iPNS

APNS Push Library 연동 가이드

버전	1.7.3
작성일자	2019.11.21
작성자	신동현

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		AppRegist에 PushYN파라미터 추가	신동현	2016.02.02
1.6.5	전체	FCM 추가	신동현	2018.01.10
1.6.6	3.8, 4.8	Message History 가져오는 기능 추가	신동현	2018.01.24
1.6.7	3.8, 4.8	Message History 조회시에 Title, RecvYn, ReadYn	신동현	2018.03.12
1.7.3	전체	Android 와 iOS 분리	신동현	2019.11.21

목차

1. 연동 시스템 정의 4
2. 연동 인터페이스 정의 4
3. IOS용 LIBRARY 4
3.1. iPNS Push Library 구성 4
3.2. Library추가 및 설정 4
3.3. Library 초기화 및 Callback구현 6
3.4. APNS 등록 및 App등록 7
3.5. Push Message수신시 처리 추가 9
3.6. Push Message 읽음 확인 보고 10
3.7. Push 수신여부 등 설정 전달 및 수신 11
3.8. Push Message History 조회 13

1. 연동 시스템 정의

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

2. 연동 인터페이스 정의

iOS 의 경우 Library(.a 파일) 파일이 제공되며, 이 라이브러리를 업무용 App 혹은 3rd Party App에 import 하여 Library 내의 API 들을 호출하는 방식으로 동작됩니다.

3. iOS용 Library

3.1. iPNS Push Library 구성

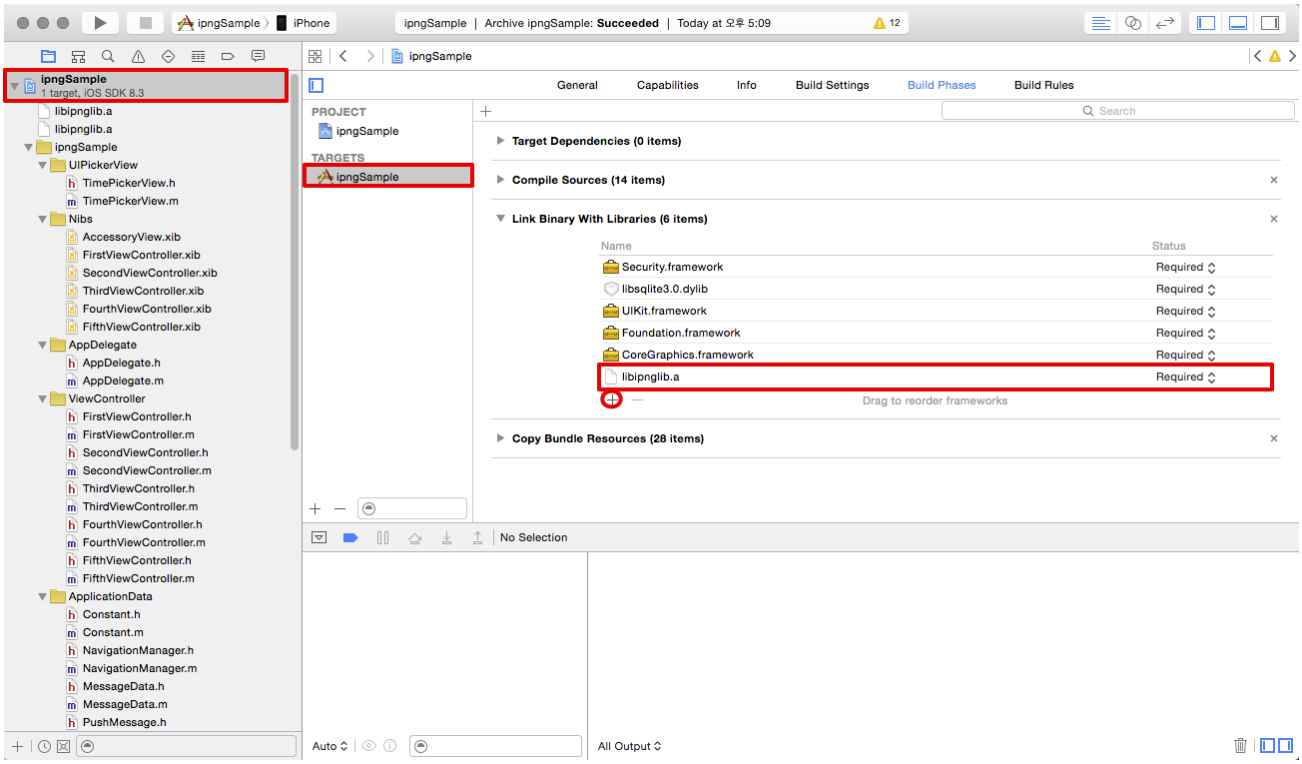
iPNS Push Library는 libipnglib.a파일과 헤더들로 이루어져 있습니다.

제품 구성물	내용
libipnglib.a	Object-C Static Library
ipnglib_extern.h	iPNS Library의 함수 헤더 파일
RequestCallbacks.h	iPNS Library에서 호출하는 Callbak 함수들의 Prototype
MessageItem.h	푸쉬 메시지 클래스 헤더파일
MessageItemList.h	푸쉬 메시지 리스트 클래스 헤더파일
ServerDefines.h	각종 define이 모여있는 헤더파일

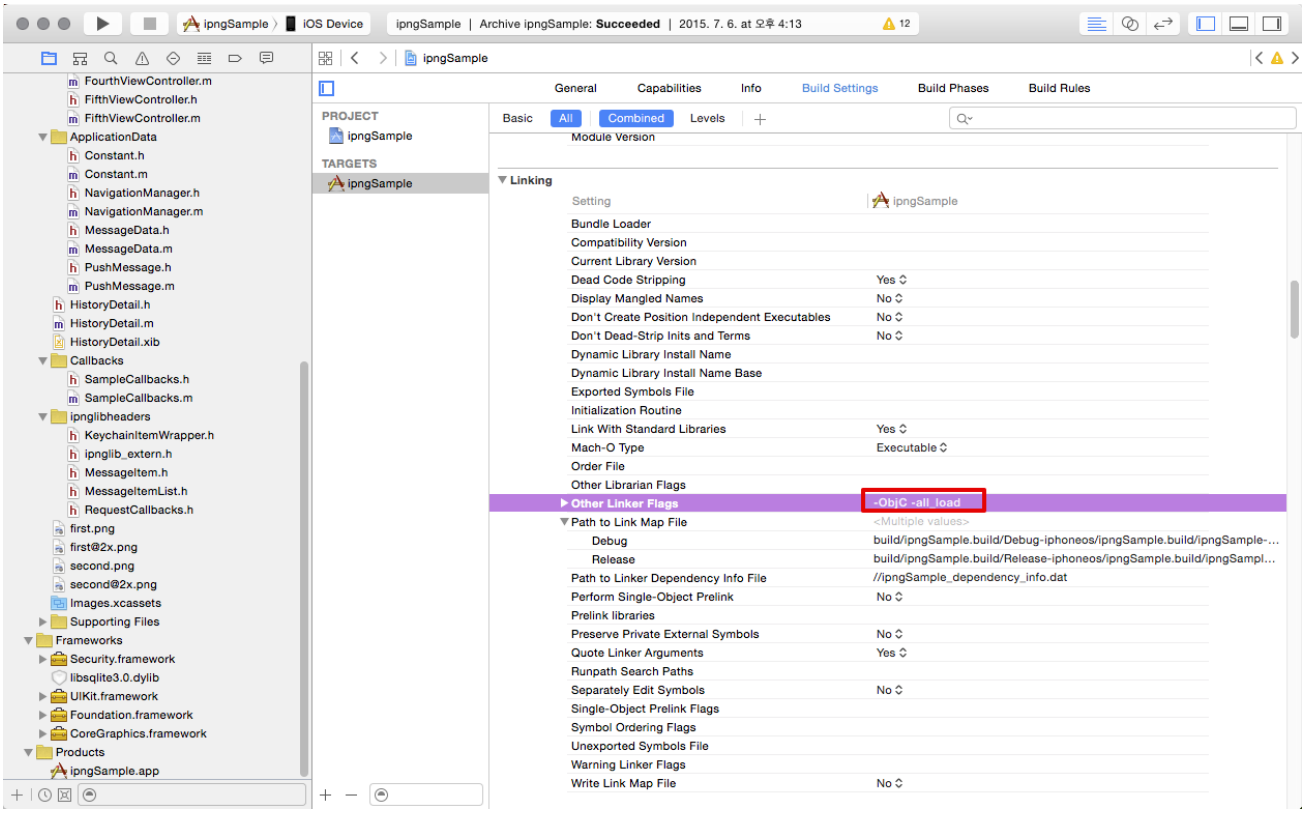
3.2. Library추가 및 설정

라이브러리를 iOS Application Project에 포함시키기 위해서는 아래와 같은 절차를 걸쳐야 합니다.

빌드 설정 화면에서 Build Phrase 탭을 선택 > Link Binray with Libraries항목을 열고 > +버튼을 눌러 libipnglib.a를 추가



Build Settings > Linking메뉴 > Other Linker Flags에 “-ObjC”를 추가.



3.3. Library 초기화 및 Callback구현

Library의 기능을 사용하기 위해서는 아래와 같이 Library를 초기화 해줘야 합니다.
initWithCallbacks API 를 이용해서 초기화가 진행됩니다.

Library초기화와 Callback 구현에 대한 자세한 내용은 Sample Project를 참고하시기 바랍니다.

Library 초기화 및 Callback구현

```
// AppDelegate.h
@property (nonatomic, strong) ipnglib *lib;
@property (nonatomic, strong) SampleCallbacks *callbacks;

// AppDelegate.m
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
        //callback초기화.
        callbacks = [SampleCallbacks alloc];
        //library초기화.
        lib = [[ipnglib alloc]initWithCallbacks:callbacks SvrAddr:[[Constant getInstance]getServerURLtoFile]];
}

// SampleCallbacks.h - 이름은 바꿔도 상관없습니다.
//RequestCallbacks.h를 임포트하고 <RequestCallbacks>를 상속받아야 함.
#import "RequestCallbacks.h"
@interface SampleCallbacks : NSObject <RequestCallbacks>
...

// SampleCallbacks.m - 이름은 바꿔도 상관없습니다.
//m파일에는 아래 interface들을 구현해줌.
//onRegist는 iPNS서버의 Registration결과를 처리.
-(void)onRegist:(NSString *)rtCode Error:(NSString *)error
//iPNS서버로 부터 수신된 Push Message를 처리.
-(void)onMessageReceived:(NSString *)msg_id Message:(NSString *)msg Alert:(NSString *)alert;
//iPNS서버로 부터 수신된 Push Message History List수신 결과를 처리.
-(void)onListReceived:(MessageItemList *)list Return:(NSString *)rtCode Error:(NSString *)error
//iPNS서버로 부터 수신된Push Message 상세 조회 결과를 처리.
-(void)onDetailReceived:(MessageItem *)message Return:(NSString *)rtCode Error:(NSString *)error
//iPNS서버로 부터 수신된Polling결과를 처리.
-(void)onPollingReceived:(MessageItemList *)list Remain:(NSInteger)remain Return:(NSString *)rtCode
Error:(NSString *)error
//iPNS서버로 부터 Push설정값 조회 결과를 처리.
-(void)onConfigReceived:(BOOL)enabled StartTime:(NSString *)sTime EndTime:(NSString *)eTime Return:
(NSString *)rtCode Error:(NSString *)error
//iPNS서버로 부터 Push설정값 업데이트 결과를 처리.
-(void)onConfigSet:(NSString *)rtCode Error:(NSString *)error
//iPNS서버로 부터 Push읽음 보고에 대한 결과를 처리.
-(void)onSetReadReport:(NSString *)rtCode Error:(NSString *)error;
```

3.4. APNS 등록 및 App등록

Library가 초기화된 상태라면 APNS 등록과 App등록이 이루어져야 합니다. APNS 등록 완료시에 setSvrAddress와 registiPNG API를 순차적으로 호출해줘야 합니다.

자세한 내용은 샘플 프로젝트를 참조바랍니다.

APNS 등록 및 App등록

```
// APNS 등록
//다른 방법으로 등록해도 되나 iOS 8.0이후에 API가 바뀐 부분을 꼭 처리해야 합니다.
- (void)registerApns {
    if(SYSTEM_VERSION_GRATERTHAN_OR_EQUALTO(@"10.0")){
        NSLog(@"ios10.0 registration");
        UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
        center.delegate = self;
        [center requestAuthorizationWithOptions:(UNAuthorizationOptionSound |
UNAuthorizationOptionAlert | UNAuthorizationOptionBadge) completionHandler:^(BOOL granted,
NSError * _Nullable error){
            if(!error){
                [[UIApplication sharedApplication] registerForRemoteNotifications];
            }
        }];
    } else if (SYSTEM_VERSION_GRATERTHAN_OR_EQUALTO(@"8.0")) {
        // iOS 8 Notifications
        NSLog(@"ios8.0 registration");

        // iOS 8 Notifications
        [[UIApplication sharedApplication] registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:(UIRemoteNotificationTypeSound | UIRemoteNotificationTypeAlert |
UIRemoteNotificationTypeBadge) categories:nil]];

        [[UIApplication sharedApplication] registerForRemoteNotifications];
    } else {
        [[UIApplication sharedApplication] registerUserNotificationSettings:[UIUserNotificationSettings
settingsForTypes:(UIUserNotificationTypeSound | UIUserNotificationTypeAlert |
UIUserNotificationTypeBadge) categories:nil]];

        [[UIApplication sharedApplication] registerForRemoteNotifications];
    }
}
// App등록 :  APNS가 정상적으로 등록될 때 호출되는 didRegisterForRemoteNotificationsWithDeviceToken에 아
래와 같이 구현합니다.
// AlertView는 테스트 용도입니다.
- (void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:
(NSData *)deviceToken{
    //NSLog(@"Device Token:%@", deviceToken);
    NSMutableString *tokenString = [NSMutableString string];
    unsigned char *ptr = (unsigned char *)[deviceToken bytes];
    for (NSInteger i=0; i<32; i++) {
        [tokenString appendString:[NSString stringWithFormat:@"%02x", ptr[i]]];
    }
    UIAlertView *alertView = [[UIAlertView alloc]initWithTitle:@"IpngTest"
                                                       message:[NSString stringWithFormat:@"Device Token Received:%@₩n",
tokenString] delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alertView show];
    @try{
        if (lib) {
            NSLog(@"appid = %@, usercn = %@, tokenString = %@", [[Constant getInstance]appid],
[[Constant getInstance]usercn], tokenString);
            [lib setSvrAddress:[[Constant getInstance]getServerURLtoFile]];
            [lib registiPNG:[NSString stringWithString:[Constant getInstance].appid] UserCN:[NSString
stringWithString:[Constant getInstance].usercn] DeviceToken:[NSString stringWithString:tokenString]];
        }
    }@catch (NSException *e) {
        NSLog(@"Exception = %@", e);
    }
}
```

3.5. Push Message수신시 처리 추가

Library기능이 정상적으로 동작하려면 APNS Push Message를 수신할 때 Library의 API를 Call 해줘야 합니다.

pushReceived API는 Push Server로부터 APNS 서버를 통해 단말로 도달된 메시지를 파싱하고 Callback의 onMessageReceived를 최종적으로 호출해주는 역활을 합니다.

APNS 수신시 처리

```
// APNS 수신 시에 아래와 같이 library의 pushReceived를 호출해야 합니다.
/**
 *  background nofication에 대한 처리.
 *  content-available : 1인 경우 이 함수를 타게된다.
 **/
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(nonnull NSDictionary
*)userInfo fetchCompletionHandler:(nonnull void (^)(UIBackgroundFetchResult))completionHandler{
    NSLog(@"Background Notification : didReceiveRemoteNotification!!!! FetchCompletionHandler!!!!!");
    [self handleUserInfo:userInfo];
    completionHandler(UIBackgroundFetchResultNewData);
}

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo{
    NSLog(@"didReceiveRemoteNotification!!!!");
    [self handleUserInfo:userInfo];
}

-(void)handleUserInfo:(NSDictionary *)userInfo{
    if(userInfo){
        NSLog(@"userInfo = %@", userInfo);
        if(lib==nil){
            NSLog(@"library is nil!!!!! re-initializing library.");
            if(callbacks==nil){
                NSLog(@"callbacks is nil!!!!! re-initializing callbacks.");
                callbacks = [SampleCallbacks alloc];
            }
            lib = [[ipnglib alloc]initWithCallbacks:callbacks SvrAddr:[[Constant
getInstance]getServerURLtoFile]];
        }
        //라이브러리에서 전달된 push메시지에 대한 처리를 진행
        [lib pushReceived:[[Constant getInstance]appid] Data:userInfo];

        NSString *alert = [[userInfo valueForKey:@"aps"] valueForKey:@"alert"];
        NSString *message = [userInfo valueForKey:@"message"];
        NSString *time = [self getTime];
        if (!alert) {
            alert = @"";
        }
        if (!message){
            message = @"";
        }
        if (!time){
            time = @"";
        }
    }
}
```

3.6. Push Message 읽음 확인 보고

Push Library **reportReadResultToServer** API를 통하여 특정 Message에 대해서 읽음 확인 보고를 Server로 전달 가능합니다.

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

4.5항목에서 pushReceived를 정상적으로 호출 한경우 Callback의 onMessageReceived가 호출되게 됩니다. onMessageReceived의 파라미터는 reqID, message, alert 세 개인데, reqID가 MessageID를 의미합니다.

위의 MessageID를 확인한 후 아래와 같은 방법으로 해당 Message에 대한 읽음 확인을 Push Server로 전달 가능합니다.

Push Message 읽음 확인 보고

```
// appDelegate에 library가 있는 경우.
AppDelegate *appDelegate = (AppDelegate *)[[UIApplication sharedApplication] delegate];
// 첫번째 파라미터는 APPID이며 두번째 파라미터가 msg_id(혹은 request_id)입니다.
[appDelegate.lib reportReadResultToServer:APPID ReqID:msg_id];

// 결과 값은 아래의 Callback으로 전달되며 rtCode가 0000인 경우 성공 그렇지 않은 경우 실패입니다.
-(void)onSetReadReport:(NSString *)rtCode Error:(NSString *)error
{
    [self ShowAlert:[NSString stringWithFormat:@"Return Code:%@, Reason:%@", rtCode, error]];
}
```

3.7. Push 수신여부 등 설정 전달 및 수신

PushLibrary는 Server로 해당 단말 혹은 사용자의 Push설정 값을 전달/조회할 수 있는 기능을 클라이언트 레벨에서 제공합니다.

4.7.1. 설정 전달

Push등록 이후에 **updateConfig** API를 호출함으로써 해당 기능을 사용가능하며
설정 가능 값은 아래와 같습니다.

- Push 수신 여부 (Default : true)
- CP User ID 변경 (Default : Push등록시의 CP User ID)
- Push 수신 시작/종료시간 (Default : “0000”/”2359”)

아래는 Server로 설정값을 전달하는 방법입니다.

Push 설정 전달

```
// 자세한 내용은 Sample Project를 참조하시기 바랍니다.
[lib updateConfig:[[Constant getInstance]appid] Start:startTime.text End:endTime.text YN:[pushSwitch isOn] CpUserId:cpuserid.text];

// 위 요청에 대한 결과는 RequestCallbacks.h에  onConfigSet함수로 전달됩니다.
// 아래와 같이 RT값에 대한 처리가 가능합니다.
-(void)onConfigSet:(NSString *)rtCode Error:(NSString *)error
{
    [self ShowAlert:[NSString stringWithFormat:@"Return Code:%@, Reason:%@", rtCode, error]];
}
```

4.7.2. 설정 수신

Push등록 이후에 **GetPushConfig**를 호출하면 서버에 저장되어 있는 해당 단말의 설정값을 가져올 수 있습니다.

가져올 수 있는 설정값은 4.1.1의 설정값과 동일합니다.

다음은 Server로부터 설정값을 수신하는 방법입니다.

Push 설정 전달

```
// 자세한 내용은 Sample Project를 참조하시기 바랍니다.
[lib getConfig:[[Constant getInstance]appid]];

// 위 요청에 대한 결과는 RequestCallbacks.h에  onConfigReceived함수로 전달됩니다.
// 아래와 같이 설정값에 대한 처리가 가능합니다.
-(void)onConfigReceived:(BOOL)enabled StartTime:(NSString *)sTime EndTime:(NSString *)eTime Return:
(NSString *)rtCode Error:(NSString *)error
{
    [self ShowAlert:[NSString stringWithFormat:@"Return Code:%@, Reason:%@", rtCode, error]];
    if(sTime)
        self.config_sTime = [NSString stringWithString:sTime];
    else {
        self.config_sTime = [NSString stringWithString:@""];
    }
    if(eTime){
        self.config_eTime = [NSString stringWithString:eTime];
    }
    else{
        self.config_eTime = [NSString stringWithString:@""];
    }
    //Push수신 여부
    self.config_bEnabled = enabled;
    if (target && selector)
        [target performSelectorOnMainThread:selector withObject:nil waitUntilDone:NO];

}
```

3.8. Push Message History 조회

PushLibrary의 **getMsgs** API를 통하여 Server에 저장된 기 발송Message들에 대한 조회가 가능합니다.

기본적으로 PushLibrary 초기화 시에 등록한 callback 의 onListReceived 가 호출됩니다. 하지만 이 이벤트를 ViewController 에서 받고 싶을 때는 Sample Project 처럼 RequestCallbacks 를 구현하는 클래스를 아래처럼 구현하고

SampleCallbacks.h

```
#import <Foundation/Foundation.h>
#import "RequestCallbacks.h"
#import "MessageData.h"

//아래와 같이 target,selector를 멤버로 선언
@interface SampleCallbacks : NSObject <RequestCallbacks>
{
    id target;
    SEL selector;
}
@property (nonatomic, strong)MessageItemList *history;
@property (nonatomic, strong)MessageItem *detail;
@property (nonatomic, strong)MessageItemList *polling;
@property (nonatomic) NSInteger pollingRemain;
@property (nonatomic) BOOL config_bEnabled;
@property (nonatomic, strong)NSString *config_sTime;
@property (nonatomic, strong)NSString *config_eTime;
@property (nonatomic, strong)NSString *received;

//target,selector의 setter
-(void)setTarget:(id)theTarget Selector:(SEL)theSelector;
@end
```

m파일에서 아래와 같이 구현해주면 onListReceived 가 호출될 때 ViewController의 selector를 호출하도록 할 수 있습니다.

SampleCallbacks.m

```
-(void)setTarget:(id)theTarget Selector:(SEL)theSelector
{
    target = theTarget;
    selector = theSelector;
}

-(void)onListReceived:(MessageItemList *)list Return:(NSString *)rtCode Error:(NSString *)error
{
    [self ShowAlert:[NSString stringWithFormat:@"onListReceived Return Code:%@, Reason:%@", rtCode,
error]];
    self.history = list;
    if (target && selector)
        [target performSelectorOnMainThread:selector withObject:nil waitUntilDone:NO];
}
```

getMsgs는 Android의 GetPushHistoryList와 비슷하게 appID, pushIdType, reqId, reqType, count 를 파라미터로 가지고 있습니다.

- appID : regitstiPNG 호출시에 입력한 appID 값
- pushIdType : ServerDefines.h 의 PUSH_ID_TYPE_DEVICE_ID(@”00”) PUSH_ID_TYPE_USER_CN(@”01”) 중 하나를 선택할 수 있습니다.

* Device ID로 선택시에 동일 단말에서 다른 UserCN 값으로 전달된 메시지 까지 가져올 수 있습니다.

- reqId : 조회의 기준이 될 메시지의 Request Id 혹은 Message Id. (onMessageReceived의 파라미터인 msg_id)
- reqType : @“N”(New), @“O”(Old) 중에 하나를 선택할 수 있습니다. Old 인 경우 reqId 를 기준으로 이보다 더 전송된지 오래된 항목을 조회합니다. New 인 경우, reqId는 무시되며 가장 최신으로부터 더 오래된 순으로 Message를 조회합니다.
- Count : 위의 파라미터 조건에 따른 Message 조회 시 요청할 Message의 개수를 의미합니다. 최대 50개까지 요청 가능합니다.(50이상을 요청해도 Server에서 50개만 전달됩니다.)

getMsgs 의 호출은 아래와 같이 할 수 있습니다.

Push Message History 조회

```
-(void)getMsgList:(NSString *)theAppid More:(BOOL)bMore Reference:(NSString *)reqid Count:
(NSInteger)theCount
{
    @try{
        //라이브러리 객체 가져옴
        ipnglib *lib = ((AppDelegate *)[[UIApplication sharedApplication] delegate]).lib;
        //라이브러리 초기화 시에 선언된 callback 객체를 소환
        SampleCallbacks *callback = ((AppDelegate *)[[UIApplication sharedApplication] delegate]).callbacks;
        //기본적으로 callback 의 onListReceived 로 결과값이 전달되나, ViewController 상에서 직접 받아서 처리해야
할 경우
        //아래 처럼 callback 의 setTarget API로 selector 를 지정해두면
        //정상 요청 시에 selector 가 호출된다.
        if (callback) {
            [callback setTarget:self Selector:@selector(actionDone:)];
        }
        if (lib) {
            [lib getMsgs:[[Constant getInstance]
                  appid] PushIdType:PUSH_ID_TYPE_USER_CN ReqID:reqid ReqType:@"N" Count:theCount];
        }
    }@catch (NSException *e) {
        NSLog(@"Exception = %@", e);
    }
}


//selector 의 구현부
-(void)actionDone:(id)dumb
{
    SampleCallbacks *callback = ((AppDelegate *)[[UIApplication sharedApplication] delegate]).callbacks;

    MessageItemList *result = callback.history;
    if (result && [result count]) {
        if (dataList && [dataList count]) {
            for (NSInteger i=0; i<[result count]; i++) {
                MessageItem *item = [result getMessageAtIndex:i];
                if (![dataList searchListWithReqID:item.requestID]) {
                    //not duplicated message
                    NSLog(@"ReqID:%@", item.requestID);
                    [dataList addMessage:item];
                }
                else
                {
                    NSLog(@"Duplicated Message:%@", item.requestID);
                }
            };
        }
        else
        {
            //currently has no data
            dataList = callback.history;

        }
    }
    [myTableView reloadData];
    [self showFooterBtn];
}
```