

Execute a system command, bash script via REST API

vutil provides a REST API to execute commands on the test machine in which test application's data resides. Now with the help of this API, we can insert database dumps through commands and these commands can be invoked via REST APIs.

This API execute commands on machines on which this utility is installed.

API Endpoint:

The API format is

```
POST {{vutilBaseURL}}/execute/command
```

Here {{vutilBaseURL}} is the base URL of the vutil server.

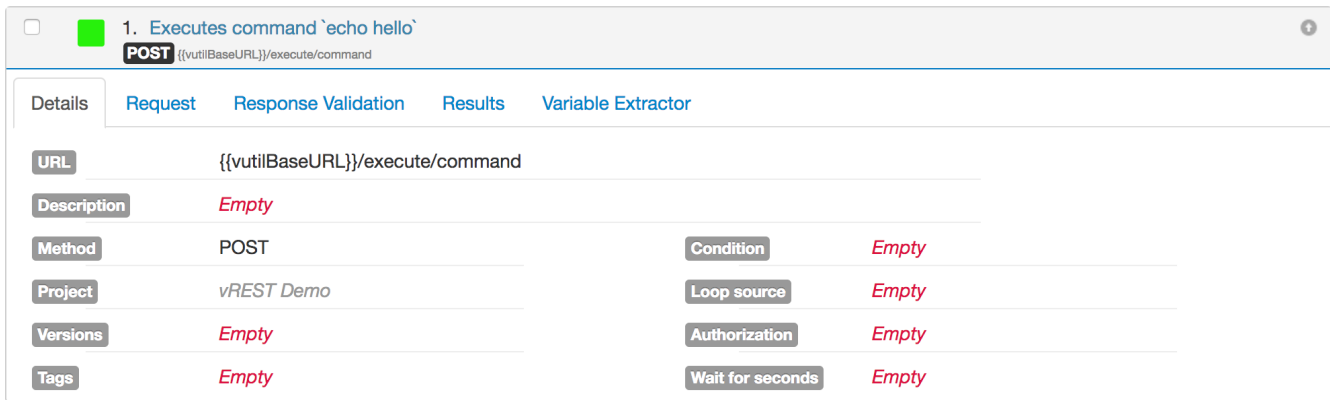
and this API accepts a single body parameter "command" and its value is the command which you want to execute on your system.

How to use this API

Assuming that vutil is running on port 4080 and vutil base URL is "<http://localhost:4080>"

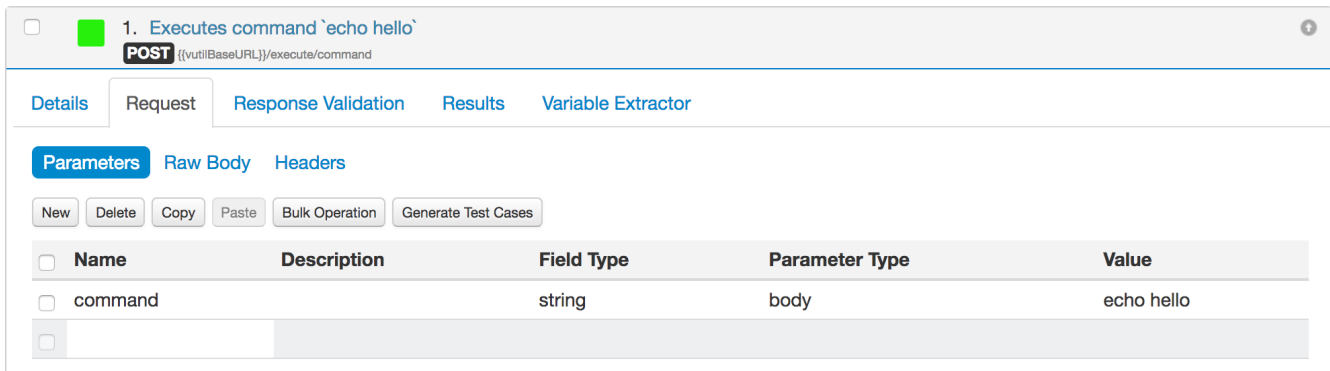
Let us try to execute a simple command `echo hello` via this utility. Simply create a test case with URL "<http://localhost:4080>".

Your test case might look like:



The screenshot shows a REST client interface for a test case titled "1. Executes command `echo hello`". The request method is POST and the URL is {{vutilBaseURL}}/execute/command. The interface includes tabs for Details, Request, Response Validation, Results, and Variable Extractor. The Details tab is active, showing various fields: URL, Description (Empty), Method (POST), Project (vREST Demo), Versions (Empty), Tags (Empty), Condition (Empty), Loop source (Empty), Authorization (Empty), and Wait for seconds (Empty).

and provide a request parameter "command" as below screenshot describes:



The screenshot shows the same REST client interface, but with the Parameters tab selected. It displays a table of parameters for the request. The table has columns for Name, Description, Field Type, Parameter Type, and Value. One parameter is defined: "command" with a description, field type of "string", parameter type of "body", and value of "echo hello".

Name	Description	Field Type	Parameter Type	Value
command		string	body	echo hello

and provide some assertions to validate whether the command executed successfully or not.

1. Executes command `echo hello`
POST {{vutilBaseUrl}}/execute/command

Details Request Response Validation Results Variable Extractor

Assertions Expected Body Expected Schema

New Delete Copy Paste

#	Source	Property	Comparison	Expected Value
1	Status Code		Equal to number	200
2	Text Body		Call Default Validator	Show expected body

and finally the expected body for the test case.

1. Executes command `echo hello`
POST {{vutilBaseUrl}}/execute/command

Details Request Response Validation Results Variable Extractor

Assertions Expected Body Expected Schema

JSON Save Changes

```

1 {
2   "pid": "{{*}}",
3   "output": "hello\n"
4 }
```

Now, if we execute this test case. Then the above REST API executed the command `echo hello` on the remote machine and returned the command output as response as shown in below figure.

1. Executes command `echo hello`
POST {{vutilBaseUrl}}/execute/command

Details Request Response Validation Results Variable Extractor 7:23:59.015 pm, Apr 15, 2017

Test Results Pre Hooks Post Hooks

Result : **Passed** Test Run : 7:23:59.015 pm, Apr 15, 2017 Show variables used Show Execution time details

Execution time details :

Details Request Headers Response Headers

POST http://localhost:4080/execute/command
Status code : 200 Response Time : 23 ms

Actual request body :

```
command=echo%20hello
```

Actual response body :

JSON

```

1 {
2   "pid": 23182,
3   "output": "hello\n"
4 }
```

Hope you got the idea.

Now, let us take a more practical example of using this utility. Suppose, I want to restore my MongoDB database before I execute my test suite. And my MongoDB dump is stored at `{{dataDir}}/xyz-ts-dump` directory. Then I can write a separate test case in vREST like below:

The image shows two screenshots of the vREST interface. The top screenshot displays the 'Details' tab for a test case titled '2. Restore database for Test Suite `XYZ`'. The URL is `{{vutilBaseUrl}}/execute/command` and the method is POST. The description is 'Empty'. Other fields like Method, Project, Versions, Tags, Condition, Loop source, Authorization, and Wait for seconds are also shown as 'Empty'. The bottom screenshot shows the 'Request' tab for the same test case. It has tabs for 'Parameters', 'Raw Body', and 'Headers'. Below these are buttons for 'New', 'Delete', 'Copy', 'Paste', 'Bulk Operation', and 'Generate Test Cases'. A table lists the parameters:

Name	Description	Field Type	Parameter Type	Value
command		string	body	mongorestore --db dbname --drop {{dumpDir}}/xyz-ts-dump

Similarly other details can be provided as described in previous example.

You can even execute some custom script via this API to initialize your Test Suite data depending upon your context.