The application is implemented as a single instance application, meaning that there will only be one instance of vaConnect.exe running at any time.

To register a new protocol handler all you need to do is add a few lines to the registry:

**[HKEY\_CLASSES\_ROOT]**

**[vaconnect]**

**(Default) = “URL:vaConnect”**

**URL Protocol = “”**

**[DefaultIcon]**

**(Default) = “vaConnect.exe”**

**[shell]**

**[open]**

**[command]**

**(Default) = “c:\whatever\vaConnectt.exe “%1””**

The most important part is the URL Protocol value under the “vaconnect” key. This will register the vaconnect:// protocol with “Url.dll”.

When ever Windows encounters a vaconnect:// link “Url.dll” will execute your vaConnect.exe file passing the URL as a command argument.

So when “Url.dll” executes a link to “vaconnect” it’ll execute “c:\whatever\vaConnect.exe vaconnect://.......”. In main method we need to parse the

arguments and take appropriate action.

To make things easy the application can write/delete the registry keys needed to register the protocol handler.

The code to do that looks as follow:

**RegistryKey helpDesk = Registry.ClassesRoot.CreateSubKey("vaConnect");**

**helpDesk.SetValue("", "URL:vaConnect");**

**helpDesk.SetValue("URL Protocol", "");**

**RegistryKey defaultIcon = vaConnect.CreateSubKey("DefaultIcon");**

**defaultIcon.SetValue("", Path.GetFileName(Application.ExecutablePath));**

**RegistryKey shell = vaConnect.CreateSubKey("shell");**

**RegistryKey open = shell.CreateSubKey("open");**

**RegistryKey command = open.CreateSubKey("command");**

**command.SetValue("", Application.ExecutablePath + " %1");**

When you have your protocol handler registered you need to parse the command line arguments.

In this application I’ve created a single instance application, meaning that any attempt to start “vaConnect.exe” while the application

is running will not create a new instance. Instead it’ll show the all ready running application.

There are several ways to implement single instance applications.

The easiest way however is to use.NET 2.0 code - to derive the WindowsFormsApplicationBase class which is part of the “special” VB.NET libraries.

All you have to do is to add a reference to “Microsoft.VisualBasic.dll” in C# application.

All this implemented in SingleInstanceApplication.cs and Program.cs class.

Application structure in general is the same as for Android application.

**Main classes**:

Onboarding service - singletone class. Main method - public async Task< WiFiProfile> getWiFiProfileAsync(String token, String identifier), get WiFi profile from onboarding services.

To manage and control profiles application use managedwifi package.