Postman

Automation with Postman

1. Postman Overview
2. Postman vs ACI
3. Postman Glossary
4. Automation with Postman

Postman Overview

- Testing
- Collaboration
- Documentation
- Mocks
- Monitoring

Postman vs ACI

- Testing
- ~~Collaboration~~
- ~~Documentation~~
- ~~Mocks~~
- ~~Monitoring~~

Note: functionality is not available because you cannot sign in to Postman and use its cloud to store collections.

Postman Glossary

- Collection - executable description of API. Includes:
  - API specification
  - API documentation (description of each method, examples)
  - Tests
- Variables:
  - Global - variable initialized in the code in a global scope or from special file
  - Collection - collection level variable, can be initialized only from UI
  - Environment - variable initialized with value from environment definition / special file
  - Data - variable initialized with value from data file (see Control flow)
  - Local - variable initialized in the code in a local scope of pre-request script / test / function

Automation with Postman

- Code review
- Pipeline integration
- Debugging
- Sandbox
- Authorization
- Control flow
- Shared functions
- Error handling
- Connection to database?

Code review

```
"name": "UI endpoint",
"event": [
    {
        "listen": "test",
        "script": {
            "id": "94b2a259-fb8e-4c6e-97fd-aa5975ce6536",
            "exec": [
                "pm.test(request.name + ' - Status code is 200', function () {",
                "    pm.response.to.have.status(200);",
                "});",
                "",
                "pm.test(request.name + ' - Response is valid', function() {",
                "    let expectedResponse = {",
                "        message: 'UI endpoint'",
                "    }",
                "",
                "    pm.expect(pm.response.json()).to.deep.equal(expectedResponse);",
                "});"
            ],
            "type": "text/javascript"
        }
    }
],
"request": {
    "method": "GET",
    "header": [],
    "body": {
        "mode": "raw",
        "raw": ""
    },
    "url": {
        "raw": "{{baseUrl}}/",
        "host": [
        "path": [
    }
},
```

- Collection is stored as a single JSON file.
- Environment specific variables are stored as separate JSON files.

Examples can be found in [Postman API Network]

Pipeline integration

```
newman run /collection.json
    -e /environment.json
    --timeout-script 1000
    ~r html
    ~~reporter-html-export report.html
```

- command line test runner
- contains environment specific variables
- fails test if no response after 1000 ms
- allows to generate html report, also available:
  - CLI report
  - JUnit report
  - JSON report

Debugging during development

- Debugging using Request and Response body in Collection Runner
- Postman Console
  - View requests and responses
  - Log whatever you want with `console.log()`

Debugging of tests running through pipeline

- Reporters:
  - CLI - allows to see in job console logs whatever you print with `console.log()`
  - JSON - provides full information about run including actually sent request and received response:
    - Note: response is stored as a stream, so you will need to decode it
- Exporting state:
  - `--export-environment` - allows to save environment variables file before completing a run
  - `--export-globals` - allows to save global variables file before completing a run

Sandbox

- chai - assertion library
- lodash - dozens of methods to simplify work with arrays, objects, strings, etc
- xml2js - xml to json converter
- crypto-js - encoding/decoding with popular crypto algorithms
- moment - work with date and time
- tv4 - allows to validate json object against json schema
- uuid - uuid generation

More info: Postman Sandbox and Postman Sandbox API reference

Authorization

- Bearer Token
- Basic auth
- Digest Auth
- OAuth 1.0
- OAuth 2.0
- Hawk Authentication
- AWS Signature
- NTLM Authentication

More details here

Control flow

Control flow is managed by

- collection structure (folders, order of scripts in folders) - linear execution
- `postman.setNextRequest()` function - branching and looping
  - `.setNextRequest('request_name')` - sets next request to execute
  - `.setNextRequest(null)` - stops workflow
- data files - multiple executions

Note: no parallel execution available.

Control flow: pm.sendRequest anti-pattern

The `pm.sendRequest` function allows sending HTTP/HTTPS requests asynchronously. Simply put, with asynchronous scripts, you can now execute logic in the background if you have a heavy computational task or are sending multiple requests.

Some things to know about pm.sendRequest():

1. It will break your test report - see newman issue #1303
2. ...

More info on Postman Sandbox API reference.

Shared functions

Pre-request script of the first request:

```
pm.globals.clear();

pm.globals.set('loadUtils', function loadUtils() {
    let utils = {};

    utils.yourFunction = function yourFunction() {
        console.log('foobar');
    }

    return utils;
} + '; loadUtils();');
```

Pre-request script or test of any request:

```
const utils = eval(pm.globals.get('loadUtils'));
utils.yourFunction();
```

Note: according to documentation “Global variables are always be stored as strings. If you’re storing objects or arrays, be sure to JSON.stringify() / JSON.parse() them.”

But if you don’t want to export your globals in files it’s totally fine to save objects in them.

Error handling

To avoid interruption of collection execution all the code that can fail should be located either in:

- `pm.test`
- or `try ... catch`

```
try {
    const res = pm.response.json();
} catch (e) {
    console.log('Unable to parse response');
}

pm.test(request.name + ' - Status code is 422', function() {
    pm.response.to.have.status(422);
});

if (res && res.id) {
    try {
        const utils = eval(pm.globals.get('loadUtils'));
        utils.deleteObject(res.id);
    } catch (e) {
        console.error(`Not able to delete obj ${res.id}`);
    }
}
```

Connection to database?

Not supported. Possible workaround - REST-enabled database, e.g.:

- PostgREST or pREST for Postgres
- Eve, RESTHeart and others for MongoDB

Q&A