Response Validators

In this section, response validators can be defined. Default response validators are already provided for reference. A custom response validator can be created by clicking on "New Validator" link in left hand side.

In vREST, assertions can further invoke validators to validate the test case responses. A response validator is simply a JavaScript function. A test case passes if the assigned response validator returns true otherwise it fails.

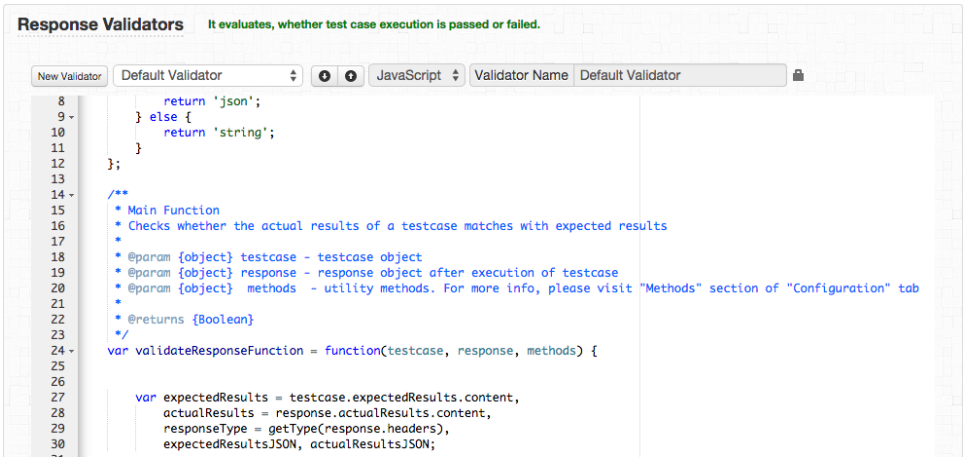
Types of Validators

vREST provides the following default response validators, and also provide you a way to define your own custom validator.

1. **Default Validator** validates the expected and actual responses (response body) for exact match. If the value doesn't match, it simply fails the test case. We need to define **Expected Body** for the test case, if we choose this. Any variables in expected body are first replaced and then provided to the validator for response validation.

2. **Default Schema Validator** validates the schema of actual response instead of checking for exact content match. We need to define **Expected Schema** for the test case, if we choose this. If we use this validator, then we don't need to define the Expected Body.

3. **Custom Validator** validates the actual response by your own way. Most of the cases of response validation can be handled with the above default validators. However, you may define your own custom validators for customized response validation. vREST also provides the Javascript code for the default validators in **Project Configuration** > **Response Validators** section for reference on how you can define your own custom validator. A custom validator can be created by clicking on the "New Validator" button as shown in the image below:



A response validator function will get the following input parameters:

1. testcase
   First parameter is test case, having all the details of a test case which has been provided in the test cases tab.
2. response
   Second parameter is actual response of the HTTP request. It is a JSON object with the following keys:
   a. headers: HTTP Response headers retrieved
   b. actualResults: HTTP Response result retrieved. It is also a JSON object with the following keys:
      i. content: (String) Actual HTTP Response body
      ii. resultType: (String) Content type of HTTP Response body
3. methods
   Third parameter is methods object. This parameter is a JSON object where key is the utility method name and value is the utility method reference.
   a. So, you may write your own custom utility methods and use them in your custom response validators.
4. **vars**
   Fourth parameter is variables map object. This parameter is a JSON object where key is the variable name and value is the variable's value.
   a. Note: You may directly change the variable's value in response validator. Please be cautious while changing variable's value in response validator because it may affect the subsequent test cases.