Abstract

This article describes code sample illustrating using Azure Notification Hub to deliver push notification messages to mobile applications on iOS, Android and Windows Phone platforms

Notification Hub Sample PushMeNow

Table of Contents

[Introduction 1](#_Toc444384190)

[Use case scenario: 2](#_Toc444384191)

[Components and Implementation details 4](#_Toc444384192)

[User registration implementation 4](#_Toc444384193)

[.NET Core 1.0 Web Application 4](#_Toc444384194)

[Notes 4](#_Toc444384195)

[Creating a site 5](#_Toc444384196)

[Node.JS Web App 13](#_Toc444384197)

[Create a Node.JS app and connect to the Notification Hub 13](#_Toc444384198)

[Create pages to display the list of registrations and send notification to different platforms 14](#_Toc444384199)

[Creation Azure Web App and configuration the continuous deployment for Node.JS Web App 17](#_Toc444384200)

[Publish Node.JS application to Azure using Git 18](#_Toc444384201)

[Windows Phone Client 19](#_Toc444384202)

[Android Client 25](#_Toc444384203)

[iOS Client 26](#_Toc444384204)

[Summary 29](#_Toc444384205)

# Introduction

The idea of this sample is to give an ability Azure Web App visitor to send a push notification to any registered mobile device having installed mobile application from this sample.

Web Application displays a list of registered devices, user selects a device from the list then types in a free text message and sends it.

To cover more audience this sample includes two Web Applications which are equivalent from functional prospective, but developed in two different technologies:

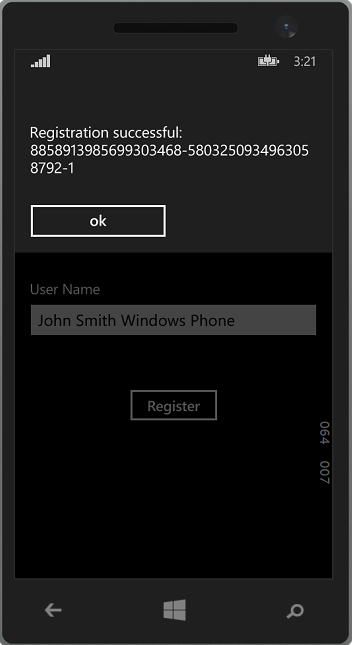
* Node.JS
* Latest Microsoft .NET Core 1.0 (which was originally introduced as ASP.NET 5.0)

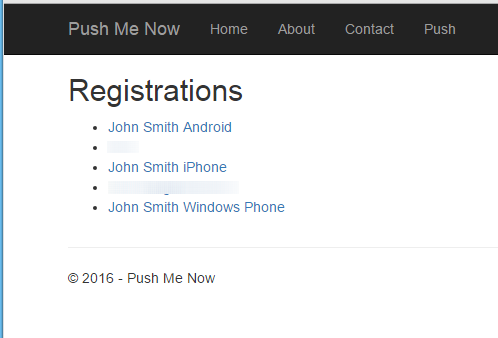
Node.JS is very popular in a startup world and .NET Core 1.0 is becoming new wave of technology evolution from Microsoft. Azure Notification Hub client library has not provided by Microsoft yet. This fact makes this sample valuable, because it demonstrates how to use native REST API and this approach can applied for ANY technology stack.

Mobile application allows user to describe himself in simple string format and receive push notification messages sent to this device from Azure Notification Hub. This sample contain simple native applications for three major platforms: Windows Mobile, iOS and Android.

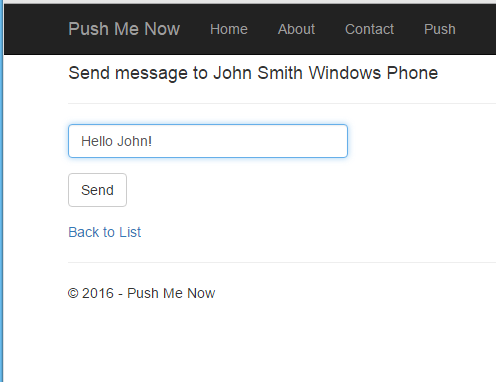
## Use case scenario:

We are considering two actors (***Subscriber*** and ***Publisher***) and three major use cases for push notifications (screenshots are done using Windows Phone, but Android and iOS versions are also available and you can find screens below in the section describing iOS and Android implementation details):

* ***Subscriber***: Subscribe for notifications
  + ***Subscriber*** downloads and installs application built from the sample. On the first application start ***subscriber*** enters tag to identify the device (like *John Smith*). This tag is just a free text string which will be used by ***Publisher*** to identify user’s device.  
     
* ***Publisher:*** Send notification to selected user (device)
  + ***Publisher*** visits *Azure Web App* page and gets the list of registered devices (list of tags provided by ***subscribers***):



On the page user selects one of them which is target for notification, types message and presses a send button. Selected device will display push notification message:



* ***Subscriber:*** Receive notification and read message
  + ***Subscriber***  receives push notification message in a way standard for specific mobile platform and can read text message delivered.



# Components and Implementation details

## User registration implementation

System needs to have a list of registered devices, at the same time we would like to keep this sample as simple as it possible to demonstrate push notification technology. To keep things simple we will not introduce any database backend which will be a common in more real life applications. We will place the user information into a tag which represents registered device in Azure Notification Hub. User information is a JSON object containing required information, some salt data to make user with same information have a unique tag. For this sample user information will look like this: {"name":"John Smith","s":"54b9d84a32"} (assuming ***Subscriber*** entered *John Smith* during device registration)

The Notification Hub’s notification registration tag can contain only alphanumeric symbols. So we will encode a JSON as a base64 string increasing a salt field size to be a 8 or more to avoid an ending ‘=’ symbols in the base64 string.

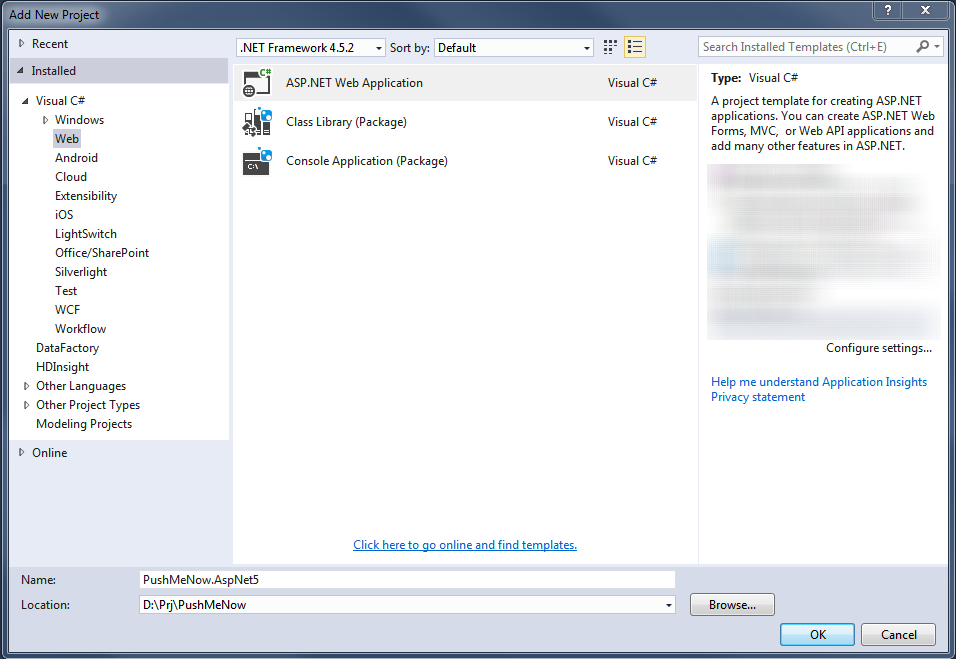
## .NET Core 1.0 Web Application

### Notes

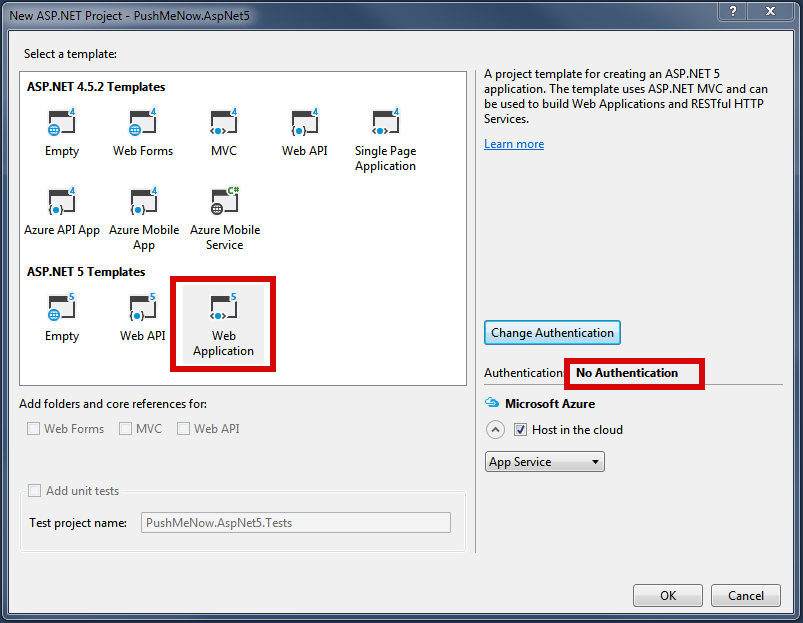
Please note that .NET Core 1.0 is in the release candidate RC1 state at the moment the sample is being created. Also there is no official notification hub client working with DNX.Core 5.0. So a simple rest API client NotificationHubApiClient is implemented in this sample. It does have a minimum required notification hub client functionality, but limited error handling and does not covers all possible cases, that is why it is strongly recommended to use an official client library in the production as soon as it will be available.

### Creating a site

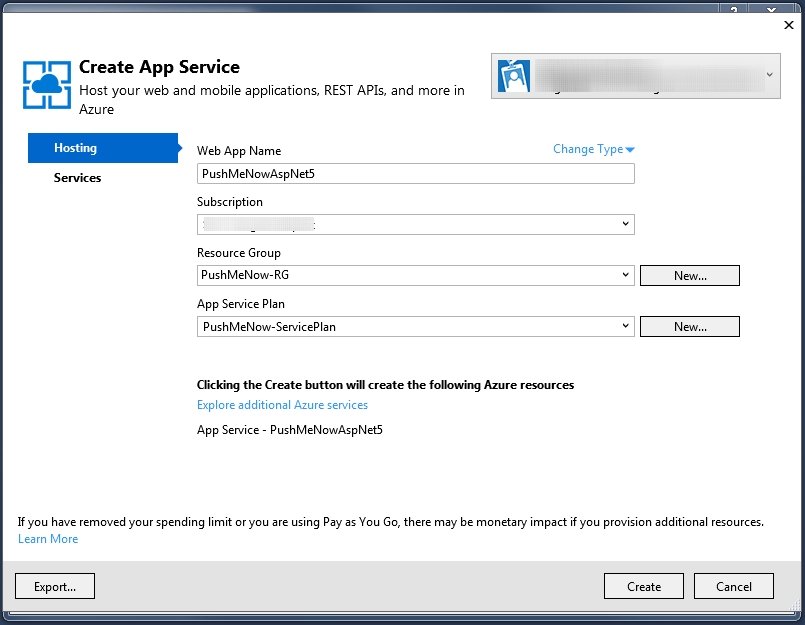
* Create a new ASP.NET web application (please note that Visual Studio is still using the old name for the technology – ASP.NET 5.0, so please don’t be confused):



* Select an ASP.NET 5 web application template and turn off authentication and keep Host in the Cloud selected.



* Setup an Azure deployment as a Web App:



* Add *System.Net.Http* and *Newtonsoft.Json* Nuget packages.
* Add a new project folder Models for models and create a new model class UserInfo:

**using System;**

**using Newtonsoft.Json;**

**namespace PushMeNow.AspNet5.Models**

**{**

**public class UserInfo**

**{**

**[JsonProperty("name")]**

**public string Name { get; set; }**

**}**

**}**

* Add a model class Registration:

using System;

using System.Text;

using Newtonsoft.Json;

namespace PushMeNow.AspNet5.Models

{

enum RegistrationType

{

Windows,

Apple,

Gcm,

}

public class Registration

{

public Registration()

{ }

public Registration(Registration other)

{

RegistrationType = other.RegistrationType;

Tags = other.Tags;

}

[JsonProperty("registration\_type")]

public string RegistrationType { get; set; }

[JsonProperty("tags")]

public string Tags { get; set; }

public string Name

{

get

{

try

{

return

JsonConvert.DeserializeObject<UserInfo>(Encoding.UTF8.GetString(Convert.FromBase64String(Tags)))

.Name;

}

catch

{

return "Unknown";

}

}

}

}

}

### 

* Add a model class Message:

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

using Newtonsoft.Json;

namespace PushMeNow.AspNet5.Models

{

public class Message : Registration

{

public Message()

{

}

public Message(Registration registration)

: base(registration)

{

}

[JsonProperty("text")]

[Required]

public string Text { get; set; }

}

}

### 

* Add a new controller PushController and modify it like this:

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.AspNet.Mvc;

using PushMeNow.AspNet5.Models;

// For more information on enabling MVC for empty projects, visit http://go.microsoft.com/fwlink/?LinkID=397860

namespace PushMeNow.AspNet5.Controllers

{

public class PushController : Controller

{

public async Task<IActionResult> Index()

{

var registrations = await GetRegistrationsAsync();

return View(registrations);

}

[HttpGet]

public IActionResult Send([Bind("Tags,RegistrationType")] Registration registration)

{

return View(new Message(registration));

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Send([Bind("Tags,Text,RegistrationType")] Message message)

{

if (ModelState.IsValid)

{

NotificationHubApiClient notificationHubApiClient = GetClient();

await notificationHubApiClient.SendNotificationAsync(message);

return RedirectToAction("Index");

}

return View(message);

}

private async Task<IEnumerable<Registration>> GetRegistrationsAsync()

{

var notificationHubApiClient = GetClient();

var registrationTags = await notificationHubApiClient.GetRegistrationsAsync();

return registrationTags;

}

private static NotificationHubApiClient GetClient()

{

string hubName = "<HUB-NAME>";

string fullConnectionString = "<HUB-FULL-CONNECTION\_STRING>";

return new NotificationHubApiClient(hubName, fullConnectionString);

}

}

}

### 

* Add a new Views subfolder Push and add a pair of views for registration selection and push message sending.
  + Index.cshtml like this:

@using System.Threading.Tasks

@using Newtonsoft.Json

@using PushMeNow.AspNet5.Models

@model IEnumerable<Registration>

@{

ViewData["Title"] = "Push Page";

}

<div class="row">

<div class="col-md-12">

<h2>Registrations</h2>

<ul>

@foreach (var registration in Model)

{

<li><a asp-controller="Push" asp-action="Send" asp-route-tags="@registration.Tags" asp-route-registrationType="@registration.RegistrationType">@registration.Name</a></li>

}

</ul>

</div>

</div>

* + Send.cshtml like this:

@using System.Threading.Tasks

@using PushMeNow.AspNet5.Models

@model Message

@{

ViewData["Title"] = "Send message to " + Model.Name;

}

<form asp-action="Send">

<div class="form-horizontal">

<h4>@ViewData["Title"]</h4>

<hr />

<div asp-validation-summary="ValidationSummary.ModelOnly" class="text-danger" />

<input type="hidden" asp-for="Tags" />

<input type="hidden" asp-for="RegistrationType" />

<div class="form-group">

<div class="col-md-12">

<input asp-for="Text" class="form-control" />

<span asp-validation-for="Text" class="text-danger" />

</div>

</div>

<div class="form-group">

<div class="col-md-12">

<input type="submit" value="Send" class="btn btn-default" />

</div>

</div>

</div>

</form>

<div>

<a asp-action="Index">Back to List</a>

</div>

### 

* Edit a Shared/\_Layout.cshtml
  + Add a line to nav bar in the to

<li><a asp-controller="Push" asp-action="Index">Push</a></li>

* + Add script references in order to enable validation

<script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>

* Replace <HUB-NAME> and <HUB-FULL-CONNECTION\_STRING> with your settings.
* Add a NotificationHub client. An official client is not ready at the moment so a class NotificationHubApiClient is used as a workaround. See a sample sources to get more details about this class.
* Build and run a web application



Complete sources of the project are in the PushMeNow.AspNet5 sub-folder of sample sources.

## Node.JS Web App

### Create a Node.JS app and connect to the Notification Hub

First step is a creation Node.JS application. Epress.JS framework v4.13.1 is used for this sample.

The detailed step by step guide is available here: <http://expressjs.com/en/starter/generator.html>.

Next step is installation Azure npm package. It is done using the following command:

npm install azure --save

Add the following string to the top of the app.js file of the application:

var azure = require('azure');

In order to connect to the Notification Hub the NotificationHubService object is used. It is created in app.js file:

var notificationHubService = azure.createNotificationHubService('hubname','connectionstring');

### Create pages to display the list of registrations and send notification to different platforms

Create a push.js file in the routes folder and write the following code:

var express = require('express');

var router = express.Router();

module.exports = router;

Add the following strings to the app.js file to configure routes and set up handlers:

var push = require('./routes/push');

app.use('/push', push);

Add the methods for displaying registration list in push.js:

router.get('/', function(req, res, next) {

// getting the registrations list

notificationHubService.listRegistrations(null, function (error, response) {

if (!error) {

var regisrations = response.map(function (item) {

return {

name: getNameByRegistrationTags(item.Tags),

tags: item.Tags,

// identify registration type

registrationType: item.\_.ContentRootElement.replace('RegistrationDescription', '')

};

});

res.render('push', { title: 'Registrations', regisrations: regisrations });

} else {

throw new Error(error);

}

});

});

Add function to get devices by Notification Hub’s registration tags:

function getNameByRegistrationTags(tags) {

try {

return JSON.parse(new Buffer(tags, 'base64').toString('utf8')).name;

}

catch (e) {}

return tags;

}

Add view to display the registration list. All views use the Jade engine. Create a push.jade file in views folder and add the following code to it:

extends layout

block content

h2 #{title}

div

div

- each registration in regisrations

li

a(href='/push/send?tags=#{registration.tags}&registrationType=#{registration.registrationType}') #{registration.name}

Open http://localhost:<port>/push and check you already can see the list of registrations.

Add the new route to send notification to a user in the push.js:

// GET view for sending notification

router.get('/send/', function(req, res, next) {

var tags = req.query.tags;

var name = getNameByRegistrationTags(tags);

res.render('send', { title: 'Send message', name: name, tags: tags, registrationType: req.query.registrationType, message: '' });

});

Create view sand.jade for displaying message control with following code extends layout

block content

h4 Send message to #{name}

form(role='form'

method='post')

input(type='hidden'

name='tags'

value='#{tags}')

input(type='hidden'

name='registrationType'

value='#{registrationType}')

input(class='form-control'

type='text'

name='text'

id='text'

value='#{message}')

input(type='submit'

value='Send'

style='margin: 15px 0 0 0')

div

a(href='/push') Back to list:

Add handler for sending message request and place it in the push.js file:

// POST send notification to use

router.post('/send/', function(req, res, next) {

var message = req.body.text;

var tags = req.query.tags;

var name = getNameByRegistrationTags(tags);

var registrationType = req.query.registrationType;

var payload;

// callback method after finished sending notification

var onSendingCallback = function(error) {

if (!error) {

res.redirect('/push');

} else {

res.render('send', { title: 'Send message', name: name, tags: tags, registrationType: req.query.registrationType, message: message });

}

};

switch (registrationType) {

// sending notification to the Windows platform

case 'Windows':

payload = '<toast><visual><binding template="ToastText01"><text id="1">' + message + '</text></binding></visual></toast>';

notificationHubService.wns.send(tags, payload, 'wns/toast', onSendingCallback);

break;

// sending notification to the Android platform

case 'Gcm':

payload = {

data: {

msg: message

}

};

notificationHubService.gcm.send(tags, payload, onSendingCallback);

break;

// sending notification to the Apple platform

case 'Apple':

payload = {

alert: message

};

notificationHubService.apns.send(null, payload, onSendingCallback);

break;

default:

throw new Error('Unknowwn registration type');

}

});

### Creation Azure Web App and configuration the continuous deployment for Node.JS Web App

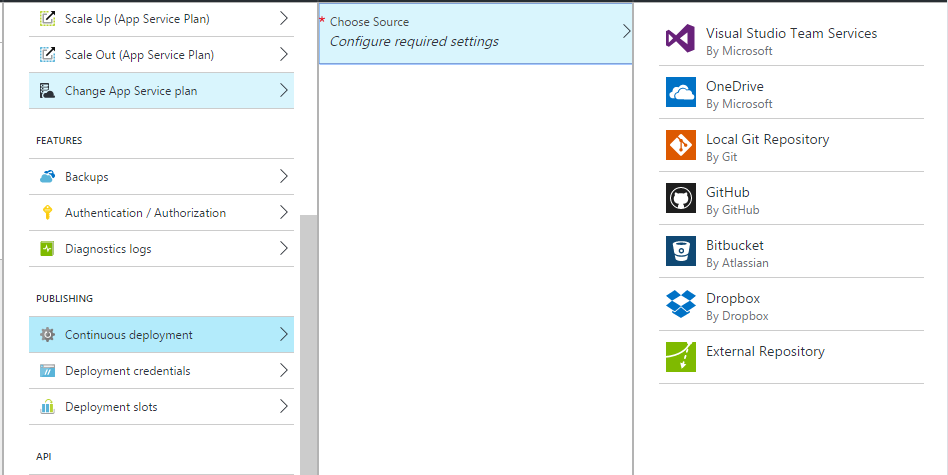
Please follow the steps below:

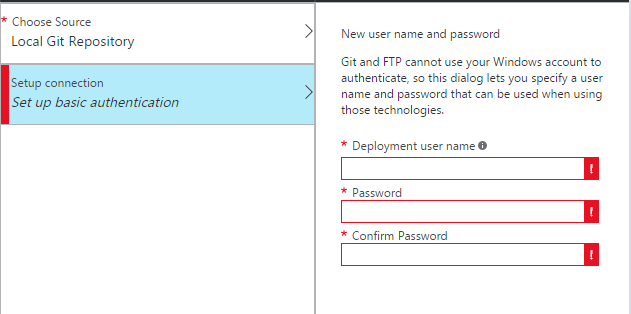
Sign in Azure portal (<https://portal.azure.com/>).

Click on the NEW button then click on the Web + Mobile and then click on the Web App.

Fill necessary fields and click on the Create.

Select this web app and click on the All settings and then click on the Continues deployment and configure settings by Local Git Repository



Click on the “Setup connection” and create deployment account

### Publish Node.JS application to Azure using Git

Use command line change directory to the project directory and enter the following command to initialize Git repository

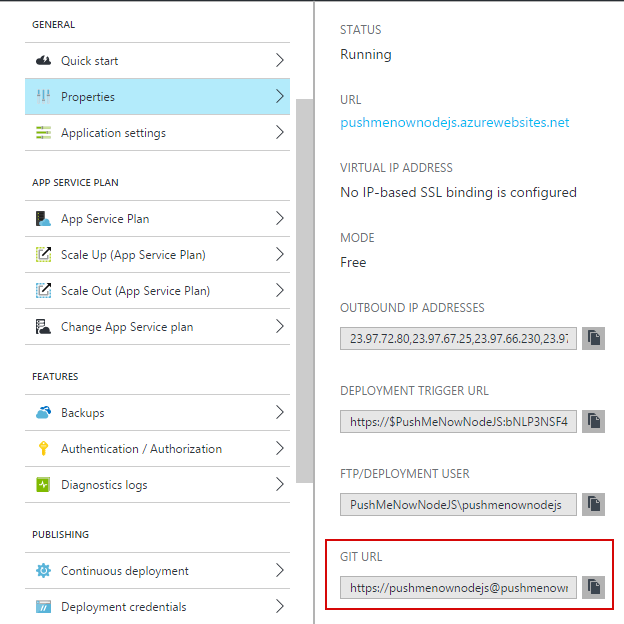
git init

then add files to the repository:

git add .

git commit -m "initial commit"

To get Url for remote repository you open all settings and click on the Properties and you can find “Git URL”



Add a Git remote for pushing updates to the web app that you created previously, by using the following command:

git remote add azure [URL for remote repository]

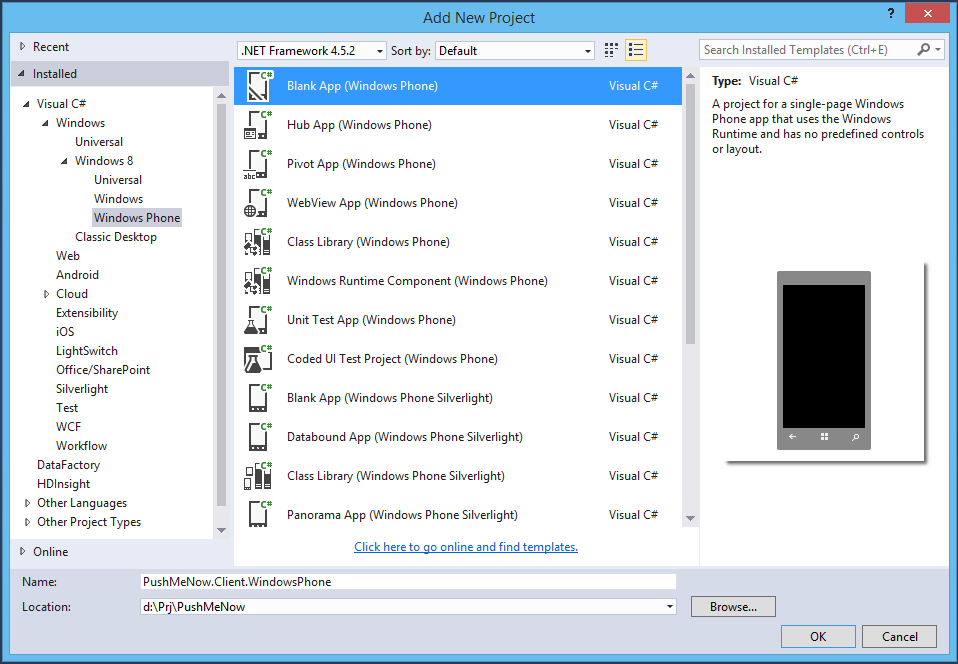
Push your changes to Azure by following command:

git push azure master

## Windows Phone Client

There is a great starter guide available here <https://azure.microsoft.com/en-us/documentation/articles/notification-hubs-windows-store-dotnet-get-started/>. We will follow this guide in the sample.

Create a new Windows Phone project. For Visual Studio 2015 use a template shown on the image below.



Register your application in the Store and tune a notification hub settings as this is described in the article.

Add WindowsAzure.Messaging.Managed and Newtonsoft.Json Nuget packages.

Create a new folder Models and put a new class UserInfo like this:

using Newtonsoft.Json;

namespace PushMeNow.Client.WindowsPhone.Models

{

public class UserInfo

{

private string \_name;

private string \_salt;

[JsonProperty("name")]

public string Name

{

get { return \_name; }

set

{

\_name = value;

int saltLength = 7;

do

{

++saltLength;

\_salt = Guid.NewGuid().ToString("N").Substring(0, saltLength);

} while (AsTag().Contains("="));

}

}

[JsonProperty("s")]

public string Salt

{

get { return \_salt; }

}

public string AsTag()

{

return Convert.ToBase64String(Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(this)));

}

}

}

Note. User information is stored into the tag content as a json containing required information. Some salt data is added to make a tag with same user information unique. The notification registration tag should contain alphanumeric symbols only. So the json is encoded as a base64 string increasing a salt field size to be 8 or more to avoid an ending ‘=’ symbols in the base64 string.

Open a MainPage.xml and add a stackpanel instead of grid. Put a minimum of required UI elements like these:

<StackPanel Margin="20">

<TextBlock HorizontalAlignment="Left"

TextWrapping="Wrap"

Text="Register your device for Push Me Now"

VerticalAlignment="Top"

Style="{StaticResource HeaderTextBlockStyle}" />

<TextBlock Margin="0,50,0,0"

HorizontalAlignment="Left"

TextWrapping="Wrap"

Text="User Name"

VerticalAlignment="Top"

Style="{StaticResource BaseTextBlockStyle}" />

<TextBox x:Name="textBoxName"

HorizontalAlignment="Stretch"

TextWrapping="Wrap"

Text="" VerticalAlignment="Top" />

<Button Margin="50"

x:Name="buttonRegister"

Content="Register"

HorizontalAlignment="Center"

VerticalAlignment="Bottom"

Click="buttonRegister\_Click" />

</StackPanel>

### 

Open a MainPage.xaml.cs and a set of usings:

using Microsoft.WindowsAzure.Messaging;

using PushMeNow.Client.WindowsPhone.Models;

using Windows.Networking.PushNotifications;

using Windows.UI.Popups;

Create an event handler for a registration button:

private async void buttonRegister\_Click(object sender, RoutedEventArgs e)

{

string message;

try

{

IsEnabled = false;

var userInfo = new UserInfo { Name = textBoxName.Text.Trim() };

if (string.IsNullOrWhiteSpace(userInfo.Name))

{

throw new Exception("Please enter your name");

}

string hubName = "<HUB-NAME>";

string listenConnectionString = "<HUB-LISTEN-CONNECTION\_STRING>";

var channel = await PushNotificationChannelManager.CreatePushNotificationChannelForApplicationAsync();

var hub = new NotificationHub(hubName, listenConnectionString);

string tag = userInfo.AsTag();

var result = await hub.RegisterNativeAsync(channel.Uri, new[] { tag });

if (result.RegistrationId == null)

{

throw new Exception("No registration id");

}

message = "Registration successful: " + result.RegistrationId;

}

catch (Exception exception)

{

message = "Registration failed: " + exception.Message;

}

finally

{

IsEnabled = true;

}

var dialog = new MessageDialog(message);

dialog.Commands.Add(new UICommand("OK"));

await dialog.ShowAsync();

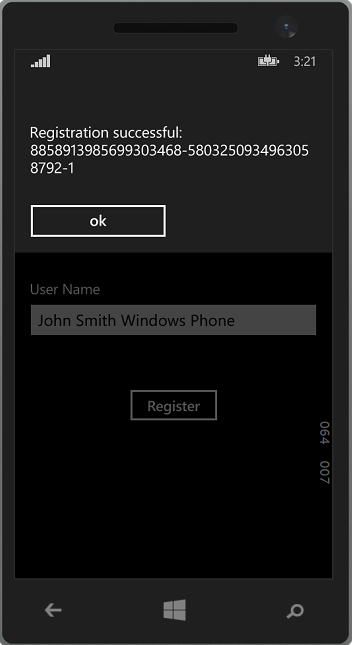
}

Replace <HUB-NAME> and <HUB-LISTEN-CONNECTION\_STRING> with settings for your notification hub.

Double check Toast capable in enabled in the appmanifest.

Build and run an application on Device or Windows Phone Emulator.





Now you can close your application, open a Web Application and send a push message to just registered user.

You will receive a push sent.



## Android Client

Official Notification Hub documentation for Android platform is available here: <https://azure.microsoft.com/en-us/documentation/articles/notification-hubs-android-get-started/>

Class UserInfo handles device registration data and very similar to one described in Windows Phone section. Device registration is performed in a body of AsyncTask:

***private void*** *registerPushNotification() {*

***private final*** *String SENDER\_ID =* ***"<Your project number>"****;*

***private final*** *String HubName =* ***"<Your hub name>"****;*

***private final*** *String HubListenConnectionString =* ***"<Your default listenconnection sctring>"****;*

***final*** *UserInfo userInfo =* ***new*** *UserInfo();*

*userInfo.setName(\_nameEditText.getText().toString());*

***new*** *AsyncTask() {*

*@Override*

***protected*** *Object doInBackground(Object... params) {*

***try*** *{*

*GoogleCloudMessaging gcm = GoogleCloudMessaging.getInstance(MainActivity.****this****);*

*String regid = gcm.register(SENDER\_ID);*

*NotificationHub hub =* ***new*** *NotificationHub(HubName, HubListenConnectionString, MainActivity.****this****);*

*String registrationId = hub.register(regid, userInfo.getAsTag()).getRegistrationId();*

*ToastNotify(****"Registered Successfully - RegId : "*** *+ registrationId);*

*}* ***catch*** *(Exception e) {*

*ToastNotify(****"Registration Exception Message - "*** *+ e.getMessage());*

***return*** *e;*

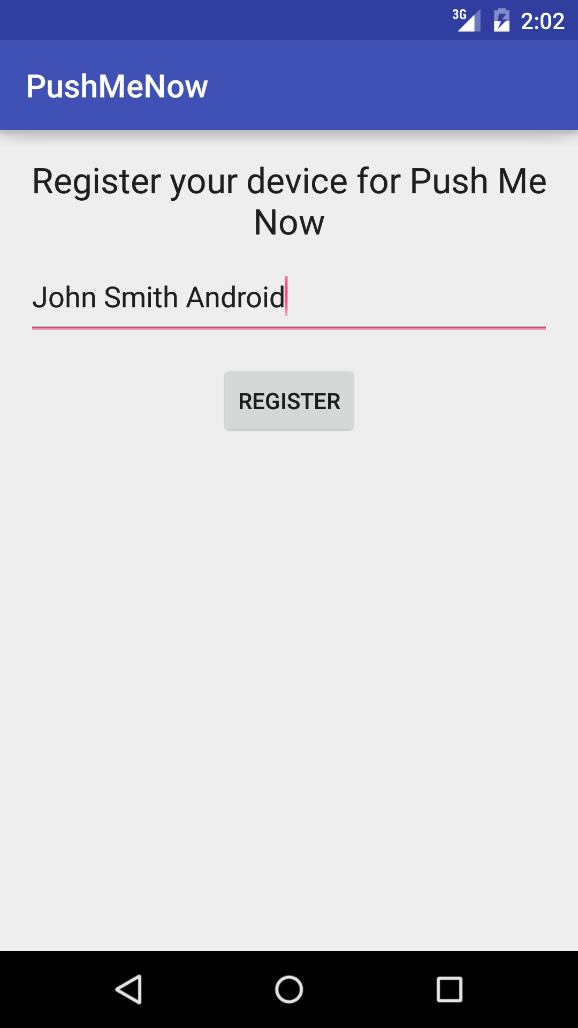
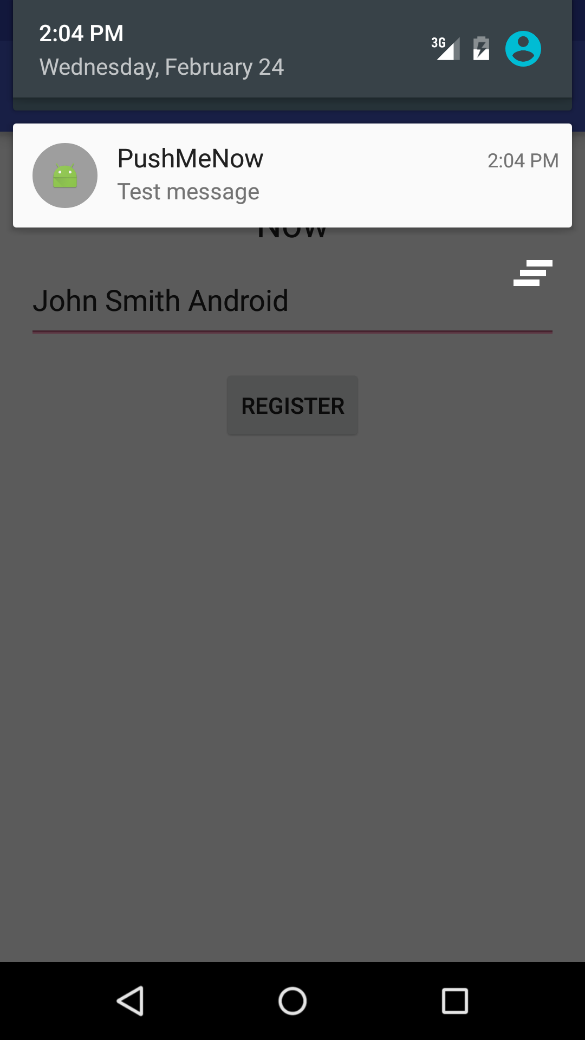
*}*

***return null****;*

*}*

*}.execute(****null****,* ***null****,* ***null****);*

*}*

## iOS Client

Official Notification Hub documentation for Android platform is available here: <https://azure.microsoft.com/en-us/documentation/articles/notification-hubs-ios-get-started/>

Class UserInfo handles device registration data and very similar to one described in Windows Phone section. Device registration is performed in class AppDelegate, because deviceToken is available only in the method application:didRegisterForRemoteNotificationsWithDeviceToken: of class AppDelegate

Register to receive remote notifications via Apple Push Notification service.

-(void) registerPushNotificationForUser:(UserInfo\*) userInfo {

self.userInfo = userInfo;

UIUserNotificationSettings\* notificationSettings = [UIUserNotificationSettings settingsForTypes:UIUserNotificationTypeAlert | UIUserNotificationTypeBadge | UIUserNotificationTypeSound categories:nil];

[[UIApplication sharedApplication] registerUserNotificationSettings:notificationSettings];

[[UIApplication sharedApplication] registerForRemoteNotifications];

}

This code connects to the notification hub. Then it sends the device token to the notification hub to register for notifications:

- (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*) deviceToken {

static NSString\* hubName = @"<Your hub name>";

static NSString\* listenConnectionString = @""<Your default listenconnection sctring>";

SBNotificationHub\* hub = [[SBNotificationHub alloc] initWithConnectionString:listenConnectionString

notificationHubPath:hubName];

NSSet\* tags = nil;

if (self.userInfo) {

tags = [NSSet setWithObjects:self.userInfo.asTag, nil];

}

[hub registerNativeWithDeviceToken:deviceToken tags:tags completion:^(NSError\* error) {

if (error != nil) {

NSLog(@"Error registering for notifications: %@", error);

}

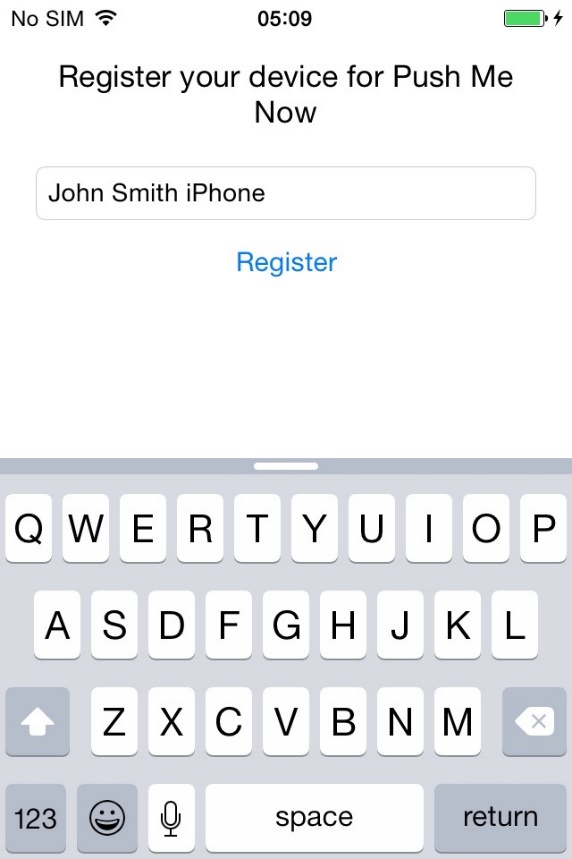
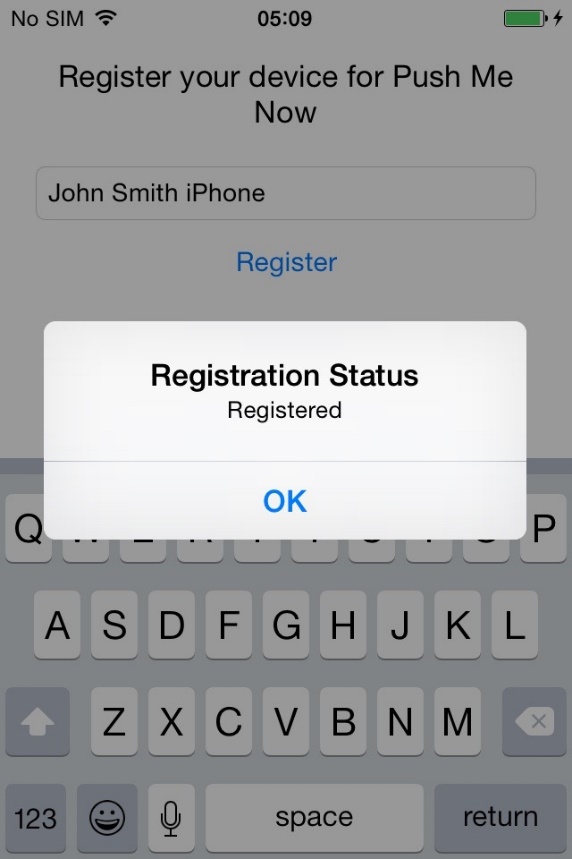
else {

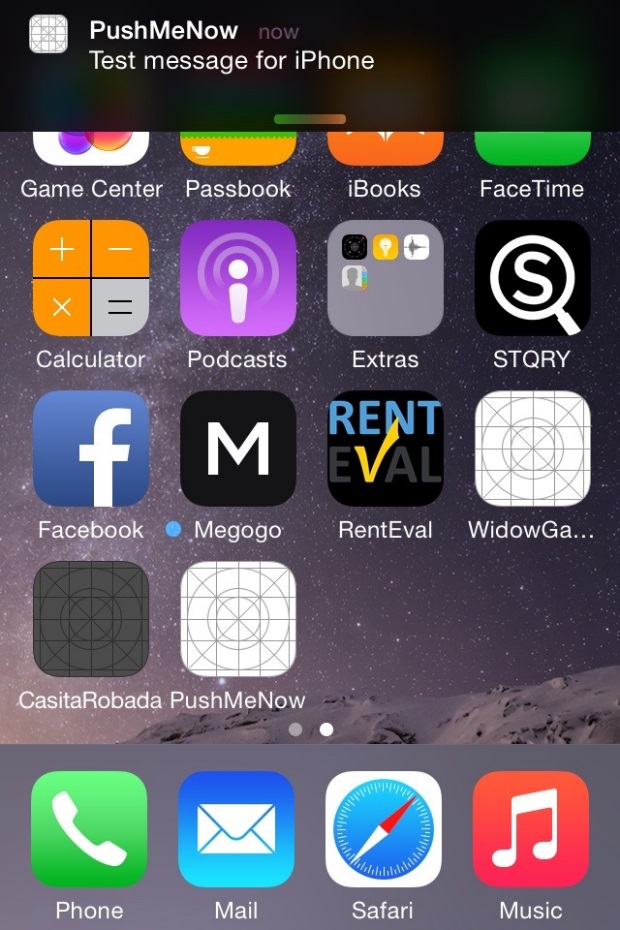
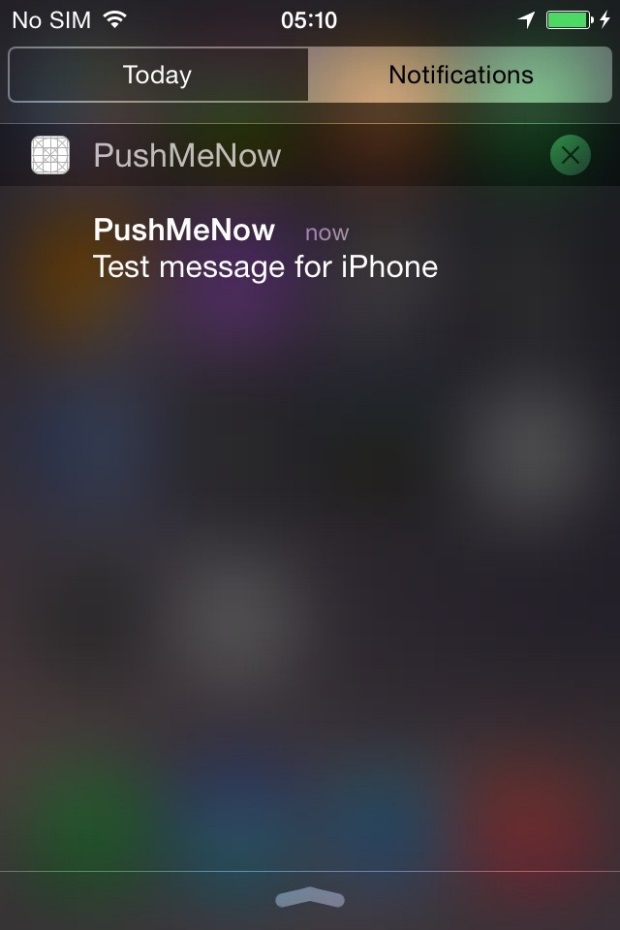
[self MessageBox:@"Registration Status" message:@"Registered"];

}

}];

}

# Summary

In the article we described, how to get started with Azure Notification Hub to build your own push notification solution using Node.JS or .NET Core 1.0 for Android, iOS and Windows phone mobile applications

A startup can just take this sample as a starting point and customize to match business model into a specific application.