**Swagger code converted to C#**

**Please flow the below details.**

Installation Tool

1. JDK17

2. swagger-codegen-cli-3.0.34.jar

3.Visual Studio 2022

Step:

1. Code generation command 🡪java -jar swagger-codegen-cli-3.0.34.jar generate -i swagger.yaml -l csharp -o GeneratedCSharpAPI
2. Open VS2022->Openfolder->Select project folder then open

Open CustomersAddressesApiController-> CreateCustomerAddress Method

Code:

Convert the C# object to a JSON string using XsdHelper

public string ConvertObjectToJson<T>(T objectToConvert)

{

if (objectToConvert == null)

throw new ArgumentNullException(nameof(objectToConvert));

// Serialize the object to JSON

return JsonConvert.SerializeObject(objectToConvert);

}

// Method to convert JSON string to XML string

public string ConvertJsonToXml(string json)

{

XmlDocument doc = JsonConvert.DeserializeXmlNode(json, "Root");

return doc.OuterXml;

}

public async Task<string> SendSoapRequest(string soapRequest)

{

// Authorize.Net API credentials

string apiLoginID = ""; // Your actual API Login ID

string transactionKey = ""; // Your actual Transaction Key

// Create the authorization value in base64

string authValue = Convert.ToBase64String(Encoding.UTF8.GetBytes($"{apiLoginID}:{transactionKey}"));

// Set the Base URL for the Authorize.Net API

httpClient.BaseAddress = new Uri("https://apitest.authorize.net");

// Set the Authorization header

httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", authValue);

// The SOAP/REST API endpoint URL for the request

//string apiEndpoint = "/xml/v1/customers/924517851/addresses";

string apiEndpoint = "https://apitest.authorize.net/xml/v1/customers/924517851/addresses";

try

{

// Set up the content for the SOAP request

var httpContent = new StringContent(soapRequest, Encoding.UTF8, "text/xml");

// Send POST request

var response = await httpClient.PostAsync(apiEndpoint, httpContent);

// Check if the response is successful

if (response.IsSuccessStatusCode)

{

// Read the response content

string result = await response.Content.ReadAsStringAsync();

// Convert XML to JSON if needed (you may use the method provided in your codebase)

string jsonResult = ConvertXmlToJson(result); // Assuming you have a ConvertXmlToJson method

return jsonResult;

}

else

{

//Console.WriteLine("Error in the API request: " + response.ReasonPhrase);

//return null;

string jsonResult = ConvertXmlToJson(soapRequest); // Assuming you have a ConvertXmlToJson method

return jsonResult;

}

}

catch (Exception ex)

{

Console.WriteLine("Exception occurred while calling the API: " + ex.Message);

return null;

}

}

Change in JASON order not impact XML order

1.Orginal Jason

{

"address1": "19 A",

"administrativeArea": "KOL",

"company": "LMC",

"country": "test",

"\_Default": true,

"lastName": "MM",

"firstName": "SM",

"locality": "KOL",

"phoneNumber": "1234567890",

"postalCode": "700000"

}

2.Change Jason:

{

"phoneNumber": "1234567890",

"lastName": "MM",

"administrativeArea": "KOL",

"company": "LMC",

"country": "test",

"\_Default": true,

"firstName": "SM",

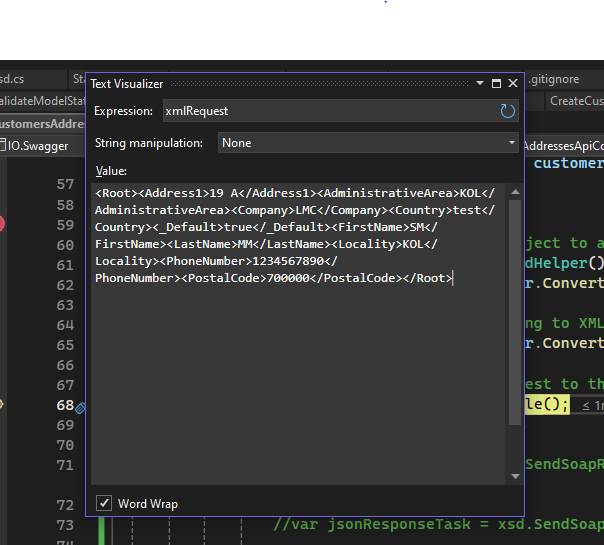
"locality": "KOL",

"postalCode": "700000"

"address1": "19 A",

}

3.XML result:

  
Send SOP request:

public async Task<string> SendSoapRequest(string soapRequest)

{

// Authorize.Net API credentials

string apiLoginID = ""; // Your actual API Login ID

string transactionKey = ""; // Your actual Transaction Key

// Create the authorization value in base64

string authValue = Convert.ToBase64String(Encoding.UTF8.GetBytes($"{apiLoginID}:{transactionKey}"));

// Set the Base URL for the Authorize.Net API

httpClient.BaseAddress = new Uri("https://apitest.authorize.net");

// Set the Authorization header

httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", authValue);

// The SOAP/REST API endpoint URL for the request

//string apiEndpoint = "/xml/v1/customers/924517851/addresses";

string apiEndpoint = "https://apitest.authorize.net/xml/v1/customers/924517851/addresses";

try

{

// Set up the content for the SOAP request

var httpContent = new StringContent(soapRequest, Encoding.UTF8, "text/xml");

// Send POST request

var response = await httpClient.PostAsync(apiEndpoint, httpContent);

// Check if the response is successful

if (response.IsSuccessStatusCode)

{

// Read the response content

string result = await response.Content.ReadAsStringAsync();

// Convert XML to JSON if needed (you may use the method provided in your codebase)

string jsonResult = ConvertXmlToJson(result); // Assuming you have a ConvertXmlToJson method

return jsonResult;

}

else

{

//Console.WriteLine("Error in the API request: " + response.ReasonPhrase);

//return null;

string jsonResult = ConvertXmlToJson(soapRequest); // Assuming you have a ConvertXmlToJson method

return jsonResult;

}

}

catch (Exception ex)

{

Console.WriteLine("Exception occurred while calling the API: " + ex.Message);

return null;

}

}

For XSD implementation:

1. Add create XSD file:

<updateCustomerShippingAddressRequest xmlns="AnetApi/xml/v1/schema/AnetApiSchema.xsd"> <merchantAuthentication> <name>5KP3u95bQpv</name> <transactionKey>346HZ32z3fP4hTG2</transactionKey> </merchantAuthentication> <customerProfileId>10000</customerProfileId> <address> <firstName>Newfirstname</firstName> <lastName>Doe</lastName> <company></company> <address>123 Main St.</address> <city>Bellevue</city> <state>WA</state> <zip>98004</zip> <country>US</country> <phoneNumber>000-000-0000</phoneNumber> <faxNumber></faxNumber> <customerAddressId>30000</customerAddressId> </address> <defaultShippingAddress>false</defaultShippingAddress> </updateCustomerShippingAddressRequest>

2. public async Task<string> SendSoapRequest(string soapRequest, string xsdFilePath= "C:/Temp/New.xsd")

{

// Authorize.Net API credentials

string apiLoginID = ""; // Your actual API Login ID

string transactionKey = ""; // Your actual Transaction Key

// Create the authorization value in base64

string authValue = Convert.ToBase64String(Encoding.UTF8.GetBytes($"{apiLoginID}:{transactionKey}"));

// Set the Base URL for the Authorize.Net API

using (var httpClient = new HttpClient())

{

httpClient.BaseAddress = new Uri("https://apitest.authorize.net");

// Set the Authorization header

httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Basic", authValue);

// The SOAP/REST API endpoint URL for the request

string apiEndpoint = "https://apitest.authorize.net/xml/v1/customers/924517851/addresses";

//Validate the SOAP request XML against the XSD schema

if (!ValidateXmlAgainstXsd(soapRequest, xsdFilePath))

{

Console.WriteLine("XML validation failed against the XSD schema.");

return "XML validation failed.";

}

try

{

// Set up the content for the SOAP request

var httpContent = new StringContent(soapRequest, Encoding.UTF8, "text/xml");

// Send POST request

var response = await httpClient.PostAsync(apiEndpoint, httpContent);

// Check if the response is successful

if (response.IsSuccessStatusCode)

{

// Read the response content

string result = await response.Content.ReadAsStringAsync();

// Convert XML to JSON if needed (you may use the method provided in your codebase)

string jsonResult = ConvertXmlToJson(result); // Assuming you have a ConvertXmlToJson method

return jsonResult;

}

else

{

Console.WriteLine("Error in the API request: " + response.ReasonPhrase);

return null;

}

}

catch (Exception ex)

{

Console.WriteLine("Exception occurred while calling the API: " + ex.Message);

return null;

}

}

}

private string ConvertXmlToJson(string xml)

{

var xmlDoc = new XmlDocument();

xmlDoc.LoadXml(xml); // Load the XML string into XmlDocument

string jsonText = JsonConvert.SerializeXmlNode(xmlDoc, Newtonsoft.Json.Formatting.Indented);

return jsonText; // Return the JSON string

}

public bool ValidateXmlAgainstXsd(string xmlContent, string xsdFilePath)

{

try

{

XmlSchemaSet schema = new XmlSchemaSet();

// Specify the targetNamespace that matches your XSD

string targetNamespace = "AnetApi/xml/v1/schema/AnetApiSchema.xsd";

schema.Add(targetNamespace, xsdFilePath); // Load XSD file with the correct targetNamespace

XmlReaderSettings settings = new XmlReaderSettings();

settings.Schemas.Add(schema);

settings.ValidationType = ValidationType.Schema;

settings.ValidationEventHandler += (sender, args) =>

{

Console.WriteLine($"Validation error: {args.Message}");

};

using (StringReader stringReader = new StringReader(xmlContent))

using (XmlReader reader = XmlReader.Create(stringReader, settings))

{

while (reader.Read()) { } // Perform validation

}

return true;

}

catch (XmlException ex)

{

Console.WriteLine("XML Exception: " + ex.Message);

return false;

}

catch (Exception ex)

{

Console.WriteLine("Exception during XML validation: " + ex.Message);

return false;

}

}

public async Task<string> SendSoapRequestxsd(string soapRequest)

{

string[] paths = { "C:/Temp" };

// Ensure the directory path where the XSD file will be placed

string directoryPath = Path.Combine(paths);

// Check if the directory exists, if not, create it

if (!Directory.Exists(directoryPath))

{

Directory.CreateDirectory(directoryPath);

}

// Define the full path to the XSD file

//string xsdPath = Path.Combine(directoryPath, "out.xsd");

string xsdPath = "C:/Temp/New.xsd";

// Check if the XSD file doesn't exist, create it

if (!File.Exists(xsdPath))

{

string xsdContent = @"<?xml version='1.0' encoding='utf-8'?>

<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema' targetNamespace='AnetApi/xml/v1/schema/AnetApiSchema.xsd' xmlns='AnetApi/xml/v1/schema/AnetApiSchema.xsd' elementFormDefault='qualified'>

<xs:element name='createCustomerShippingAddressRequest'>

<xs:complexType>

<xs:sequence>

<xs:element name='merchantAuthentication'>

<xs:complexType>

<xs:sequence>

<xs:element name='name' type='xs:string' />

<xs:element name='transactionKey' type='xs:string' />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name='customerProfileId' type='xs:string' />

<xs:element name='address'>

<xs:complexType>

<xs:sequence>

<xs:element name='firstName' type='xs:string' />

<xs:element name='lastName' type='xs:string' />

<xs:element name='company' type='xs:string' minOccurs='0' />

<xs:element name='address' type='xs:string' />

<xs:element name='city' type='xs:string' />

<xs:element name='state' type='xs:string' />

<xs:element name='zip' type='xs:string' />

<xs:element name='country' type='xs:string' />

<xs:element name='phoneNumber' type='xs:string' />

<xs:element name='faxNumber' type='xs:string' minOccurs='0' />

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name='defaultShippingAddress' type='xs:boolean' />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:schema>";

// Write the XSD content to the file

File.WriteAllText(xsdPath, xsdContent);

}

// Assuming SendSoapRequest is already implemented and accepts two parameters: XML request and XSD path

string result = await SendSoapRequest(soapRequest, xsdPath);

// Return the result

return result;

}

}