Creating A System Monitor API Client

McLaren Applied

Version 1.0 | Presented by Roman Stec

Introduction

The following is a guide to create a System Monitor Configuration API Client in C# with .Net 6 in Visual Studio 2022.

The API uses gRPC with protobuf so is accessible with any programming language that supports gRPC, e.g.: Python, GO etc. Adjustments to the instructions for other languages are not provided but should be able to be inferred from the context.

Processing Protobuf files in an API dll

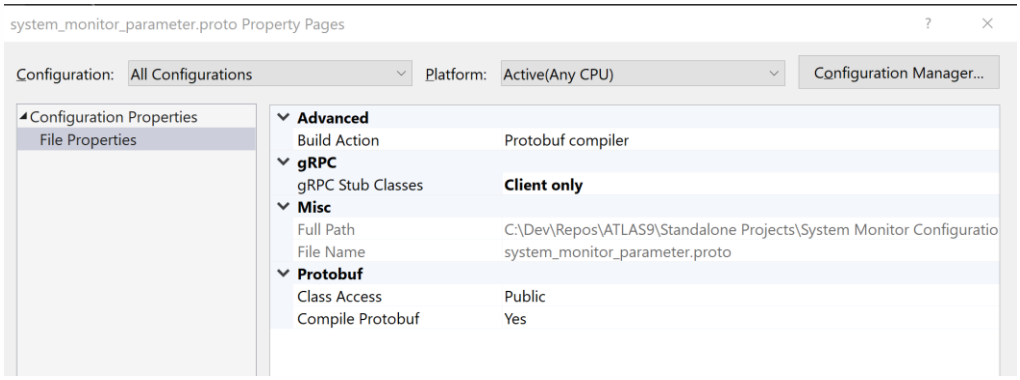
Due to the implementation of gRPC by Microsoft, including Protobuf files directly into a client is not always possible as the numerous auto-generated files can generate namespace errors when an attempt is made to use them directly. To get around this an API dll with the processed protobuf can be created and then included in your client. This step may not be necessary if a different programming language is used.

McLaren Applied supplies a Nuget package with the API dll pre-prepared along with Protobuf files should they be required for use elsewhere. If you need to create your own API dll for some reason, please refer to the following instructions:

Creating an API dll.

Instructions assume knowledge of Visual Studio and c# development

- In Visual studio create a new 'Class Library' project.
- Select the appropriate framework target for your use, (supplied API dll uses .Net 6)
- Add the following Nuget packages:
  - Google.Protobuf
  - Grpc.Net.Client
  - Grpc.Tools
- Add a Protobuf folder to the project and add the Configuration API Protobuf files, which, at this time, consist of the following files:
  - system_monitor_common.proto
  - system_monitor_logging.proto
  - system_monitor_parameter.proto
  - system_monitor_project.proto
  - system_monitor_system.proto
  - system_monitor_virtual.proto
- Select each Protobuf File Properties in turn:
  - Set 'Build Action' to 'Protobuf compiler'
  - Set 'gRPC Stub Classes' to 'Client only'



- Build the project to create the API dll.

Creating a Client

Client Project.

- Create a new project in Visual Studio.
- If you are using the McLaren Applied supplied Configuration API load the 'SystemMonitorConfigurationAPI' Nuget package.
- If you have created your own API to add a project reference to the appropriate dll.
- Create a gRPC Channel to the API server:
  - `var channel = GrpcChannel.ForAddress("127.0.0.1");`
- Initialise each of the Protobuf clients with the channel:
  - `var systemClient = new SystemMonitorSystem.SystemMonitorSystemClient(channel);`
  - `var projectClient = new SystemMonitorProject.SystemMonitorProjectClient(channel);`
  - `var virtualClient = new SystemMonitorVirtual.SystemMonitorVirtualClient(channel);`
  - `var loggingClient = new SystemMonitorLogging.SystemMonitorLoggingClient(channel);`
  - `var paramClient = new SystemMonitorParameter.SystemMonitorParameterClient(channel);`
- You are now able to call the methods exposed by the API

Calling an API method

- To call an API method:
  - Create a request object.
  - Call the Client Method.
  - Handle the reply object.
- Taking the System Client Method 'SetOnline' as an example:

```
var state = new OnlineRequest()
{
    State = true
};

var error = this.systemClient.SetOnline(state, header);
if (error.ReturnCode == 0)
{
}
```

- If the method does not require a request method pass in an empty request:
  - `var state = this.systemClient.GetStatus(new Empty(), header);`
- All methods implemented in the Confiuration API an ReturnCode variable of type ErrorCode.
- ErrorCode is an Enum defining the return state of the call to the API.
- An ErrorCode value of 'NoError' indicates the API call was successful.
- Any other value indicates System Monitor was unable to process the request with its value indicating a reason for the failure

- The header contains optional information for authentication, as described below.
- If authentication is not required the header variable can be null or emitted.

Client Authentication.

Authentication is the process that helps identify if a user is allowed access. On the other hand, authorization is the process of determining what a user can do. The Configuration API only supports authentication at this time.

The use of authentication for the API is optional, it is controlled via the Configuration API Setup dialog in System Monitor, and implementation in the client is only required if enabled.

The client contains an example of using Auth0 to manage client authentication via the Auth0 Authorization Server. System Monitor needs to be configured with the appropriate authorisation server details but the client must also know how to obtain an access token.

This document assumes Auth0 is used to manage Token assignments and code should be amended appropriately if another provider is used.

Obtaining a Token

- To request a token for an API defined in Auth0 the following information is required:
  - Token URI: Address to send the token request to.
    E.G.: https://dev-xxxxx.uk.auth0.com/
  - Audience: Unique identifier for the API as defined in Auth0.
    E.G.: https://xxxxx-api-xxxxxx.com/
  - Client Id: API Id as defined in Auth0.
  - Client Secret: API Secret as defined in Auth0
- A Rest request is sent to the URI with the appropriate information and a Bearer Token is returned.
- Example code for requesting the Token using a variety of languages is available on the Auth0 website.

Using the Authentication Token

- Once a Token has been obtained a Metadata Header can be created with the Token in a "Authorization" field.

```
var header = new Metadata
{
        { "Authorization", $"Bearer {token}" }
};
```

- The header should then be used for each call to the API:
  - `var error = this.systemClient.SetOnline(state, header);`

API Authentication of Token

- Authentication must be enabled in the System Monitor Configuration API Setup dialog. The following information must be provided:
  - Authority: Address of the authority providing authentication services.
    E.G.: https://dev-xxxxx.uk.auth0.com/
  - Audience: Unique identifier for the API as defined in Auth0.
    Matching the value used in the client.
    E.G.: https://xxxxx-api-xxxxxx.com/

- Checks are applied when each API method call is received to ensure the Token received is valid, this includes a check to ensure the token has not expired. The live time for a Token is defined within the Auth0 settings.
- If the Token fails authentication access to the API is blocked and an error is returned