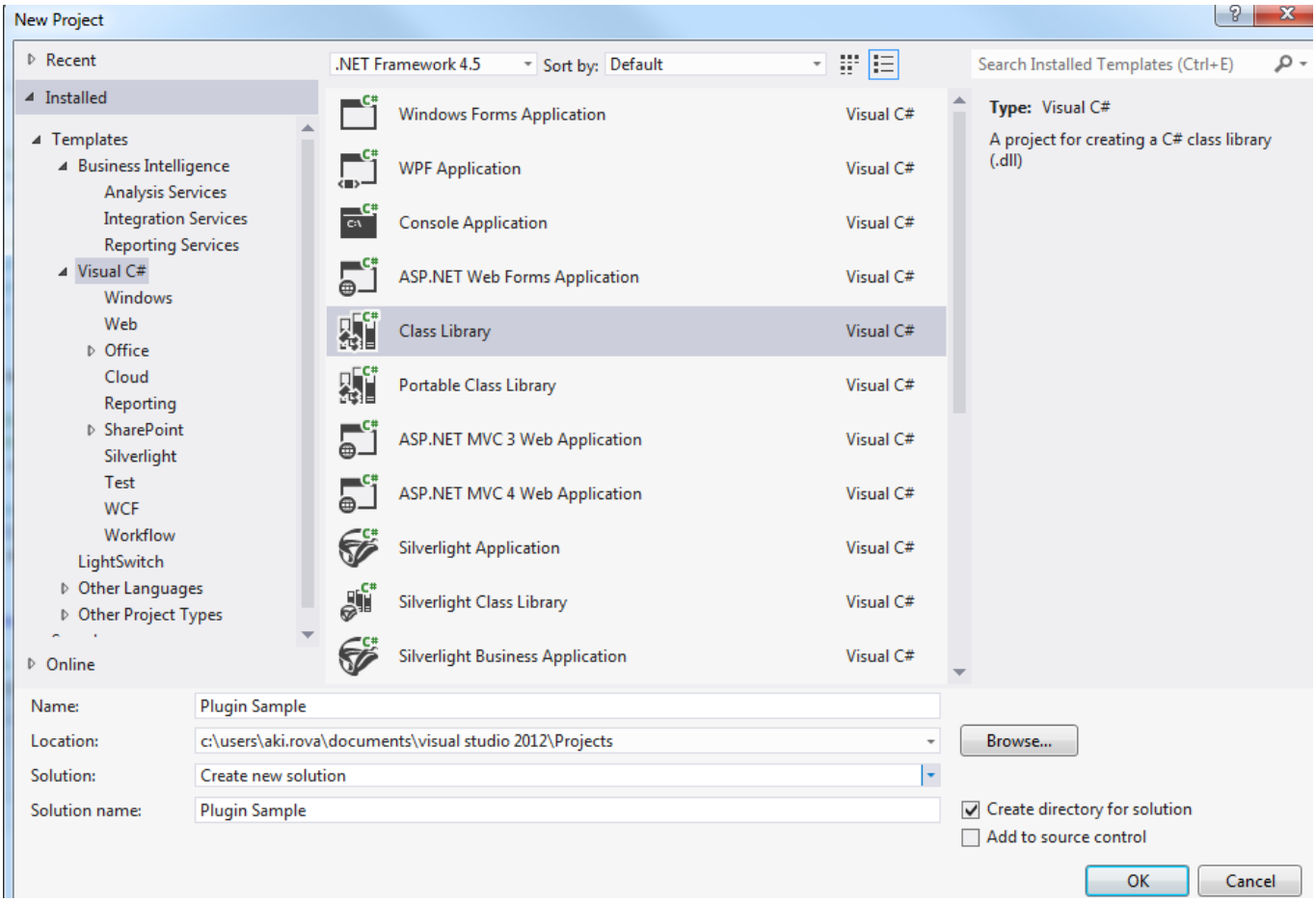
Developing a Plug-in

How to Start Developing a CRM Plug-in

1. Download CRM SDK. This will provide you all the information, required SDK assemblies and many helpful samples.

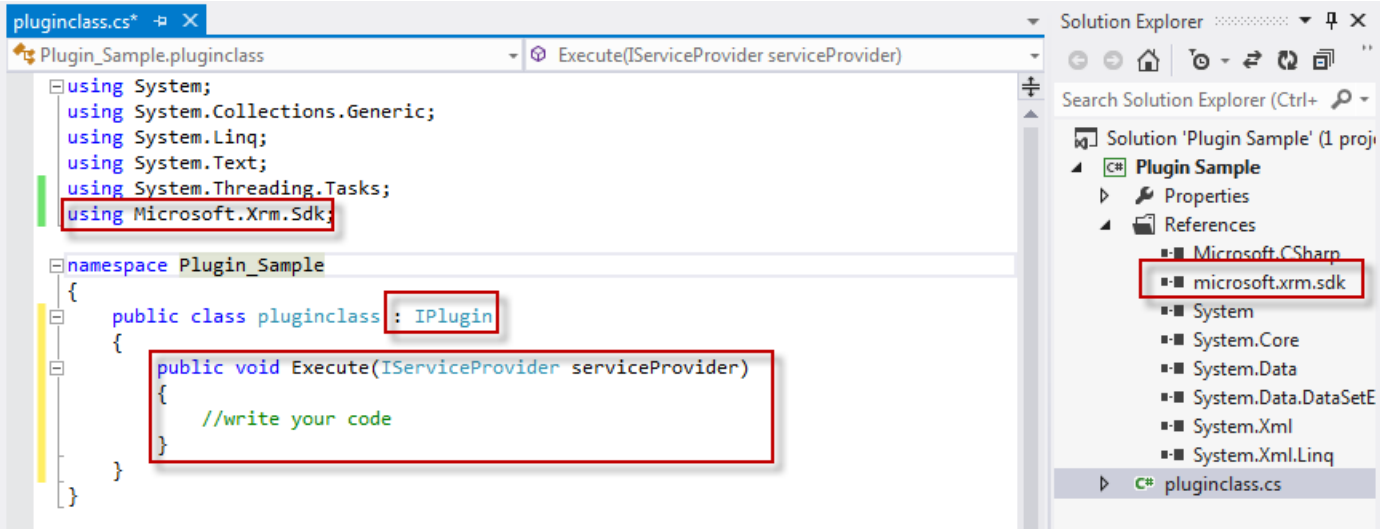
2. Set up your plug-in project in Visual Studio. This will be a .NET class library.



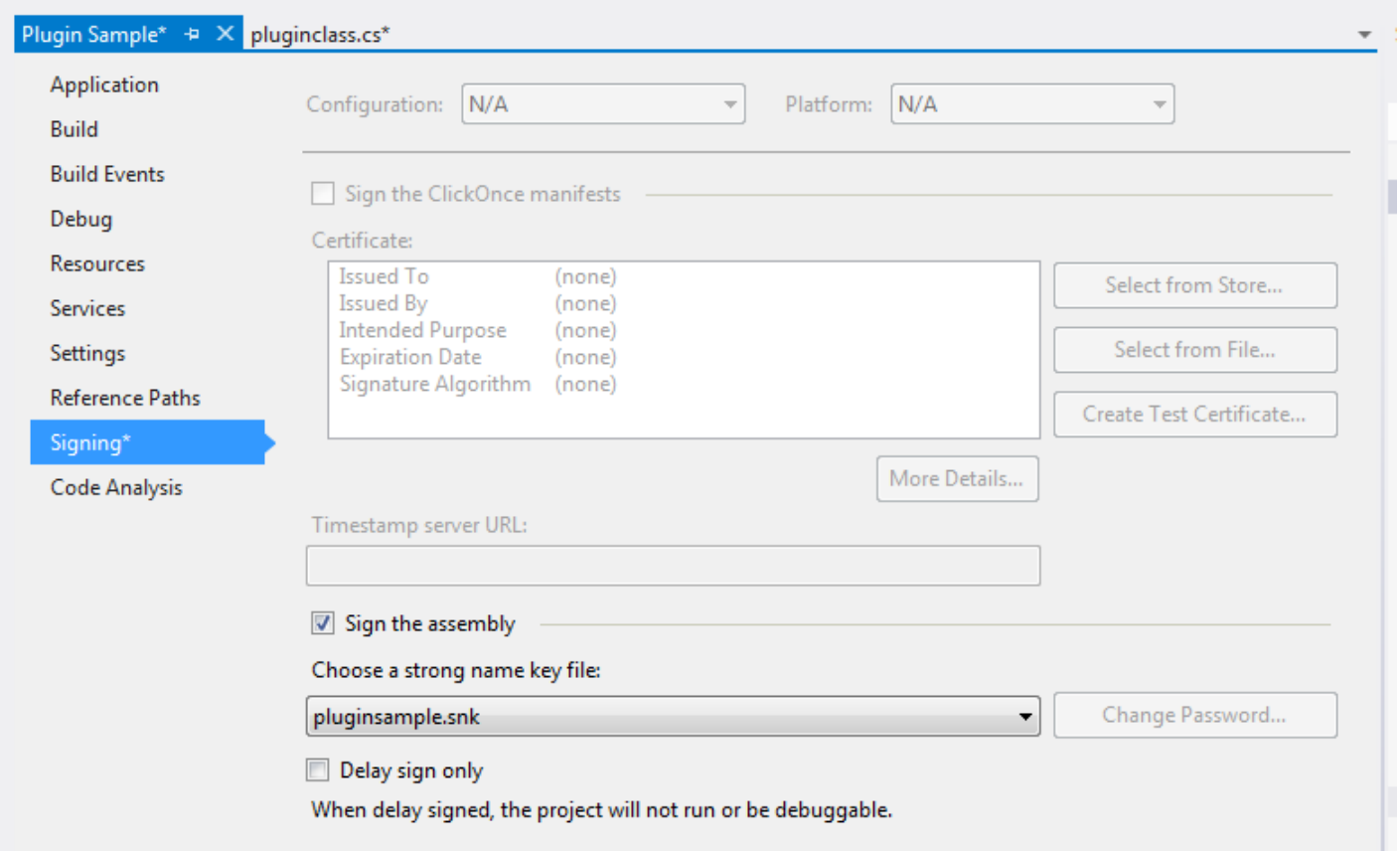
3. Add References. At minimum, you will need Microsoft.Xrm.Sdk, obtained from CRM SDK.

4. Extend the class from Microsoft.Xrm.Sdk.IPlugin

5. Write your code



6. At the project, sign the assembly. This is required in order to be able to deploy the plugin.



7. Compile the assembly and deploy using Plugin Registration Tool.

Example 1: Update parent record based on multiple fields

Here's an extremely simplified example of a plug-in. We have parent record with a field for total, and a single child with unit and rate. When either unit or rate changes, we want to multiply these and update to the total of the parent.

Design Notes

- Because user may only update units and not update rate, we cannot use target which would only include the attributes that were actually updated. Therefore, we use image to make sure we get the necessary attributes, without having to do additional data retrieve.
- The plug-in is post-update, so we will have access to post-image, showing all values like they are after update.
- When registering the plugin, we set filtering attributes to include only rate and units. Therefore, plugin will not trigger needlessly if something unrelated updates.
- When setting up the Image, we only need rate, units and parent ID.

Plug-in Code

```
using System;
using Microsoft.Xrm.Sdk;

namespace CRMBook
{
   public class Update : IPlugin
   {
      public void Execute(IServiceProvider serviceProvider)
      {

         // Obtain the execution context from the service provider.
         IPluginExecutionContext context =

(IPluginExecutionContext)serviceProvider.GetService(typeof(IPluginExecutionContext));

         // Get a reference to the Organization service.
         IOrganizationServiceFactory factory =

(IOrganizationServiceFactory)serviceProvider.GetService(typeof(IOrganizationServiceFactory));
         IOrganizationService service = factory.CreateOrganizationService(context.UserId);

         if (context.InputParameters != null)
         {
            //entity = (Entity)context.InputParameters["Target"];
```

```
            //Instead of getting entity from Target, we use the Image
            Entity entity = context.PostEntityImages["PostImage"];

            Money rate = (Money)entity.Attributes["po_rate"];
            int units = (int)entity.Attributes["po_units"];
            EntityReference parent = (EntityReference)entity.Attributes["po_parentid"];

            //Multiply
            Money total = new Money(rate.Value * units);

            //Set the update entity
            Entity parententity = new Entity("po_parententity");
            parententity.Id = parent.Id;
            parententity.Attributes["po_total"] = total;

            //Update
            service.Update(parententity);

        }

    }

  }
}
```

Example 2: Update same record based on multiple fields

This example is similar to the one above, but let's assume our total is on same entity.

- In order to change the total on the fly, the plug-in needs to be pre-update
- This means we do not have access to post image, and have to use the pre-image instead. Therefore, we need to get all the changed values from the target, and unchanged values from the image.

Plug-in Code

```
if (context.InputParameters != null)
        {
            //Get Target - includes everything changed
            Entity entity = (Entity)context.InputParameters["Target"];

            //Get Pre Image
            Entity image = context.PreEntityImages["PreImage"];
```

```
    //If value was changed, get it from target.
    //Else, get the value from preImage
    Money rate = null;
    if(entity.Attributes.Contains("po_rate"))
       rate = (Money)entity.Attributes["po_rate"];
    else
       rate =  (Money)image.Attributes["po_rate"];
    int units = 0;
    if (entity.Attributes.Contains("po_units"))
       units = (int)entity.Attributes["po_units"];
    else
       units = (int)image.Attributes["po_units"];

    //Multiply
    Money total = new Money(rate.Value * units);

    //Set the value to target
    entity.Attributes.Add("po_total",total);

    //No need to issue additional update
}
```