Below is the reformatted text document with improved wording and structure. You can save the content as a plain text file (or Markdown file) and use it as a guide for implementing API versioning in an ASP.NET Core 8.0 project.

**API Versioning in ASP.NET Core 8.0**

This document describes two approaches to API versioning: **by folder structure** and **by NuGet package**. Follow the steps below to implement versioning in your project.

**Versioning by Folder Structure**

1. **Create Folders**
   * Create two folders in your project: **V1** and **V2**.
2. **Copy Controllers**
   * Copy the entire **Controllers** folder and paste it into both the **V1** and **V2** folders.
3. **Remove the Original Controllers Folder**
   * Delete the original **Controllers** folder from the project.
4. **Change the Namespaces**
   * In the controllers within the **V1** folder, update the namespace to:
   * namespace Countrie.V1.Controllers
   * In the controllers within the **V2** folder, update the namespace to:
   * namespace Countrie.V2.Controllers
5. **Change the Route Attribute**
   * For version 1 controllers, use:
   * [Route("api/v1/[controller]")]
   * For version 2 controllers, use:
   * [Route("api/v2/[controller]")]
6. **Update DTOs and Domain Models**
   * Duplicate the DTOs and domain models similarly, e.g., create **DTOV1** and **DTOV2** folders, and update their namespaces accordingly.

**Versioning by NuGet Package**

**A. Install Required NuGet Packages**

* Install the following NuGet packages:
  + Microsoft.AspNetCore.Mvc.Versioning
  + Microsoft.AspNetCore.Mvc.Versioning.ApiExplorer

**B. Organize Domain Models**

* Organize your domain models into version-specific folders (e.g., **DTOV1** and **DTOV2**).

**C. Add API Versioning Services in Program.cs**

1. Register controllers and API versioning:
2. builder.Services.AddControllers();
3. builder.Services.AddApiVersioning();
4. Set up the default version so that if the client does not specify one, version 1.0 is used:
5. builder.Services.AddControllers();
6. builder.Services.AddApiVersioning(options =>
7. {
8. options.AssumeDefaultVersionWhenUnspecified = true;
9. });

**D. Create a Versioned Controller**

Below is an example controller that supports two API versions.

**CountriesController.cs:**

namespace versioningByNuget.Controllers

{

[Route("api/v{version:apiVersion}/[controller]")]

[ApiController]

[ApiVersion("1.0")]

[ApiVersion("2.0")]

public class CountriesController : ControllerBase

{

[MapToApiVersion("1.0")]

[HttpGet]

public IActionResult GetV1()

{

var countries = CountrieData.GetCountries();

var DTO = new List<CountrieDTOV1>();

foreach(var cntry in countries)

{

DTO.Add(

new CountrieDTOV1

{

Id = cntry.Id,

Name = cntry.Name

});

}

return Ok(DTO);

}

[MapToApiVersion("2.0")]

[HttpGet]

public IActionResult GetV2()

{

var countries = CountrieData.GetCountries();

var DTO = new List<CountrieDTOV2>();

foreach (var cntry in countries)

{

DTO.Add(

new CountrieDTOV2

{

Id = cntry.Id,

CountrieName = cntry.Name

});

}

return Ok(DTO);

}

}

}

**E. Set Up Swagger Support for Versioning**

1. **Configure API Versioning in Program.cs:**
2. builder.Services.AddControllers();
3. builder.Services.AddApiVersioning(options =>
4. {
5. options.AssumeDefaultVersionWhenUnspecified = true;
6. options.DefaultApiVersion = new Microsoft.AspNetCore.Mvc.ApiVersion(1, 0);
7. options.ReportApiVersions = true;
8. });
9. builder.Services.AddVersionedApiExplorer(options =>
10. {
11. options.GroupNameFormat = "'v'VVV";
12. options.SubstituteApiVersionInUrl = true;
13. });
14. // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
15. builder.Services.AddEndpointsApiExplorer();
16. builder.Services.AddSwaggerGen();
17. var app = builder.Build();
18. var versionDescriptionProvider = app.Services.GetRequiredService<IApiVersionDescriptionProvider>();
19. // Configure the HTTP request pipeline.
20. if (app.Environment.IsDevelopment())
21. {
22. app.UseSwagger();
23. app.UseSwaggerUI(options =>
24. {
25. foreach (var description in versionDescriptionProvider.ApiVersionDescriptions)
26. {
27. options.SwaggerEndpoint($"/swagger/{description.GroupName}/swagger.json", description.GroupName.ToUpperInvariant());
28. }
29. });
30. }
31. **Configure Swagger Options**

Create a new class **ConfigureSwaggerOptions.cs** in the root:

namespace versioningByNuget

{

public class ConfigureSwaggerOptions : IConfigureOptions<SwaggerGenOptions>

{

private readonly IApiVersionDescriptionProvider apiVersionDescriptionProvider;

public ConfigureSwaggerOptions(IApiVersionDescriptionProvider apiVersionDescriptionProvider)

{

this.apiVersionDescriptionProvider = apiVersionDescriptionProvider;

}

public void Configure(string? name, SwaggerGenOptions options)

{

Configure(options);

}

public void Configure(SwaggerGenOptions options)

{

foreach (var item in apiVersionDescriptionProvider.ApiVersionDescriptions)

{

options.SwaggerDoc(item.GroupName, CreateVersionInfo(item));

}

}

private OpenApiInfo CreateVersionInfo(ApiVersionDescription description)

{

var info = new OpenApiInfo

{

Title = "Your versioned API",

Description = description.ApiVersion.ToString(),

};

return info;

}

}

}

1. **Inject Swagger Options in Program.cs:**
2. // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
3. builder.Services.AddEndpointsApiExplorer();
4. builder.Services.AddSwaggerGen();
5. builder.Services.ConfigureOptions<ConfigureSwaggerOptions>(); // Add this line
6. var app = builder.Build();

**F. Usage in Swagger**

* Once everything is set up, open Swagger (usually at http://localhost:5000/swagger).
* You will see a **version dropdown** in the top right corner.
* You can switch between different versions (e.g., v1.0 and v2.0) and test the endpoints accordingly.

**Example URLs**

* **Version 1**:
* https://localhost:7082/api/Countries?api-version=1.0
* **Version 2**:
* https://localhost:7082/api/Countries?api-version=2.0

**End of Document**

You can now save the text above into a file (e.g., APIVersioningGuide.txt or APIVersioningGuide.md) and use it as a reference to understand and implement API versioning in an ASP.NET Core 8.0 project.