제공되며, 이 라이브러리를 업무용 App혹은 3rd Party App에 import 하여 Library내의 API들을 호출하는 방식으로 동작됩니다.

\* FCM 은 GOOGLE CLOUD MESSAGE 를 뜻하며 이는 FIREBASE CLOUD MESSAGE 로 전환 되었습니다. 이에 서버와 클라이언트의 규격이 변경되었습니다.

## 3. Android용 Library

### 3.1. FCM Push Library 구성

FCM Push Library는 IpnsFcmLib-3.1-release-**<생성날짜>**.aar 파일로 이루어져 있습니다.

\* FCM PUSH LIBRARY 는 아래 ANDROID STUDIO 의 각종 기준 버전에 맞춰서 작성되었습니다. 따라서 모든 내용은 기준 버전에 대한 설명으로만 제공됩니다.(자세한 설정은 샘플 앱 PROJECT 설정을 참고하시면 됩니다.)

각종 기준 버전	버전 상세 내용
Android Studio 버전	Android Studio 3.3 Build #AI-182.5107.16.33.5199772, built on December 25, 2018 JRE: 1.8.0_152-release-1248-b01 x86_64 JVM: OpenJDK 64-Bit Server VM by JetBrains s.r.o macOS 10.14.3
Gradle 버전	3.3.0
buildToolVersion	28.0.3
compileSdkVersion	28
minSdkVersion	14

targetSdkVersion	28
com.google.gms:google-services 버전	4.2.0
com.google.firebase 버전	com.google.firebase:firebase-core:16.0.6 com.google.firebase:firebase-analytics:16.0.6 com.google.firebase:firebase-messaging:17.3.4
retrofit, okhttp 버전	com.squareup.retrofit2:retrofit:2.3.0 com.squareup.retrofit2:converter-gson:2.3.0 com.squareup.okhttp3:okhttp:3.10.0 com.squareup.okhttp3:logging-interceptor:3.8.0

제품 구성물	내용
IpnsFcmLib-3.0-release-<생성날짜>.aar	3rd party App과 FCM Push Library와 연동하기 위한 인터페이스 라이브러리.

추가 구성물	내용
PUSH_FCMLibSample	FCM Library사용 Sample Project

## 3.2. Firebase Console 에서 Project 등록

FCM을 이용하기 위해서는 Google Firebase Console 상에 Project 가 존재해야 합니다. 이 문서는 Firebase console 상에서 신규로 Project 생성하는 과정을 기준으로 기술되었습니다.

\* Google Firebase Cloud 에 접근하기 위해서는 Google 계정이 반드시 있어야 합니다. 생성 방법은 어렵지 않고 본 제품과 직접적인 연관이 없기 때문에 생략됩니다.

### 3.2.1. Firebase Console 에 접속

아래 사이트를 브라우저에서 열고, 새로 생성한 혹은 기 보유한 Google 계정으로 로그인 합니다.

<https://console.firebase.google.com>

\* 기존 GCM 을 사용하던 앱이 FCM 으로 마이그레이션을 하는 경우, 반드시 동일한 Google 계정으로 console 에 로그인 후, 동일한 Project 에서 설정을 구성해야 합니다.

### 3.2.2. 새 Project 를 생성

프로젝트 추가 버튼을 눌러서 새 프로젝트를 추가합니다.



## 프로젝트 추가

✕

프로젝트 이름

프로젝트 이름

▼

📱 + iOS + </>

도움말: 프로젝트는 여러 플랫폼의 여러 앱을 포괄합니다.

?

프로젝트 ID ?

my-awesome-project-id

국가/지역: ?

미국

▼

기본적으로 Analytics 데이터는 다른 Firebase 기능과 Google 제품을 개선합니다. 언제든지 설정에서 Analytics 데이터가 공유되는 방식을 관리할 수 있습니다. [자세히 알아보기](#)

취소

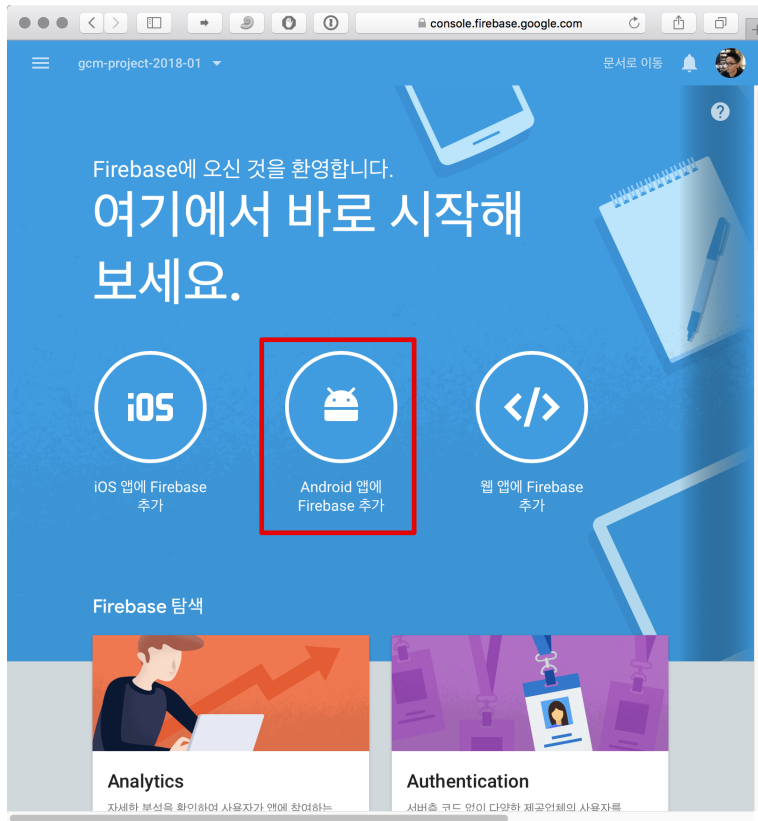
프로젝트 만들기

적당한 이름을 입력하고, 국가를 대한민국으로 지정한 후 프로젝트 만들기 버튼을 눌러서 프로젝트 생성을 마무리합니다.



### 3.2.3. Android App 등록

이제 FCM을 사용할 Android 앱을 등록합니다.



Android 앱에 Firebase 추가 버튼을 누릅니다.



FCM 라이브러리를 포함시켜서 실제로 Push Message 를 수신 할 앱의 Package Name 을 적고 앱 등록 버튼을 누릅니다.

여기까지 진행하면 실제로 필요한 작업은 거의 수행되었습니다.

## Android 앱에 Firebase 추가

1
2
3

앱 등록
구성 파일 다운로드
Firestore 추가

### Android Studio 안내

다른 방법: [Unity](#) [C++](#)

- 다운로드 google-services.json
- Android 스튜디오에서 프로젝트 보기로 전환하여 프로젝트 루트 디렉토리를 표시하세요.
- 방금 다운로드한 **google-services.json** 파일을 Android 앱 모듈 루트 디렉토리로 이동하세요.

google-services.json

이미 종속 항목을 추가했나요?  
[콘솔로 건너뛰기](#)

계속

위 화면의 가이드대로 google-services.json 파일을 앱의 app 디렉토리 아래로 복사하고 계속 버튼을 눌러줍니다.

Android 앱에 Firebase 추가

1

2

3

앱 등록

구성 파일 다운로드

3

Firestore SDK 추가

Firestore SDK 추가

Firestore SDK 추가

Gradle 안내

다른 방법: [Unity](#) [C++](#)

Gradle 용 Google 서비스 플러그인에서 방금 다운로드한 google-services.json 파일을 로드합니다. build.gradle 파일을 수정하여 플러그인을 사용하세요.

1. 프로젝트 수준 build.gradle(<project>/build.gradle):

```

buildscript {
    dependencies {
        // Add this line
        classpath 'com.google.gms:google-services:3.1.0'
    }
}

```

2. 앱 수준 build.gradle(<project>/<app-module>/build.gradle):

```

...
// Add to the bottom of the file
apply plugin: 'com.google.gms.google-services'

```

기본적으로 Analytics 포함 ⓘ

3. 마지막으로 IDE의 표시줄에 있는 '지금 동기화'를 누르세요.

Gradle files have changed since last sync

Sync now

완료

그리고 위의 설명과 동일하게 프로젝트 설정을 수정합니다.

단, 1번 항목은 최신 버전인 classpath 'com.google.gms:google-services:4.2.0' 으로 바꿔서 입력해야 합니다.

-11-

☰


gcm-project-2018-01 ▾

문서로 이동

개요

프로젝트의 앱 1개

다른 앱 추가



gcm-sample01

com.dkitech.gcmlibrary.sample01

⋮

일일 활성 사용자 수

0

월간 활성 사용자 수

0

비정상 종료가 발생하지 않은 사용자

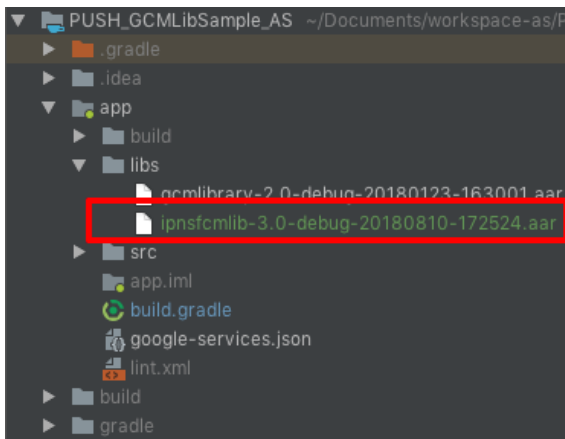
0%

비정상 종료

0

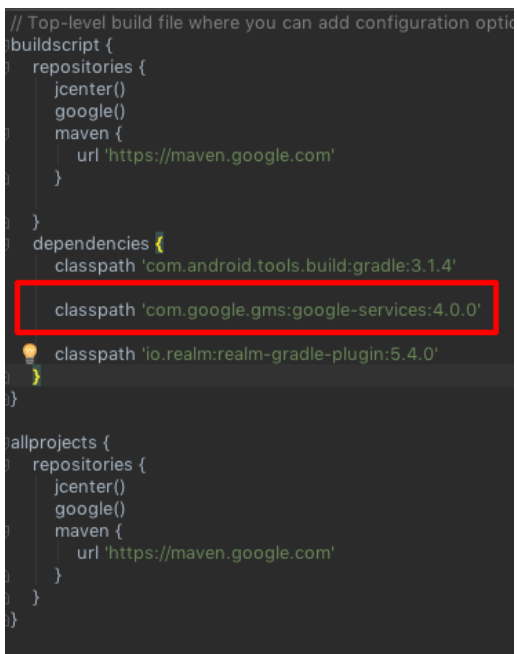
이제 위와 같이 프로젝트 생성이 완료되었습니다.

### 3.3. FCM Library 를 App Project 에 적용



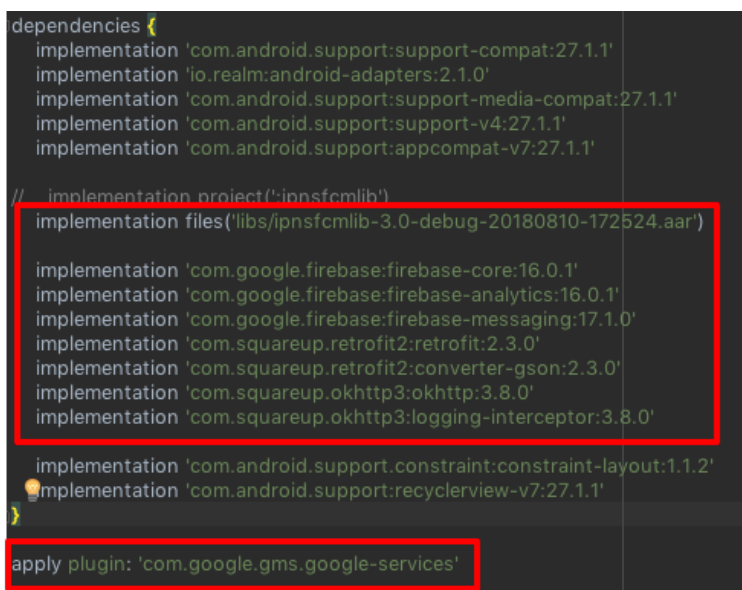
라이브러리 파일을 실제 프로젝트에 적용할 차례입니다. 좌측 그림과 같이 libs 디렉토리에 aar 파일을 추가합니다.

\* 샘플 프로젝트의 경우 aar 파일이 libs 디렉토리에 이미 추가되어 있습니다.



프로젝트 레벨의 build.gradle 에 좌측 붉은 색 박스안의 내용이 추가되어 있는지 확인하고 없으면 추가합니다.

(가이드를 순서대로 진행한 경우, 이미 이전 단계에서 추가되어 있어야 합니다.)



이제 app 레벨의 build.gradle 을 열어서 dependency 부분 중 좌측 그림의 붉은 색 항목을 추가해 줍니다.

\* 이 절차는 샘플 프로젝트에서 해당 항목을 복사해서 사용해도 됩니다.

### 3.4.Permission추가 및 Receiver 등록

라이브러리를 정상적으로 사용하기 위해서 필수 Permission 과 Broadcast Receiver 가 AndroidManifest.xml 에 선언되어야 합니다.

단, AAR 파일 내부에 자체 AndroidManifest.xml 을 포함하고 있고 Android Studio 에서는 AndroidManifest.xml 을 자동으로 Merge 하기 때문에 아래 항목만 추가하면됩니다.

아래 코드의 receiver 는 FCM 을 수신하기 위해서 반드시 선언 및 구현되어야 하는 Broadcast Receiver 입니다. 따라서 샘플 코드를 그대로 사용하거나 아래 예제에서 android:name 을 실제 FCM 수신할 receiver 의 class 이름으로 변경하면 됩니다.

(build 시에 AndroidManifest.xml 관련 error 나 conflict 발생 시에는 [mdm@dkitec.com](mailto:mdm@dkitec.com) 으로 문의 부탁드립니다.)

#### AndroidManifest.xml

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

<application
    ....>

<receiver
    android:name=".receiver.MyIpnsReceiver"
    android:enabled="true"
    android:exported="false"
    android:permission="{applicationId}.permission.SEND_TO_MYSELF">
    <intent-filter>
        <action android:name="com.dkitec.ipnsfcmlib.intent.RECEIVE" />
        <category android:name="{applicationId}" />
    </intent-filter>
</receiver>
</application>
```

아래에는 BroadcastReceiver 를 실제로 구현한 예제 소스 코드에 대한 설명입니다.

#### MyIpnsReceiver.java

```

public class MyIpnReceiver extends IpnReceiver {
    private final static String TAG = MyIpnReceiver.class.getSimpleName();

    @Override
    public void onNotiReceive(Context context, Intent intent) {
        IpnPublicMessage ipnPublicMessage = handleIntent(intent);
        if (ipnPublicMessage != null) {
            Log.i(TAG, Log.getMethod() + ", parcel : " + ipnPublicMessage.toString());
            Toaster.showToastOnUiThread(context, Log.getMethod() + " : " +
            ipnPublicMessage.getAlert(), Toast.LENGTH_SHORT);

            saveMessage(ipnPublicMessage);
        }
    }

    @Override
    public void onMsgReceive(Context context, Intent intent) {
        IpnPublicMessage ipnPublicMessage = handleIntent(intent);
        if (ipnPublicMessage != null) {
            Log.i(TAG, Log.getMethod() + ", parcel : " + ipnPublicMessage.toString());
            Toaster.showToastOnUiThread(context, Log.getMethod() + " : " +
            ipnPublicMessage.getAlert(), Toast.LENGTH_SHORT);

            saveMessage(ipnPublicMessage);
        }
    }

    private IpnPublicMessage handleIntent(Intent intent) {
        IpnPublicMessage ipnPublicMessage = null;
        if (intent != null) {
            ipnPublicMessage = intent.getParcelableExtra(IpnConstants.PARCEL);
        }
        return ipnPublicMessage;
    }
}

```

Push 메시지를 수신하기 위해서는 위와 같이 IpnReceiver 를 extend 한 BroadcastReceiver 를 구현해야 합니다. 이 때 onNotiReceive 와 onMsgReceive 2가지 API 를 의무적으로 구현해야 합니다.

onNotiReceive 를 통해 전달되는 메시지는 iPNS 서버 상에서 NOTI\_YN 값이 'Y' 인 상태로 발송된 메시지이며 그 이외의 메시지는 모두 onMsgReceive 로 전달 됩니다.

위 코드에서 handleIntent API 를 보시면 손쉽게 Intent 에서 IpnPublicMessage class 로 메시지 데이터를 가져올 수 있습니다. IpnPublicMessage 는 라이브러리 내부에 포함된 class 로써 수신된 메시지를 각 멤버변수에 저장할 수 있습니다.

\* 단, FCM 메시지의 수신은 아래 App 등록 절차가 이루어지고 난 다음에야 가능합니다.

### 3.5. App 등록

3rd party 앱이 FCM 및 Push Library의 기능을 정상적으로 이용하기 위해서는 App 등록 절차가 이루어져야 합니다.

App 등록은 아래 순서로 이루어 집니다.

1. FCM 등록 요청
2. FCM 등록 완료
3. iPNS 서버에 FCM 토큰과 단말, 사용자 정보(옵션)를 등록

위 절차는 아래 API 를 호출하면 내부적으로 이루어 집니다.

#### 3.5.1. requestRegistration 호출

requestRegistration 을 호출하면 IpnsErrorCode 중 하나의 결과 값이 리턴되며, 이 값이 성공인 경우 IpnsAppRegistrationResultListener 로 전달됩니다.

\* 정상적으로 등록완료 되는 경우 Library 는 serverUrl, appId, userId 값을 저장합니다.

#### - Parameter

Parameter	필수 여부	설명
context	Y	API 를 호출하는 context (일반적으로 Activity)
listener	Y	IpnsAppRegistrationResultListener 의 onRegistrationResult 로 등록 결과가 리턴됨
serverUrl	Y	iPNS 서버 주소 ( <a href="http://[아이피 혹은 도메인]:[포트]">http://[아이피 혹은 도메인]:[포트]</a> ) ex) <a href="http://push.server.com:8080">http://push.server.com:8080</a>
appId	Y	iPNS 서버 관리자 페이지에 등록된 app id
userId	N	사용자의 id. 만약, 입력 값이 없을 경우 device id 만으로 앱 등록이 진행됨.
pushYn	N	앱 등록 시에 메타 정보만 전달하고, Push 수신은 하지 않도록 세팅 하는 케이스에 사용됨. null 입력 시 Default 값인 Y 로 세팅됨. Y : Push 수신 허용(Default) N : Push 수신 차단
e2eParams	N	E2E 암호화 사용하는 경우 serviceID, 미사용 시 null 입력

#### - Return Value

전체 ErrorCode 표 이며 셀 색상이 마킹되어 있는 것만 이 API 의 리턴값으로 전달 가능합니다. 초록색만 정상이며 나머지는 오류입니다.



ErrorCode	Member Variable Name	설명
1400	OK	정상 호출(앱 등록이 완료된 것은 아닙니다.)
1401	CONTEXT_NOT_VALID	context 파라미터가 유효하지 않음
1402	SERVER_ADDRESS_NOT_VA	iPNS 서버 주소가 유효하지 않음
1403	APP_ID_NOT_VALID	app id 가 유효하지 않음
1404	DEVICE_ID_NOT_VALID	단말의 DeviceID(Android ID) 가 유효하지
1408	REQUEST_ID_MISSING	request id 가 존재하지 않습니다.
1409	RESULT_DATA_MISSING	result data 가 존재하지 않습니다.
1410	SENDING_MSG_NULL	전송 메시지가 null 입니다.
1411	LISTENER_IS_NULL	listener 가 null 입니다.
1412	PARAMETER_INVALID	Parameter 가 잘못되었습니다.
1413	APP_REGISTRATION_REQUI	앱 등록이 필요합니다.
1499	UNKNOWN_ERROR	알 수 없는 오류가 발생하였습니다.
1501	GOOGLE_PLAY_SERVICE_N	Google Play Service 상태가 유효하지 않습

코드 사용 예는 아래와 같습니다.

#### App Registration

```
int result = IpnsFcmLib.getInstance().requestRegistration(this, this, svrUrl, APP_ID, userCn, null);
if (result == IpnsErrorCode.OK) {
    // 정상 호출
}
```

위와 같이 정상 호출된 경우 결과 값은 아래처럼 리스너의 onRegistrationResult 로 전달됩니다.

#### App Registration Result

```
@Override
public void onRegistrationResult(IpnsGeneralResponse response){
    Log.d(TAG, Log.getMethod() + ", response : " + response.toString() + ", rt : " +
response.getRT() + ", rtMsg : " + response.getRT_MSG());
    if (IpnsConstants.PUBLIC_RT_OK.equals(response.getRT())) {
        // 정상 앱 등록 완료
    }
}
```

리스너의 파라미터인 response 의 구조는 아래와 같습니다. 이 값들은 response 의 getter 함수로 가져올 수 있습니다.

항목	설명
RT	결과 코드
RT_MSG	결과 메시지

전달 가능한 RT 값들은 아래와 같습니다.

RT	Member Variable Name	설명
0000	PUBLIC_RT_OK	성공
1501	PUBLIC_RT_FCM_REG_CANCELLED	fcm 등록이 취소됨
1599	PUBLIC_RT_FCM_REG_FAILED_UNKNOWN	fcm 등록이 알 수 없는 이유로 실패됨
9000	PUBLIC_RT_REQUEST_FAILED_UNKNOWN	알 수 없는 이유로 요청이 실패됨
9999	PUBLIC_RT_NOT_OK	실패(서버로 부터 실패가 전달된 경우 등)

### 3.6. Push 수신여부 등 설정 전달 및 수신

iPNS FCM Library 통해서 App 등록을 하는 경우 iPNS 서버상에 해당 단말(혹은 사용자 아이디)의 Push 수신 여부 관련 설정값이 자동으로 저장됩니다.

설정 가능한 값들은 Push 수신여부, 수신 가능 시작,종료 시간 그리고 userId 입니다.

\* 설정 값의 자세한 설명은 아래 requestSetConfig 의 Parameter 를 참조하세요.

아래 내용은 이 설정 값을 전달/조회하는 API 에 대한 설명입니다.

#### 3.6.1. requestSetConfig 호출(설정 전송)

App 등록 이후 SetPushConfig를 호출할 수 있습니다. 성공할 경우 파라미터로 전달한 값들이 서버에 저장됩니다.

Config 설정 값은 서버 상에서 deviceId 기준으로 저장됩니다.

##### - Parameter

Parameter	필수 여부	설명
context	Y	API 를 호출하는 context (일반적으로 Activity)
listener	Y	IpnsSetConfigResultListener 의 instance, 이 리스너의 onSetConfigResult 로 결과가 리턴됨
userId	N	사용자의 id 값(변경을 원할 때 사용됨)
pushYn	N	앱 등록 시에 정보만 올려놓고 Push 수신은 하지 않는 등의 경우에 사용됨 Y : Push 수신 허용 N : Push 수신 차단
startTime	N	Push 수신 가능 시작시간, default : 0000 (00시 00분) * startTime, endTime 은 pushYn 이 Y 인 경우에만 유효함
endTime	N	Push 수신 가능 종료시간, default : 2359 (23시 59분) * startTime, endTime 은 pushYn 이 Y 인 경우에만 유효함

##### - Return

전체 ErrorCode 표 이며 셀 색상이 마킹되어 있는 것만 이 API 의 리턴값으로 전달 가능합니다. 초록색만 정상이며 나머지는 오류입니다.

ErrorCode	Member Variable Name	설명
1400	OK	정상 호출(앱 등록이 완료된 것은 아닙니다.)
1401	CONTEXT_NOT_VALID	context 파라미터가 유효하지 않음
1402	SERVER_ADDRESS_NOT_VALID	iPNS 서버 주소가 유효하지 않음

1403	APP_ID_NOT_VALID	app id 가 유효하지 않음
1404	DEVICE_ID_NOT_VALID	단말의 DeviceID(Android ID) 가 유효하지 않습니다.
1408	REQUEST_ID_MISSING	request id 가 존재하지 않습니다.
1409	RESULT_DATA_MISSING	result data 가 존재하지 않습니다.
1410	SENDING_MSG_NULL	전송 메시지가 null 입니다.
1411	LISTENER_IS_NULL	listener 가 null 입니다.
1412	PARAMETER_INVALID	Parameter 가 잘못되었습니다.
1413	APP_REGISTRATION_REQUIRED	앱 등록이 필요합니다.
1499	UNKNOWN_ERROR	알 수 없는 오류가 발생하였습니다.
1501	GOOGLE_PLAY_SERVICE_NOT_AVAILABLE	Google Play Service 상태가 유효하지 않습니다.

코드 사용 예는 아래와 같습니다.

#### SetPushConfig

```
ret = IpnsFcmLib.getInstance().requestSetConfig(this, this, userCnStr,
isChecked?"Y":"N", startTime, endTime);
if (ret == IpnsErrorCode.OK) {
    // 정상 호출
    Log.v(TAG, Log.getMethod() + ", requestSetConfig successful");
}
```

위와 같이 정상 호출된 경우 결과 값은 아래처럼 리스너의 onSetConfigResult 로 전달됩니다.

#### SetPushConfig Result

```
@Override
public void onSetConfigResult(IpnsGeneralResponse response) {
    Log.v(TAG, Log.getMethod() + ", response : " + response.toString());
    if (IpnsConstants.PUBLIC_RT_OK.equals(response.getRT())) {
        // 성공
    }
}
```

리스너의 파라미터인 response 의 구조는 아래와 같습니다. 이 값들은 response 의 getter 함수로 가져올 수 있습니다.

항목	설명
RT	결과 코드
RT_MSG	결과 메시지

전달 가능한 RT 값들은 아래와 같습니다.

RT	Member Variable Name	설명
0000	PUBLIC_RT_OK	성공
1501	PUBLIC_RT_FCM_REG_CAN CELED	fcm 등록이 취소됨
1599	PUBLIC_RT_FCM_REG_FAIL _UNKNOWN	fcm 등록이 알 수 없는 이유로 실패됨
9000	PUBLIC_RT_REQUEST_FAIL ED_UNKNOWN	알 수 없는 이유로 요청이 실패됨
9999	PUBLIC_RT_NOT_OK	실패(서버로 부터 실패가 전달된 경우 등)

### 3.6.2. requestGetConfig 호출(설정 수신)

App 등록 이후 requestGetConfig 를 호출해서 서버에 저장되어 있는 해당 단말의 설정값을 가져올 수 있습니다.

가져올 수 있는 설정값은 pushYn(수신여부), startTime(시작시간), endTime(종료시간) 3가지 입니다.

#### - Parameter

Parameter	필수 여부	설명
context	Y	API 를 호출하는 context (일반적으로 Activity)
listener	Y	IpnsGetConfigResultListener 의 instance, 이 리스너의 onGetConfigResult 로 결과가 리턴됨

#### - Return

전체 ErrorCode 표 이며 셀 색상이 마킹되어 있는 것만 이 API 의 리턴값으로 전달 가능합니다. 초록색만 정상이며 나머지는 오류입니다.

ErrorCode	Member Variable Name	설명
-----------	----------------------	----

1400	OK	정상 호출(앱 등록이 완료된 것은 아닙니다.)
1401	CONTEXT_NOT_VALID	context 파라미터가 유효하지 않음
1402	SERVER_ADDRESS_NOT_VALID	iPNS 서버 주소가 유효하지 않음
1403	APP_ID_NOT_VALID	app id 가 유효하지 않음
1404	DEVICE_ID_NOT_VALID	단말의 DeviceID(Android ID) 가 유효하지 않습니다.
1408	REQUEST_ID_MISSING	request id 가 존재하지 않습니다.
1409	RESULT_DATA_MISSING	result data 가 존재하지 않습니다.
1410	SENDING_MSG_NULL	전송 메시지가 null 입니다.
1411	LISTENER_IS_NULL	listener 가 null 입니다.
1412	PARAMETER_INVALID	Parameter 가 잘못되었습니다.
1413	APP_REGISTRATION_REQUIRED	앱 등록이 필요합니다.
1499	UNKNOWN_ERROR	알 수 없는 오류가 발생하였습니다.
1501	GOOGLE_PLAY_SERVICE_NOT_AVAILABLE	Google Play Service 상태가 유효하지 않습니다.

코드 사용 예는 아래와 같습니다.

#### SetPushConfig

```
int ret = IpnsFcmLib.getInstance().requestGetConfig(this, this);
if (ret == IpnsErrorCode.OK) {
    // 정상 호출
    Log.v(TAG, Log.getMethod() + ", calling requestGetConfig successfully.");
}
```

위와 같이 정상 호출된 경우 결과 값은 아래처럼 리스너의 onGetConfigResult 로 전달됩니다.

#### SetPushConfig Result

```

@Override
public void onGetConfigResult(IpnsGeneralResponse response) {
    Log.v(TAG, Log.getMethod() + ", response : " + response.toString());
    if (IpnsConstants.PUBLIC_RT_OK.equals(response.getRT())) {
        // 성공
        String startTime = response.getPUSH_STIME();
        String endTime = response.getPUSH_ETIME();
        boolean isPushAllowed = response.getPUSH_YN() != null &&
        "Y".equals(response.getPUSH_YN());
    }
}

```

리스너의 파라미터인 response 의 구조는 아래와 같습니다. 이 값들은 response 의 getter 함수로 가져올 수 있습니다.

항목	설명
RT	결과 코드
RT_MSG	결과 메시지
PUSH_YN	Push 수신 여부
PUSH_STIME	Push 수신 시작시간
PUSH_ETIME	Push 수신 종료시간

전달 가능한 RT 값들은 아래와 같습니다.

RT	Member Variable Name	설명
0000	PUBLIC_RT_OK	성공
1501	PUBLIC_RT_FCM_REG_CANCELED	fcm 등록이 취소됨
1599	PUBLIC_RT_FCM_REG_FAILED_UNKNOWN	fcm 등록이 알 수 없는 이유로 실패됨
9000	PUBLIC_RT_REQUEST_FAILED_UNKNOWN	알 수 없는 이유로 요청이 실패됨
9999	PUBLIC_RT_NOT_OK	실패(서버로 부터 실패가 전달된 경우 등)

### 3.7.Message 읽음 확인 보고

Library API를 통하여 특정 Message 에 대한 읽음 확인 보고를 Server로 전달 가능합니다.

UI 를 통해서 사용자가 단말로 전달된 특정 Message 를 확인 하는 경우, 이API를 내부적으로 Call하면 사용자의 Push 확인 여부를 Push Server로 전달하고 정상적으로 도달된 경우 이 내역을 관리 서버에서 확인할 수 있습니다.

#### 3.7.1. requestReadReport 호출

##### - Parameter

Parameter	필수 여부	설명
context	Y	API 를 호출하는 context (일반적으로 Activity)
listener	Y	IpnsReadReportResultListener 의 instance, 이 리스너의 onReadReportResult 로 결과가 리턴됨
messageId	Y	읽음 확인 보고를 보낼 메시지의 id 값

##### - Return

전체 ErrorCode 표 이며 셀 색상이 마킹되어 있는 것만 이 API 의 리턴값으로 전달 가능합니다. 초록색만 정상이며 나머지는 오류입니다.

ErrorCode	Member Variable Name	설명
1400	OK	정상 호출(앱 등록이 완료된 것은 아닙니다.)
1401	CONTEXT_NOT_VALID	context 파라미터가 유효하지 않음
1402	SERVER_ADDRESS_NOT_VALID	iPNS 서버 주소가 유효하지 않음
1403	APP_ID_NOT_VALID	app id 가 유효하지 않음
1404	DEVICE_ID_NOT_VALID	단말의 DeviceID(Android ID) 가 유효하지 않습니다.
1408	REQUEST_ID_MISSING	request id 가 존재하지 않습니다.
1409	RESULT_DATA_MISSING	result data 가 존재하지 않습니다.
1410	SENDING_MSG_NULL	전송 메시지가 null 입니다.
1411	LISTENER_IS_NULL	listener 가 null 입니다.
1412	PARAMETER_INVALID	Parameter 가 잘못되었습니다.



1413	APP_REGISTRATION_REQUIRED	앱 등록이 필요합니다.
1499	UNKNOWN_ERROR	알 수 없는 오류가 발생하였습니다.
1501	GOOGLE_PLAY_SERVICE_NOT_AVAILABLE	Google Play Service 상태가 유효하지 않습니다.

코드 사용 예는 아래와 같습니다.

(아래는 파라미터에 IpnsReadReportResultListener 를 직접 구현한 예입니다.)

callback 인 onReadReportResult 로 결과 값이 전달됩니다.

#### SetReadReport 와 그 결과

```
int ret = IpnsFcmLib.getInstance().requestReadReport(context, new
IpnsReadReportResultListener() {
    @Override
    public void onReadReportResult(IpnsGeneralResponse response) {
        Log.d(TAG, Log.getMethod() + " , response : " + response.toString());
    }
}, reqId);
if (IpnsErrorCode.OK == ret) {
    Log.d(TAG, "requestReadReport OK");
}
```

리스너의 파라미터인 response 의 구조는 아래와 같습니다. 이 값들은 response 의 getter 함수로 가져올 수 있습니다.

항목	설명
RT	결과 코드
RT_MSG	결과 메시지

전달 가능한 RT 값들은 아래와 같습니다.

RT	Member Variable Name	설명
0000	PUBLIC_RT_OK	성공
1501	PUBLIC_RT_FCM_REG_CANCELLED	fcm 등록이 취소됨
1599	PUBLIC_RT_FCM_REG_FAIL_UNKNOWN	fcm 등록이 알 수 없는 이유로 실패됨

9000	PUBLIC_RT_REQUEST_FAILED_UNKNOWN	알 수 없는 이유로 요청이 실패됨
9999	PUBLIC_RT_NOT_OK	실패(서버로 부터 실패가 전달된 경우 등)

### 3.8. Message History 조회

Library API를 통하여 특정 기준에 전달된 Message 들을 Server 로 부터 조회가 가능합니다.

UI 를 통해서 사용자가 단말로 전달된 특정 Message 를 확인 하는 경우, 이API를 내부적으로 Call하면 사용자의 Push 확인 여부를 Push Server로 전달하고 정상적으로 도달된 경우 이 내역을 관리서버에서 확인할 수 있습니다.

#### 3.8.1. requestGetHistoryList 호출

##### - Parameter

Parameter	필수 여부	설명
context	Y	API 를 호출하는 context (일반적으로 Activity)
listener	Y	IpnsMessageHistoryListResultListener 의 instance, 이 리스너의 onMessageHistoryListResult 로 결과가 리턴됨
pushIdType	Y	IpnsPushIdType(ENUM), 다음 파라미터의 타입을 결정 USER_ID("01"), DEVICE_ID("02") 중 하나의 값을 가짐.
pushId	Y	위 파라미터에 따라 userId/deviceId 중 하나의 값
count	Y	조회 건 수
requestId	Y	요청의 기준이 되는 messageId 혹은 requestId
requestType	Y	IpnsRequestType(ENUM), 요청의 종류를 의미 NEW("N") : requestId 와 관계 없이 가장 최신 건을 가져옴 OLD("O") : requestId 로부터 count 만큼의 메시지를 조회

##### - Return

전체 ErrorCode 표 이며 셀 색상이 마킹되어 있는 것만 이 API 의 리턴값으로 전달 가능합니다. 초록색만 정상이며 나머지는 오류입니다.

ErrorCode	Member Variable Name	설명
1400	OK	정상 호출(앱 등록이 완료된 것은 아닙니다.)
1401	CONTEXT_NOT_VALID	context 파라미터가 유효하지 않음
1402	SERVER_ADDRESS_NOT_VALID	iPNS 서버 주소가 유효하지 않음
1403	APP_ID_NOT_VALID	app id 가 유효하지 않음
1404	DEVICE_ID_NOT_VALID	단말의 DeviceID(Android ID) 가 유효하지 않습니다.

1408	REQUEST_ID_MISSING	request id 가 존재하지 않습니다.
1409	RESULT_DATA_MISSING	result data 가 존재하지 않습니다.
1410	SENDING_MSG_NULL	전송 메시지가 null 입니다.
1411	LISTENER_IS_NULL	listener 가 null 입니다.
1412	PARAMETER_INVALID	Parameter 가 잘못되었습니다.
1413	APP_REGISTRATION_REQUIRED	앱 등록이 필요합니다.
1499	UNKNOWN_ERROR	알 수 없는 오류가 발생하였습니다.
1501	GOOGLE_PLAY_SERVICE_NOT_AVAILABLE	Google Play Service 상태가 유효하지 않습니다.

코드 사용 예는 아래와 같습니다.

#### GetPushHistoryList

```
int ret = IpnsFcmLib.getInstance().requestGetHistoryList(this, this,
IpnsPushIdType.USER_ID, !TextUtils.isEmpty(userId)?userId:"", (!
TextUtils.isEmpty(length) && TextUtils.isDigitsOnly(length))?Integer.valueOf(length):0,
reqId, reqType);
if (IpnsErrorCode.OK == ret) {
    Log.v(TAG, Log.getMethod() + ", requestGetHistory successful");
}
```

#### GetPushHistoryList Result

```

@Override
public void onMessageHistoryListResult(IpnsGeneralResponse response) {
    Log.i(TAG, Log.getMethod() + ", response : " + response.toString());
    if (IpnsConstants.PUBLIC_RT_OK.equals(response.getRT())) {
        List<IpnsHistoryMessage> list = response.getPUSH_LIST();
        if (list != null && list.size() > 0) {
            if (realm == null)
                realm = Realm.getDefaultInstance();
            realm.beginTransaction();
            for (IpnsHistoryMessage item : list) {
                Log.v(TAG, "item : " + item.toString());
                PushMessageObject pushMessageObject = new PushMessageObject();
                pushMessageObject.setRequestID(item.getREQUEST_ID());
                pushMessageObject.setAlert(item.getALERT());
                pushMessageObject.setTitle(item.getTITLE());
                pushMessageObject.setMessage(item.getMESSAGE());
                realm.insertOrUpdate(pushMessageObject);
            }
            realm.commitTransaction();
        }
    }
}

```

위의 예제는 리스너의 onMessageHistoryListResult 의 파라미터인 response 로 전달받은 history message 들을 PushMessageObject 타입의 값으로 변환해서 DB 에 저장하는 코드 입니다.

리스너의 파라미터인 response 의 구조는 아래와 같습니다. 이 값들은 response 의 getter 함수로 가져올 수 있습니다.

항목	설명
RT	결과 코드
RT_MSG	결과 메시지
COUNT	아래 파라미터인 PUSH_LIST 의 갯수
PUSH_LIST	History Message 들의 리스트

PUSH\_LIST 의 각 항목들은 아래와 같습니다.

항목	설명
TITLE	제목
ALERT	alert 메시지

MESSAGE	메시지 바디
PUSH_ID	userId 값
REQUEST_ID	메시지 아이디 혹은 request id
READ_YN	읽음 보고 여부
RECV_YN	메시지 도착 보고 여부
REQUEST_DATE	요청 시간
END_DATE	요청 완료 시간

전달 가능한 RT 값들은 아래와 같습니다.

RT	Member Variable Name	설명
0000	PUBLIC_RT_OK	성공
1501	PUBLIC_RT_FCM_REG_CANCELLED	fcm 등록이 취소됨
1599	PUBLIC_RT_FCM_REG_FAIL_UNKNOWN	fcm 등록이 알 수 없는 이유로 실패됨
9000	PUBLIC_RT_REQUEST_FAILED_UNKNOWN	알 수 없는 이유로 요청이 실패됨
9999	PUBLIC_RT_NOT_OK	실패(서버로 부터 실패가 전달된 경우 등)