

TeamCity Integration

TeamCity is a popular CI/CD tool. Now you can automatically validate your project builds using vREST test suite with TeamCity Server.

i If you are using some other Continuous Integration Server, you can integrate vREST with that in the same way as with TeamCity Server. If you face any issues then do let us know by dropping an email to support@vrest.io

Prerequisite

Let's take a sample application ([Contacts Application](#)) and step by step guide on how we can integrate vREST with TeamCity Server.

Note: You can find the source code of sample application at [Github](#).

Step by Step Guide:

Step 1: Write your automated test suite in vREST

First you need to write test cases for your test application in vREST to ensure the build quality of your web application. Then copy the test case list URL by clicking on button "Copy Current Test Case List URL" available in Continuous Integration section. You can find the test cases for this sample application in vREST [here](#).

The screenshot shows the vREST web interface. The browser address bar displays `https://vrest.io/i/demo/VC/`. The interface includes a navigation bar with 'Test Cases', 'Mock Server', 'Test Runs', and 'Project Configuration'. A sidebar on the left contains sections for 'FILTERS APPLIED', 'FOR CONTINUOUS INTEGRATION', 'CURRENT ENVIRONMENT', and 'QUICK GUIDE LINKS'. The main content area displays a list of 13 test cases with their IDs, HTTP methods, and descriptions. A red box highlights the first three test cases, and a red text overlay reads '1. Write automated Test cases in vREST'. A red box also highlights the 'Copy Current Test Case List URL' button in the sidebar.

ID	Method	Description
1.	POST	Initialize Contacts DB
2.	GET	Validate whether List Contacts API is working fine or not
3.	POST	Validate whether Add Contact API works with all valid details
4.	PUT	Validate whether Update Contact API should return error when name field is blank
5.	PUT	Validate whether Update Contact API should return error when contact's name length is greater than the permissible name length limit
6.	PUT	Validate whether Update contact API should return error when contact's designation length is greater than the permissible designation length limit
7.	PUT	Validate whether Update contact API should return error when contact's organization length is greater than the permissible organization length limit
8.	PUT	Validate whether Update Contact API should return error when contact's country length is greater than the permissible country length limit
9.	PUT	Validate whether Update Contact API should return error when contact's aboutMe length is greater than the permissible aboutMe length limit
10.	PUT	Validate whether Update Contact API should return error when setting invalid email for a contact
11.	PUT	Validate whether Update Contact API by ID works properly with all valid details
12.	GET	Validate whether View Contact API by ID is returning the correct details or not
13.		Checking the list of contacts to validate whether the new record is in the list or not

Step 2: [Optional] Learn how vranner command works and get your hands dirty on it by installing it locally first

First setup the vranner module by following the guide [Setup / install vranner](#).

Once you setup this module on your local machine, you can execute the vREST test cases by executing the following command:

```
vrrunner --email=<EMAIL_ID> --password=<PASSWORD> --url=<URL OF vREST Test Cases> --logger=xunit [--  
filepath=<absolute log file path>] [--env=<ENVIRONMENT>]
```

Options

- **email:** Email ID through which you have registered on vREST
- **password:** Password of your vREST account
- **url:** Provide the test case list URL here. You can find the test case list URL by going to your vREST instance and select Test Cases tab. Now click on button "Copy Current Test Case List URL" available in Left hand side, below the "Filters section". Provide the copied URL in this option.
- **logger:** Use xunit here. Reporter xunit will export the test case report in xml junit style, which can be used in any continuous integration server to publish the reports.
- **filepath:** Absolute path of the log file, into which execution process and result logs will be dumped. If path/file is not present, tool will try to setup that path, and create file automatically. Please note that if file already exists, that will be overwritten. By default it will be the `vrest_logs/logs.xml` in current directory.
- **env:** [Optional] Provide the environment name which you have defined in vREST (Case-sensitive).

Step 3: Add vrrunner command in the build step in TeamCity Build Configuration

In TeamCity Build Configuration, select "**Build Step: Command Line**" and then click on button "**Add Build Step**" and just fill the form and add the custom script shown in the snapshot below in the field "**Custom Script**". Please replace the path to vrrunner binary, email, password, env, url parameters of the command according to your configuration.

Note: Make Sure every % symbol in the URL must be escaped with another % symbol.

The screenshot shows the TeamCity Build Configuration page for a project named 'Contacts Application Build'. The 'Build Step' configuration is visible, with the following settings:

- Runner type:** Command Line (Simple command execution)
- Step name:** Validate build using vrrunner (Optional, specify to distinguish this build step from other steps.)
- Run:** Custom script
- Custom script:** A text area containing the following command:

```
/Users/vrest/vrunner/builds/latest/vrunner_macos_0_2_39  
--email="john.doe@example.com" --password="john doe"  
--url="https://vrest.io/i/demo/g/testcase?projectId=54955"  
--logger="xunit" --env=prod
```

Below the text area, there is a note: "A platform-specific script, which will be executed as a .cmd file on Windows or as a shell script in Unix-like environments." At the bottom of the configuration form, there are 'Save' and 'Cancel' buttons.

Step 4: Publish reports in TeamCity Server

By default, **vrrunner** command writes the report in '**vrest_logs/logs.xml**' if logger '**xunit**' is used. You may change this path by providing the **filepath** option in the vrrunner command. In TeamCity Build Configuration, select "**Build Features**" and then click on button "**Add Build Feature**" and specify the form as shown below:

TC Projects | Changes Agents 1 Build Queue 0 Admin User | Administration

Administration / <Root project> / Contacts Jenkins Run ... Actions Build Configuration Home

Contacts Application Build

- General Settings
- Version Control Settings
- Build Step: Command L
- Triggers 1
- Failure Conditions
- Build Features**
- Dependencies
- Parameters
- Agent Requirements

Last edited moments ago by Admin User (view his...)

Add Build Feature

XML report processing

Allows importing data from report files produced by an external tool in TeamCity.

⚠ Please make sure that tests are not detected automatically before using this feature.

Report type: * Ant JUnit
Choose a report type.

Monitoring rules: * Type report monitoring rules:
`%teamcity.build.workingDir%/vrest_logs/logs.xml`

Newline- or comma-separated set of rules in the form of `+|-:path`.
Ant-style wildcards supported, e.g. `dir/**/*.*xml`

Verbose output:

Save Cancel

That's it.