Windows Phone Silverlight

Getting Started with Accengage

Version

Current version of Accengage SDK is S1.0.0. This version supports:

- Windows Phone Silverlight 7.1
- Windows Phone Silverlight 8.0
- Windows Phone Silverlight 8.1

For more information about available functionalities, please find the Changelog here.

Summary

Accengage SDK allows you to track the execution and display of In-App and to handle Push notifications of your application.

Accengage SDK is provided as DLLs that your project can reference to at compile-time. It is available via NuGet Package Manager.

An installation package contains the following files/folders:

- Two DLL for each Windows Phone version (7.1, 8.0, 8.1) via NuGet Package Manager
- The source of XAML as an example for notifications customisation

Requirements

To integrate AccengageSDK on Windows Phone, you will need the following requirements:

- Visual Studio 2012 with Windows Phone SDK 7 for Windows Phone 7.1
- Visual Studio 2012/13 with Windows Phone SDK 8 for Windows Phone 8.0
- Visual Studio 2013/15 with Windows Phone SDK 8.1 for Windows Phone 8.1

Login Information

You will also need an Accengage Partner Id and a Private Key.

This information is available for registered customers at www.accengage.com

Integration

> You can download the latest version of the SDK at the following address: AccengageSDK

Import Bma4S SDK DLL into your solution

Start NuGet Packages and look for AccengageSDK. Click on Install to add the SDK reference to your project.

You can use the NuGet Console with the following command:

```
Install-Package AccengageSDK -Version 1.0.0
```

If you are experiencing some difficulties, please see our Troubleshooting section.

Editing Application Configuration

The SDK needs some capabilities to work correctly:

- ID_CAP_IDENTITY_DEVICE
- ID_CAP_IDENTITY_USER
- ID_CAP_NETWORKING
- ID_CAP_PUSH_NOTIFICATION
- ID_CAP_LOCATION
- ID_CAP_WEBBROWSERCOMPONENT
- ID_CAP_PHONEDIALER

Editing your App.xaml.cs

You need to replace the value *partner-id* and *secret-key* with your owns. The service and channel names are optional. See Push Notifications.

In your App.xaml.cs, just add this line in the **Application_Launching** method:

```
Tracker.Start(_rootFrame, @"partner-id", @"secret-key");
```

This way, you will be able to use Tracking services and In-App notifications.

Windows Phone 8+ steps

You need to add the bma4s protocol in your app to be able to extract logs of the SDK from your app. Open your **WMAppManifest.xml** in XML Editor and add the following lines in <Extensions></Extensions>, after all <Extension> tags:

```
<Protocol Name="bma4s" NavUriFragment="encodedLaunchUri=%s" TaskID="_default" />
```

For example:

```
<?xml version="1.0" encoding="utf-8"?>
<Deployment ...>
   <App ...>

      <Tokens>...</Tokens>

      <Extensions>
         ...
         <Protocol Name="bma4s" NavUriFragment="encodedLaunchUri=%s" TaskID="_default"
/>
      </Extensions>

      <ScreenResolutions>...</ScreenResolutions>

   </App>
</Deployment>
```

If you use an UriMapper to process the incoming URI, you need to add the following line in your UriMapper:

```
uri = Tracker.MapUri(uri);
```

For example:

```
class UriMapperWrapper : UriMapperBase
{
    public override Uri MapUri(Uri uri)
    {
        uri = Tracker.MapUri(uri);
        /* Your code */
        return uri;
    }
}
```

> The Accengage Uri Mapper is only for Debug Apps. It is only requested to extract logs and display DeviceID.

Encountering difficulties with our SDK? Please read our Troubleshooting section for more informations.

Testing

In order to test your integration, you can create a segment with your device id and try to activate an In-App on your segment.

Tracking

Events

If you want to track a specific event you can add to your code:

```
Tracker.TrackEvent(1000, "test");
```

Where 1000 is your eventID and “test” is an argument of this event.

Using Analytics

You can track specifics events like “Add to Cart”, “Purchase” and “Lead”. Here is how to use each of these events.

Lead

Send your Lead to A4S Servers:

```
Tracker.TrackLead("Lead Label", "Lead Value");
```

Add to Cart

First, you will need to create an item to track like this one:

```
var item = new Item
{
    Id = "ArticleID",
    Label = "Label",
    Category = "Category",
    Currency = "Currency",
    Price = 12.30,
    Quantity = 1
};
```

Now you just need to specify the cart id and give us the item:

```
Tracker.TrackCartItem("CartId", item);
```

Purchase

You have 2 ways to send a Purchase tracking, with or without Items.

You can create a Purchase to track like this one:

```
Tracker.TrackPurchase("Purchase ID", null, "Purchase Currency", 12.30);
```

Note: Currency should be a valid 3 letters ISO4217 currency (EUR,USD,..) and the last argument is the total price of this purchase.

If you want to add items belonging to this purchase, you can:

```
Tracker.TrackPurchase("Purchase ID", items, "Purchase Currency", 12.30);
```

Note: Don't forget to create each “Item” at first and add them to an “IEnumerable<Item>”.

Update Device Information

You can have a device profile for each device in order to qualify the profile.For example, this will allow the user to opt in for and opt out of some categories of notifications. Device profile is a set of key/value that are uploaded to Accengage server. In order to update information about a device profile, add the following lines:

```
var userFields = new Dictionary<string, string>();
userFields.Add("key", "value");
Tracker.UpdateUserField(userFields);
```

Replace “key” and “value” with the segmentation key name and the value you want to set.

> The keys and values must match Accengage user information field names and values to allow information to be correctly updated.

Date Format

You can send a date using UpdateUserField method.

Your date has to be in the following format: “yyyy-MM-dd HH:mm:ss zzz”

Please use the code below to format your date before sending it to Accengage servers:

```
var formattedDate = Tracker.FormatAccengageDate(dateTime);
userFields.Add("key", formattedDate);
```

Identify each Page

By default, the name of each page is the class name i.e. **MainPage** for **MainPage.xaml**.

You can customize this page name or set different name depending of the place in the XAML Page. For that, you can use this:

```
Tracker.OnNavigatedTo("your-view");
```

Where *your-view* is the name of your view.

Tracking WebBrowser

Just replace the WebBrowser you wish to track events with, the one from the Accengage SDK.

In your code:

```
WebBrowser myWebBrowser = Tracker.WebBrowser();
```

Events

If you want to track a specific event you can do a redirection to the following URL:

```
http://bma4s/event/1000/test
```

Where 1000 is your eventID and “test” is an argument of this event.

Using Analytics

You can track specifics events like “Add to Cart”, “Purchase” and “Lead”. Here is how to use each of these events.

Lead

You can send a Lead event with this link:

```
http://bma4s/event/10/{value}
```

Where 10 is the constant id for add to cart event and {value} is the following JSON template:

```
{
    "value":"leadValue",
    "label":"leadLabel"
}
```

Add to Cart

You can send a Add to Cart event with this link:

```
http://bma4s/event/30/{value}
```

Where 30 is the constant id for an add to cart event and {value} is the following JSON template:

```
{
    "cartId":"cartId",
    "quantity":1,
    "category":"catItem",
    "label":"labelItem",
    "price":23,
    "articleId":"idItem",
    "currency":"EUR"
}
```

Purchase

You can send a Purchase event with this link:

```
http://bma4s/event/50/{value}
```

Where 50 is the constant id for purchase event and {value} is the following JSON template:

```
{
   "purchaseId":"777",
   "items":[
      {
         "id":"idItem1",
         "quantity":1,
         "category":"cat1",
         "label":"item1",
         "price":23,
         "currency":"EUR"
      },
      {
         "id":"idItem2",
         "quantity":2,
         "category":"cat2",
         "label":"item2",
         "price":50,
         "currency":"EUR"
      }
   ],
   "totalPrice":123,
   "currency":"EUR"
}
```

Update Device Information

When you want to use the updateDeviceInfo function inside of your WebBrowser, you have to use the following link:

```
http://bma4s/updatedeviceinfo/[{"key":"key1","value":"vaue1"},{"key":"key2","value":"v
aue2"}]
```

Identify each Page

You can identify a Page viewed in the WebBrowser with the following link:

```
http://bma4s/view/your-view
```

Where *your-view* is the name of your view.

Deep Linking

Define custom URI Scheme

You can define an URI Scheme to be able to use deeplinking in your app.

Open your **WMAppManifest.xml** in XML Editor and add the following lines in <Extensions></Extensions> and after all <Extension> tags:

WMAppManifest.xml

```
<Extensions>
  <Protocol Name="your-uri-scheme" NavUriFragment="encodedLaunchUri=%s"
TaskID="_default" />
</Extensions>
```

After that, you need to listen for the URI. To use a URI mapper class like this one in your app, assign it to the frame of the app in the **App.xaml.cs** file. In the **InitializePhoneApplication** method, just after where **RootFrame.Navigated** is assigned, assign the **RootFrame.UriMapper** property to your URI mapper class. In the following example, the **AssociationUriMapper** class is assigned to the frame's UriMapper property:

App.xaml.cs

```
void Application_Launching(object sender, LaunchingEventArgs e)
{
    _rootFrame.UriMapper = new AssociationUriMapper();
    // Accengage Tracker launch
    Tracker.Start(_rootFrame, "PartnerId", "SecretKey", "MpnsServiceName",
"MpnsChannelName");
}
```

Then, use the URI Scheme to redirect to the right Page:

AssociationUriMapper.cs

```
class AssociationUriMapper : UriMapperBase
{
    private string tempUri;
    public override Uri MapUri(Uri uri)
    {
        #if DEBUG
        uri = Tracker.MapUri(uri);
        #endif
        tempUri = System.Net.HttpUtility.UrlDecode(uri.ToString());
        if (tempUri.Contains("your-uri-scheme:ShowProducts?CategoryID="))
        {
            // Get the category ID (after "CategoryID=").
            int categoryIdIndex = tempUri.IndexOf("CategoryID=") + 11;
            string categoryId = tempUri.Substring(categoryIdIndex);
            // Map the show products request to ShowProducts.xaml
            return new Uri("/ShowProducts.xaml?CategoryID=" + categoryId,
UriKind.Relative);
        }
        // Otherwise perform normal launch.
        return uri;
    }
}
```

Retrieving Push Custom Parameters

When the user clicks on a notification, the current page is launched or the application is started (depending on whether your application is already started or not). You can manage Custom Params in the Navigated event of the RootFrame:

App.xaml.cs

```
_rootFrame.Navigated += OnRootFrameNavigated;
```

And your OnRootFrameNavigated:

App.xaml.cs

```
 private static void OnRootFrameNavigated(object sender, NavigationEventArgs e)
{
    if (e.NavigationMode > NavigationMode.Refresh)
        return;
    var page = e.Content as Page;
    if (page != null)
    {
        var queryString = page.NavigationContext.QueryString;
        if (queryString.Count > 0)
        {
            string customParam1Value;
            if (queryString.TryGetValue("customParam1Key", out customParam1Value))
            {
                // process custom param "customParam1Key"
            }
        }
    }
}
```

Retrieving In-App Custom Parameters

When an In-App is displayed, clicked or closed, some Custom Parameters are available. To listen to these Custom Parameters, you can register them to these events:

```
Tracker.InAppNotificationDidAppear += Tracker_InAppNotificationCustomParameters;
Tracker.InAppNotificationClicked += Tracker_InAppNotificationCustomParameters;
Tracker.InAppNotificationClosed += Tracker_InAppNotificationCustomParameters;
```

And in the **Tracker_InAppNotificationCustomParameters** function:

```
private void Tracker_InAppNotificationCustomParameters(object sender,
EventArgs<Dictionary<string, string>> e)
{
    Dictionary<string, string> customParams = e.Parameter;
    if (customParams == null)
        return;
    string customParam1Value;
    if (customParams.TryGetValue("customParam1Key", out customParam1Value))
    {
        // process custom param "customParam1Key"
    }
}
```

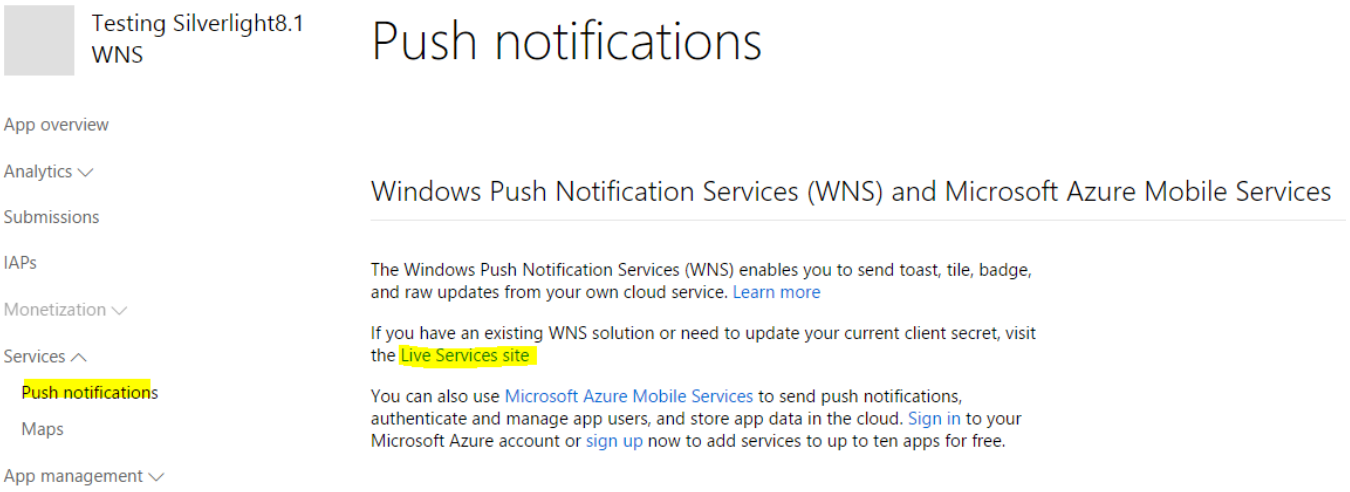
WNS Notifications Setup

Set up application in the Windows Dev Center

To be able to receive Toast and Tiles Notifications in your application using WNS, you need to set up your application in the Windows Dev Center to get your identifiers.

If your app is not already registered, you need to follow How to authenticate with the Windows Push Notification Service (WNS) on MSDN.

Once your application is registered in the Windows Dev Center, go to your Windows Dev Center Dashboard and select your application. Then go to **Services** > **Push Notifications** and find **Live Services site**:



You will need **Package SID**, **Client Secret** and **Application Identity** for later:

Package SID:
ms-app://s-1-15-2-364713165-1003059220-3599444935-2451662442-1598661904-3010795756-2285161735
Link to different app

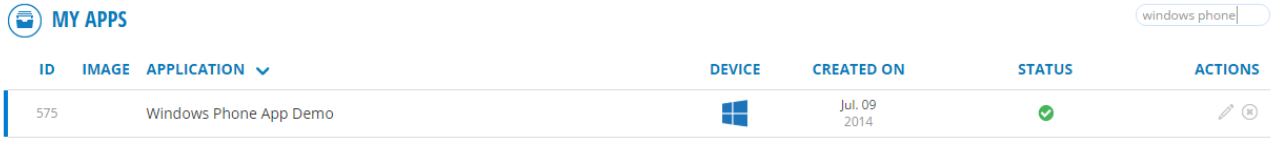
Application identity:
<Identity
Name="**Accengage.TestingSilverlight8.1WNS**"
Publisher="**CN=F8CFEDA6-B930-4C9B-73BD-B42EFDB7F437**" />

Client ID:
00000000591530F9

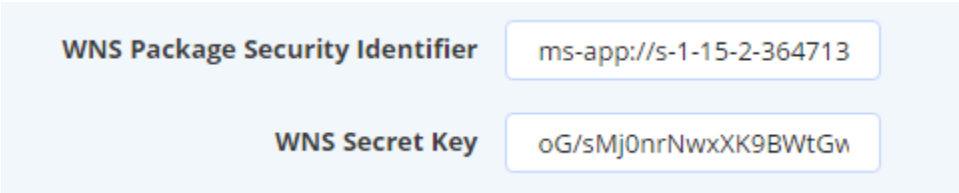
Client secret:
oG/sMg0nrNwxXT9BWtGvXSirs3TQHtrZ

Configure WNS with Accengage User Interface

1. Go to Accengage User Interface, and go to **Settings** > **Manage Application** and find your application:



2. Edit your application's settings, and fill the WNS Package Security Identifier box with your own **Package SID** and the WNS Secret Key with your own **Client Secret** from the first part.



Editing your application

Ensure that your *WMAppManifest.xml* is set to WNS.

Go to your *Package.appxmanifest* and check the following elements:

- **Internet (Client & Server)** capability is checked
- **Location** capability is checked
- In **Application** tab, Toast capable is set to **Yes**

> Please note that Accengage SDK will take care of all WNS events (including Registration, retrieving Token, …). You don't need to handle anything in your code.

Then, you need to associate your application with the application in the Windows Dev Center.

- First solution, right click on your project and select **Store** > **Associate App with the Store**. Visual Studio will automatically edit your manifest to match your application in the Windows Dev Center configuration.
- Second solution, you can manually edit your **Package.appxmanifest**. Right click on the file and select **View Code**. Then, replace the <Identity> tag with the Application Identity from the first part. For example:

```
<Identity Name="Accengage.TestingSilverlight8.1WNS"
Publisher="CN=F8CFEDA6-B930-4C9B-73BD-B42EFDB7F437" Version="1.0.0.0"/>
```

MPNS Notifications Setup

Generate MPNS certificate

You will need OpenSSL in order to generate your push certificates.

Generate a Private Key (RSA)

You need to generate a new private key (RSA). Do not forgot to securely store your password:

```
C:\OpenSSL\bin>openssl genrsa -out MpnsPrivateKey.key -passout pass:yourownpassword
-des3 2048
```

You should see:

```
Generating RSA private key, 2048 bit long modulus
.....................+++
.....................................+++
```

Generate a CSR

The CSR will be needed by your certificate provider to generate your security certificate. Execute the following line:

```
C:\OpenSSL\bin>openssl req -new -out CertificateSignRequest.csr -key
MpnsPrivateKey.key -passin pass:yourownpassword -config ..\openssl_accengage.cnf
```

Fill each option with the answer you want. The important part is the Common Name (CN). The CN must be unique to your domain and must be used as the MPNS Service Name for the Accengage SDK initialization.

```
Loading 'screen' into random state - done
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FR
State or Province Name (full name) [Some-State]:France
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Accengage
Organizational Unit Name (eg, section) []:
Common Name (eg, YOUR name) []:mpns.accengage.com
Email Address []:
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

You will generate a CSR named **CertificateSignRequest.csr** with your private key and using the previous custom configuration.

Put the CN as the MPNS Service Name.

Get the certificate from the generated CSR

When the CSR is generated, you will be able to choose your certificate provider. You cannot choose the provider that you want.

If you want to use push notifications with Windows Phone 7 and more, you need to choose a certificate with one of the following root certificate:

Approved Windows Phone 7.1 SSL Root Certificate

And for Windows Phone 8 and more:

Approved Windows Phone 8 SSL Root Certificate

Provide the previous generated CSR when ordering your certificate.

Convert certificate to PEM files

When you have your certificates files, two options:

- You have PEM files: go to the next section.
- You have other formats than PEM: you need to convert them into PEM formatted files.

Go to SSL Shopper and find the OpenSSL command to convert your files in PEM format. For example, if you have a DER file, you will need to execute:

```
openssl x509 -inform der -in certificate.cer -out certificate.pem
```

Create PEM files

When you have all PEM files, you need to create two others files with the content of PEM files.

- For Microsoft Dev Center PEM file, named **MicrosoftCertificate.cer**, copy/paste files content in this specific order:
    a. Your Web Server Certificate
    b. (Intermediate Certificate 1)
    c. (Intermediate Certificate 2)
    d. Root Certificate
- For Accengage PEM file, named **AccengageCertificate.cer**, copy/paste files content in this specific order:
    a. Your Private Key
    b. Your Web Server Certificate
    c. (Intermediate Certificate 1)
    d. (Intermediate Certificate 2)
    e. Root Certificate

You should see a similar output:

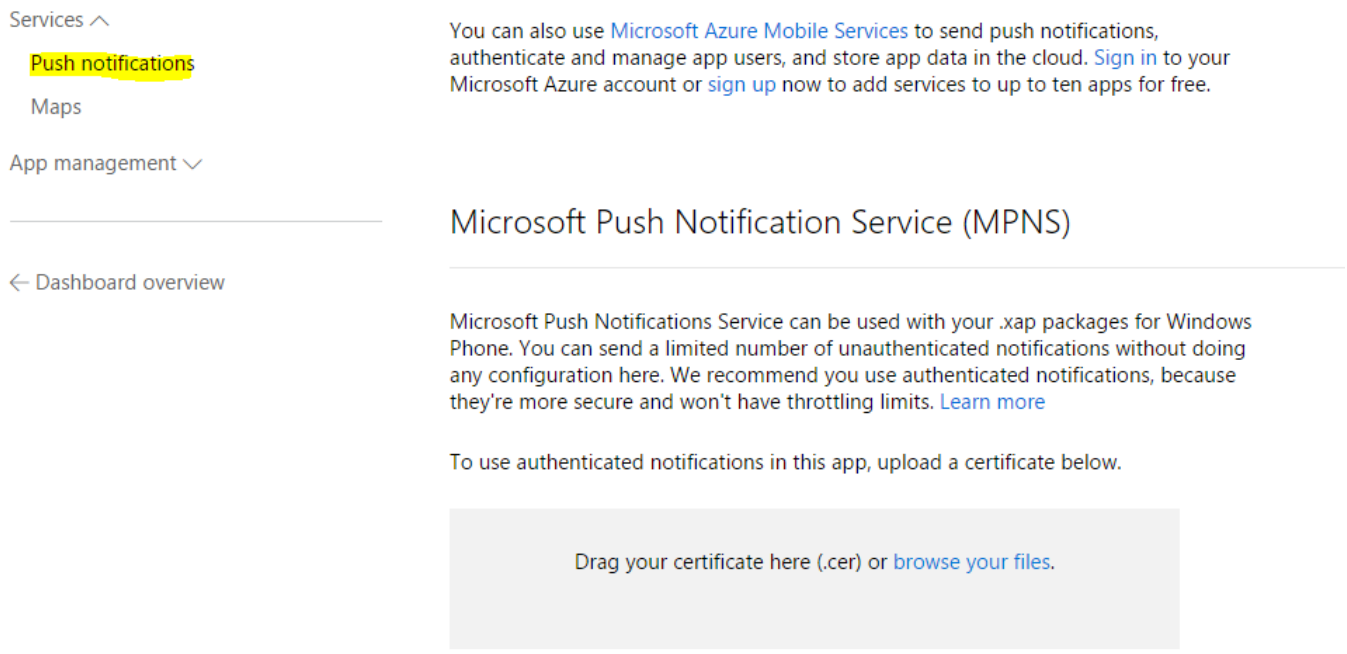
```
-----BEGIN CERTIFICATE-----
MIIFADCCA+igAwIBAgIQE5brJRm1RBnXqPiO24PGkzANBgkqhkiG9w0BAQsFADBE
                              ...
SdZjyY+wYeaH5krDdTE4PB6n4cgQ108A1Ssm+NmfWFo7bYZe
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIErzCCA5egAwIBAgIQXXL7M3Yg9kxygNvpEoH/ajANBgkqhkiG9w0BAQsFADCB
                              ...
gZmdQfLnxVcFDhnKr0I5H6cnXuAKF7iuR6uS8YoE3zDgu0+K+RuITwO0JXp43i59
KdEx
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
MIIERTCCA66gAwIBAgIQM2VQCHmtc+IwueAdDX+skTANBgkqhkiG9w0BAQUFADCB
                              ...
95OBBaqStB+3msAHF/XLxrRMDtdW3HEgdDjWdMbWj2uvi42gbCkLYeA=
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAydYGWO0Oq1ZQrjea/NbK/LbrXkex6rEtHgzkq9Xf/zawCDD+
                              ...
8rHRAQUtbnQZ+/JODhbEZKOSmsEjvoMjtUJyxFziS5Ir1dGET4yM7A==
-----END RSA PRIVATE KEY-----
```

MicrosoftCertificate.cer and **AccengageCertificate.cer** will be used in next sections.

Configure MPNS with Microsoft Dev Center

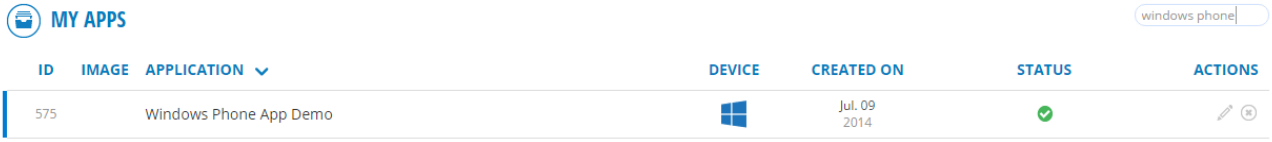
Go to Windows Dev Center Dashboard and register your application if it is not already done.

Once your application is registered in the Windows Dev Center, go to your Windows Dev Center Dashboard and select your application. Then go to **Services** > **Push Notifications**. Just drag and drop your **MicrosoftCertificate.cer** in the dedicated area:

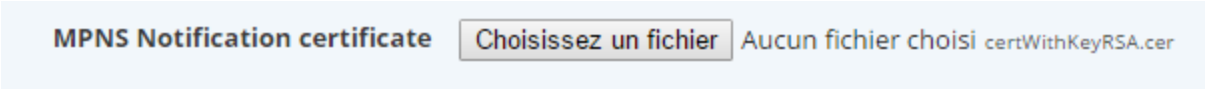


Configure MPNS with Accengage User Interface

1. Go to Accengage User Interface, and to **Settings** > **Manage Application** and find your application:



2. Edit your application's settings, and upload your own certificate in **MPNS Notification certificate** (**AccengageCertificate.cer** creation is explained in previous sections of this documentation):



Editing your application

Ensure that your **WMAppManifest.xml** is set to **MPN**.

In your **App.xaml.cs**, just add this line in the **Application_Launching** method:

```
Tracker.Start(_rootFrame, @"partner-id", @"secret-key", @"mpns-service-name");
```

Where **mpns-service-name** is the **Common Name (CN)** found in the certificate's Subject value. Example: **www.accengage.com**

Please check that **ID_CAP_PUSH_NOTIFICATION** is set in your application capabilities.

If you already use MPNS, you can provide the existing channel name.

```
Tracker.Start(_rootFrame, @"partner-id", @"secret-key", @"mpns-service-name",
@"mpns-channel-name");
```

Push Notifications

Enable/Disable Accengage Pushs

You can enable or disable Accengage notifications with the following code:

```
Tracker.DoNotUsePush = true; // Disable Pushs
Tracker.DoNotUsePush = false; // Enable Pushs
```

If you need to know if push notifications are enabled or disabled, use:

```
if (Tracker.DoNotUsePush)
{
    // Pushs disabled
}
else
{
 // Pushs enabled
}
```

Add Tile Images Hosts

In Windows Phone 7.1 and Windows Phone 8.0, you need to specify allowed hosts used by images in tiles. If you only use the Accengage User Interface to send push notifications, you can skip this section.

Otherwise, you need to specify your own hosts with the following lines:

```
if (Tracker.DefaultTileImageHosts == null || Tracker.DefaultTileImageHosts.Count == 0)
{
    // Add custom hosts
    Tracker.DefaultTileImageHosts = new List<Uri>();
    Tracker.DefaultTileImageHosts.Add(new Uri("http://static.mydomain.com"));
}
else
{
    // Hosts already set, nothing to do
}
```

In-App Notifications

The AccengageTracking SDK also allows you to display In-App notifications. There is no additional code to write to support In-App notifications. There are five different In-App notifications:

- Text: an In-App notification that will appear as a view with a height of 50. It will contain a title and a body text.

- WebView: an In-App notification that will appear as a view with a height of 50. It will contain a WebBrowser that will display the content of a URL
- Text Interstitial which will appear in full screen. It will contain a title and a body text.
- WebBrowser Interstitial: which will appear in full screen. It will contain a WebBrowser that will display the content of a URL and which will be clickable.
- WebBrowser Interstitial with NavBar: which will appear in full screen. It will contain a WebBrowser that will display the content of a URL. It will also display a navigation bar and enable browsing.

You can customize the behaviour and appearance of these In-App notifications.

For more details, please check: Advanced In-App

Advanced In-App

In-App display customization

If you want to display an In-App notification in a specific position, you just have to implement the **IElementsForTemplates** interface in your behind code of the page. The implementation should return a **IEnumerable<KeyValuePair<string, FrameworkElement»** that represents a list of elements. These elements are defined by a regex to match a template name and a **FrameworkElement** in which is placed the In-App notification:

```
public sealed partial class MainPage : IElementsForTemplates
{
    ...
    public IEnumerable<KeyValuePair<string, FrameworkElement>>
Bma4SElementsForTemplates()
    {
        return new Dictionary<string, FrameworkElement>
        {
            {"(.+)Banner", BannerPresenter}
        };
    }
}
```

If you want to use your own template, you have to add it in the Accengage user interface, create a new xaml in the namespace *Bma4S.Templates* and implement needed interface elements in the code behind.

Let's take an example: Here, we created a new template in our project: **MyTemplate.xaml** with the associated file **MyTemplate.xaml.cs**. The xaml can be based on existing templates:

```
<Grid
    x:Class="Bma4S.Templates.MyTemplate"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    Height="136"
    VerticalAlignment="Bottom"
    Background="Black">

    <phone:WebBrowser
        x:Name="WebBrowser" />

    <Button
        x:Name="ActionButton"
        Margin="-12"
        BorderThickness="0"
        Opacity="0" />

    <Button
        x:Name="CloseButton"
        HorizontalAlignment="Right"
        VerticalAlignment="Top"
        Background="{StaticResource PhoneBackgroundBrush}"
        Content="X"
        Foreground="{StaticResource PhoneForegroundBrush}" />

</Grid>
```

And the code behind:

```
public partial class MyTemplate
{
    partial void OnInitializeComponent();

    public MyTemplate()
    {
        InitializeComponent();
        OnInitializeComponent();
    }
}
```

Next, implement needed interface elements:

- **ITemplateBrowsable** if the template contains a WebBrowser. This interface is **mandatory**.
- **ITemplateProgressable** if you want a progress bar during the loading of the WebBrowser
- **ITemplateClosable** if you want a close button to close the In-App
- **ITemplateActionable** if you want to specify a specific action when the user clicks on the In-App

For example:

```
public partial class MyTemplate : ITemplateActionable, ITemplateClosable,
ITemplateBrowsable
{
    partial void OnInitializeComponent();

    public MyTemplate()
    {
        InitializeComponent();
        OnInitializeComponent();
    }

    public WebBrowser Bma4SWebBrowser
    {
        get { return WebBrowser; }
    }

    public Button Bma4SCloseButton
    {
        get { return CloseButton; }
    }

    public Button Bma4SActionButton
    {
        get { return ActionButton; }
    }
}
```

If you want to customize the LandingPage, please see: Customizing Interstitial

Prevent In-App notifications display

If you don't want to display In-App messages temporarily, use:

```
Tracker.DoNotUseInAppNotification = true;
```

This function allows you to prevent In-App visual elements to be displayed. Default value is false which means that In-App display is enabled.

> Once In-App is locked, no In-App visual element will be displayed until unlocked. Please use this function with care!

Customizing Interstitial

When you setup a Landing Page in the Accengage User Interface, you can choose to open this Landing Page in a WebBrowser.

In this case, you can customize this Landing Page template that we will call Interstitial.

Like In-App Notifications, you can customize them.

If you want to use your own template, you have to add it in the Accengage User Interface and create a new XAML file with the same name as the value you specified.

In addition to In-App customizations, you can add a NavBar in your custom template. With this NavBar, you can give the user access to certain controls of the WebBrowser like back/forward page, open in Internet Explorer and Refresh.

For that, just add the following custom control in your XAML:

```
<Grid
    ...
    xmlns:controls="clr-namespace:Bma4S.Controls"
    ... >
    <controls:WebBrowserNavBar x:Name="WebBrowserNavBar" WebBrowser="{Binding
Bma4SWebBrowser}" />
    ...
</Grid>
```

You need to implement the **ITemplateBrowsable** interface to provide a WebBrowser to the NavBar.

Troubleshooting

Restrict SDK connection

You can temporarily restrict connection in order to tell our SDK to stop flushing network requests:

```
 Tracker.NetworkRestriction = Tracker.NetworkEnum.DoNotUseNetwork;
```

You can restrict connection depending of the network used (3G/4G/Wifi):

```
Tracker.NetworkRestriction = Tracker.NetworkEnum.OnlyWiFi4G3G;
Tracker.NetworkRestriction = Tracker.NetworkEnum.OnlyWiFi4G;
Tracker.NetworkRestriction = Tracker.NetworkEnum.OnlyWiFi;
```

All requests will be locked until you use:

```
Tracker.NetworkRestriction = Tracker.NetworkEnum.UseNetwork;
```

In order to know if connection is restricted for our SDK, please use:

```
NetworkEnum restrictionLevel = Tracker.NetworkRestriction;
```

Set cache delay

If you want SDK network requests to be executed more or less often, you can set a custom delay for our cache system:

```
Tracker.QueueWaitTime = 15;
```

Where 15 is the delay in seconds before each execution of cached requests.

Enable/Disable Geolocation

You can enable or disable SDK Geolocation whenever you want with the following line:

```
Tracker.DoNotUseAutoGeoLoc = true; // Stop Geolocation and disable it for upcoming
application starts
Tracker.DoNotUseAutoGeoLoc = false; // Start Geolocation immediately and enable it for
upcoming application starts
```