# Mo.net Financial Modelling Platform
# Using Swagger for API Integration and Testing

March 2026
Revision 1

## Purpose

This datasheet describes how Swagger can be used to document and interact with API endpoints that are exposed via the Mo.net Quotations Service. It explains the background to Swagger and the OpenAPI specification and demonstrates how tools such as Swagger UI can be used to test and integrate with endpoints such as the quotation service.

## Introduction

Modern actuarial modelling environments like Mo.net increasingly expose functionality through web APIs rather than traditional desktop tools or batch processes. In this architecture, systems such as pricing engines, projection models, or quotation services can be accessed programmatically by other applications. Policy administration systems, web quotation tools, and analytical platforms can therefore trigger model calculations directly through standard HTTP requests.

While APIs provide flexibility and automation, they can also be difficult to use without clear documentation. Developers and analysts must understand the available endpoints, the required parameters, and the structure of both requests and responses. Without this information, integrating with a modelling service becomes time-consuming and error prone.

Swagger addresses this problem by providing a standardised framework for documenting, exploring, and testing APIs interactively. By describing an API using the OpenAPI specification, Swagger tools can automatically generate clear documentation and allow users to submit requests directly from a web interface.

## APIs and the OpenAPI Specification

An API endpoint is a web address that accepts requests from a client system and returns a result. Requests typically contain structured data, often in JSON format, which defines the parameters of the calculation or operation being performed.

For example, the Mo.net Quotations Service endpoint used in this environment can be accessed using the following HTTP request:

```
POST http://52.174.150.85:8099/api/v1/quotes
```

The request body contains the parameters required to perform the calculation. A simplified example is shown below:

```
{
  "Client": "Postman",
  "PackageId": "17db0911-f4a5-4632-9de5-3604ee171e3f",
  "Inputs": [
    {"key": "Age1", "value": "50"},
    {"key": "Sex1", "value": "M"},
    {"key": "AntyAmt", "value": "12000"},
    {"key": "AntyFreq", "value": "12"}
  ]
}
```

Without formal documentation, it can be difficult to determine which parameters are required, what formats are expected, or what output will be returned. To address this challenge, the software industry developed the OpenAPI specification, a standard format for describing REST APIs in a structured and machine-readable way.

An OpenAPI specification typically defines the server location, the available endpoints, the expected request bodies, and the structure of responses. Tools such as Swagger UI can then read this specification and automatically generate interactive documentation.

## What Swagger Provides

Swagger is a set of tools designed to work with OpenAPI specifications. The most widely used component is Swagger UI, which converts an OpenAPI definition into a web-based interface where users can explore and test API endpoints.

When an API is documented using Swagger, users can view the available operations and execute them directly from the browser. The interface displays the required parameters and allows the request body to be edited before sending the request to the API server.

For example, the Swagger interface might display the quotation endpoint as follows:

```
POST /api/v1/quotes
```

A user can then select Try it out, edit the JSON request body, and execute the call. Swagger sends the request to the API server and displays the response immediately, allowing users to validate both the request and the resulting output.

This interactive capability significantly reduces the effort required to understand and test API-based modelling services.

## Describing the Mo.net Quotation Service API with OpenAPI

To use Swagger with an existing API endpoint, the endpoint must be described using an OpenAPI specification. This specification defines the server location and the structure of the request and response data.

A simplified example of an OpenAPI definition for the NPIA quotation endpoint is shown below.

```
openapi: 3.0.3
info:
  title: Quotes API
  version: 1.0.0

servers:
  - url: http://52.174.150.85:8099

paths:
  /api/v1/quotes:
    post:
      summary: Generate a quotation
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/QuoteRequest'
      responses:
        '200':
          description: Successful response
```

The request structure is then defined as a reusable schema within the specification.

```
components:
  schemas:
    QuoteRequest:
      type: object
      properties:
        Client:
          type: string
        PackageId:
          type: string
        Inputs:
          type: array
          items:
            $ref: '#/components/schemas/InputItem'

    InputItem:
      type: object
      properties:
        key:
          type: string
        value:
```

```
            type: string
```

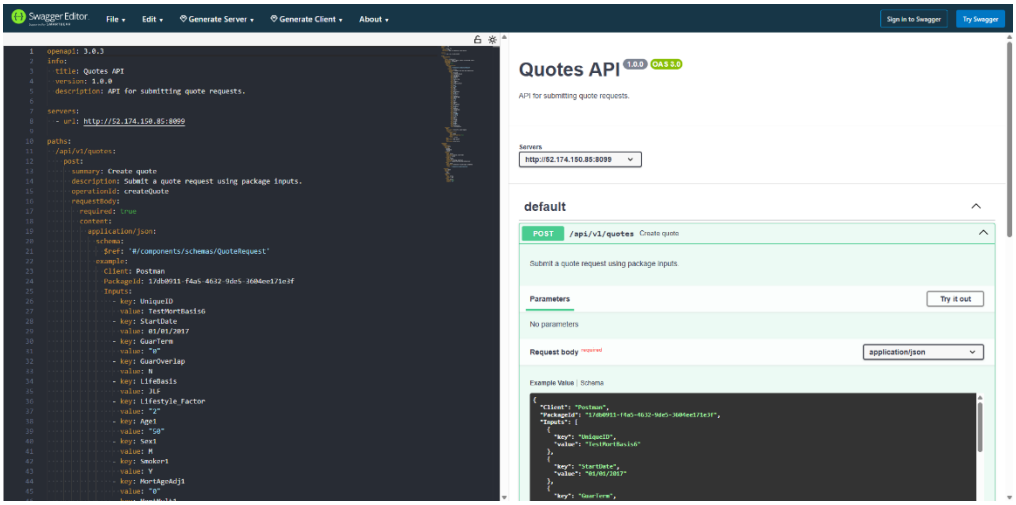
Once this specification is created, it can be loaded into Swagger UI, which will automatically generate documentation and testing functionality for the endpoint.

The easiest way to do this is to create a file (e.g. openapi_npia.yaml) in Notepad and then copy the above definition into it. Save the file and then open it using the online version of Swagger:

https://editor.swagger.io/

## Using Swagger to Test the API

After the OpenAPI specification is loaded into Swagger UI, the quotation endpoint becomes immediately accessible through the interface. Users can edit the request body and submit a test request without writing code or building a custom client.




The interaction follows a simple workflow. A user enters the required inputs into the Swagger interface and executes the request. Swagger sends the HTTP request to the API server, which processes the modelling calculation and returns the results. The response is then displayed within the Swagger interface for inspection.

This workflow allows developers and analysts to quickly validate that an endpoint is functioning correctly and that the input parameters are being interpreted as expected.

## Use Cases with Mo.net

The Mo.net modelling platform supports scalable actuarial modelling and is well suited to API-driven workflows. Swagger can play an important role in enabling teams to interact with Mo.net services efficiently and consistently.

One common use case involves exposing model calculations as APIs. In this scenario, a Mo.net model can be deployed as a service that accepts inputs such as policy characteristics, economic assumptions, or scenario parameters. Swagger documentation allows users to understand the available inputs and test the model execution directly.

Another important use case involves quotation and pricing services. Insurance pricing engines can be deployed as APIs that calculate annuity income, premium

values, or other pricing outputs based on supplied inputs. Swagger enables pricing teams and integration developers to test these calculations interactively and confirm that the correct parameters are being passed to the model.

Swagger is also valuable when integrating modelling services with other systems. Policy administration systems, customer portals, and data pipelines may need to invoke pricing or projection models as part of automated workflows. A clear OpenAPI specification provides a shared contract between systems, reducing the complexity of integration.

## Conclusion

Swagger provides a powerful and accessible framework for documenting and testing API-based modelling services. By describing endpoints using the OpenAPI specification, teams can generate interactive documentation that simplifies both development and integration.

For actuarial platforms such as Mo.net, Swagger supports a range of important activities, including testing quotation services, validating model inputs, and integrating modelling engines with external systems. As actuarial systems increasingly adopt API architectures, tools such as Swagger will play an important role in ensuring that these services remain transparent, accessible, and easy to use.

## Contact Us

For more information regarding the Mo.net platform and the use of APIs for modern financial modelling & calculations, please get in touch:

Software Alliance Limited
54 Charlotte Street, London, W1T 2NS
Tel: +44 (0) 20 3964 2755
www.softwarealliance.net

Author: Guy Shepherd