Azure Mobile Services - Implementing Push Notifications

This article explains how to implement push notifications on Windows Phone with Azure Mobile Services.

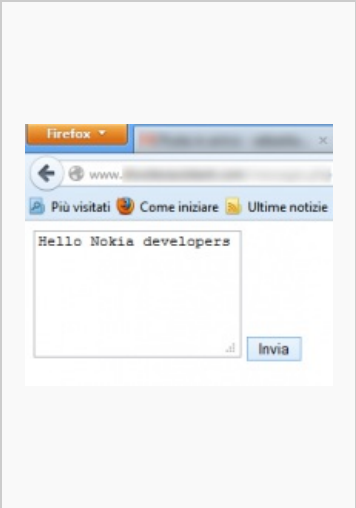
18 Aug 2013

Introduction

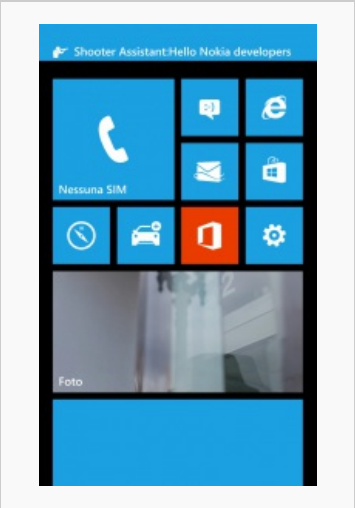
This article explains how to implement push notifications on Windows Phone using Azure Mobile Services, and provides both client code for registering to receive push notifications and also a basic HTML form and PHP script for sending them to all registered devices (the mobile service sends a push notification each time a record is inserted). The article assumes you are familiar with Mobile Services, but if not please first refer to: Azure Mobile Services on Windows Phone.

Implementing push notifications on Azure is straightforward, and the code produced here was written in less than 20 minutes. Sharing the information with you here on the wiki took longer than writing the code!

Note: At time of writing when you send less than 500 messages per user each day, you do not need to register or authenticate your mobile service app with MPNS.



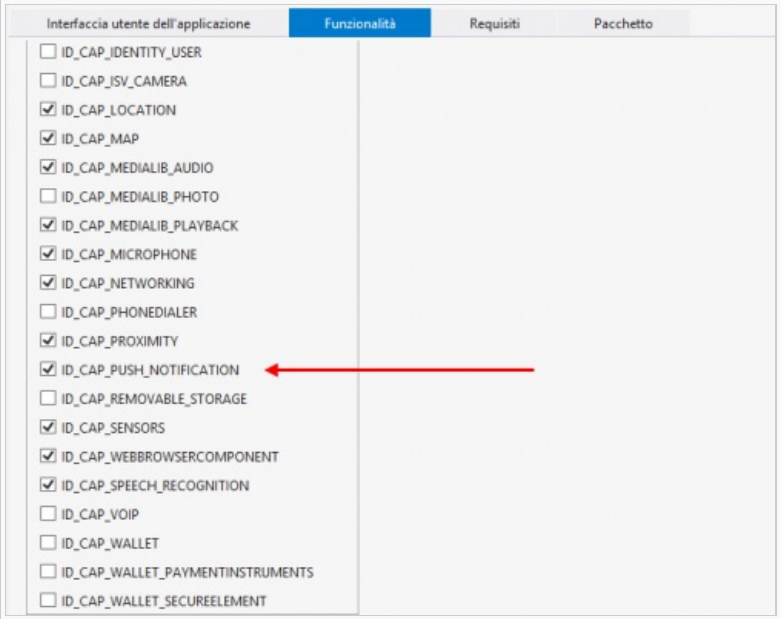
Web form for sending messages



Notification on Windows Phone

Setting Capabilities

May seem trivial, but as a first step we need to set the correct capability to use Push Notifications.

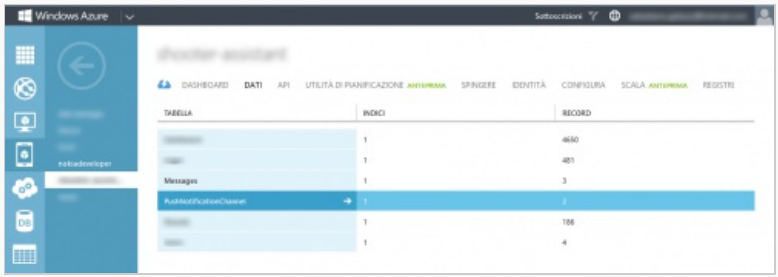


Setting capabilities

Creating Tables

(C) Copyright Nokia 2014. All rights reserved.

Login with your account into Windows Azure Management Portal, select a Mobile Service (previously created) and create two tables: one to manage Push Notification subscriptions (I called `PushNotificationChannel`), the second to manage messages you will send to your users (I called `Messages`).



Creating tables

You need also to create classes on your code to represent the newly created tables.

```
public class PushNotificationChannel
{
        public int Id { get; set; }

        [DataMember(Name = "uri")]
        public string Uri { get; set; }
}

public class Messages
{
        public int Id { get; set; }

        [DataMember(Name = "text")]
        public string Text { get; set; }
}
```

Azure Libraries

Into your **App.xaml.cs** add the namespace

```
using Microsoft.WindowsAzure.MobileServices;
```

and the object to connect to your Mobile Service

```
public static MobileServiceClient MobileService = new
MobileServiceClient("https://nokiadeveloper.azure-mobile.net/", "YOUR KEY HERE");
```

Adding Push Notifications to your app

First of all we need to import the proper namespace:

```
using Microsoft.Phone.Notification;
```

Create classes to represent the one you created

```
public static HttpNotificationChannel CurrentChannel { get; private set; }
```

`HttpNotificationChannel` creates a notification channel between the *Microsoft Push Notification Service* and the Push Client and

(C) Copyright Nokia 2014. All rights reserved.

creates a new subscription for raw notifications.

```
private void AcquirePushChannel()
{
    string PushChannelName = "MyPushChannel";

    CurrentChannel = HttpNotificationChannel.Find(PushChannelName);

    if (CurrentChannel == null)
    {
        CurrentChannel = new HttpNotificationChannel(PushChannelName);

        // Register for all the events before attempting to open the channel.
        CurrentChannel.ChannelUriUpdated += new
EventHandler<NotificationChannelUriEventArgs>(PushChannel_ChannelUriUpdated);
        CurrentChannel.ErrorOccurred += new
EventHandler<NotificationChannelErrorEventArgs>(PushChannel_ErrorOccurred);

        // Register for this notification only if you need to receive the notifications
while your application is running.
        CurrentChannel.ShellToastNotificationReceived += new
EventHandler<NotificationEventArgs>(PushChannel_ShellToastNotificationReceived);

        CurrentChannel.Open();
        CurrentChannel.BindToShellTile();
        CurrentChannel.BindToShellToast();
    }

    if (CurrentChannel != null)
    {
        try
        {
            var channel = new PushNotificationChannel { Uri =
CurrentChannel.ChannelUri.ToString() };
            App.MobileService.GetTable<PushNotificationChannel>().InsertAsync(channel);
        }
        catch (Exception e)
        {
            System.Diagnostics.Debug.WriteLine(e.Message);
        }
    }
}
```

`PushChannelName` is the name of the Push Notification service you want to create. Be sure it is unique.

`HttpNotificationChannel.Find` checks a service registered with the name already exists. If not, it returns `null` so a new `HttpNotificationChannel` object is created and initialized.

As final step we need to register the user device performing an insert into the table `PushNotificationChannel`.

```
var channel = new PushNotificationChannel { Uri = CurrentChannel.ChannelUri.ToString()
};
App.MobileService.GetTable<PushNotificationChannel>().InsertAsync(channel);
```

That's it! now your application is able to receive Push Notification from the created channel. The following code is not mandatory and would be implemented based on your needs.

(C) Copyright Nokia 2014. All rights reserved.

```
void PushChannel_ChannelUriUpdated(object sender, NotificationChannelUriEventArgs e)
{
            Dispatcher.BeginInvoke(() =>
            {
                // Display the new URI for testing purposes.   Normally, the URI would
be passed back to your web service at this point.
                MessageBox.Show(String.Format("Channel Uri is {0}",
e.ChannelUri.ToString()));
            });
}
```

`PushChannel_ChannelUriUpdated` is called each time a new channel is created.

```
void PushChannel_ErrorOccurred(object sender, NotificationChannelErrorEventArgs e)
{
    // Error handling logic for your particular application would be here.
    Dispatcher.BeginInvoke(() => MessageBox.Show(String.Format("A push notification {0}
error occurred.  {1} ({2}) {3}", e.ErrorType, e.Message, e.ErrorCode,
e.ErrorAdditionalData)) );
}
```

`PushChannel_ErrorOccurred` is self-explanatory - called when an error occurs.

```
void PushChannel_ShellToastNotificationReceived(object sender, NotificationEventArgs e)
{
    StringBuilder message = new StringBuilder();
    string relativeUri = string.Empty;

    message.AppendFormat("Received Toast {0}:\n", DateTime.Now.ToShortTimeString());

    // Parse out the information that was part of the message.
    foreach (string key in e.Collection.Keys)
    {
        message.AppendFormat("{0}: {1}\n", key, e.Collection[key]);

        if (string.Compare(
            key,
            "wp:Param",
            System.Globalization.CultureInfo.InvariantCulture,
            System.Globalization.CompareOptions.IgnoreCase) == 0)
            {
                relativeUri = e.Collection[key];
            }
    }

    // Display a dialog of all the fields in the toast.
    Dispatcher.BeginInvoke(() =>  MessageBox.Show(message.ToString()) );
}
```

`PushChannel_ShellToastNotificationReceived`, you should register for this notification only if you need to receive the notifications while your application is running.

Acquiring the channel

(C) Copyright Nokia 2014. All rights reserved.

To acquire the channel just call the `AcquirePushChannel` inside the application's Page Loaded function.

```
private void PhoneApplicationPage_Loaded(object sender, RoutedEventArgs e)
{
    AcquirePushChannel();
}
```

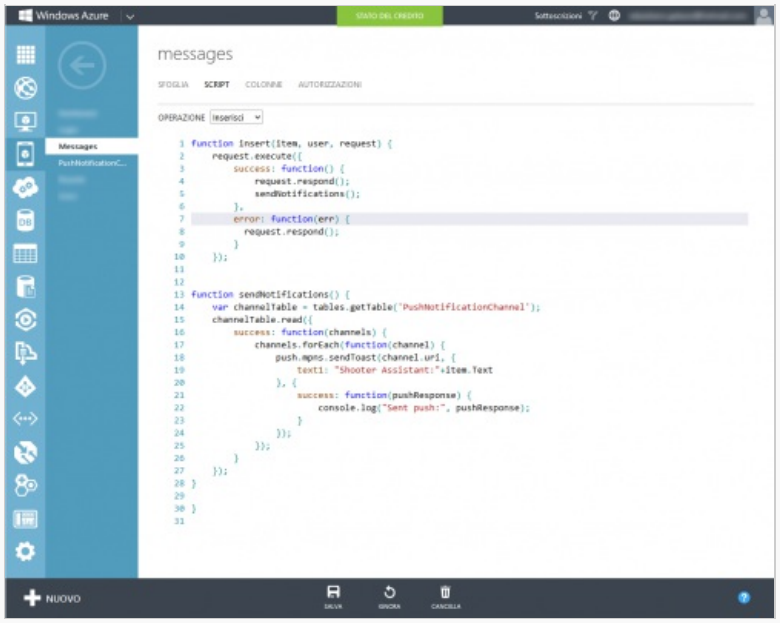
Sending Push Notifications

Above we showed how to receive push notifications. This section shows how to send them. As I prefer "real world use cases" we will send the messages to all registered devices from a simple HTML form with a textarea (for the message).

Because of my knowledge I used PHP, but any server side programming language (e.g. ASP) is good. First we need to update the registered insert script in the **Management Portal**.

Update the registered insert script in the Management Portal

Let's come back into the **Windows Azure Management Portal** and open the **Script** tab of the table "Messages" (created earlier in the article).



Updating insert script

Add the following code.

```
function insert(item, user, request) {
    request.execute({
        success: function() {
            request.respond();
            sendNotifications();
        },
        error: function(err) {
          request.respond();
        }
    });

function sendNotifications() {
    var channelTable = tables.getTable('PushNotificationChannel');
    channelTable.read({
        success: function(channels) {
            channels.forEach(function(channel) {
                push.mpns.sendToast(channel.uri, {
```


(C) Copyright Nokia 2014. All rights reserved.

```
                    text1: "Message:"+item.Text
                }, {
                    success: function(pushResponse) {
                        console.log("Sent push:", pushResponse);
                    }
                });
            });
        }
    });
}

}
```

`push.mpns.sendToast` is the method to send Toast notifications - but you can also to send other types of notifications like Flip Tile.

Your code should something like the snippet below:

```
function insert(item, user, request) {
    request.execute({
        success: function () {
            // Write to the response and then send the notification in the background
            request.respond();
            push.mpns.sendFlipTile(item.channel, {
                title: item.text
            }, {
                success: function (pushResponse) {
                    console.log("Sent push:", pushResponse);
                }
            });
        }
    });
}
```

Now each time a record is inserted into the *Messages* table, it will start the push notification process for all devices registered into table `PushNotificationChannel`. This should have a content as shown below:

Table of devices registered to the service

Server Side Script

Here the very simple PHP code I used to send Push Notifications inserting a new record into `Messages` table.

```
<?php

if( isset($_POST['message']) ) {

$endpoint = "https://YOUR-MOBILE-SERVICE-NAME.azure-mobile.net/tables/Messages";

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $endpoint);
```

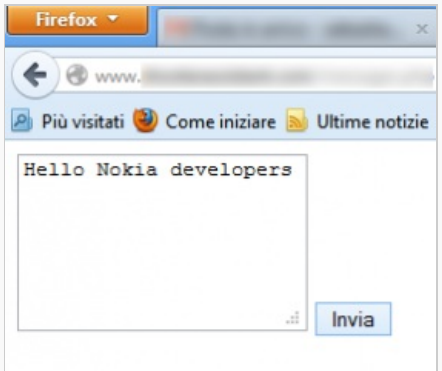

(C) Copyright Nokia 2014. All rights reserved.

```
// Set so curl_exec returns the result instead of outputting it.
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_SSL_VERIFYPEER, false);

curl_setopt($ch, CURLOPT_HTTPHEADER, array(
    'X-ZUMO-APPLICATION: YOUR KEY HERE',
    'Accept: application/json',
    'Content-Type: application/json',
    ));

$json = "{ \"Text\": \"".$_POST['message']."\" }";
echo $json;

curl_setopt($ch,CURLOPT_POSTFIELDS, $json);
// Get the response and close the channel.
$response = curl_exec($ch);
curl_close($ch);

echo $response;
}

?>

<form method="post" action="messages.php">
<textarea name="message"></textarea>
<input type="submit" value="Invia"/>
</form>
```

Without using complex SDK I used Windows Azure REST API to insert a new object record using JSON.

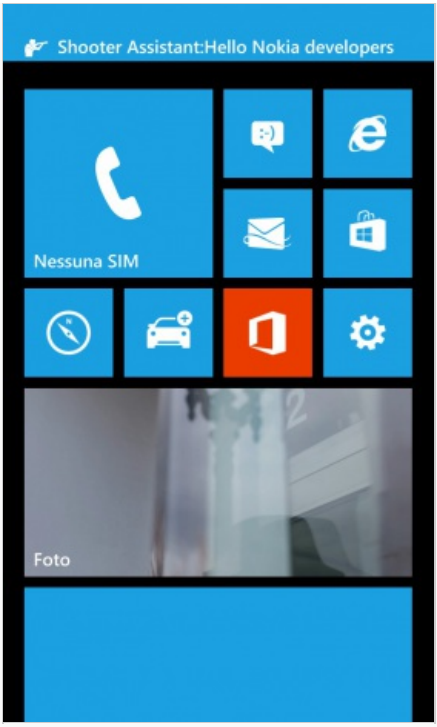
```
$json = "{ \"Text\": \"".$_POST['message']."\" }";
```

That's all! Now from your website you can send a message to everybody.



Form web

(C) Copyright Nokia 2014. All rights reserved.



Notification on Windows Phone

(C) Copyright Nokia 2014. All rights reserved.