|  |
| --- |
| Turkcell İletişim Hizmetleri A.Ş. |
| Turkcell Push SDK iOS Entegrasyon Kılavuzu |
| Turkcell Push SDK Entegrasyon Kılavuzu |

|  |
| --- |
| Abdulbasıt Tanhan  10/21/2013 |

İçindekiler

Giriş 2

SDK’nın Projeye Referans Olarak Eklenmesi 2

Turkcell Push Server Üzerinde Uygulamanın Oluşturulması 2

SDK Metodları 2

Ayarların Atanması 3

Kayıt 4

Bildirim Kategori Listesinin Alınması 4

Bildirim Kategorisine Abone Olma 4

Bildirim Kategorisi Aboneliğini Kaldırma 5

Bildirim Kategorisi Aboneliklerinin Alınması 5

Bildirim Geçmişinin Alınması 5

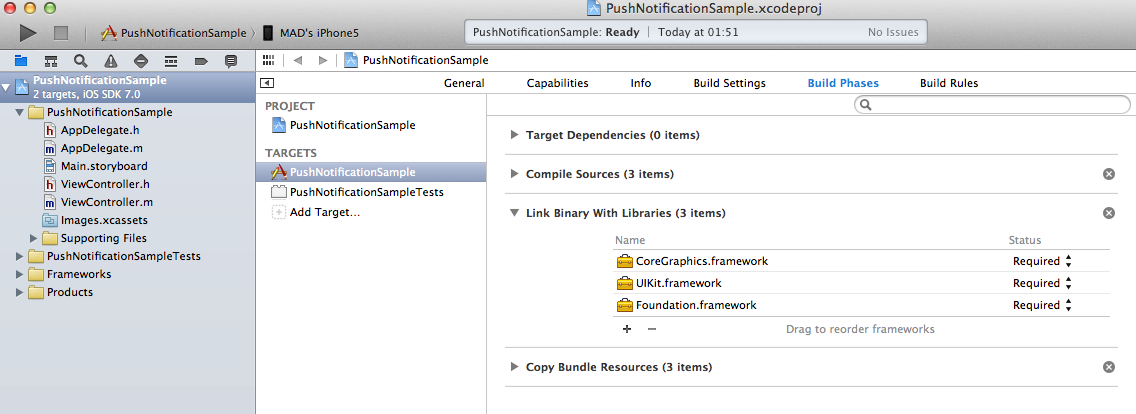
# Giriş

Turkcell Push SDK sayesinde iOS uygulamalarınıza kolayca push bildirimleri yeteneği ekleyebilirsiniz. Bu dokümanda bir iOS uygulamasına Turkcell Push SDK entegrasyonunun nasıl yapılacağı anlatılmıştır.

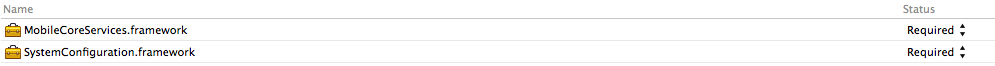
# SDK’nın Projeye Referans Olarak Eklenmesi

TCellPushNotification.framework’ u projeye eklemek için Xcode’ ta açık olan projenizi seçtiğinizde sağda beliren Targets bölümünden aşağıdaki yolu izleyerek ekleyebilirsiniz.

Targets -> Your Target -> Link Binary With Libraries -> Artı işareti -> Add Other



Ayrıca projenize MobileCoreServices.framework ve SystemConfiguration.framework kütüphanelerini eklemelisiniz.



# Turkcell Push Server Üzerinde Uygulamanın Oluşturulması

Uygulamanıza Turkcell Push SDK entegrasyonu yapabilmeniz için öncelikle Push Server üzerinde uygulama kaydı oluşturmalısınız. Bir iOS uygulamasını Push Server’a kaydetmek için aşağıdaki bilgiler ile Push Server sorumlusuna başvurmanız gerekmektedir.

|  |  |
| --- | --- |
| Alan İsmi | Alan Açıklaması |
| Uygulama Adı | Push Server üzerinde uygulamanıza vermek istediğiniz isim. Uygulamanızın birden fazla platformda yer alabileceğini düşünerek uygulama isminde platform adına da yer vermek faydalı olabilir. |
| Güvenlik Anahtarı | Sizin belirlediğiniz bir SecretKey |

Bu bilgiler ile birlikte uygulamanız oluşturulduğunda size ***ApplicationId*** bilgisi iletilecek. ***ApplicationId*** *ve* ***SecretKey*** *SDK için bulunması zorunlu iki değerdir.*

# SDK Metodları

Turkcell Push SDK’nın en önemli sınıfı **TCellNotificationManager** sınıfıdır. İşlemlerin hemen hepsi bu sınıfın üzerindeki **sharedInstance** özelliği üzerinde yapılmaktadır.

## Ayarların Atanması

**TCellNotificationManager** sınıfı üzerindeki metodlar çağırılmadan önce bu sınıfın gerekli bilgiler ile hazırlanması gerekmektedir. **TCellNotificationManager** sınıfı **sharedInstance** metodu aracılığıyla gerekli bilgileri kabul etmektedir. Ayarlar **TCellNotificationSettings** sınıfı içerisinde tutulmaktadır. Bir TCellNotificationSettings nesnesi oluşturmalı ve bu nesne üzerindeki metotlara gerekli parametreleri vermelisiniz. Örnek bir kod bloğu;

#import <TCellPushNotification/TCellNotificationSettings.h>

#import <TCellPushNotification/TCellNotificationManager.h>

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions

{

TCellNotificationSettings\* settings = [[TCellNotificationSettings alloc]

initWithAppId:@"0dc982a768ef94eea195b05c7cc405bb"

secretKey:@"7adc160c-34c7-41e2-81e2-09fc9d14b9b4"

notificationTypes:(UIRemoteNotificationTypeBadge | UIRemoteNotificationTypeAlert | UIRemoteNotificationTypeSound)];

TCellNotificationManager\* man =[TCellNotificationManager sharedInstance];

man.notificationSettings = settings;

[man registerApplicationForRemoteNotificationTypes];

return YES;

}

Buradaki **ApplicationId** ve **SecretKey** değerleri belirtilmesi zorunlu alanlardır, belirtilmedikleri takdirde hata alınması kaçınılmazdır.

TCellNotificationSettings nesnesi ile sağlayabileceğiniz değerler ve açıklamaları aşağıdaki tabloda gösterilmiştir.

|  |  |  |
| --- | --- | --- |
| Alan Adı | Açıklaması | Zorunlu mu? |
| appId | PushServer üzerinde uygulamanız oluşturulduğunda size iletilen ApplicationId değeri. | Evet |
| secretKey | PushServer üzerinde uygulamanızın oluşturulması için oluşturup yetkili kişiye bildirdiğiniz **SecretKey** değeri. | Evet |
| notificationTypes | UIRemoteNotificationTypeNone, UIRemoteNotificationTypeBadge, UIRemoteNotificationTypeSound, UIRemoteNotificationTypeAlert, UIRemoteNotificationTypeNewsstandContentAvailability değerlerinden birini veya birkaçını alabilir. | Evet |

Diğer atanması gereken zorunlu değişken TCellNotificationManager nesnesinde deviceToken değişkeni. Bunu cihazın size sağlayacağı token ile yapmanız gerekiyor. Bu tokenı alabilmeniz uygulamanızı Notification Center’ a kaydetmeniz gerekiyor. Yukarıdaki kod örneğinde bu aşağıdaki satırda yapılmış oluyor.

[[TCellNotificationManager sharedInstance] registerApplicationForRemoteNotificationTypes];

Bunu yaptığınızda uygulama ilk açıldığında sizden Notification Center’a kayıt izni isteyecektir.

Daha sonra tokenı alabilmek için AppDelegate sınıfına aşağıdaki delegate metotlarını eklemeniz gerekiyor.

- (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)newDeviceToken

{

[[TCellNotificationManager sharedInstance]

setNotificationDeviceTokenWithData:newDeviceToken];

}

- (void)application:(UIApplication \*)app didFailToRegisterForRemoteNotificationsWithError:(NSError \*)err

{

NSString \*str = [NSString stringWithFormat: @"Error: %@", err];

NSLog(@"Error %@",str);

}

## Kayıt

**TCellNotificationManager** nesnesine gerekli ayarlar sağlandıktan sonra Push bildirimleri alabilmek için yapılması gereken ilk işlem kayıt olmaktır. Push Server’ a kayıt olmak için **registerDeviceWithDelegate** metodu çağırılmalıdır. NotificationManagerDelegate protokolü üzerinden alabilirsiniz. Örnek kullanım aşağıdaki gibidir.

[[TCellNotificationManager sharedInstance] registerDeviceWithDelegate:self];

İşlem sonucunda **TCellRegistrationResult** tipinde bir nesne döner. Bu nesne üzerindeki **isSuccessfull** özelliği ile işlem sonucunu kontrol edebilirsiniz. İşlem sonucunun başarısız olması durumunda ise **TCellRegistrationResult.Error** ve **TCellRegistrationResult.StatusCode** alanları ile hata sebebi ile ilgili detaylı bilgiye ulaşabilirsiniz.

- (void)registrationResult:(TCellRegistrationResult\*)result;

Kayıt olmanın yanı sıra bir uygulama push bildirimleri almayı kesmek te gerekebilir. Push Server kaydığını kaldırmak için **unRegisterDeviceWithDelegate** metodu çağırılmalıdır. Dönen cevap nesnesi Register metodu ile aynıdır.

[[TCellNotificationManager sharedInstance] unRegisterDeviceWithDelegate:self];

Protokol metodu;

- (void)unRegistrationResult:(TCellRegistrationResult\*)result;

result değişkeninden **isSuccessfull** değerini kontrol ederek başarılı ise uygulamayı Notification Center’ dan kaldırabilirsiniz. Bunun için aşağıdaki satırı eklemek yeterli olacaktır.

[[TCellNotificationManager sharedInstance] unRegisterApplicationForRemoteNotificationTypes];

## Bildirim Kategori Listesinin Alınması

Push Server uygulama bazlı Push bildirim kategorileri oluşturma yeteneği sağlamaktadır. Bu yetenek sayesinde uygulamalar sadece belirli kategorilerdeki bildirimlere abone olabilmektedir. Örneğin bir haber uygulamasında kullanıcı sadece Spor kategorisindeki bildirimleri almak isteyebilir. Uygulamada mevcut kullanılabilir bildirim kategori listesini almak için **getCategoryListWithDelegate** metodu çağırılmalıdır. Örnek bir kod bloğu aşağıdaki gibidir.

[[TCellNotificationManager sharedInstance] getCategoryListWithDelegate:self];

Bu metot cevap olarak **TCellCategoryListQueryResult** tipinde bir nesne döndürür. Nesne üzerindeki **categories** alanı üzerinden mevcut kategori listesine ulaşabilirsiniz. İsteğin başarılı olup olmadığını **isSuccessfull** özelliğinden, başarısızlık durumunda yine **TCellCategoryListQueryResult.Error** ve **TCellCategoryListQueryResult.StatusCode** özellikleri üzerinden detaylı bilgiye ulaşabilirsiniz.

## Bildirim Kategorisine Abone Olma

Uygulama için tanımlanmış bildirim kategori listesi alındıktan sonra kullanıcıya bu kategorilere abone olabileceği bir ara yüz sunabilirsiniz. Kullanıcıyı herhangi bir kategoriye abone yapmak için **subscribeToCategoryWithDelegate** metodunu çağırmalısınız. Metot parametre olarak kategori ismini almaktadır. Örnek bir kod bloğu aşağıdaki gibidir.

[[TCellNotificationManager sharedInstance] subscribeToCategoryWithDelegate:self  
 categoryName:@"spor"];

Cevap olarak **TCellCategorySubscriptionsResult** tipinde bir nesne dönmektedir. İsteğin başarılı olup olmadığını **isSuccessfull** özelliği üzerinden kontrol edebilirsiniz.

## Bildirim Kategorisi Aboneliğini Kaldırma

Kullanıcının herhangi bir bildirim kategorisine olan aboneliğini kaldırmak için **subscribeToCategoryWithDelegate** metodu çağırılmalıdır. Metot parametre olarak kategori ismini almaktadır. Örnek bir kod bloğu aşağıdaki gibidir.

[[TCellNotificationManager sharedInstance] unSubscribeFromCategoryWithDelegate:self   
 categoryName:@"spor"];

Cevap olarak **TCellCategorySubscriptionsResult** tipinde bir nesne dönmektedir. İsteğin başarılı olup olmadığını **isSuccessfull** özelliği üzerinden kontrol edebilirsiniz.

## Bildirim Kategorisi Aboneliklerinin Alınması

Kullanıcın abone olduğu bildirim kategorilerinin listesini almak için **getCategorySubscriptionsWithDelegate** metodu çağırılmalıdır. Örnek bir kullanım aşağıdaki gibidir.

[[TCellNotificationManager sharedInstance] getCategorySubscriptionsWithDelegate:self];

Bu metod cevap olarak **TCellCategoryListQueryResult** tipinde bir nesne döndürür. Nesne üzerindeki **categories** alanı üzerinden mevcut kategori listesine ulaşabilirsiniz. İsteğin başarılı olup olmadığını **isSuccessfull** özelliğinden, başarısızlık durumunda yine **TCellCategoryListQueryResult.Error** ve **TCellCategoryListQueryResult.StatusCode** özellikleri üzerinden detaylı bilgiye ulaşabilirsiniz.

## Bildirim Geçmişinin Alınması

Kullanıcıya o zamana kadar iletilmiş olan bildirim listesini almak için **getNotificationHistoryWithDelegate** metodu çağırılmalıdır. Metot parametre olarak sayfa sayısı ve bir sayfadaki bildirim sayısını alır. Aşağıdaki örnek kod bloğunda her biri 15 bildirimden oluşan bildirim geçmişi sayfalarından ikinci sayfa isteniyor.

[[TCellNotificationManager sharedInstance] getNotificationHistoryWithDelegate:self offSet:2  
 listSize:15];

Bu metoda cevap olarak **notificationHistoryResult** tipinde bir nesne döner. Bildirimlere bu nesne üzerindeki **messages** özelliğinden ulaşabilirsiniz.