Getting Started with Microsoft MVP API

There are a few instructions that need to be followed to get up and running to use the Microsoft MVP API as a developer. The developers will access the API through Azure API Management service and the endpoints exposed there. The developer’s applications will also need to manage the Microsoft Account (aka. Live ID) authentication process.

**Feedback**

Please share your feedback, suggestions or ideas via the Yammer channel: <http://aka.ms/mvpapiyammer> OR fill out this simple feedback form: <http://aka.ms/mvpapifeedback>.

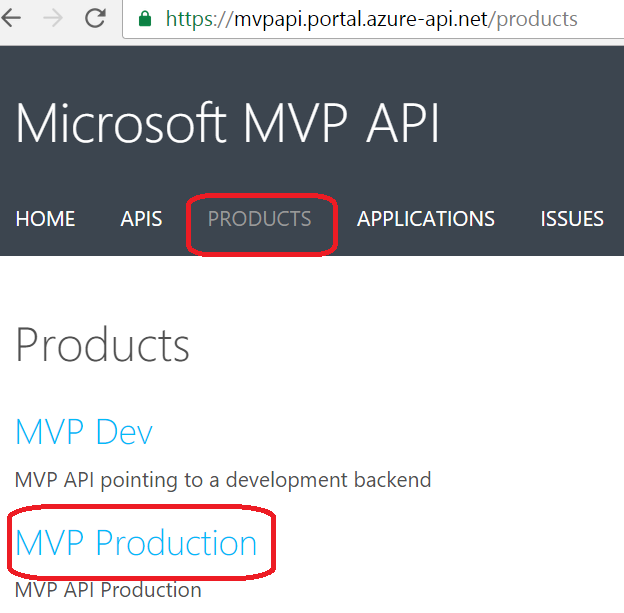
**Step 1. Whitelist yourself for using the Microsoft MVP API**

This step is needed only during the Hackathon for us to control who can log in and preview the Microsoft MVP APIs. When we eventually launch the Microsoft MVP API, this step won’t be needed because all ‘active’ MVPs will be auto-whitelisted.

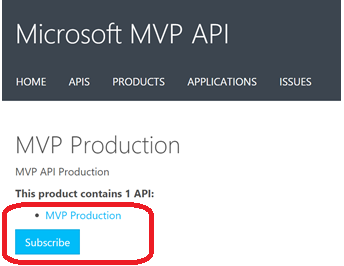
1. Go to <https://mvp.microsoft.com/en-us/benefits/ApiRequest>
2. Log in with the Microsoft Account used for your MVP Site access (MUST be active MVP).
   1. If you forgot your login account, or you get any errors, please let us know.

**Step 2. Request Developer access to Microsoft MVP API via the Developer Portal**

1. Go to <https://mvpapi.portal.azure-api.net/>
2. Choose “Sign In” (Not “Sign Up”! ☺) with your Microsoft Account.
   * **Strongly recommended** the same as you use for your MVP Site access (in above Step 1)
3. Subscribe to the **MVP Production** product
   * Go to the PRODUCTS tab, and choose ‘MVP Production’.

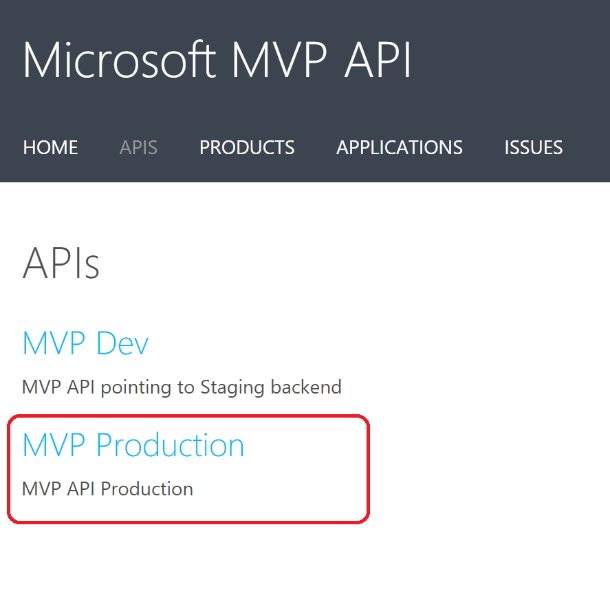


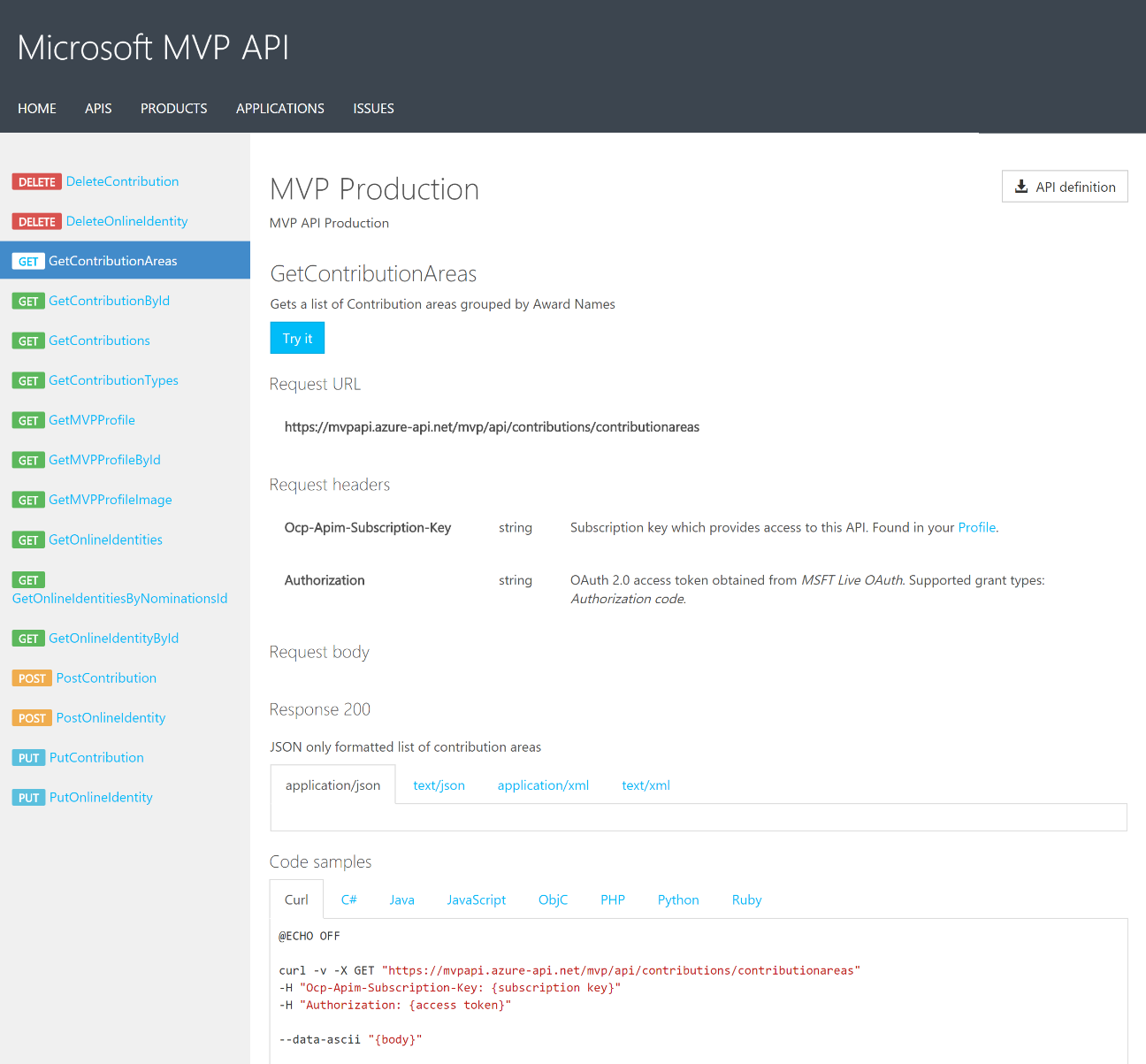
* + Click the ‘Subscribe’ button



* + This request will be reviewed and Accepted / Rejected by the admin. During the Hackathon, we may auto-approve all requests, so you should be able to access and try API actions.

**Step 3. Explore, Learn and Try the APIs on the Developer Portal**

1. After you are approved in Step 2, go to <https://mvpapi.portal.azure-api.net/docs/services/>, and select ‘MVP Production’.
2. You can explore all the APIs. Thanks to Azure API Management, the Developer Portal even provides a nice ‘Try it’ function, and auto-generated code snippets in various programming languages.



**Step 4. Create your own app**

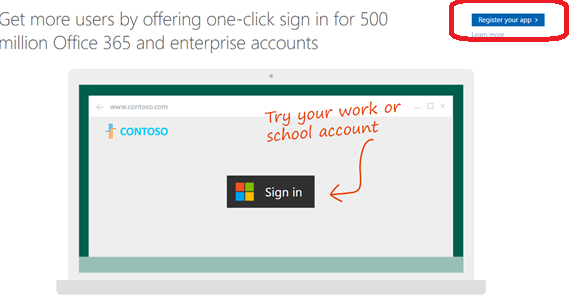
1. Get your Primary key (**Ocp-Apim-Subscription-Key**) from the Developer Portal (<https://mvpapi.portal.azure-api.net/developer> )

In the Developer Portal, you need to click your name and choose ‘PROFILE’. Unveil the Primary key of the API Subscription. Copy this key and save it somewhere. Later, you will need to pass it to the **Ocp-Apim-Subscription-Key** header when calling the API.

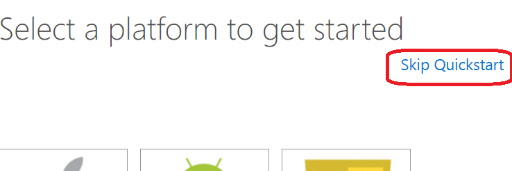


1. Get Microsoft Account client id and client secret for the Microsoft Account login

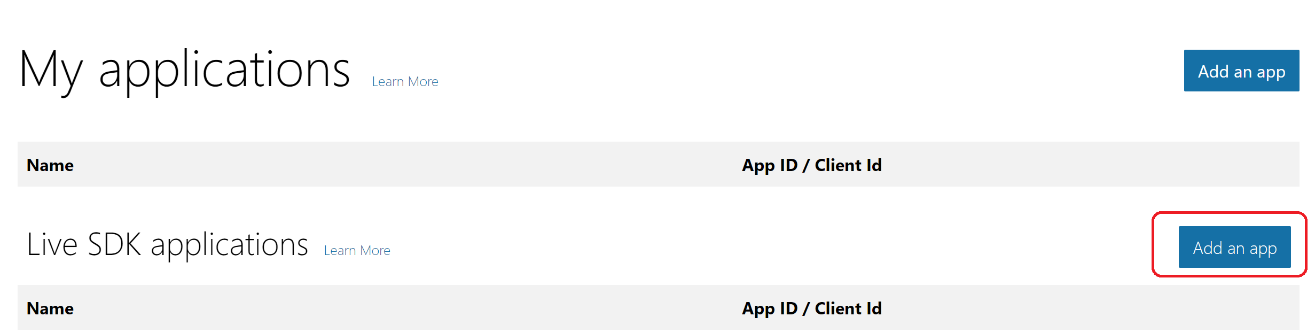
First, open <https://apps.dev.microsoft.com> and click ‘Register your app’.



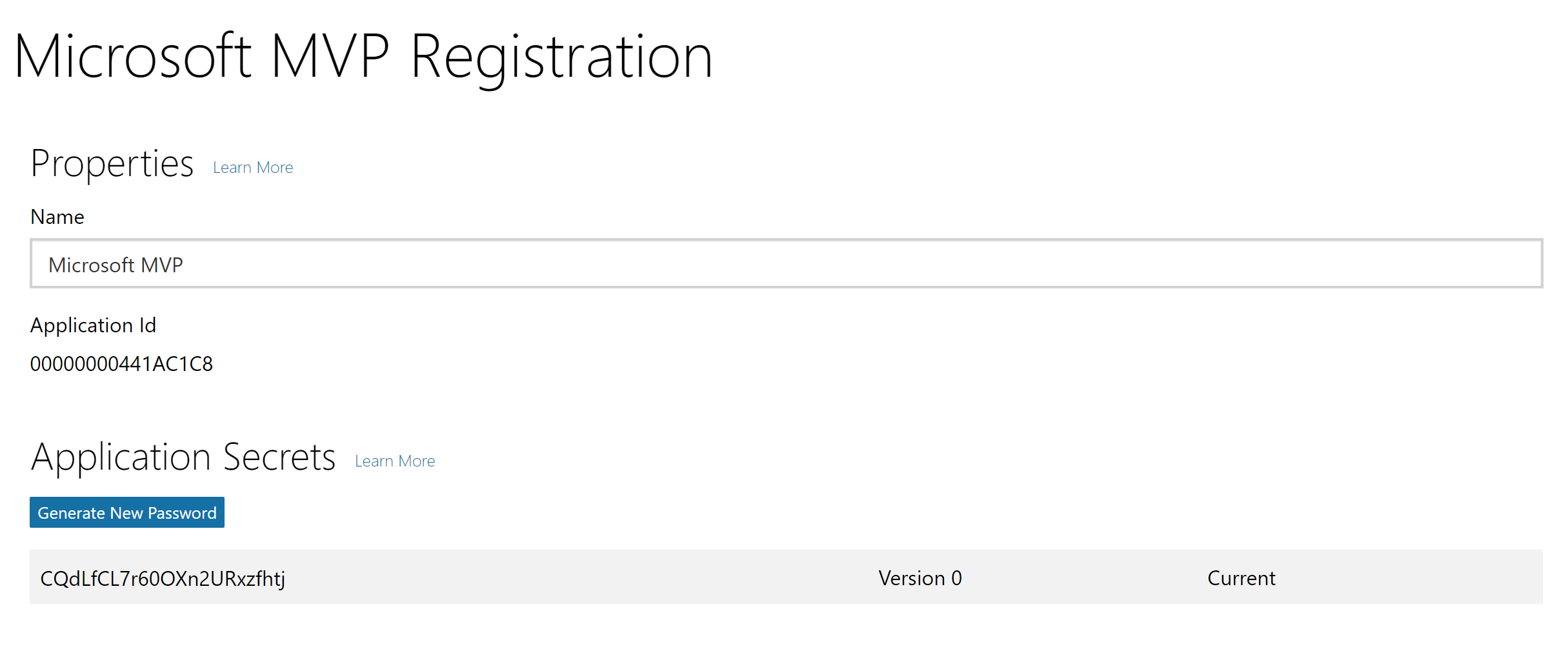
Choose to “skip Quickstart”.



Under the **‘Live SDK applications’**, click ‘Add an app’.



After you finish the wizard, you will get an Application Id, and an Application Secret key. Copy them somewhere as you will need them in your code later.



1. Now let’s write some code to call the API.

The following code handles the ‘login’ with the Live ID authentication and get the Live ID access token.

|  |
| --- |
| static string scope = "wl.emails%20wl.basic%20wl.offline\_access%20wl.signin";  static string client\_id = "<paste your application id from step 2>";  static string client\_secret = "<paste your application secret from step 2";  static Uri signInUrl = new Uri(  String.Format(@"https://login.live.com/oauth20\_authorize.srf?client\_id={0}&redirect\_uri=https://login.live.com/oauth20\_desktop.srf&response\_type=code&scope={1}", client\_id, scope));  static string accessTokenUrl = String.Format(@"https://login.live.com/oauth20\_token.srf?client\_id={0}&client\_secret={1}&redirect\_uri=https://login.live.com/oauth20\_desktop.srf&grant\_type=authorization\_code&code=", client\_id, client\_secret);  static string refreshTokenUrl = String.Format(@"https://login.live.com/oauth20\_token.srf?client\_id={0}&client\_secret={1}&redirect\_uri=https://login.live.com/oauth20\_desktop.srf&grant\_type=refresh\_token&refresh\_token=", client\_id, client\_secret);  Next, in a WebBrowser control, navigate to the signInUrl  webBrowser.Navigate(signInUrl);  After the user signs in, grab the **auth code** from the browser.  private void webBrowser\_LoadCompleted(object sender, NavigationEventArgs e)  {  if (e.Uri.AbsoluteUri.Contains("code=")) // This means that the user signed in.  {  // get the auth code  string auth\_code = Regex.Split(e.Uri.AbsoluteUri, "code=")[1];  // and store it somewhere as an example in the application scope  if (App.Current.Properties.Contains("auth\_code"))  App.Current.Properties.Clear();  App.Current.Properties.Add("auth\_code", auth\_code);      // close the browser control window  this.Close();  }  }  void browser\_Closed(object sender, EventArgs e)  {  if (App.Current.Properties.Contains("auth\_code"))  {  makeAccessTokenRequest(accessTokenUrl + App.Current.Properties["auth\_code"]);  }  }  Next, use the **auth code** to get the Live ID **access token**:  public void makeAccessTokenRequest(string requestUrl)  {  HttpWebRequest request = WebRequest.Create(requestUrl) as HttpWebRequest;    string responseTxt = String.Empty;  using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)  {  var reader = new StreamReader(response.GetResponseStream());  responseTxt = reader.ReadToEnd();    var tokenData = JsonConvert.DeserializeObject<Dictionary<string, string>>(responseTxt);  if (tokenData.ContainsKey("access\_token"))  {  Application.Current.Properties.Add("access\_token", tokenData["access\_token"]);    // refresh token  Application.Current.Properties.Add("refresh\_token", tokenData["refresh\_token"]);  }  }  } |

After getting the access token, you can call the Microsoft MVP APIs now.

|  |
| --- |
| try  {  HttpWebRequest request = WebRequest.Create("https://mvpapi.azure-api.net/mvp/api/profile") as HttpWebRequest;  request.Method = "GET";  request.Headers["Ocp-Apim-Subscription-Key"] = "<Paste the subscription key from step 1 in Developer Portal>";  request.Headers["Authorization"] = "paste the access token here from developer portal, under Authorization Headers or use: App.Current.Properties["access\_token"].ToString();"  request.Headers["api-version"] = "1"; // optional  request.ContentType = "application/json"; // application/xml    string responseTxt = String.Empty;  using (HttpWebResponse response = request.GetResponse() as HttpWebResponse)  {  var reader = new StreamReader(response.GetResponseStream());  responseTxt = reader.ReadToEnd();  response.Close();    // write to ui or do something with responseTxt    }  }  catch (WebException ex)  {  var reader = new StreamReader(ex.Response.GetResponseStream());  var responseTxt = reader.ReadToEnd();  } |

**Step 5. Play with the Xamarin app**

1. Install Visual Studio 2015 with ‘Cross Platform Development’. This will install Xamarin for you.
2. Download the Xamarin app source code from <http://aka.ms/mvpxamarinapp>
3. Install ‘Add java option.reg’ in the download package and restart your computer.
4. Open the solution file and choose the Droid or Windows Phone projects to deploy.