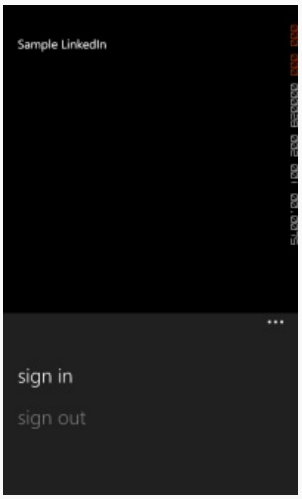
LinkedIn authentication and own basic profile view using Hammock

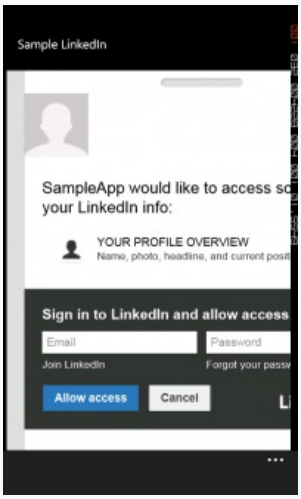
This article explains how to make a LinkedIn OAuth 1.0 authentication process in the Windows Phone application using Hammock library. It will also help you in fetching and then viewing your own basic profile information.

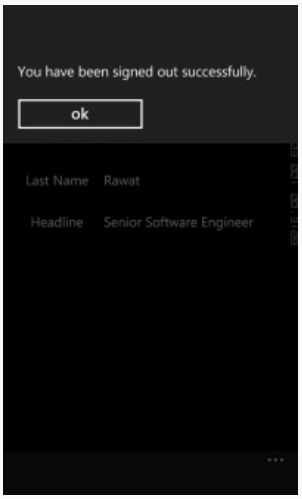
Introduction

LinkedIn API uses OAuth 1.0 authentication process. In this article, we'll use Hammock library to implement OAuth functionality.

The screens in this project are as shown below.







Note: As you can see in Image # 2, our app is asking the user to access his/her basic profile only. If you want you may access full profile as well. To read more about it, you may refer this link

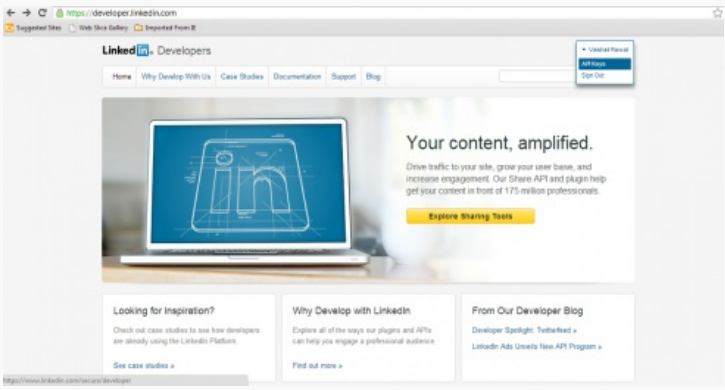
Prerequisites

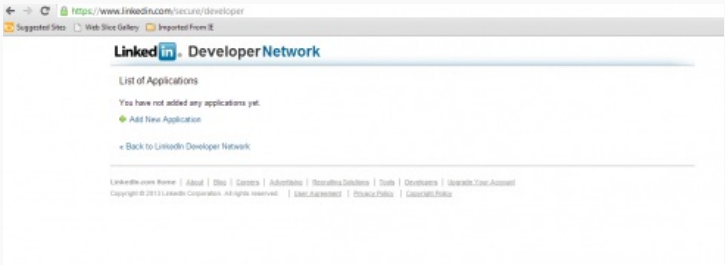
- Windows Phone development environment
- Hammock library

Registering App in LinkedIn

The first step is to register our application at LinkedIn Developer Site. After successfully login, please visit '*API Keys | Add New Application* section' and fill the details of the application.

(C) Copyright Nokia 2014. All rights reserved.





Click to enlarge the screenshots for the above steps.

After registering the app, we will be provided with the Consumer key and Consumer Secret Key. We need to save the two keys somewhere as we will require them later in the project.

Installing Hammock Library

Please read this article which explains how to install Hammock library in WP project.

Creating UI

 (C) Copyright Nokia 2014. All rights reserved.

We will just make one screen which will have two menu items named *Sign In* and *Sign Out*. When a user will press *Sign In* button, a web browser will be shown which will be redirected to the LinkedIn login page. After successful authentication, we will hide the web browser control and will make a server hit to fetch current profile information by using LinkedIn Profile API. After getting successful response, we will parse the response string and display the **first name, last name and headline of the user's profile**. So, the xaml code would be like:

```
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <Grid.RowDefinitions>
                <RowDefinition Height=".10*"/>
                <RowDefinition Height="*"/>
            </Grid.RowDefinitions>

            <phone:WebBrowser Grid.Row="0" Grid.RowSpan="2" Margin="-6,3,0,1"
Name="loginBrowserControl" Visibility="Collapsed"
                              Navigated="loginBrowserControl_Navigated"
Navigating="loginBrowserControl_Navigating"
                              IsScriptEnabled="True"/>

          <TextBlock Grid.Row="0" x:Name="txtWelcomeText" Visibility="Collapsed"
VerticalAlignment="Top" HorizontalAlignment="Center" FontSize="26" FontFamily="Segoe WP
Bold" Foreground="Red"/>

            <Grid x:Name="UserPanel" Grid.Row="1" Visibility="Collapsed">
                <Grid.RowDefinitions>
                    <RowDefinition Height=".20*"/>
                    <RowDefinition Height=".20*"/>
                    <RowDefinition Height=".35*"/>
                    <RowDefinition Height="*"/>
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width=".50*"/>
                    <ColumnDefinition Width="*"/>
                </Grid.ColumnDefinitions>

                <TextBlock Grid.Row="0" Grid.Column="0" Text="First Name"
VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontFamily="Segoe
WP Bold"/>
                <TextBlock Grid.Row="0" Grid.Column="1" x:Name="txtFirstName"
VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="22" FontFamily="Segoe WP
Bold"/>
                <TextBlock Grid.Row="1" Grid.Column="0" Text="Last Name"
VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontFamily="Segoe
WP Bold"/>
                <TextBlock Grid.Row="1" Grid.Column="1" x:Name="txtLastName"
VerticalAlignment="Center" HorizontalAlignment="Left" FontSize="22" FontFamily="Segoe WP
Bold"/>
                <TextBlock Grid.Row="2" Grid.Column="0" Text="Headline"
VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="22" FontFamily="Segoe
WP Bold"/>
                <TextBlock Grid.Row="2" Grid.Column="1" x:Name="txtHeadline"
VerticalAlignment="Center" HorizontalAlignment="Left" TextWrapping="Wrap" FontSize="22"
FontFamily="Segoe WP Bold"/>
            </Grid>
        </Grid>
    </Grid>

    <!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
```


(C) Copyright Nokia 2014. All rights reserved.

```
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Sign In"
Click="MenuItemSignIn_Click"/>
                <shell:ApplicationBarMenuItem Text="Sign Out"
Click="MenuItemSignOut_Click" IsEnabled="False"/>
            </shell:ApplicationBar.MenuItems>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
```

As shown above, the `txtWelcomeText` and `UserPanel` are initially set to `Collapsed` state. We have also kept the web browser's visibility to collapsed by default, so that it can be visible on pressing *Sign In* menu item. We have also defined the Web Browser's events `Navigated` and `Navigating`.

Code Behind

The following .cs files have been created for this application:

- **AppSettings.cs** - to keep the LinkedIn related URLs and other keys.
- **MainUtils.cs** - to set/get keys in/from Isolated Storage.
- **OAuthUtil.cs** - to fetch the request token query. This query would be of *OAuthWebQuery* type (defined in Hammock), inherited from WebQuery class.
- **person.cs** - Helper file to parse the XML response

AppSettings.cs File's contents

This file has following declaration.

```
  // linked in
        public static string RequestTokenUri =
"https://api.linkedin.com/uas/oauth/requestToken";
        public static string AuthorizeUri =
"https://www.linkedin.com/uas/oauth/authorize";
        public static string AccessTokenUri =
"https://api.linkedin.com/uas/oauth/accessToken";

        public static string CallbackUri = "http://www.google.com";
        Enter your keys below
        public static string consumerKey = "";
        public static string consumerKeySecret = "";

        public static string oAuthVersion = "1.0";

        //general Strings
        public static string TEXT_WELCOME = "Welcome";
        public static string SIGNED_OUT_SUCCESSFULLY = "You have been signed out
successfully.";
```

MainUtil.cs File's contents

This file has following declaration.

```
public class MainUtil
    {
        public static Dictionary<string, string> GetQueryParameters(string response)
        {
            Dictionary<string, string> nameValueCollection = new Dictionary<string,
string>();
            string[] items = response.Split('&');

            foreach (string item in items)
```


(C) Copyright Nokia 2014. All rights reserved.

```
            {
                if (item.Contains("="))
                {
                    string[] nameValue = item.Split('=');
                    if (nameValue[0].Contains("?"))
                        nameValue[0] = nameValue[0].Replace("?", "");
                    nameValueCollection.Add(nameValue[0],
System.Net.HttpUtility.UrlDecode(nameValue[1]));
                }
            }
            return nameValueCollection;
        }

        internal static T GetKeyValue<T>(string key)
        {
            if (IsolatedStorageSettings.ApplicationSettings.Contains(key))
                return (T)IsolatedStorageSettings.ApplicationSettings[key];
            else
                return default(T);
        }

        internal static void SetKeyValue<T>(string key, T value)
        {
            if (IsolatedStorageSettings.ApplicationSettings.Contains(key))
                IsolatedStorageSettings.ApplicationSettings[key] = value;
            else
                IsolatedStorageSettings.ApplicationSettings.Add(key, value);
            IsolatedStorageSettings.ApplicationSettings.Save();
        }
    }
```

In this class, the method of get/set keys in Isolated Storage defined. Also, one method is also defined to fetch the key pair values from a String.

OAuthUtil.cs File's contents

This file has following declaration.

```
 public class OAuthUtil
    {
        internal static OAuthWebQuery GetRequestTokenQuery()
        {
            var oauth = new OAuthWorkflow
            {
                ConsumerKey = AppSettings.consumerKey,
                ConsumerSecret = AppSettings.consumerKeySecret,
                SignatureMethod = OAuthSignatureMethod.HmacSha1,
                ParameterHandling = OAuthParameterHandling.HttpAuthorizationHeader,
                RequestTokenUrl = AppSettings.RequestTokenUri,
                Version = AppSettings.oAuthVersion,
                CallbackUrl = AppSettings.CallbackUri
            };

            var info = oauth.BuildRequestTokenInfo(WebMethod.Get);
            var objOAuthWebQuery = new OAuthWebQuery(info, false);
            objOAuthWebQuery.HasElevatedPermissions = true;
            objOAuthWebQuery.SilverlightUserAgentHeader = "Hammock";
            return objOAuthWebQuery;
        }
```


(C) Copyright Nokia 2014. All rights reserved.

```
    }
```

Via above defined function, we will fetch the Request Token query. For that, Hammock code is used.

person.cs File's contents

This file has following declaration.

```
 [XmlRoot("person")]
    public class person
    {
        [XmlElement("first-name")]
        public string FirstName { get; set; }

        [XmlElement("last-name")]
        public string LastName { get; set; }

        [XmlElement("headline")]
        public string Headline { get; set; }
    }
```

The above file is used to get and set the attributes values which we will get after parsing the profile request's response.

OAuth Authentication Process

- A request is sent to server to fetch the Request Token and Request Token Secret Keys.
- By Request Token and its key, LinkedIn Login page is opened for the user to enter the credentials.
- On successful login, a Verifier Pin is returned in a HTML response which needs to be parsed and pin needs to be saved.
- A request is sent using Request Token, Request Token Secret Key and the Verifier Pin, this time to fetch the permanent Access Token and Access Token Key. In case all goes well, then the two keys are returned back to the user.
- LinkedIn APIs methods can be invoked by using the Access Token and Access Token Key.

So, next we'll write code to do all the above process.

MainPageXaml.cs File's contents

- Using Directives:

```
using Hammock.Web;
using System.Text.RegularExpressions;
using Hammock.Authentication.OAuth;
using Hammock;
using System.Windows.Resources;
using SampleLinkedApp.Data;
using System.Xml.Serialization;
using System.Text;
```

- Variables Declared:

```
        string OAuthTokenKey = string.Empty;
        string tokenSecret = string.Empty;
        string accessToken = string.Empty;
        string accessTokenSecret = string.Empty;

        XmlSerializer personXerializer;
        person currentPerson;
```

When a user presses *Sign In*, the application checks whether the user is already logged in or not. If the use is logged in already, the application sends a server request to fetch the current user profile, otherwise it creates an asynchronous request to fetch the Request Token and Request Token Secret Key. The code is shown below:

 (C) Copyright Nokia 2014. All rights reserved.

```
 private void MenuItemSignIn_Click(object sender, EventArgs e)
        {
            accessToken = MainUtil.GetKeyValue<string>("AccessToken");
            accessTokenSecret = MainUtil.GetKeyValue<string>("AccessTokenSecret");

            if (string.IsNullOrEmpty(accessToken) ||
string.IsNullOrEmpty(accessTokenSecret))
            {
                var requestTokenQuery = OAuthUtil.GetRequestTokenQuery();
                requestTokenQuery.RequestAsync(AppSettings.RequestTokenUri, null);
                requestTokenQuery.QueryResponse += new
EventHandler<WebQueryResponseEventArgs>(requestTokenQuery_QueryResponse);
            }
            else
            {
                Dispatcher.BeginInvoke(() =>
                {
                    // finding current user's profile if already signed in
                    getCurrentUserProfile();

                    var SignInMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[0];
                    SignInMenuItem.IsEnabled = false;

                    var SignOutMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[1];
                    SignOutMenuItem.IsEnabled = true;

                    txtWelcomeText.Visibility = System.Windows.Visibility.Visible;
                    UserPanel.Visibility = System.Windows.Visibility.Visible;
                    txtWelcomeText.Text = AppSettings.TEXT_WELCOME;
                });
            }
        }

void requestTokenQuery_QueryResponse(object sender, WebQueryResponseEventArgs e)
        {
            try
            {
                StreamReader reader = new StreamReader(e.Response);
                string strResponse = reader.ReadToEnd();
                var parameters = MainUtil.GetQueryParameters(strResponse);
                OAuthTokenKey = parameters["oauth_token"];
                tokenSecret = parameters["oauth_token_secret"];
                var authorizeUrl = AppSettings.AuthorizeUri + "?oauth_token=" +
OAuthTokenKey;

                Dispatcher.BeginInvoke(() =>
                {
                    this.loginBrowserControl.Navigate(new Uri(authorizeUrl,
UriKind.RelativeOrAbsolute));
                });
            }
            catch (Exception ex)
            {
                Dispatcher.BeginInvoke(() =>
                {
                    MessageBox.Show(ex.Message);
```


(C) Copyright Nokia 2014. All rights reserved.

```
                });
            }
        }

        private void loginBrowserControl_Navigated(object sender,
System.Windows.Navigation.NavigationEventArgs e)
        {
            this.loginBrowserControl.Visibility = Visibility.Visible;
            this.loginBrowserControl.Navigated -= loginBrowserControl_Navigated;
        }

        private void loginBrowserControl_Navigating(object sender, NavigatingEventArgs
e)
        {
            if (e.Uri.ToString().StartsWith(AppSettings.CallbackUri))
            {
                var AuthorizeResult = MainUtil.GetQueryParameters(e.Uri.ToString());
                var VerifyPin = AuthorizeResult["oauth_verifier"];
                this.loginBrowserControl.Visibility = Visibility.Collapsed;

                getAccessToken(VerifyPin);
            }
        }
```

On getting the response of our first request (request to fetch Request Token & Request Token Key), we're saving the two keys in Isolated storage and redirecting the web browser to the authorization URL. In the web browser control's `Navigating` event, we're fetching the **Verifier Pin**, setting the browser's Visibility to `Collapsed` state again & also making a Hammock request to fetch the access token.

The code to fetch a access code is shown below.

```
 private void getAccessToken(String aVerifyPin)
        {
            var credentials = new OAuthCredentials
            {
                Type = OAuthType.AccessToken,
                SignatureMethod = OAuthSignatureMethod.HmacSha1,
                ParameterHandling = OAuthParameterHandling.HttpAuthorizationHeader,
                ConsumerKey = AppSettings.consumerKey,
                ConsumerSecret = AppSettings.consumerKeySecret,
                Token = this.OAuthTokenKey,
                TokenSecret = this.tokenSecret,
                Verifier = aVerifyPin,
                Version = "1.0"
            };

            var client = new RestClient
            {
                Authority = "https://api.linkedin.com/uas/oauth",
                HasElevatedPermissions = true
            };

            var request = new RestRequest
            {
                Credentials = credentials,
                Path = "/accessToken",
                Method = WebMethod.Post
            };
```

 (C) Copyright Nokia 2014. All rights reserved.

```
            client.BeginRequest(request, new RestCallback(AccessTokenRequestCallback));
        }

        private void AccessTokenRequestCallback(RestRequest request, RestResponse
response, object obj)
        {
            try
            {
                string respContent = response.Content;
                var parameters = MainUtil.GetQueryParameters(respContent);
                accessToken = parameters["oauth_token"];
                accessTokenSecret = parameters["oauth_token_secret"];

                MainUtil.SetKeyValue<string>("AccessToken", accessToken);
                MainUtil.SetKeyValue<string>("AccessTokenSecret", accessTokenSecret);

                Dispatcher.BeginInvoke(() =>
                {
                    // finding current user's profile
                    getCurrentUserProfile();

                    var SignInMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[0];
                    SignInMenuItem.IsEnabled = false;

                    var SignOutMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[1];
                    SignOutMenuItem.IsEnabled = true;

                    txtWelcomeText.Visibility = System.Windows.Visibility.Visible;
                    txtWelcomeText.Text = AppSettings.TEXT_WELCOME;
                });
            }
            catch (Exception ex)
            {
                Dispatcher.BeginInvoke(() =>
                {
                    MessageBox.Show(ex.Message);
                });
            }
        }
```

As soon as we get the response of above server hit, we will save the access token and access token secret key in Isolated storage. Simultaneously, we're making our third and final server hit to fetch the currently logged in user's basic profile information. For this purpose, the code is mentioned below:

```
private void getCurrentUserProfile()
        {
            var credentials = new OAuthCredentials
            {
                Type = OAuthType.ProtectedResource,
                SignatureMethod = OAuthSignatureMethod.HmacSha1,
                ParameterHandling = OAuthParameterHandling.UrlOrPostParameters,
                ConsumerKey = AppSettings.consumerKey,
                ConsumerSecret = AppSettings.consumerKeySecret,
                Token = this.accessToken,
                TokenSecret = this.accessTokenSecret,
                Version = "1.0"
```


(C) Copyright Nokia 2014. All rights reserved.

```
            };

            var client = new RestClient
            {
                Authority = "https://api.linkedin.com",
                HasElevatedPermissions = true
            };

            var request = new RestRequest
            {
                Credentials = credentials,
                Path = "/v1/people/~",
                Method = WebMethod.Get
            };

            client.BeginRequest(request, new
RestCallback(CurrentUserProfileRequestCallback));
        }

        private void CurrentUserProfileRequestCallback(RestRequest request, RestResponse
response, object obj)
        {
            try
            {
                string respContent = response.Content;

                Deployment.Current.Dispatcher.BeginInvoke(() =>
                 {
                     // updating UI
                     UserPanel.Visibility = System.Windows.Visibility.Visible;
                     parseCurrentPersonData(respContent);
                 });
            }
            catch (Exception ex)
            {
                Dispatcher.BeginInvoke(() =>
                {
                    MessageBox.Show(ex.Message);
                });
            }
        }
```

On receiving the response successfully above, we're setting the visibility of our `UserPanel` to . Also, we are passing the response string to a function to parse the response which is in XML format. To parse the XML, we are using following references in our project:

- System.Xml
- System.Xml.Linq
- System.Xml.Serialization

To add above references, simply use the **Solution Explorer window >> your project | References | Add Reference**.

The code to parse the XML response is mentioned below.

```
  private void parseCurrentPersonData(string aRespString)
        {
            // converting string to stream
            byte[] byteArray = Encoding.UTF8.GetBytes(aRespString);
            MemoryStream personXml = new MemoryStream(byteArray);
```

 (C) Copyright Nokia 2014. All rights reserved.

```
            if (personXml != null)
            {
                personXerializer = new XmlSerializer(typeof(person));
                currentPerson = (person)personXerializer.Deserialize(personXml);
                updateUI();
            }
        }
```

As soon as the XML data is parsed, we are updating the UI using the following code snippet:

```
private void updateUI()
        {
            Deployment.Current.Dispatcher.BeginInvoke(() =>
           {
               if (currentPerson != null)
               {
                   txtFirstName.Text = currentPerson.FirstName;
                   txtLastName.Text = currentPerson.LastName;
                   txtHeadline.Text = currentPerson.Headline;
               }
           });
        }
```

To carry out Sign Out functionality, we just have to clear out the saved values in Isolated Storage as shown below:

```
 private void MenuItemSignOut_Click(object sender, EventArgs e)
        {
            MainUtil.SetKeyValue<string>("AccessToken", string.Empty);
            MainUtil.SetKeyValue<string>("AccessTokenSecret", string.Empty);

            Dispatcher.BeginInvoke(() =>
            {
                var SignInMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[0];
                SignInMenuItem.IsEnabled = true;

                var SignOutMenuItem =
(Microsoft.Phone.Shell.ApplicationBarMenuItem)this.ApplicationBar.MenuItems[1];
                SignOutMenuItem.IsEnabled = false;

                txtWelcomeText.Visibility = System.Windows.Visibility.Collapsed;
                UserPanel.Visibility = System.Windows.Visibility.Collapsed;
                txtFirstName.Text = "";
                txtLastName.Text = "";
                txtHeadline.Text = "";

                MessageBox.Show(AppSettings.SIGNED_OUT_SUCCESSFULLY);
            });
        }
```

Build and Run

Now you may build the app and try to run it.

References

- http://nuget.org/

(C) Copyright Nokia 2014. All rights reserved.

- http://nuget.org/packages/Hammock
- http://oauth.net/documentation/getting-started/
- http://developer.linkedin.com/documents/profile-api
- http://developer.linkedin.com/apis

(C) Copyright Nokia 2014. All rights reserved.