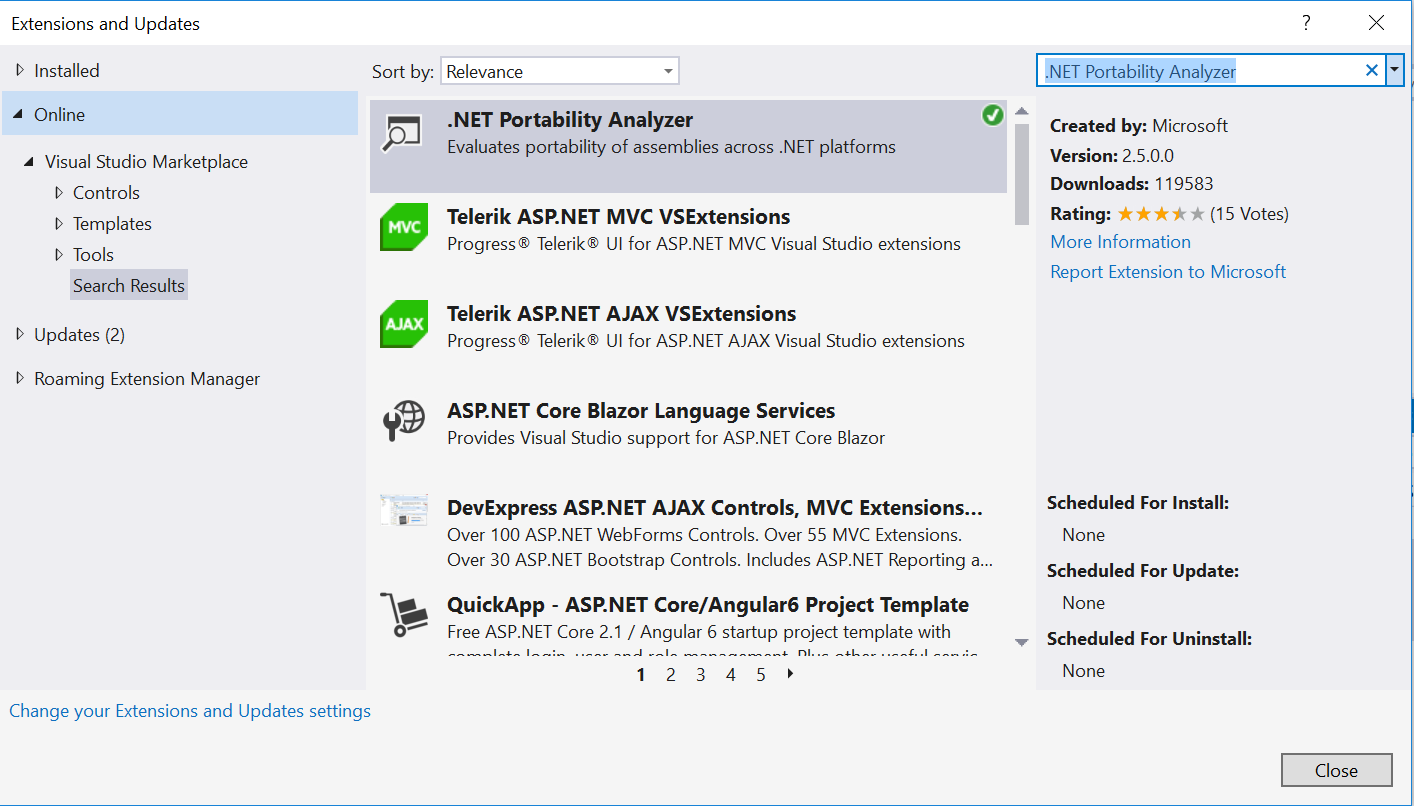
# .NET Portability Analyzer

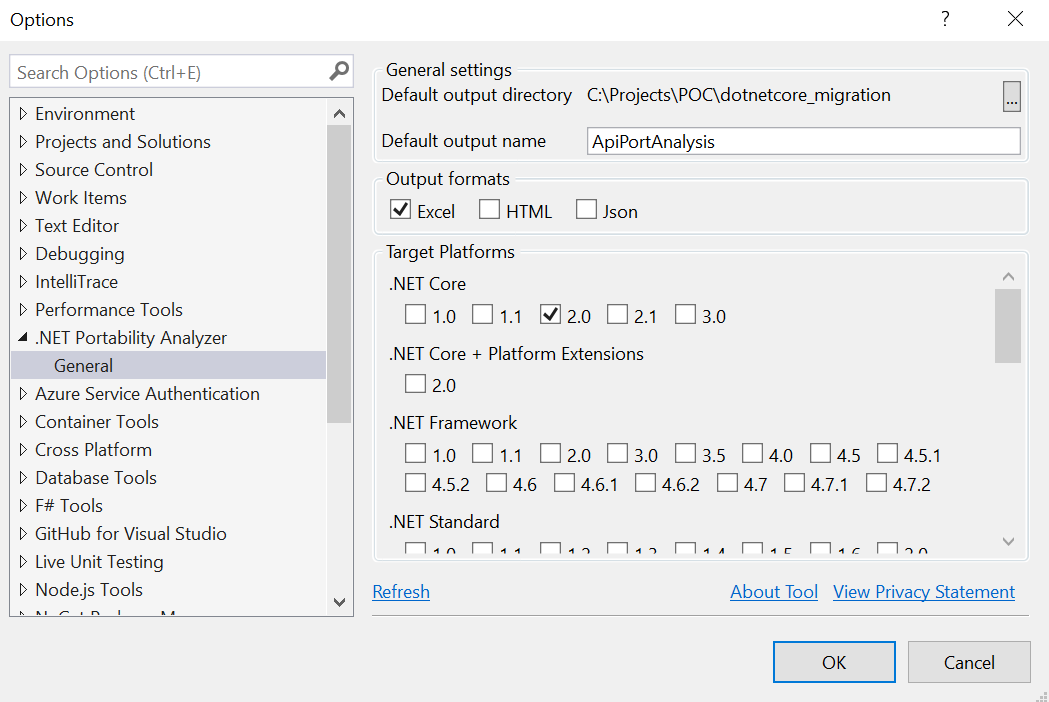
Download and install the visual studio extension for .NET Portability Analyzer from the following [link](https://marketplace.visualstudio.com/items?itemName=ConnieYau.NETPortabilityAnalyzer).

You can also install the extension from **Tools =>** **Extensions & Updates**.

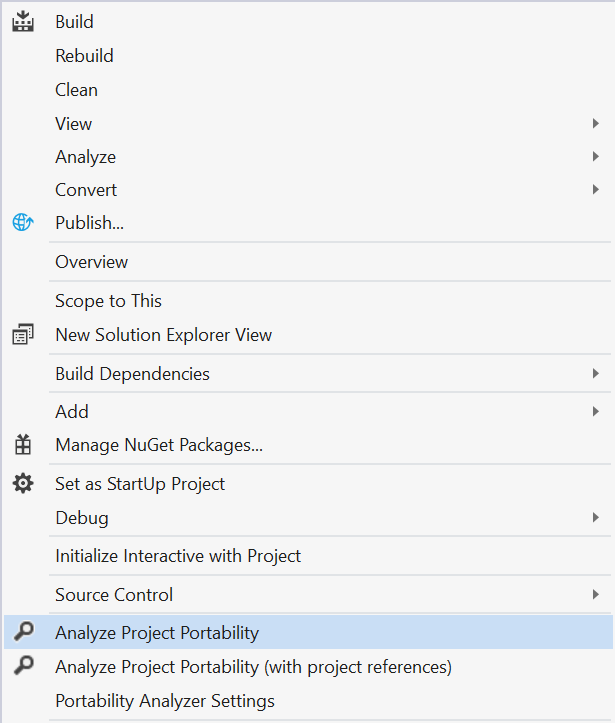


Once the extension is installed, configure the settings of .NET Portability Analyzer to define the output file name and output directory where the analyzer report will get stored, the format of the report (Excel, HTML, etc) and the Target Platform (.NET Core 2.0).

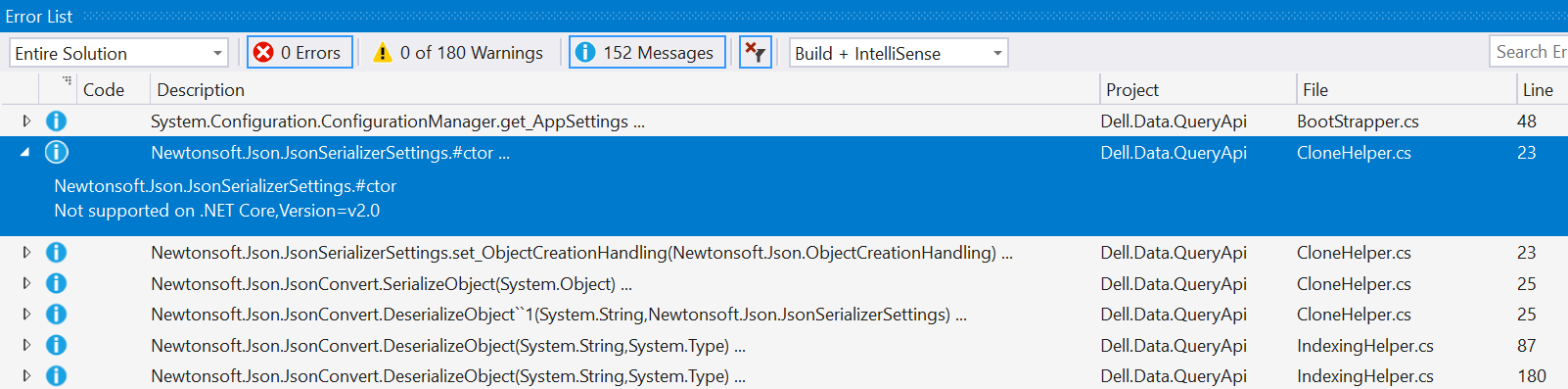
Select **Tools => Options => .NET Portability Analyzer**



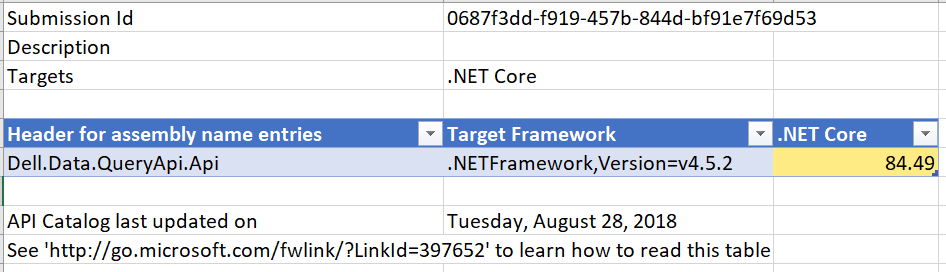
Once the settings are set, right-click individual project which you want to analyze and select **Analyze Project Portability** to build and provide you the details along with generating the report.

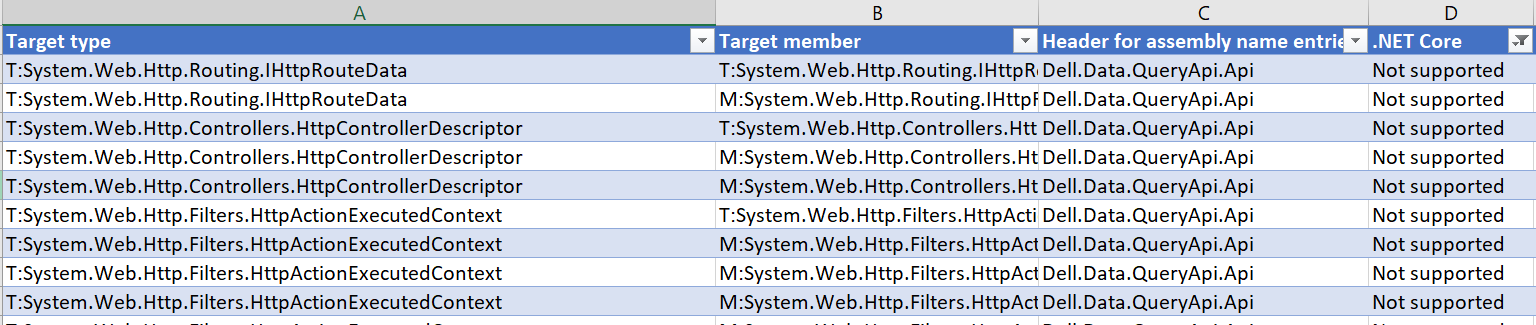


The error list will give you more details of the libraries and packages not supported in .NET Core



You can also view the full report which will provide you the list of assemblies not supported in .NET Core





# Unsupported Libraries or Packages

|  |
| --- |
|  |
| Newtonsoft.Json is not supported. |
| Need to find out the possible solution to use this library  Ref: <https://medium.com/@agriciuc/working-with-json-net-in-net-core-rc2-part-1-5f3a65f4e11> |
| System.Configuration.ConfigurationManager.get\_AppSettings is not supported |
| ConfigurationManager is not available under System.Configuration in .NET Core  Solution: Install NUGET package for the Configuration Manager  Install-Package System.Configuration.ConfigurationManager -Version 4.5.0 |
| System.Web.Mvc.HandleErrorAttribute not supported |
| Ref: <https://forums.asp.net/t/2125006.aspx?How+to+use+exception+filters+in+the+asp+net+core+web+project>  Need to use ExceptionFilterAttribute  Adding an ApplicationLogging static class that includes a static ILoggerFactory instance:  public static class ApplicationLogging  {    public static ILoggerFactory LoggerFactory {get;} = new LoggerFactory();    public static ILogger CreateLogger<T>() =>      LoggerFactory.CreateLogger<T>();  }  public class CustomExceptionFilterAttribute : ExceptionFilterAttribute { ILogger Logger { get; } =     ApplicationLogging.CreateLogger<CustomExceptionFilterAttribute>();// to tell where we log   public override void OnException(ExceptionContext context) {        using (Logger.BeginScopeImpl(       $"=>{ nameof(OnException) }")) // to tell which method we log     {       Logger.LogInformation("Log Message"); // to tell what exception we log  }  } }  Or to  use ****Microsoft.IdentityModel.Logging****directly:  public override void OnException(ExceptionContext context) {  Microsoft.IdentityModel.Logging.LogHelper.LogExceptionMessage(context.Exception );  install*****Microsoft.IdentityModel.Logging*****Nuget; |
| System.Web.Http.Description.ApiParameterDescription or ApiDescription is not supported |
| Need to install package  Install-Package Microsoft.AspNetCore.Mvc.ApiExplorer -Version 2.1.2 |
| Configuring route using MapHttpRoute in WebApiConfig.cs is not supported |
| We need to define the default routes in Startup.cs file  public class Startup  {      public void ConfigureServices(IServiceCollection services)      {          services.AddMvc();      }        public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)      {  app.UseMvc(routes =>  {      //New Route      routes.MapRoute(         name: "about-route",         template: "about",         defaults: new { controller = "Home", action = "About" }      );    routes.MapRoute(      name: "default",      template: "{controller=Home}/{action=Index}/{id?}");  });      }  } |
| System.Web.Mvc.Controller is not supported |
| Install the NUGET package  Install-Package Microsoft.AspNetCore.Mvc -Version 2.1.2 |
|  |

# Internal System Dependencies

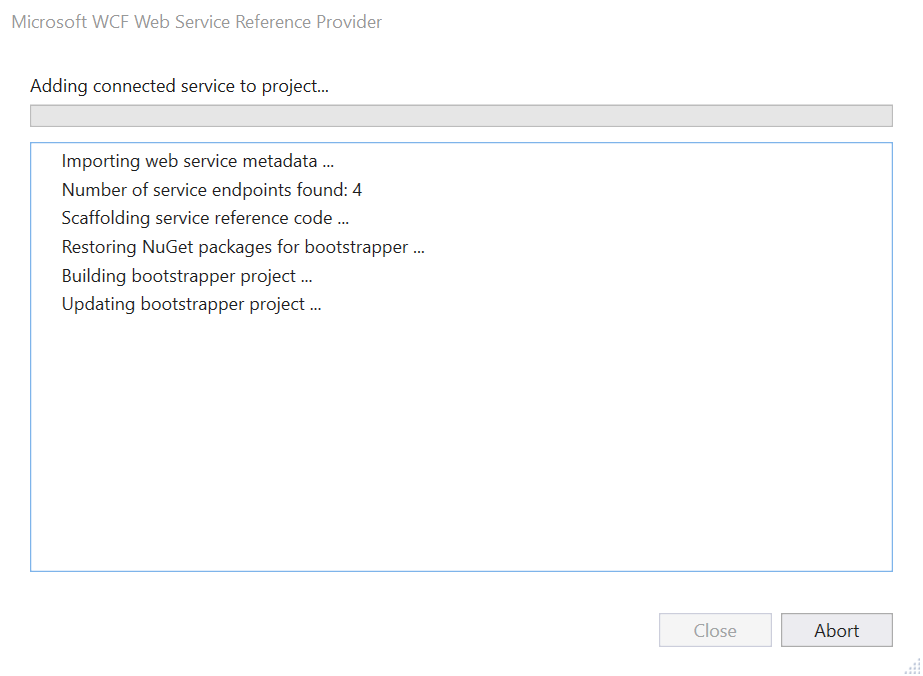
|  |  |
| --- | --- |
| ConfigurationManager.AppSettings |  |
| ConfigurationManager.OpenExeConfiguration |  |
| System.Configuration.KeyValueConfigurationCollection.AllKeys | Configuration config = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);  if (!config.AppSettings.Settings.AllKeys.Contains(settingName)) |
| System.Configuration.AppSettingsSection |  |
| System.Configuration.Configuration |  |
| System.Configuration.ConfigurationManager |  |
| System.Web.Hosting.HostingEnvironment.MapPath |  |
| System.Web.Mvc.HandleErrorAttribute |  |
| System.Web.Mvc.GlobalFilterCollection.Add() |  |
| System.Web.Mvc.RouteCollectionExtensions.IgnoreRoute() |  |
| System.Web.Mvc.RouteCollectionExtensions.MapRoute() |  |
| System.Web.Http.GlobalConfiguration |  |
| System.Web.Http.HttpConfiguration |  |
| System.Net.Http.Formatting.MediaTypeFormatterCollection |  |
| System.Net.Http.Formatting.BaseJsonMediaTypeFormatter |  |
| System.Web.HttpRequest |  |
| System.Web.HttpApplication |  |
| System.Web.Routing |  |
| System.Web.Mvc.GlobalFilters |  |
| System.Web.Mvc.AreaRegistration |  |
| System.Web.Http.Routing |  |
| System.Web.Http.Controllers |  |
| System.Web.Http.Filters |  |
| System.Net.Http.HttpRequestMessageExtensions |  |
| System.Web.Http.ApiController |  |
| System.Web.Caching.Cache |  |
| System.Web.HttpRuntime |  |
| System.Web.Http.Controllers.ReflectedHttpActionDescriptor |  |
| System.Web.Http.Controllers.ReflectedHttpParameterDescriptor |  |
| System.Web.Http.Controllers.HttpControllerDescriptor |  |
| System.Web.Http.Controllers.HttpParamterDescriptor |  |
| System.Net.Http.Formatting.MediaTypeFormatter |  |
| System.Net.Http.ObjectContent |  |
| System.Web.Http.Description.ApiDescription |  |
| System.Web.Http.Description.ApiParameterDescription |  |
| System.Web.Http.Description.ApiExplorerSettingsAttribute |  |
| System.Web.Http.Description.IApiExplorer |  |
|  |  |

# External Dependencies

|  |  |
| --- | --- |
| AppFabric | Not required |
| AutoMapper | Required, Support in .NET core 2.0 |
| ElasticSearch.Net | Required, Support in .NET Core  Ref: <https://medium.com/@cecildt/quick-start-using-elasticsearch-kibana-with-asp-net-core-2-0-6f179b6d2d02> |
| Log4Net | Required, Support in .NET Core  Ref: <https://stackoverflow.com/questions/46169606/how-to-use-log4net-in-asp-net-core-2-0> |
| Log4net.ElasticSearch | Not Required. Doesn’t support in .NET core |
| NEST | Required, Support .NET core  Ref: <https://damienbod.com/2017/02/28/implementing-an-audit-trail-using-asp-net-core-and-elasticsearch-with-nest/> |
| Newtonsoft.Json | Required, Doesn’t support in .NET Core.  Ref: <https://stackoverflow.com/questions/46192854/how-to-work-with-asp-net-core-2-0-and-newtonsoft-json> |
| RestSharp | Required, Not supported in .NET Core  Some Observations:  <https://stackoverflow.com/questions/41390647/how-to-use-restsharp-netcore-in-asp-net-core>  <https://stackoverflow.com/questions/46954398/rest-client-for-net-core-2> |
| StructureMap | Required, Support .NET Core  Ref: <https://andrewlock.net/getting-started-with-structuremap-in-asp-net-core/> |
| Angularjs | Not Required |
| Antlr | Not Required |
| Bootstrap | Not Required |
| Dotless | Not Requried |
| DotNetZip | Not Required |
| jQuery | Not Required |
| Modernizr | Not Required |
| Respond | Not Required |
| Swashbuckle | Requried, Supported in ASP.NET Core  Ref: <https://docs.microsoft.com/en-us/aspnet/core/tutorials/web-api-help-pages-using-swagger?view=aspnetcore-2.1>  <https://docs.microsoft.com/en-us/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-2.1&tabs=visual-studio%2Cvisual-studio-xml>  <https://elanderson.net/2017/10/swagger-and-swashbuckle-with-asp-net-core-2/> |
| WebActivator | Might be Required for Swashbuckle but not support in Asp.NET core  Can implement ReactJS  <https://reactjs.net/getting-started/aspnetcore.html> |
| **Dell Components need to get reviewed** | |
| Dell.Authoring.CoreV5 |  |
| Dell.Authoring.SecurityV5 |  |
| Dell.Data.ElasticClient.PersisterV5 |  |
| Dell.MetaRiver.ComponentsV5 |  |
|  |  |

Adding Connected Service like WCF service has a issue if the version of the application is .NET Core 2.1. Error says “An unknown error occurred while invoking the service metadata component. Failed to generate service reference.”

Changed the .NET Core version from 2.1 to 2.0 and it will work.



Elasticsearch.Net (>= 2.4.1)

NEST (>= 2.4.1)

Newtonsoft.Json (>= 9.0.1)

structuremap (>= 3.1.6.186)

log4net (>= 2.0.5)

log4net.ElasticSearch (>= 2.3.3)

AppFabric (>= 2.1.0.1)

# Application Changes for .NET Core 2.1 Compatibility

## Dell.Data.QueryApi.Api (Web API)

**Step 1:** Copy all the appSettings (key/value) pair from web.config to appsettings.json file of .net core application

**Web.Config -**

<appSettings>

<add key="ApplicationID" value="44ddf7eb-1e13-44a9-bcdf-c22455fa983e" />

</appSettings>

**Appsettings.json**

{

"appSettings": {

"ApplicationID": "44ddf7eb-1e13-44a9-bcdf-c22455fa983e"

}

}

**Step 2:** Access the appSettings value using the following piece of code

**ASP.NET –**

ConfigurationManager.AppSettings["CacheRefreshInterval"]

**.NET Core –**

public static class AppConfiguration

{

public static string GetSettingsValue(string settingName)

{

IConfiguration config = new ConfigurationBuilder().SetBasePath(System.AppContext.BaseDirectory).AddJsonFile("appsettings.json").Build();

return config[$"Logging:{settingName}"];

}

}

var cacheRefreshInterval = AppConfiguration.GetSettingsValue("appSettings:CacheRefreshInterval");

**Step 3:** Enabling Swagger capabilities.

After installation of Swashbuckle.AspNetCore, you need to add the following settings in Startup.cs file.

public void ConfigureServices(IServiceCollection services)

{

services.AddMvc();

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new Info { Title = "QueryApi", Version = "v1" });

c.IncludeXmlComments(GetXmlCommentsPath());

c.ResolveConflictingActions(apiDescriptions => apiDescriptions.First());

c.DescribeAllEnumsAsStrings();

});

}

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

app.UseSwagger();

app.UseStaticFiles();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "Dell Query API V1");

c.RoutePrefix = string.Empty;

});

app.UseMvc();

}

private string GetXmlCommentsPath()

{

return System.IO.Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "Dell.Data.QueryApi.Api.xml");

}

Change your appSettings.js file to provide the launchUrl to Index.html which will be the endpoint for swagger.

"profiles": {

"IIS Express": {

"commandName": "IISExpress",

"launchBrowser": true,

"launchUrl": "index.html",

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

}

In your controller, you need to provide the following attributes for your controller class as well as for actions.

[Produces("application/json")]

[Route("api/[controller]/[action]")]

public class MicroContentController : Controller

{

/// Your block of code

}

[HttpPost()]

[ActionName("GetMicrocontent")]

[ProducesResponseType(typeof(MicroContentRequest), 200)]

[ProducesResponseType(typeof(void), 400)]

[ProducesResponseType(typeof(void), 404)]

public async Task<IActionResult> GetMicroContent(MicroContentRequest microContentRequest)

{

/// Your block of code

}

**Step 4:**

Existing implementation of StructureMapDependencyResolver and StructureMapScope can be depreciated as System.Web.Http.Dependencies is not available in .NET Core.

You can use IServiceCollection to add all the dependencies required in StartUp.cs file

public void ConfigureServices(IServiceCollection services)

{

services.AddSingleton<IMicroContentProvider, MicroContentProvider>();

services.AddSingleton<IQueryProcessor, ContentQueryProcessor>();

var container = new Container();

container.Configure(config =>

{

config.Populate(services);

});

var ApplicationId = AppConfiguration.GetSettingsValue("ApplicationID");

var IsApprovedContentRequest = Core.Utils.Utils.IsApprovedContentRequest();

container

.With("applicationId").EqualTo(ApplicationId)

.With("approvedContentOnly").EqualTo(IsApprovedContentRequest);

Core.ServiceLocator.Container = container;

}

Here **Core.ServiceLocator** is the static class in Dell.Data.QueryApi.Core that can be used anywhere in the application.

public static class ServiceLocator

{

/// <summary>

/// Gets or sets the container.

/// </summary>

/// <value>

/// The container.

/// </value>

public static IContainer Container { get; set; }

}

**Step 5:** Using RestSharp in .NET Core

Dell.QueryApi.External.Common has ApiHelper which is using RestSharp in following format.

public static IRestResponse Execute(string apiUrl, string request)

{

try

{

var client = new RestClient(apiUrl);

var apiRequest = new RestRequest(Method.POST);

apiRequest.AddHeader("Content-Type", "application/json");

apiRequest.AddHeader("Accept", "application/json");

apiRequest.RequestFormat = DataFormat.Json;

client.Timeout = 120000;

apiRequest.AddParameter("application/json", request, ParameterType.RequestBody);

return client.Execute(apiRequest);

}

catch (Exception ex)

{

//todo::Logging

throw;

}

}

After installation of RestSharp.NetCore, we need to change the implementation in following way.

Create a static class extending the functionality of RestSharp

public static class RestClientExtensions

{

public static async Task<RestResponse> ExecuteAsync(this RestClient client, RestRequest request)

{

TaskCompletionSource<IRestResponse> taskCompletion = new TaskCompletionSource<IRestResponse>();

RestRequestAsyncHandle handle = client.ExecuteAsync(request, r => taskCompletion.SetResult(r));

return (RestResponse)(await taskCompletion.Task);

}

}

Now modify the implementation of RestSharp in ApiHelper

public static async Task<IRestResponse> ExecuteAsync(string apiUrl, string request)

{

try

{

var client = new RestClient(apiUrl);

var apiRequest = new RestRequest(Method.POST);

apiRequest.AddHeader("Content-Type", "application/json");

apiRequest.AddHeader("Accept", "application/json");

apiRequest.RequestFormat = DataFormat.Json;

client.Timeout = 120000;

apiRequest.AddParameter("application/json", request, ParameterType.RequestBody);

return await client.ExecuteAsync(apiRequest);

}

catch (Exception ex)

{

//todo::Logging

throw;

}

}

Change response.StatusCode == HttpStatusCode.OK to response.Result.StatusCode == HttpStatusCode.OK

And response.Content to response.Result.Content in ContentStudioClient.

Step 6:

Project: Dell.Authoring.ContractsV5.EntitiesV2

Collection (Class) –

System.ComponentModel.DataAnnotations has been replaced by System.ComponentModel.Annotations. Add this as a package.

Step 7:

PersistTranslatedTargetDocumentHelper in Dell.Authoring.CoreV5.Helper is using HostingEnvironment.MapPath for identifying the path of a file stored in App\_Data. HostingEnvironment use System.Web.Hosting which is not available for .NET Core

Instead we can use Microsoft.AspNetCore.Hosting to inject the IHostingEnvironment and use the dependency for getting the file path.

private readonly IHostingEnvironment \_hostingEnvironment;

public PersistTranslatedTargetDocumentHelper(IHostingEnvironment hostingEnvironment)

{

\_hostingEnvironment = hostingEnvironment;

}

Then replace the following code –

var filePath = HostingEnvironment.MapPath(@"~/App\_Data/TranslationOverrideConfig.txt");

However since it is a static class where we cannot inject the dependencies, the other way to go for it is creating an extension like this

public static class FilePathExtension

{

public static string ToApplicationPath(this string fileName)

{

var exePath = Path.GetDirectoryName(System.Reflection

.Assembly.GetExecutingAssembly().CodeBase);

Regex appPathMatcher = new Regex(@"(?<!fil)[A-Za-z]:\\+[\S\s]\*?(?=\\+bin)");

var appRoot = appPathMatcher.Match(exePath).Value;

return Path.Combine(appRoot, fileName);

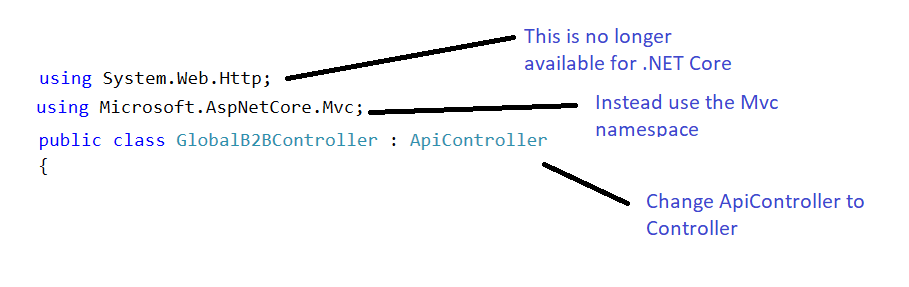
}

}

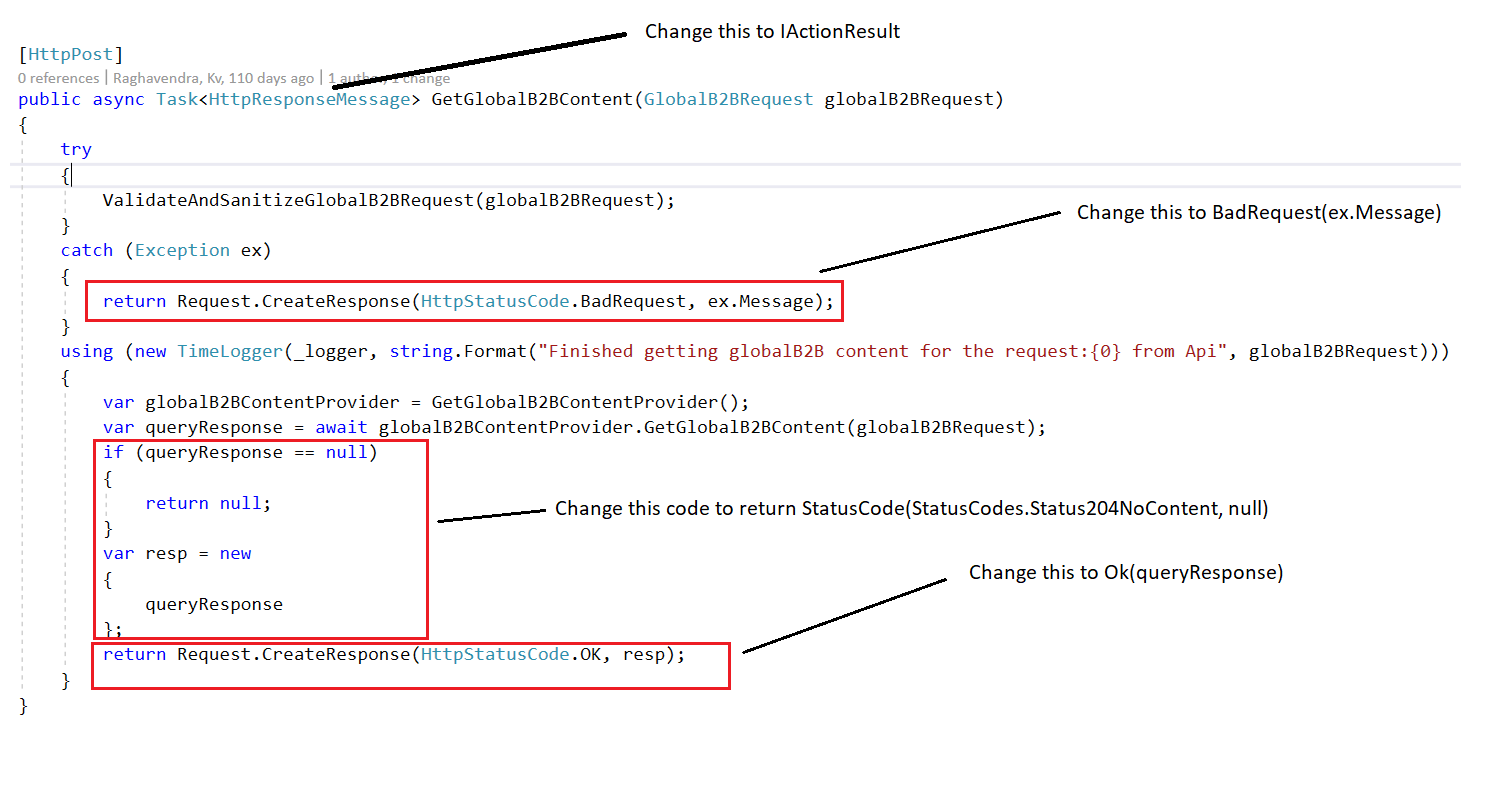
And then using it to get the file path.

var filePath = @"App\_Data\TranslationOverrideConfig.txt".ToApplicationPath();

**Controller Change:**



Old code



New Code



For returning **Internal Server Error**, you can use



**StatusCodes** like StatusCodes.Status204NoContent are derived from the following namespace



Remove **JavaScriptSerializer()** since System.Web.Script.Serialization under System.Web.Extensions is no longer supported in .NET Core



To **Newtonsoft JsonConvert**

