Creating Tests from an OpenAPI/Swagger Definition

SOAtest can automatically create a .tst based on the endpoints found in an OpenAPI/Swagger description. Each .tst generated using the OpenAPI/Swagger wizard will contain REST Clients for all of the defined endpoints.

The following specifications are supported:

- Swagger 1.0 - 2.0. This includes any Swagger core versions that are compatible with these specs (e.g., Swagger core 1.0.0, 1.2.4, 1.3.12, 1.5.0-M1, 1.5.0).
- OpenAPI/Swagger 3.0.

Java 7 or newer is required to use this functionality (Java 8 or newer on Mac).

To automatically create a test suite from a valid OpenAPI/Swagger definition:

1. Right-click the project in the Test Case Explorer and choose **Add New> Test (.tst)...**
2. Specify a name for the file and click **Next**.
3. Choose the **REST> OpenAPI/Swagger** option and click **Next**. For details on accessing the wizards, see Adding a New .tst File to an Existing Project and Adding a New Test Suite.
4. Enter the absolute URI for the OpenAPI/Swagger definition file or click **Browse** to find it on the local machine.
5. Click **Next**. The **Create Environment** dialog opens.
6. (Optional) Specify whether you want to reference an existing environment or create a new one.
    - To create a new environment:
        1. Select the **Create a new environment for your project** checkbox.
        2. Enter an **Environment Name** and **Variable Prefix**.
    - To reference an existing environment, select **Reference an existing environment** then specify the appropriate environment file.
    - For more information on environments, see Configuring Testing in Different Environments.
7. Click the **Finish** button.

When a .tst is generated, it includes one REST Client for each resource/method pair in the OpenAPI/Swagger definition.

- Each REST Client is set to be constrained to the specified service definition and schema (if applicable).
- Its resource URL, HTTP method, and payload (if applicable) are configured accordingly.
- The service's base URL is configured as a "BASEURL" variable, and each resource URL is parameterized with the "BASEURL" variable.
- Query parameters are included with default or sample values (if available) as defined by the service definition.
- If the service definition includes a schema, a sample payload is constructed from (and constrained to) that schema.