# How To: Add Profiles to the Ed-Fi ODS / API Solution

This example outlines the steps necessary to integrate and activate Ed-Fi Profile definitions for use in an Ed-Fi ODS / API. It is assumed that the Ed-Fi ODS / API has been successfully set up and is running in a local environment per the instructions in the Getting Started documentation.

To add Profiles to an Ed-Fi ODS / API, you can use the recommended approach of installing the Ed-Fi API Profiles project template for Visual Studio, or you can perform the corresponding steps manually. The instructions to use the Visual Studio template follow, and the manual steps are listed in Appendix A of this page.

The steps in Visual Studio can be summarized as:

**Downloads**

The following github link contains source files for this Profile sample:

Profile Source Files

- Step 1: Create the Profiles Project
- Step 2. Integrate Profiles into the Solution
- Step 3. Run Code Generation and Verify Changes

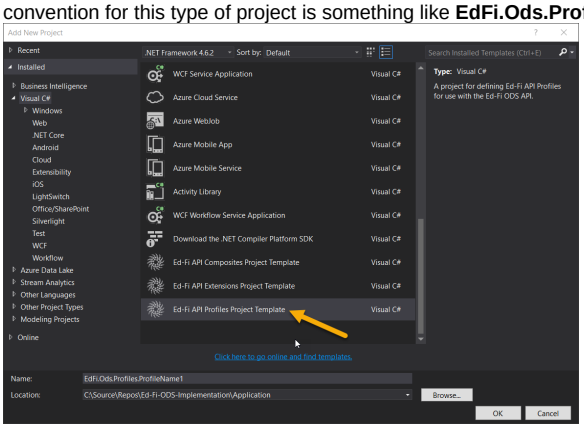
Detail on each step follows.

## Step 1: Create the Profiles Project

### Create the Visual Studio Project

1. **Add a Profiles Project Using the Visual Studio Project Template.** Visual Studio Project Template can be installed by following steps in Getting Started - Project Templates Installation s ection of this documentation.

    a. Add the Profiles project, right-click on the "Profiles" folder and selecting **File > Add > New Project...**
    b. In the "Add New Project" dialog, find the "Ed-Fi API Profiles Project Template" entry at the bottom of the Visual C# section, as shown below. Make sure you choose Microsoft .NET Framework 4.6.2 or above.
    c. Enter the project name for the new project and click **OK**. The suggested naming convention for this type of project is something like **EdFi.Ods.Profiles.MyProfiles**.




2. **Update the Marker Interface File.** To integrate the Profiles with the API, start by ensuring you have a marker interface in the root of your Profiles project.

    This interface is an idiom used in the Ed-Fi Visual Studio Solution to enable a strongly typed mechanism for obtaining a reference to the .NET assembly. If you used the Visual Studio Project template to create your profile, a file will already exist — but you'll need to rename the interface and the file to match the convention (e.g., **Marker_EdFi_Ods_Profiles_MyProfiles.cs**) . The marker interface file should have the following code:

```
namespace EdFi.Ods.Profiles.MyProfiles
{
    public interface Marker_EdFi_Ods_Profiles_MyProfiles { }
}
```

3. **Add references to extension entities**. If any of the Profile resources in the Profiles.xml document are extended, or extension entities are being constrained by a particular Profile, then project references are needed for the assemblies containing the extensions. As an example, in the Profile.xml file created by the Visual Studio template, School-and-Student-Sample has the resource School, which is extended in the **EdFi.Ods.Extensions.Sample** and **EdFi.Ods.Extensions.GrandBend** sample extensions. A hard project reference to this Ed-Fi Extension assembly must be added to the newly created Profile project.
4. **Review and modify Profiles.xml file**. The Visual Studio Project Template creates a sample **Profiles.xml** file. You should open it and modify it to meet the needs of your Profile. Consult API Profiles for guidance.
5. **Save** the Project.

# Step 2. Integrate Profiles into the Solution

To integrate the Profiles into the solution, perform the following tasks in the **EdFi.Ods.WebApi** project (located in the "Entry Points" folder):

1. Add a reference to the Profiles project you constructed in the previous step.
2. Open the **EdFi.Ods.WebApi.Startup.ConfigurationSpecificSandboxStartup.cs** file and add the code in the following steps.
    a. Add a line to include the new Profiles assembly:

**EdFi.Ods.WebApi.Startup.ConfigurationSpecificSandboxStartup.cs**

```
using EdFi.Ods.Profiles.MyProfiles;
```

    b. Locate the Configuration method in the same file and add a line for the new Profiles assembly marker to ensure it is loaded at runtime:

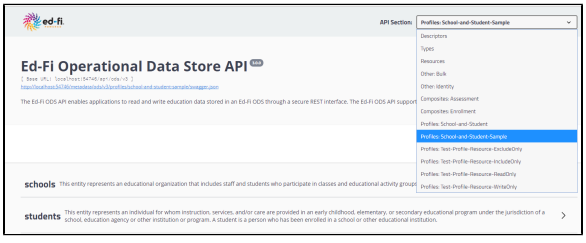
**EdFi.Ods.WebApi.Startup.ConfigurationSpecificSandboxStartup.cs**

```
Container.Register(
    Classes.
FromAssemblyContaining<Marker_EdFi_Ods_Profiles_MyProfiles>()
        .BasedOn<ApiController>()
        .LifestyleScoped());
```

# Step 3. Run Code Generation and Verify Changes

Save all modified files, close the Ed-Fi-ODS Visual Studio Solution, then re-run the code generation steps outlined in the Getting Started Guide (i.e., from a PowerShell prompt run `Initialize-PowershellForDevelopment.ps` script followed by the `initdev` command). Then run the application and view the Ed-Fi ODS / API using Swagger.

The new Profile should be visible:




# Profiles in Use

This section covers additional information and settings useful after development is complete.

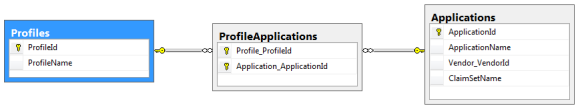
## Confirming Profile Settings

Profile settings are flexible. With this flexibility comes some complexity — so platform hosts will want to confirm that the deployed API Profiles behave as expected, exposing exactly the right resources. This can be done manually or using SDK created by utilizing code generation techniques.

## Adding Profiles to the API Administration Database

The EdFi_Admin database stores the data required to manage API keys and secrets, as well as Education Organization and Profile assignments. When the API is initialized, the names of the Profiles that have been configured into the API will be published to the Profiles table. This process performs a one-way publishing and will not remove existing Profile names that are no longer contained within the API configuration.

The EdFi_Admin tables related to this process are shown below.




# Appendix A: Adding a Profiles Project Manually

Alternatively, you can perform Step 1 manually without using the project template. To add a Profiles project to the Ed-Fi ODS / API solution manually, perform the following steps:

1. Add the Profiles project.
    a. Right-click on the "Profiles" folder and selecting **File > Add > New Project...**
    b. In the "Add New Project" dialog, find the "Class Library" entry in the Visual C# section, Make sure you choose Microsoft .NET Framework 4.6.2 or above.
    c. Enter the project name for the new project and click **OK**. The suggested naming convention for this type of project is something like **EdFi.Ods.Profiles.MyProfiles**.

2. Delete the automatically added Class1.cs file. To integrate the Profiles with the API, start by ensuring you have a marker interface in the root of your Profiles project.

    This interface is an idiom used in the Ed-Fi Visual Studio Solution to enable a strongly typed mechanism for obtaining a reference to the .NET assembly. Create an interface file to match the convention (e.g., **Marker_EdFi_Ods_Profiles_MyProfiles.cs**). The marker interface file should have the following code:

    ```
    namespace EdFi.Ods.Profiles.MyProfiles
    {
        public interface Marker_EdFi_Ods_Profiles_MyProfiles { }
    }
    ```

3. Add the following Project references:

    EdFi.Ods.Api
    EdFi.Ods.Common
    EdFi.Ods.Standard
4. Add references to extension entities:

    If any of the Ed-Fi API Profile resources in the Profiles.xml document are extended, or extension entities are being constrained by a particular Profile, then project references are needed for the assemblies containing the extensions. As an example, in the Profile.xml file provided in the following steps, School-and-Student-Sample has the resource School, which is extended in the **EdFi.Ods.Extensions.Sample** and **EdFi.Ods.Extensions.GrandBend** sample extensions. A hard project reference to this Ed-Fi Extension assembly must be added to the newly created Profile project.
5. Add a .NET assembly reference for the following:
    System.Runtime.Serialization
    System.ComponentModel.DataAnnotations

6. Save the Project and then edit the EdFi.Ods.Api.MyProfiles.csproj file in a text editor. Add the following line at the bottom:

```
  <!-- Add this line to enable code generation -->
  <Import Project="$(SolutionDir)..\..\Ed-Fi-
ODS\Application\T4TextTemplating.Targets" />
  <!-- To modify your build process, add your task inside one of the
targets below and uncomment it.
       Other similar extension points exist, see Microsoft.Common.
targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

7. Add an XML file named Profiles.xml to the root of the project, and add the appropriate Profile definitions. It should look something like the following:

```
<?xml version="1.0" encoding="utf-8" ?>
<Profiles>
  <Profile name="School-and-Student-Sample">
    <Resource name="School">
      <ReadContentType memberSelection="IncludeAll" />
      <WriteContentType memberSelection="IncludeAll" />
    </Resource>
    <Resource name="Student">
      <ReadContentType memberSelection="IncludeAll" />
      <WriteContentType memberSelection="IncludeAll" />
    </Resource>
  </Profile>
</Profiles>
```

8. In the Profiles.xml Properties, change the "Build Action" setting to "Embedded Resource".
9. Open the Package Manager Console in Visual Studio and install the following packages (consider using the `-version` flag to avoid introducing multiple versions of assemblies into the solution):

**Installing Fluent Validation**

```
Install-Package FluentValidation -ProjectName EdFi.Ods.Profiles.
MyProfiles
```

**Installing the JSON serializer**

```
Install-Package Newtonsoft.Json -ProjectName EdFi.Ods.Profiles.
MyProfiles
```

**Installing System.Web.Http**

```
Install-Package Microsoft.AspNet.WebApi.Core -ProjectName EdFi.Ods.
Profiles.MyProfiles -Version 5.2.4
```

10. Create the following folders at the root of the project: Controllers, Pipelines, Requests, and Resources. In each of the folders, add a corresponding T4 template file, as shown below:

**Controllers.tt**

```
<#@ template debug="true" hostspecific="true" language="C#" #>
<#@ include file="$(ttIncludeFolder)\Controllers.ttinclude" #>
```

**CreateOrUpdatePipelines.tt**

```
<#@ template debug="true" hostspecific="true" language="C#" #>
<#@ include file="$(ttIncludeFolder)\CreateOrUpdatePipelines.
ttinclude" #>
```

**Requests.tt**

```
<#@ template debug="true" hostspecific="true" language="C#" #>
<#@ include file="$(ttIncludeFolder)\Requests.ttinclude" #>
```

**Resources.tt**

```
<#@ template debug="true" hostspecific="true" language="C#" #>
<#@ include file="$(ttIncludeFolder)\Resources.ttinclude" #>
```

Continue through Step 2 and Step 3