How to run blackbox tests

This document describes how to run EdgeX black-box test by Newman and Postman.

Prerequisite

1. Install docker, docker-compose
2. Install postman https://www.getpostman.com/apps
3. Clone repo from https://github.com/edgexfoundry/blackbox-testing

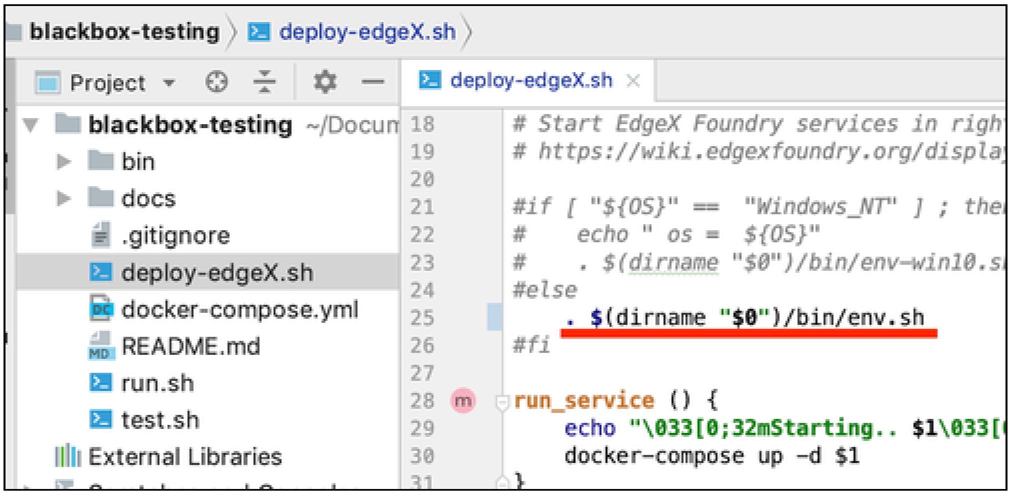
Setup env.sh

We need to set up the environment variables for the test script.

1.Open file your_path/to/blackbox-testing/deploy-edgeX.sh and uncomment below code snippet:

deploy-edgeX.sh

```
. $(dirname "$0")/bin/env.sh
```

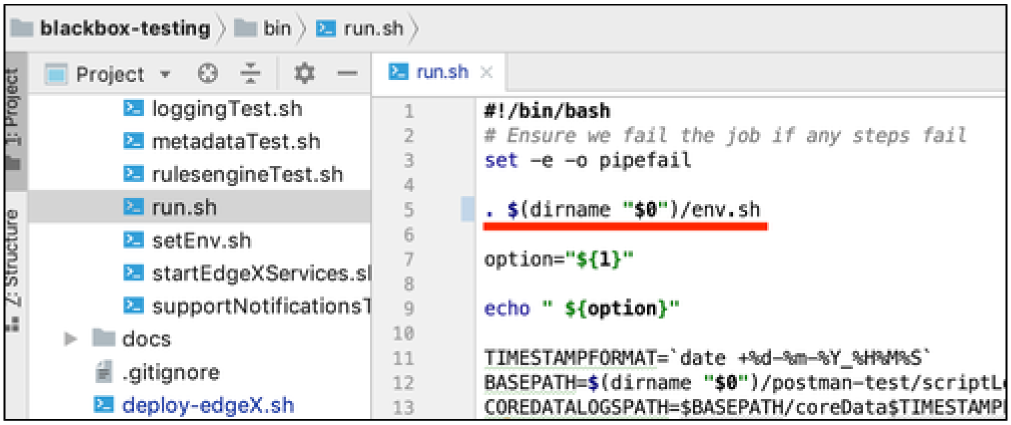


```
# Start EdgeX Foundry services in righ
# https://wiki.edgexfoundry.org/displa

#if [ "${OS}" ==  "Windows_NT" ] ; the
#    echo " os =  ${OS}"
#    . $(dirname "$0")/bin/env-win10.s
#else
    . $(dirname "$0")/bin/env.sh
#fi

run_service () {
    echo "\033[0;32mStarting.. $1\033[
    docker-compose up -d $1
}
```

2. Open file your_path/to/blackbox-testing/bin/run.sh and uncomment below code snippet:

run.sh

```
. $(dirname "$0")/env.sh
```



```
#!/bin/bash
# Ensure we fail the job if any steps fail
set -e -o pipefail

. $(dirname "$0")/env.sh

option="${1}"

echo " ${option}"

TIMESTAMPFORMAT=`date +%d-%m-%Y_%H%M%S`
BASEPATH=$(dirname "$0")/postman-test/scriptL
COREDATALOGSPATH=$BASEPATH/coreData$TIMESTAMP
```

Deploy EdgeX

Change directory to your_path/to/blackbox-testing/, execute below command:

command line tool

```
$ bash deploy-edgeX.sh
```

Then we will see the console output like:

command line tool

```
$ bash deploy-edgeX.sh
Starting.. volume
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
Creating network "blackbox-testing_edgex-network" with driver "bridge"
Creating blackbox-testing_volume_1 ... done
Starting.. consul
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
blackbox-testing_volume_1 is up-to-date
Creating blackbox-testing_consul_1 ... done
...
...
Starting.. command
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
blackbox-testing_volume_1 is up-to-date
blackbox-testing_mongo_1 is up-to-date
blackbox-testing_consul_1 is up-to-date
blackbox-testing_logging_1 is up-to-date
blackbox-testing_metadata_1 is up-to-date
Creating blackbox-testing_command_1 ... done
```

We can check EdgeX services by command docker ps:

command line tool

```
$ docker ps
CONTAINER ID        IMAGE
COMMAND                  CREATED             STATUS              PORTS
NAMES
f1b490b3e0c3
nexus3.edgexfoundry.org:10004/docker-core-command-go:0.5.2
"/core-command --con…"   3 minutes ago       Up 3 minutes
0.0.0.0:48082->48082/tcp
blackbox-testing_command_1
055dc4d48442
nexus3.edgexfoundry.org:10004/docker-core-data-go:0.5.2
"/core-data --consul…"   3 minutes ago       Up 3 minutes
0.0.0.0:48080->48080/tcp, 0.0.0.0:32781->5563/tcp
blackbox-testing_data_1
e0aa5a675d7d
nexus3.edgexfoundry.org:10004/docker-core-metadata-go:0.5.2
"/core-metadata --co…"   3 minutes ago       Up 3 minutes
0.0.0.0:48081->48081/tcp, 48082/tcp
blackbox-testing_metadata_1
deaa276c175f
nexus3.edgexfoundry.org:10004/docker-support-notifications:0.5.0
"/bin/sh -c 'java -j…"   3 minutes ago       Up 3 minutes
0.0.0.0:48060->48060/tcp
blackbox-testing_notifications_1
18336488dba4
nexus3.edgexfoundry.org:10004/docker-support-logging-go:0.5.2
"/support-logging --…"   4 minutes ago       Up 4 minutes
0.0.0.0:48061->48061/tcp
blackbox-testing_logging_1
05896fe88e17
nexus3.edgexfoundry.org:10004/docker-edgex-mongo:master
"docker-entrypoint.s…"   4 minutes ago       Up 4 minutes
0.0.0.0:27017->27017/tcp
blackbox-testing_mongo_1
85d0e33c4924
nexus3.edgexfoundry.org:10004/docker-core-config-seed-go:master
"docker-entrypoint.s…"   4 minutes ago       Up 4 minutes
8300-8302/tcp, 8400/tcp, 8500/tcp, 8301-8302/udp, 8600/tcp, 8600/udp
blackbox-testing_config-seed_1
9e4bf85969d7        consul:1.1.0
"docker-entrypoint.s…"   4 minutes ago       Up 4 minutes
0.0.0.0:8400->8400/tcp, 8301-8302/udp, 0.0.0.0:8500->8500/tcp,
8300-8302/tcp, 8600/udp, 0.0.0.0:8600->8600/tcp
blackbox-testing_consul_1
47f532468383
nexus3.edgexfoundry.org:10004/docker-edgex-volume:master
"/bin/sh -c '/usr/bi…"   4 minutes ago       Up 4 minutes
blackbox-testing_volume_1
```

Run test by Newman

First of all, the script logic is:

1. Import test data into Edgex
2. Run test script by Newman
3. Clean test data

For example, when we execute `bash ./bin/run.sh -cd`, then the script logic is:

1. Import core-data's test data into Edgex
2. Run core-data's test script by Newman
3. Clean core-data's test data

And the output will like:

command line tool

So after deploy services, we can test service's API by the command:

test service	command
core-data	bash ./bin/run.sh -cd
core-metadata	bash ./bin/run.sh -md
core-command	bash ./bin/run.sh -co
support-logging	bash ./bin/run.sh -log
support-notification	bash ./bin/run.sh -sn
support-rulesengine	bash ./bin/run.sh -ru
export-client	bash ./bin/run.sh -exc
All	bash ./bin/run.sh -all

Or we can just run `bash ./bin/run.sh` to see all usages:

command line tool

```
$ bash ./bin/run.sh

*********************************************
    ______   _    _    _____   ______
   |  ____| | |  | |  / ____| |  ____|
   | |__    | |  | | | (___   | |__
   |  __|   | |  | |  \___ \  |  __|
   | |      | |__| |  ____) | | |____
   |_|       \____/  |_____/  |______|

*********************************************
      COMPONENT BLACKBOX TEST HARNESS
              Version: Alpha
*********************************************

[INFO] Init postman test data .
run.sh:usage: [-cd Coredata] | [-md Metadata] | [-co Command] | [-sn
SupportNotification] | [-lo Logging] | [-exc Export Client] | [-ru
Rulesengine] | [-all All]
```

Run test by Postman

Postman also a good tool to run the tests or add more test APIs and update test assertions.

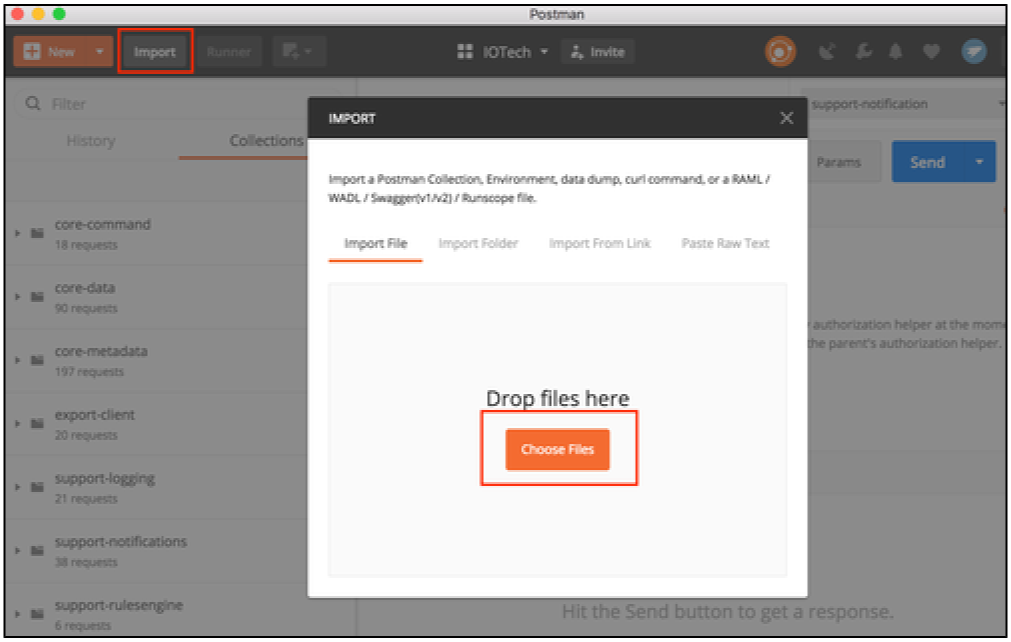
The test is same logic, just like execute bash ./bin/run.sh -cd. But there are many detail steps need to do.

And we choose core-data test as below example.

Import test data into Edgex

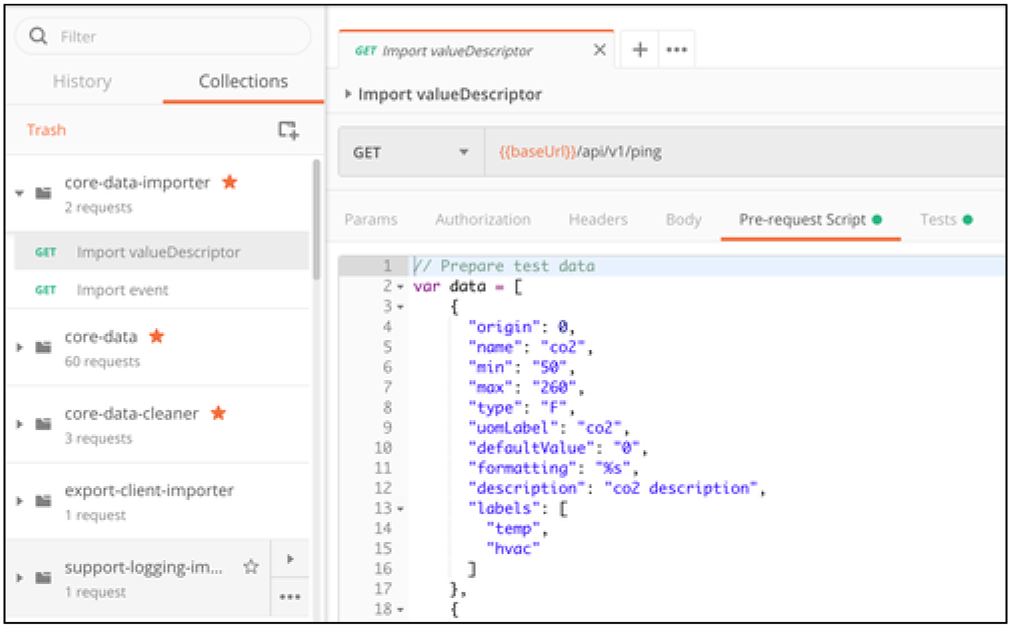
1. Add script which locates at your_path/to/blackbox-testing/bin/postman-test/collections/core-data-importer.postman_collection.json

1-1. Drop or choose the file



Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.

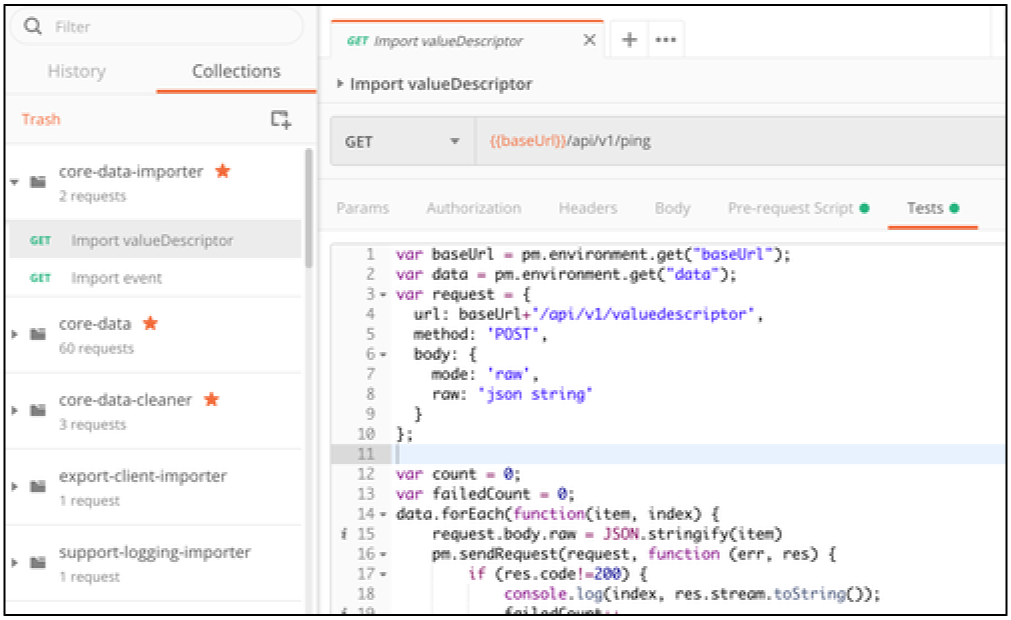
1-2. Prepared test data under the tab of Pre-request Script



{{baseUrl}}/api/v1/ping

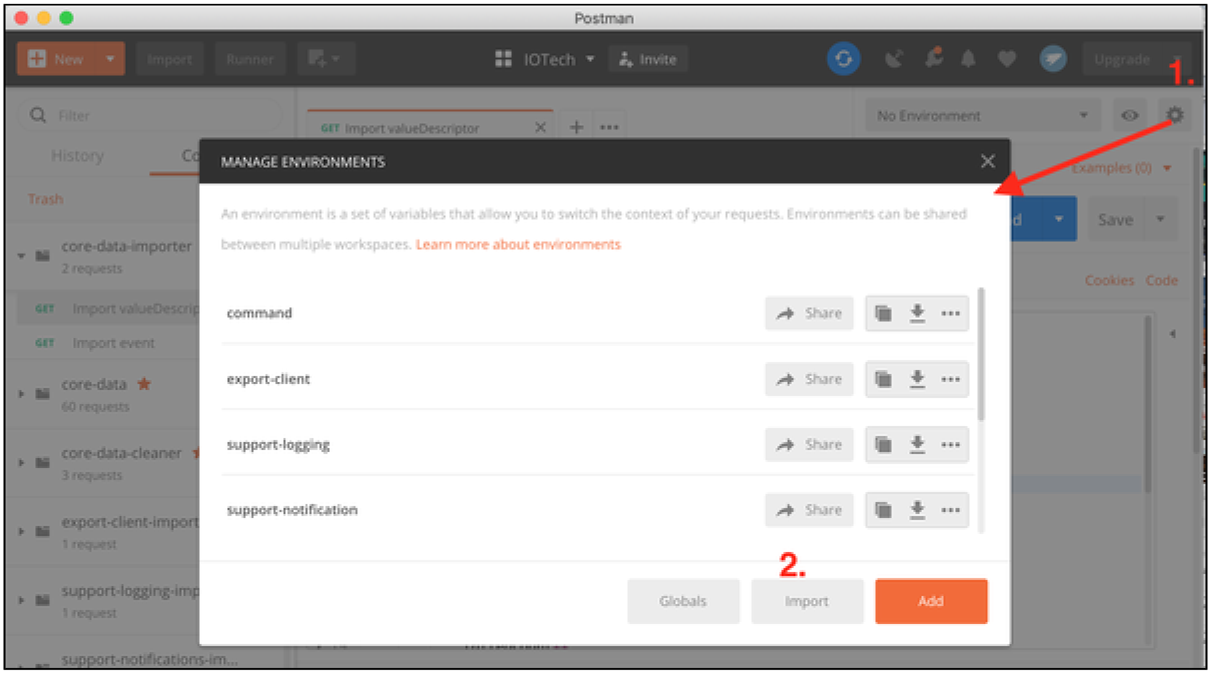
```
// Prepare test data
var data = [
    {
      "origin": 0,
      "name": "co2",
      "min": "50",
      "max": "260",
      "type": "F",
      "uomLabel": "co2",
      "defaultValue": "0",
      "formatting": "%s",
      "description": "co2 description",
      "labels": [
        "temp",
        "hvac"
      ]
    },
    {
```

1-3. Script under the tab of Tests. This script executes POST API for add new test data.

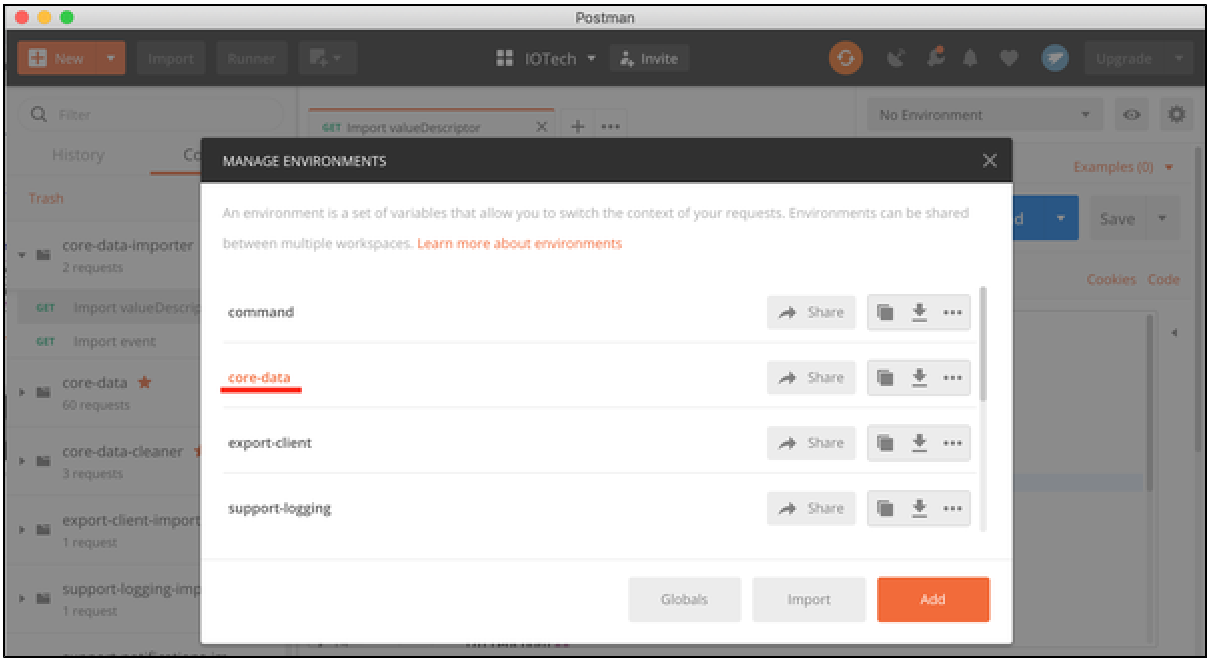


```
var baseUrl = pm.environment.get("baseUrl");
var data = pm.environment.get("data");
var request = {
  url: baseUrl+'/api/v1/valuedescriptor',
  method: 'POST',
  body: {
    mode: 'raw',
    raw: 'json string'
  }
};

var count = 0;
var failedCount = 0;
data.forEach(function(item, index) {
    request.body.raw = JSON.stringify(item)
    pm.sendRequest(request, function (err, res) {
        if (res.code!=200) {
            console.log(index, res.stream.toString());
```

2. Import environment which locates at your_path/to/blackbox-testing/bin/postman-test/environment/core-data.postman_environment.json



An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

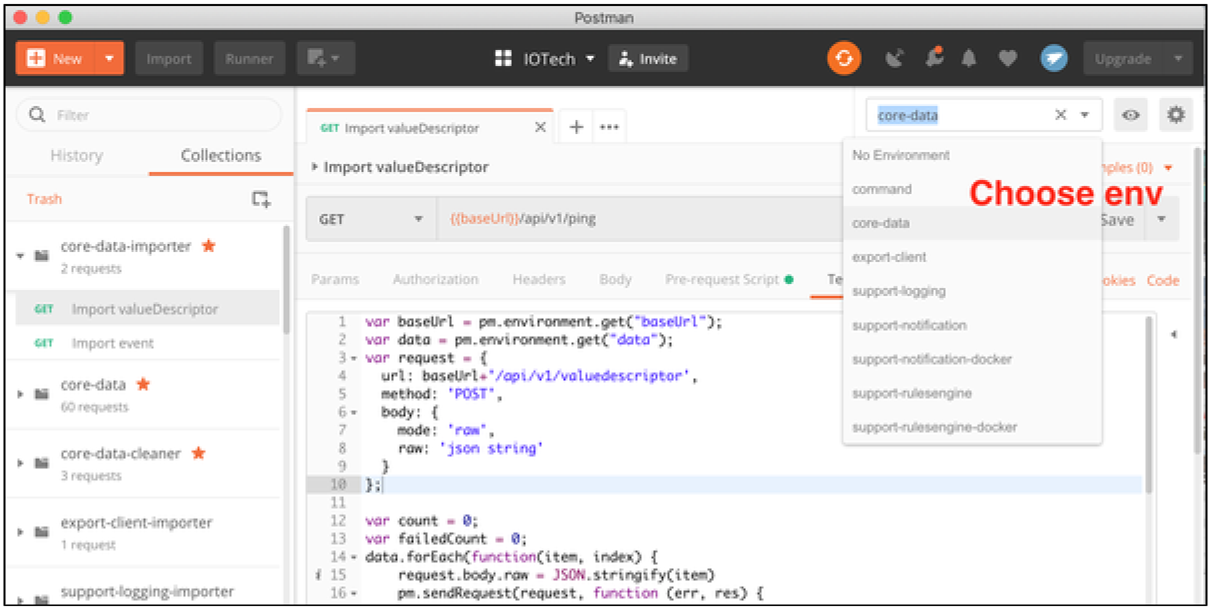


MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces. Learn more about environments

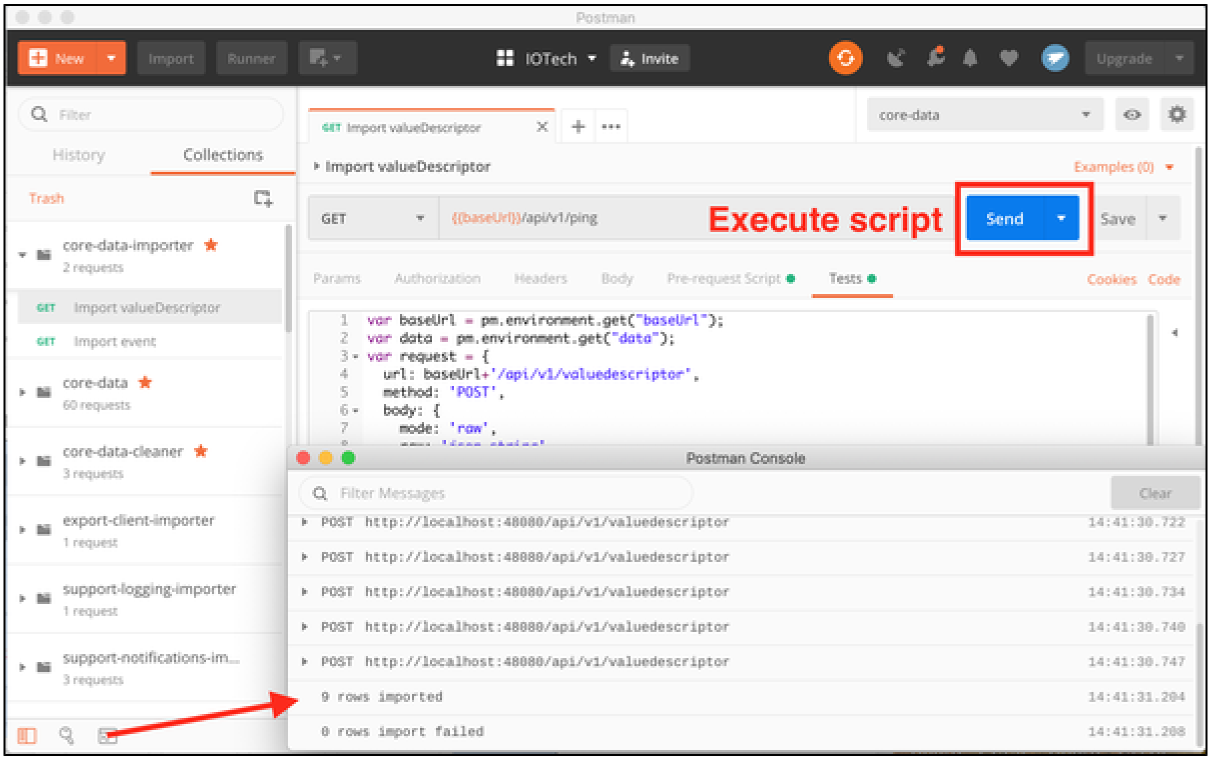
3. Execute the import script

3-1. Execute the import script one by one



GET {{baseUrl}}/api/v1/ping

```
var baseUrl = pm.environment.get("baseUrl");
var data = pm.environment.get("data");
var request = {
  url: baseUrl+'/api/v1/valuedescriptor',
  method: 'POST',
  body: {
    mode: 'raw',
    raw: 'json string'
  }
};

var count = 0;
var failedCount = 0;
data.forEach(function(item, index) {
    request.body.raw = JSON.stringify(item)
    pm.sendRequest(request, function (err, res) {
```



Import valueDescriptor

GET {{baseUrl}}/api/v1/ping

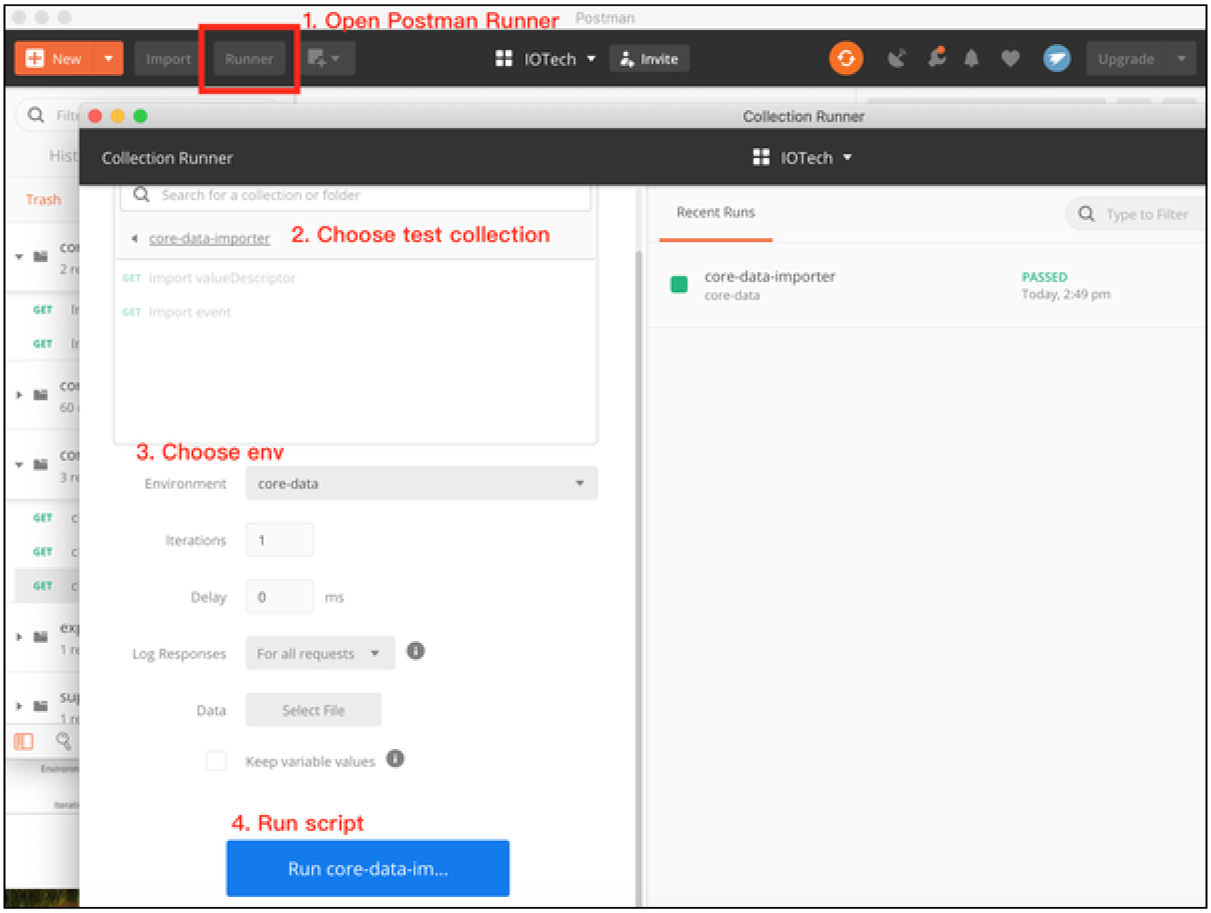
Execute script

Send

```
var baseUrl = pm.environment.get("baseUrl");
var data = pm.environment.get("data");
var request = {
  url: baseUrl+'/api/v1/valuedescriptor',
  method: 'POST',
  body: {
    mode: 'raw',
```

```
POST http://localhost:48080/api/v1/valuedescriptor      14:41:30.722
POST http://localhost:48080/api/v1/valuedescriptor      14:41:30.727
POST http://localhost:48080/api/v1/valuedescriptor      14:41:30.734
POST http://localhost:48080/api/v1/valuedescriptor      14:41:30.740
POST http://localhost:48080/api/v1/valuedescriptor      14:41:30.747
9 rows imported                                         14:41:31.204
0 rows import failed                                    14:41:31.208
```

3-2. Or you can execute the import scripts by Postman Runner



1. Open Postman Runner

2. Choose test collection

core-data-importer

3. Choose env

Environment: core-data

Iterations: 1

Delay: 0 ms

Log Responses: For all requests

Data

□ Keep variable values

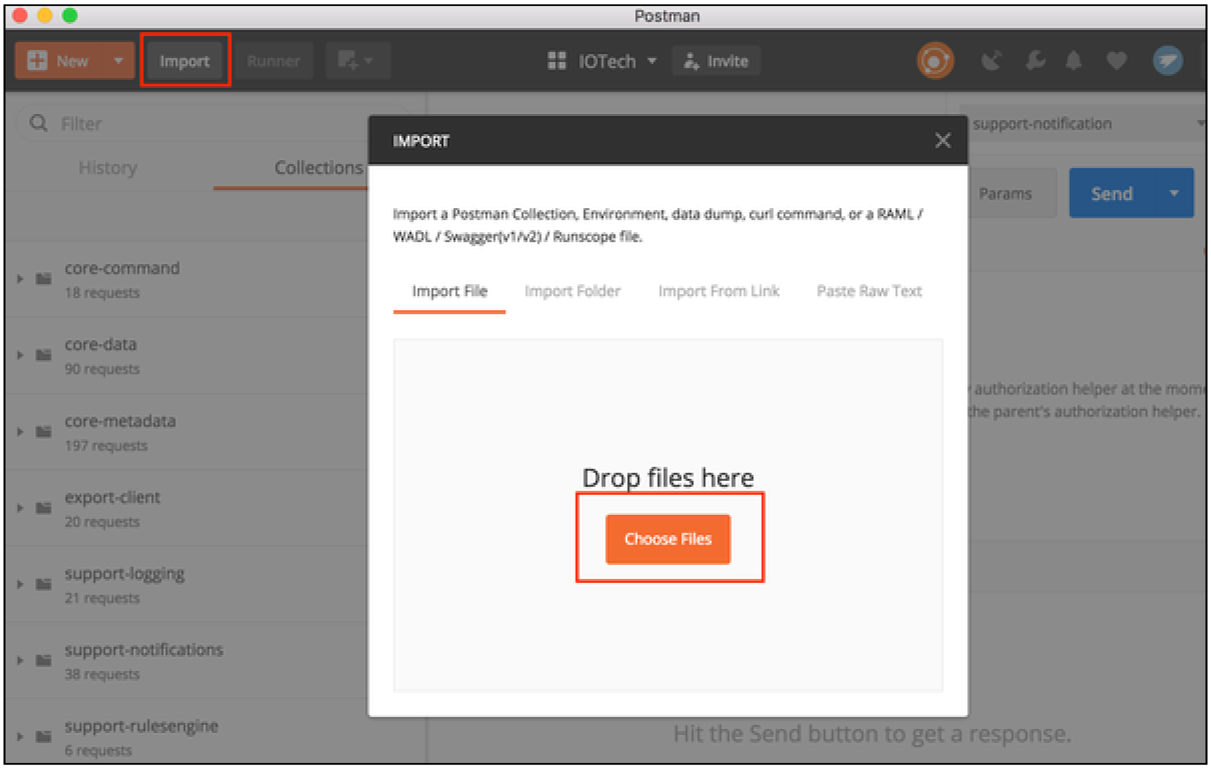
4. Run script

Run core-data-im...

core-data-importer — core-data — PASSED — Today, 2:49 pm

Run test script by Postman

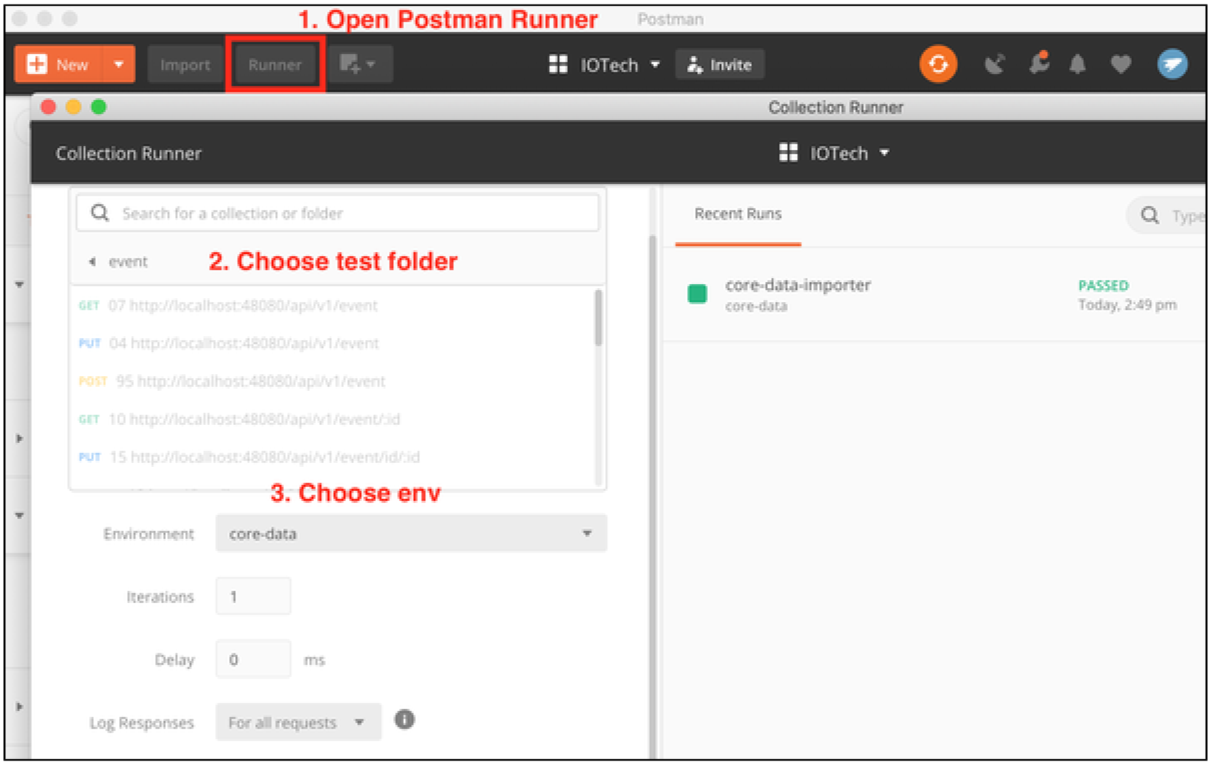
1. Import collection.json file your_path/to/blackbox-testing/bin/postman-test/collections/core-data.postman_collection.json , this json file describes the testing APIs and test assertions.



IMPORT

Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.

2. Choose a test folder and env



1. Open Postman Runner

2. Choose test folder

3. Choose env

Environment: core-data

Iterations: 1

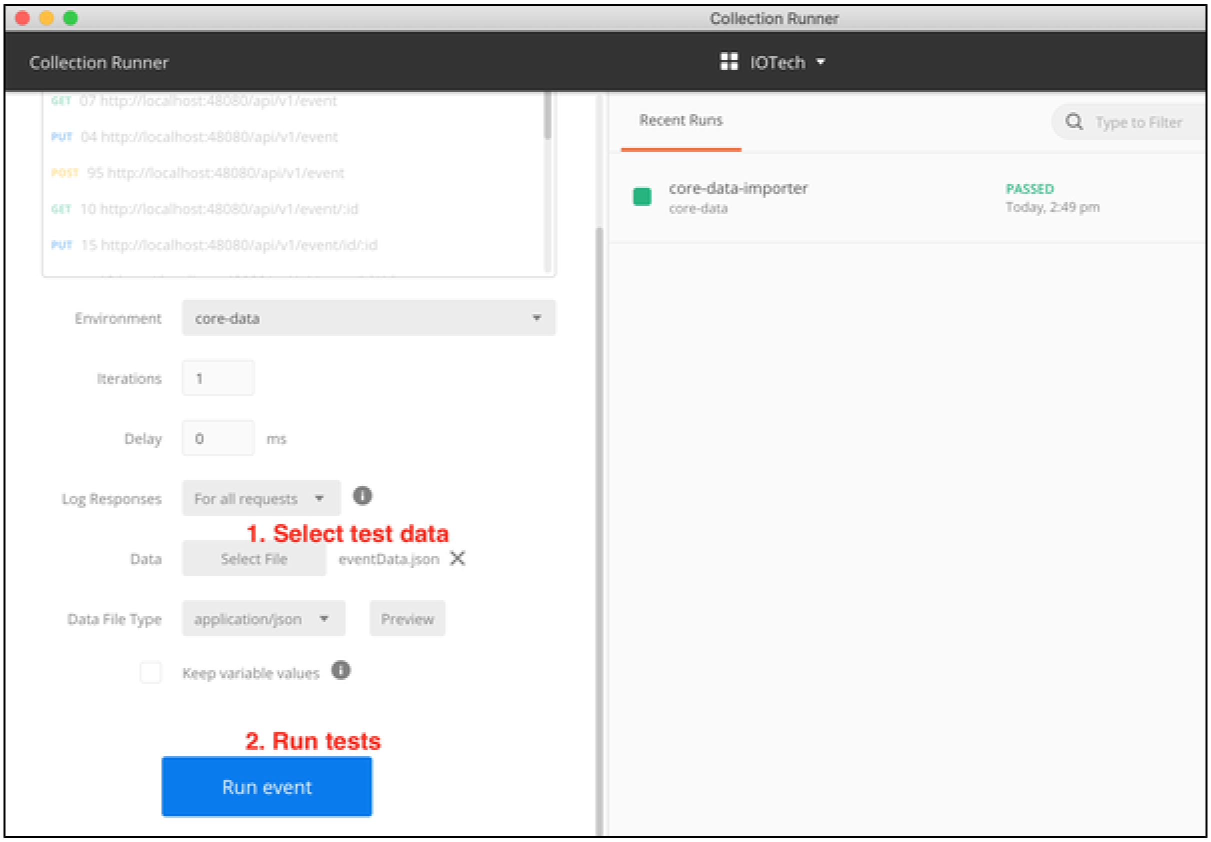
Delay: 0 ms

Log Responses: For all requests

Recent Runs: core-data-importer, core-data — PASSED, Today, 2:49 pm

Postman
New
Import
Runner
IOTech
Invite
Collection Runner
Search for a collection or folder
event
GET 07 http://localhost:48080/api/v1/event
PUT 04 http://localhost:48080/api/v1/event
POST 95 http://localhost:48080/api/v1/event
GET 10 http://localhost:48080/api/v1/event/:id
PUT 15 http://localhost:48080/api/v1/event/id/:id
Type

3. Select test data file your_path/to/blackbox-testing/bin/postman-test/data/eventData.json and run tests.



GET 07 http://localhost:48080/api/v1/event
PUT 04 http://localhost:48080/api/v1/event
POST 95 http://localhost:48080/api/v1/event
GET 10 http://localhost:48080/api/v1/event/:id
PUT 15 http://localhost:48080/api/v1/event/id/:id

core-data-importer PASSED
core-data Today, 2:49 pm

Environment: core-data

Iterations: 1

Delay: 0 ms

Log Responses: For all requests

1. Select test data

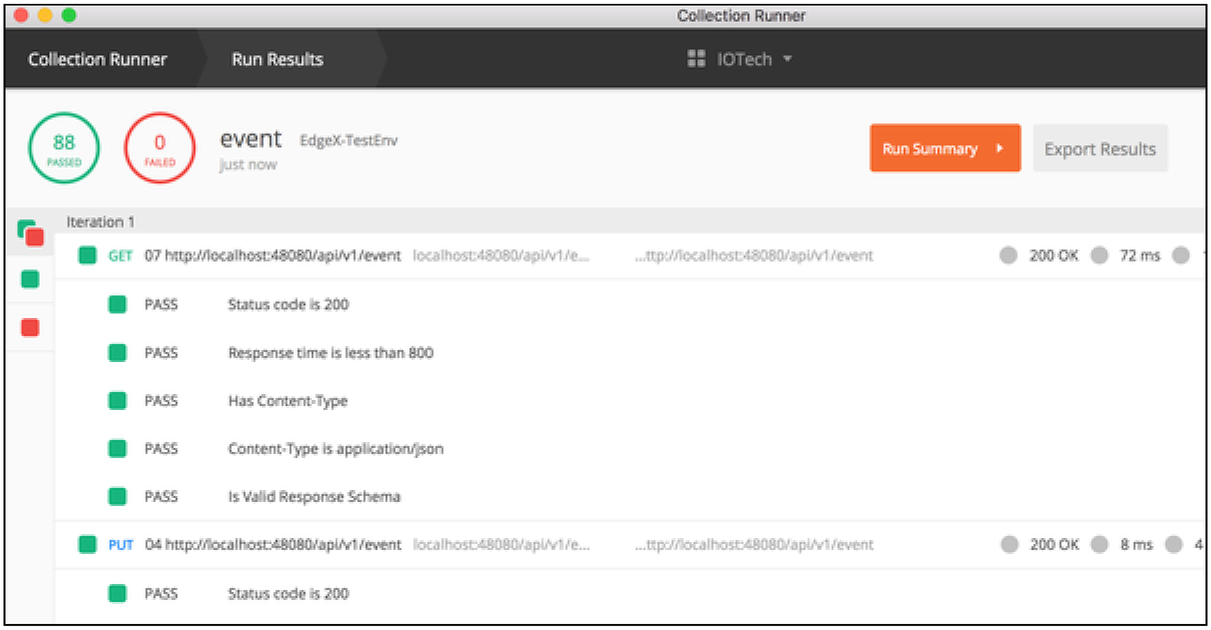
Data: Select File eventData.json

Data File Type: application/json Preview

□ Keep variable values

2. Run tests

4. Then we will see the test result.



88 PASSED | 0 FAILED | event EdgeX-TestEnv Just now

Iteration 1

GET 07 http://localhost:48080/api/v1/event localhost:48080/api/v1/e... ...ttp://localhost:48080/api/v1/event 200 OK 72 ms

PASS Status code is 200

PASS Response time is less than 800

PASS Has Content-Type

PASS Content-Type is application/json

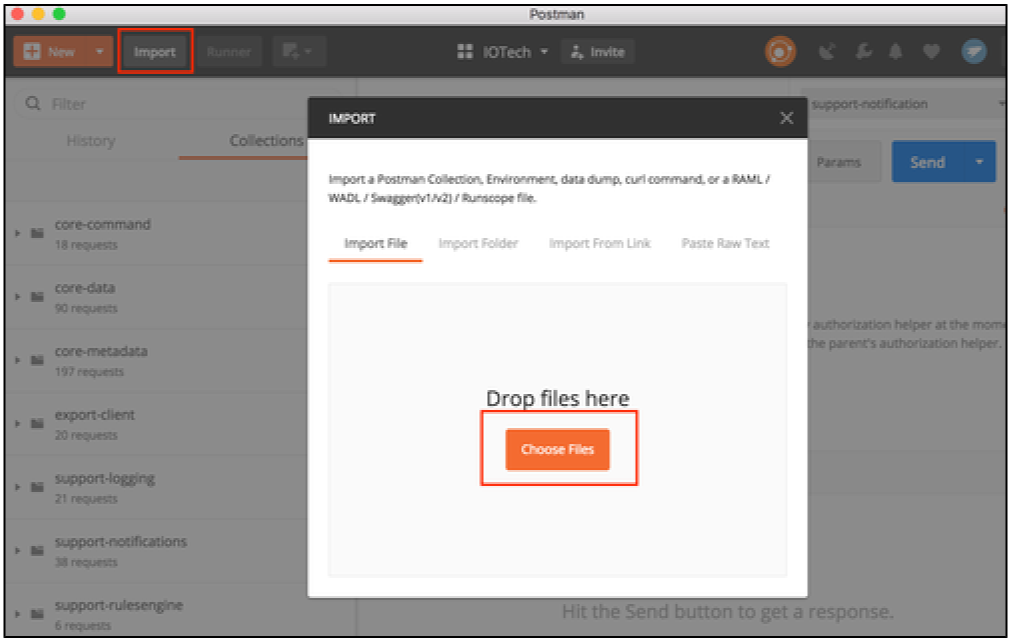
PASS Is Valid Response Schema

PUT 04 http://localhost:48080/api/v1/event localhost:48080/api/v1/e... ...ttp://localhost:48080/api/v1/event 200 OK 8 ms 4

PASS Status code is 200

Clean test data

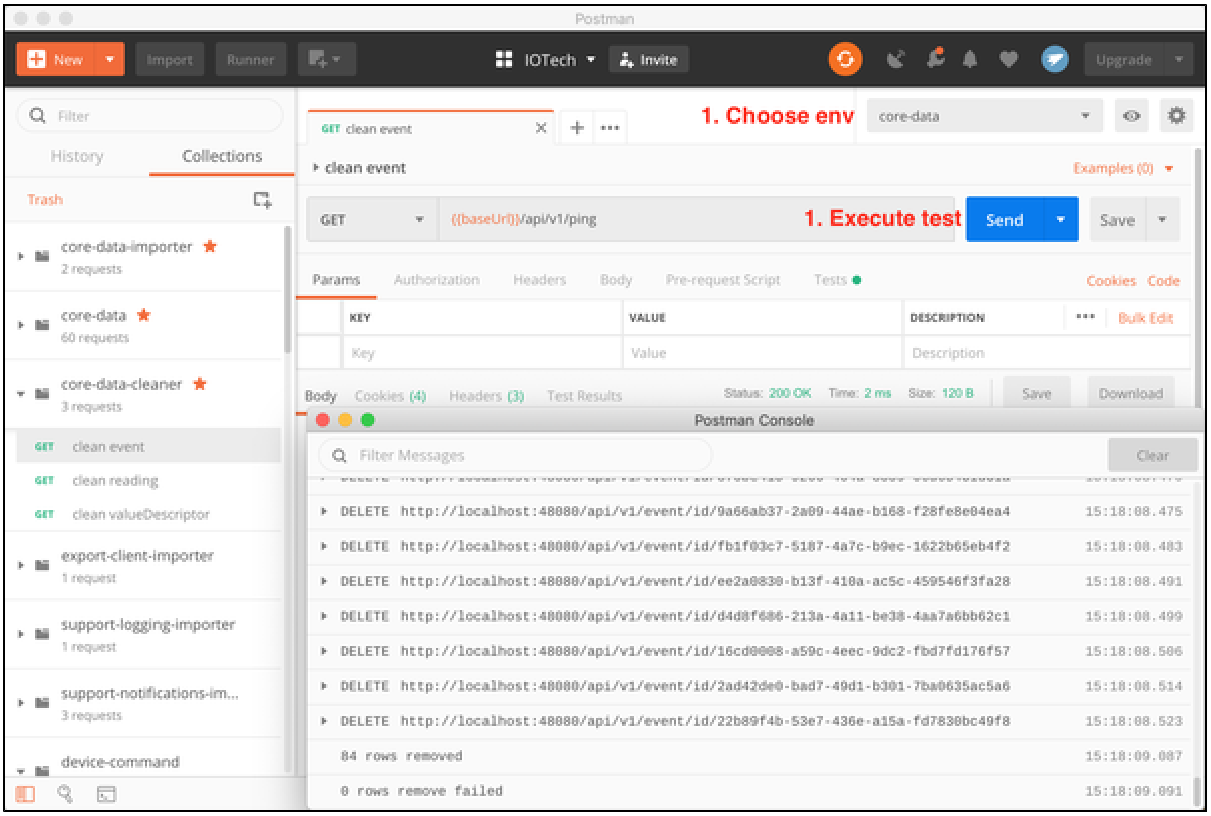
1. Add script which locates at your_path/to/blackbox-testing/bin/postman-test/collections/core-data-cleaner.postman_collection.json



Import a Postman Collection, Environment, data dump, curl command, or a RAML / WADL / Swagger(v1/v2) / Runscope file.

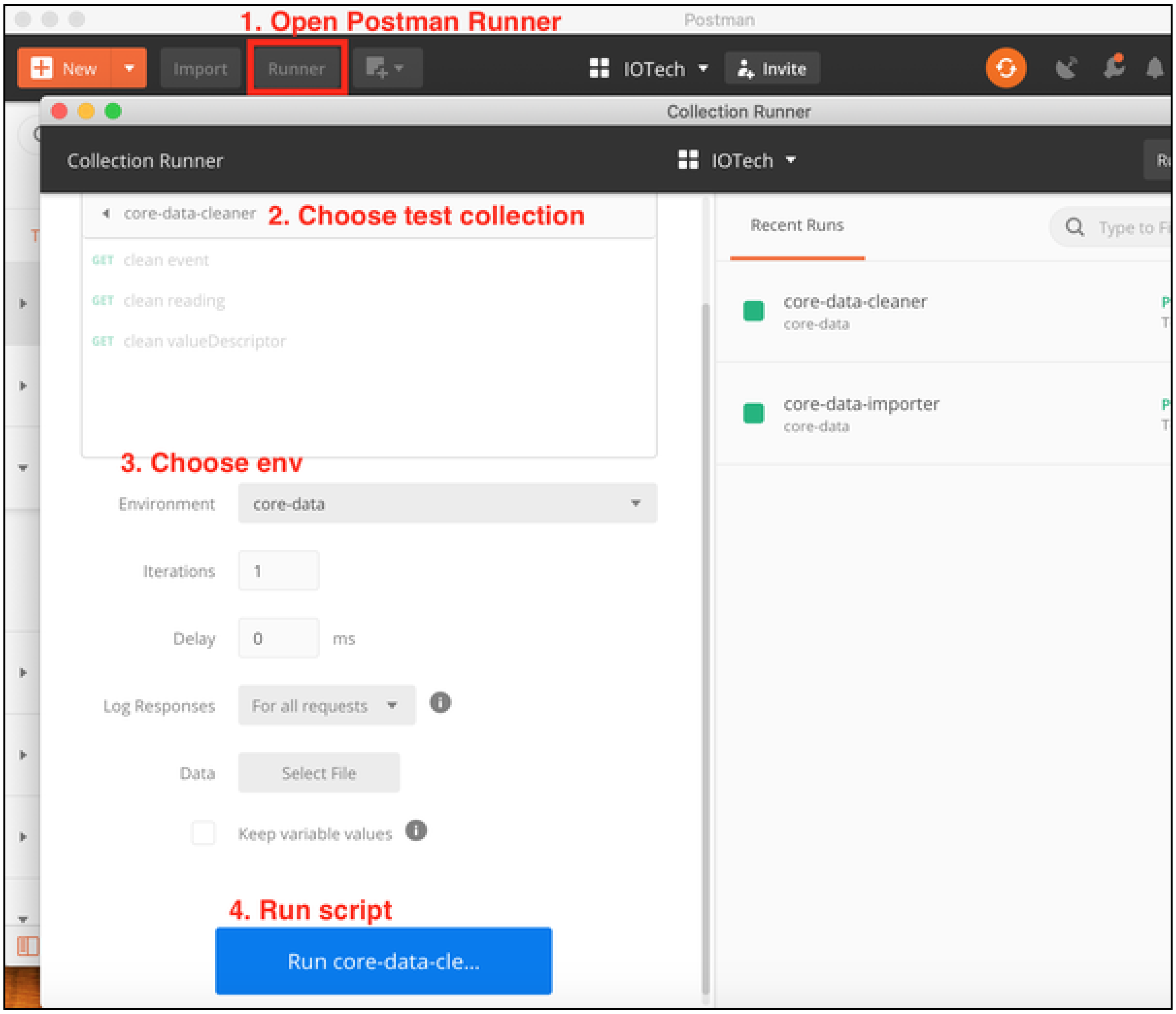
2. Execute the clean script

2-1. Execute the clean script one by one



```
DELETE  http://localhost:48080/api/v1/event/id/9a66ab37-2a09-44ae-b168-f28fe8e04ea4   15:18:08.475
DELETE  http://localhost:48080/api/v1/event/id/fb1f03c7-5187-4a7c-b9ec-1622b65eb4f2   15:18:08.483
DELETE  http://localhost:48080/api/v1/event/id/ee2a0830-b13f-418a-ac5c-459546f3fa28   15:18:08.491
DELETE  http://localhost:48080/api/v1/event/id/d4d8f686-213a-4a11-be38-4aa7a6bb62c1   15:18:08.499
DELETE  http://localhost:48080/api/v1/event/id/16cd0008-a59c-4eec-9dc2-fbd7fd176f57   15:18:08.506
DELETE  http://localhost:48080/api/v1/event/id/2ad42de0-bad7-49d1-b301-7ba0635ac5a6   15:18:08.514
DELETE  http://localhost:48080/api/v1/event/id/22b89f4b-53e7-436e-a15a-fd7830bc49f8   15:18:08.523
84 rows removed                                                                       15:18:09.087
0 rows remove failed                                                                  15:18:09.091
```

2-2. Or you can execute the clean scripts by Postman Runner



1. Open Postman Runner

2. Choose test collection

core-data-cleaner

- GET clean event
- GET clean reading
- GET clean valueDescriptor

3. Choose env

Environment: core-data

Iterations: 1

Delay: 0 ms

Log Responses: For all requests

Data: Select File

□ Keep variable values

4. Run script

Run core-data-cle...

Recent Runs

- core-data-cleaner, core-data
- core-data-importer, core-data