

Database Validation

vutil provides a REST API to access database which can be used in validating the database state. With the help of this REST API, you may perform CRUD operations on various databases. As of now, following databases are supported:

1. [MySQL](#)
2. [Postgres](#)
3. [MongoDB](#)
4. [MSSQL](#)
5. [Oracle](#)

We are also working on integration with other databases as well. Please stay tuned.

Now, with the help of this API, one can perform the database validation after a test case is executed.

API Endpoint:

API Endpoint for DB Validation API is

```
POST {{vutilBaseURL}}/execute/dbquery/{{connection}}
```

Here

1. {{vutilBaseURL}} is the variable, and the value is base URL of vutil server.
2. {{connection}} is the path variable, and the value is the connection name of the database which is configured in config.json file.

And this API accepts a single body parameter "query" which can be either in STRING format or in JSON Object depending on the database vendor.

Steps:

Step by step guide to perform database validation:

Step 1: Configure connection details in vutil

First you need to configure database connection details in config.json file. If you have not created a config.json file, then create one by copying the config.sample.json into config.json file and provide the connection details as per the samples provided.

Step 2: Execute your API test case and validate its response. e.g. for this guide, we are creating a contact with our sample contact API as follows:

The screenshot shows a REST client interface for a test case titled "1. Validate Create Contact API". The request is a POST to the endpoint `{{contactsDBBaseURL}}/contacts`. The interface includes tabs for "Details", "Request", "Response Validation", "Results", and "Variable Extractor". The "Details" tab is active, displaying the following configuration:

URL	<code>{{contactsDBBaseURL}}/contacts</code>		
Description	Empty		
Method	POST	Condition	Empty
External Id	Empty	Loop source	Empty
Versions	Empty	Authorization	Empty
Tags	Empty	Wait for seconds	Empty

1. Validate Create Contact API

POST {{contactsDBBaseURL}}/contacts

Details Request Response Validation Results Variable Extractor

Parameters Raw Body Headers

JSON Enabled Save Changes

```

1 - {
2   "_id": null,
3   "name": "John Doe",
4   "email": "john.doe@example.com",
5   "designation": "Chief Technical Officer",
6   "organization": "Example.com",
7   "country": "India",
8   "aboutMe": "My name can be used as a placeholder name and I don't have any identity.",
9   "twitterId": "fake.john.doe",
10  "facebookId": "fake.john.doe",
11  "githubId": "fake.john.doe"

```

1. Validate Create Contact API

POST {{contactsDBBaseURL}}/contacts

Details Request Response Validation Results Variable Extractor

Variables from API Response Variables from Results

New Delete Help

Name	JSON Path / XML Path / Utility Method
<input type="checkbox"/> contactId	_id
<input type="checkbox"/>	

Note:

- For defining JSON path, follow the guide.
- For defining X-Path for XML, follow the guide.
- For defining Utility method, follow the guide.
- These variables will be available in all subsequent test Cases within a test run.

Step 3: Now, to validate the database state, we need to write a separate test case in vREST. With the help of DB API provided by vutil, we can write the test case for db validation as follows:

1. First create a test case with DB Validation API as follows:

2. Validate Database state after create contact API

POST {{vutilBaseURL}}/execute/dbquery/{{connection}}

Details Request Response Validation Results Variable Extractor

URL	{{vutilBaseURL}}/execute/dbquery/{{connection}}		
Description	Empty		
Method	POST	Condition	Empty
External Id	Empty	Loop source	Empty
Versions	Empty	Authorization	Empty
Tags	Empty	Wait for seconds	Empty

2. Now, provide the connection name as follows:

Name	Description	Field Type	Parameter Type	Value
connection	Database connection name configured in config.json	string	path	contactdb

Body Parameters (highlighted with light-red background) will be ignored. Because "Raw Body" is currently enabled.

3. Now, provide the request body for the DB Validation API:

```

1- {
2-   "query": {
3-     "collection": "contacts",
4-     "command": "find",
5-     "args": [
6-       {},
7-       {}
8-     ],
9-     "cursorMethods": [
10-      {
11-        "method": "skip",
12-        "params": 0
13-      },
14-      {
15-        "method": "limit",
16-        "params": 1
17-      },
18-      {
19-        "method": "sort",
20-        "params": {
21-          "createdOn": -1
22-        }
23-      }
24-    ]
25-  }
26- }
  
```

4. Provide assertions as you provide for other test cases

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

5. And expected body as per your need.

```

1- {
2-   "output": [
3-     {
4-       "_id": "{{contactId}}",
5-       "name": "John Doe",
6-       "email": "john.doe@example.com",
7-       "designation": "Chief Technical Officer",
8-       "organization": "Example.com",
9-       "country": "India",
10-      "aboutMe": "My name can be used as a placeholder name and I don't have any identity.",
11-      "twitterId": "fake.john.doe",
  
```

That's it. In this way, you may validate the DB state.