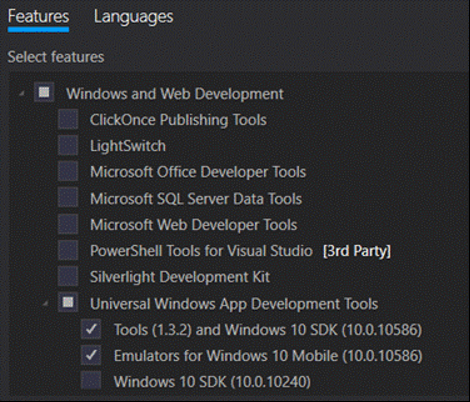
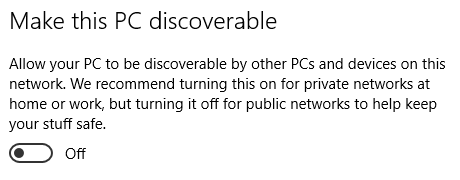
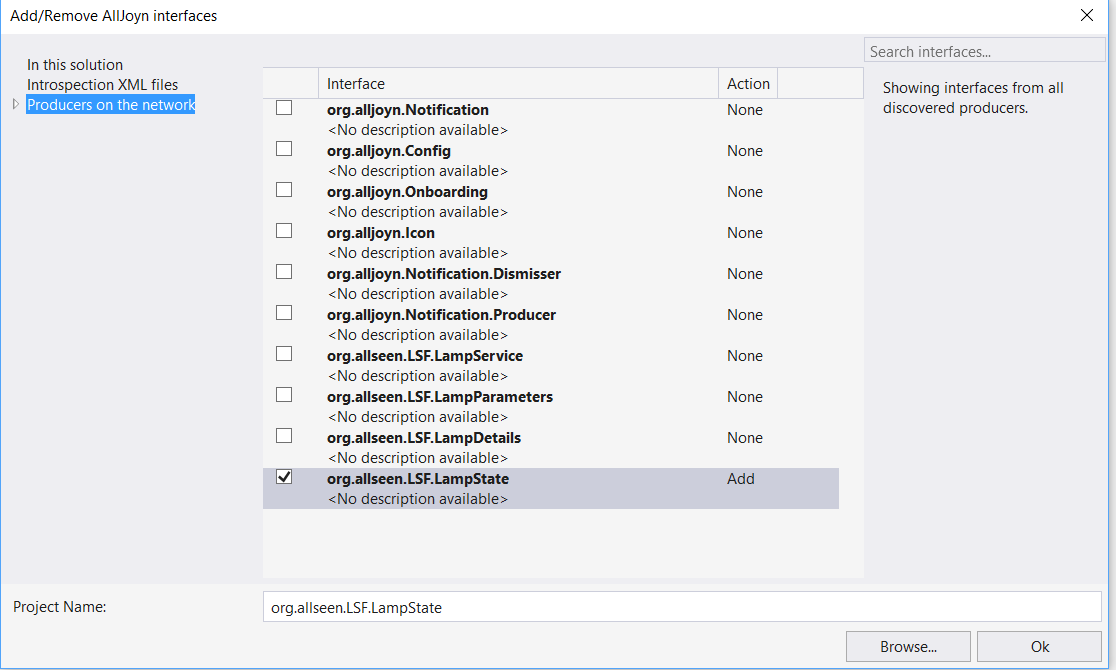
# Setup and Pre-requisites:

1. Install [VS Update 2](http://go.microsoft.com/fwlink/?LinkId=691129) with custom installation as shown below

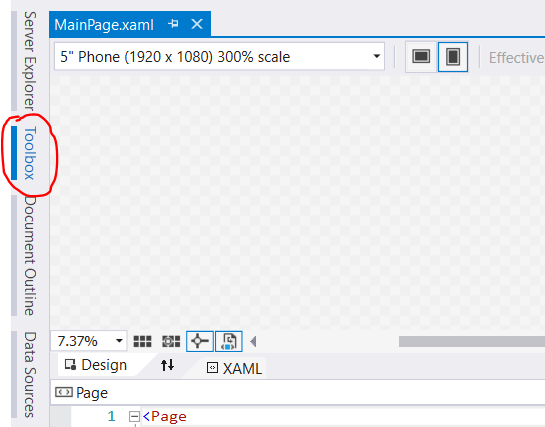


1. Install [AllJoyn Studio VS Extension](https://visualstudiogallery.msdn.microsoft.com/064e58a7-fb56-464b-bed5-f85914c89286)
2. Connect to the AllJoyn\_WiFi (password - alljoynpassword)
3. Go to Settings (shortcut: Win + i) 🡪 Network & Internet 🡪 Wi-Fi 🡪 Advanced options
4. Make sure the toggle switch of Make PC discoverable is turned on

# Creating a new AllJoyn project

1. Open VS 2015.
2. Create New Project.
3. Under Visual C#, select AllJoyn App.
4. Give the project a name, and click OK.
5. Click Ok on popup window of “New Universal Windows Project”.
6. On the “Add/Remove AllJoyn interfaces”, select Producers on the network on the left panel.
7. Check the checkbox next to org.allseen.LSF.LampState interface, as shown below, and click Ok.
8. Build and deploy the app by hitting F5.
9. At this stage you should have a blank app running on your machine. Go back to VS and stop debugging (Shift + F5).

# Adding On/Off functionality to the app

1. From the Solution Explorer open Mainpage.xaml
2. From the left most pane – open Toolbox as shown below
3. Under the All XAML Controls drop down, select Toggle Switch.
4. Drag and drop the Toggle Switch anywhere in your Design pane.
5. Double click on the Toggle Switch in the design page. This will open Mainpage.xaml.cs and will add the following code

**private** void toggleSwitch\_Toggled**(object** sender**,** RoutedEventArgs e**)**

**{**

**}**

1. Now in Mainpage.xaml.cs add three private members under Mainpage class like shown below

private AllJoynBusAttachment busAttachment = null;

private LampStateConsumer consumer = null;

private string lampDeviceId = "<Insert your lamp device id>";

1. For any code that has the red squiggle lines under like shown below



Click anywhere in the red squiggle text area and hit Ctrl + . (dot) and from the drop down menu select the using <namespace> option. Repeat this for every squiggle underlined code.

1. In the lampDeviceId member – insert the device id of the lamp given to you.
2. In the constructor of Mainpage class, initialize the AllJoyn bus attachment – like shown below

busAttachment = new AllJoynBusAttachment();

1. Then, initialize an object of the LampStateWatcher class. This generated class will help us to listen for producers on the network that advertise the LampState interface.

LampStateWatcher watcher = new LampStateWatcher(busAttachment);

1. Then, subscribe to the Added event using the watcher object just created. This event will be fired whenever a producer advertising the LampState interface is discovered on the network.

watcher.Added += Watcher\_Added;

1. Then, start the watcher

watcher.Start();

1. Add an event handler for the Added event above

private void Watcher\_Added(LampStateWatcher sender, AllJoynServiceInfo args)

{

}

1. Now within the handler for the added event, we will first get the About data of the discovered device.

AllJoynAboutDataView aboutData = await AllJoynAboutDataView.GetDataBySessionPortAsync(args.UniqueName, busAttachment, args.SessionPort);

1. Also to await on an async method we will change the containing scope of the await call to be async. This is as easy as hitting Ctrl + . on the red squiggle line and selecting – “Make the containing scope ‘async’” option from the drop down.
2. Now if the received aboutData is not null, and the device id in the about data is not a null or empty string and the device id is equal to the lamp device id of our lamp, we want to join session with this device.

if (aboutData != null && !string.IsNullOrWhiteSpace(aboutData.DeviceId) && string.Equals(aboutData.DeviceId, lampDeviceId))

{

// Join session with the producer of the LampState interface.

LampStateJoinSessionResult joinSessionResult = await LampStateConsumer.JoinSessionAsync(args, sender);

}

1. If the join session operation is successful, we will get the consumer object that can control the lamp.

if (joinSessionResult.Status == AllJoynStatus.Ok)

{

consumer = joinSessionResult.Consumer;

}

1. OPTIONAL: If after discovering the lamp and joining session, we want to get the current state of the lamp to update our UI, we can do the following

// Get the current On/Off state of the lamp.

LampStateGetOnOffResult onOffResult = await consumer.GetOnOffAsync();

if (onOffResult.Status == AllJoynStatus.Ok)

{

toggleSwitch.IsOn = onOffResult.OnOff;

}

1. Now the only thing left, is to handle the user toggling the switch in the UI. Inside the toggle switch toggled event handler, we will add the following code

if (consumer != null)

{

await consumer.SetOnOffAsync(((ToggleSwitch)sender).IsOn);

}

1. And again we will make the containing scope to be async as shown before.
2. At this point you should have a working app which can control your lamp on and off. ☺