Got it — you have a **C# API client generated by** `openapi-generator-cli` and the generated API class constructor looks something like this:

```
public MyApi(
    ILogger<MyApi> logger,
    ILoggerFactory loggerFactory,
    HttpClient httpClient,
    JsonSerializerOptions jsonSerializerOptions,
    IApiEvents apiEvents
)
```

You want to **pass a custom** `baseUrl` when creating the API client.

1️⃣ Understanding the Generated Client

When you generate a C# client with `openapi-generator-cli` (especially with `--library httpclient` or `restsharp`), the `HttpClient` you pass to the constructor determines the base URL.

So, to set the base URL, you **configure the** `HttpClient.BaseAddress` before passing it to the API class.

2️⃣ Example: Passing `baseUrl` with All Constructor Parameters

```
using System;
using System.Net.Http;
using System.Text.Json;
using Microsoft.Extensions.Logging;
using MyGeneratedApiNamespace; // Replace with your generated namespace
```

```csharp
class Program
{
    static void Main()
    {
        // 1. Create a logger factory
        using var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });

        // 2. Create a logger for the API
        var logger = loggerFactory.CreateLogger<MyApi>();

        // 3. Create and configure HttpClient with base URL
        var httpClient = new HttpClient
        {
            BaseAddress = new Uri("https://api.example.com/") // <-- Your base URL here
        };

        // 4. Configure JSON serializer options (optional)
        var jsonOptions = new JsonSerializerOptions
        {
            PropertyNamingPolicy = JsonNamingPolicy.CamelCase,
            WriteIndented = true
        };

        // 5. Implement or use a provided IApiEvents (if generated)
```

```
        IApiEvents apiEvents = new MyApiEventsImplementation();

        // 6. Create the API client
        var api = new MyApi(
            logger,
            loggerFactory,
            httpClient,
            jsonOptions,
            apiEvents
        );

        // 7. Use the API
        var result = api.GetSomeDataAsync().Result;
        Console.WriteLine(result);
    }
}

// Example implementation of IApiEvents
public class MyApiEventsImplementation : IApiEvents
{
    public void OnRequest(HttpRequestMessage request)
    {
        Console.WriteLine($"Request: {request.Method} {request.RequestUri}");
    }

    public void OnResponse(HttpResponseMessage response)
    {
```

```
        Console.WriteLine($"Response: {response.StatusCode}");
    }
}
```

3️⃣ Key Points

- `HttpClient.BaseAddress` is the correct way to set the base URL.
- You **must** pass all required constructor parameters — `logger`, `loggerFactory`, `httpClient`, `jsonSerializerOptions`, and `apiEvents`.
- If you don't have a real `IApiEvents` implementation, you can pass a **no-op** implementation.

✅ This approach works regardless of whether your generated client uses `System.Net.Http.HttpClient` or `RestSharp`, as long as the constructor accepts an `HttpClient`.

If you want, I can also show you **how to wrap this in a dependency injection setup** so that the base URL is configured automatically in `Startup` or `Program.cs`.
Do you want me to prepare that DI-based version?