**Postman Collection: API Challenges by EvilTester**

**Collection Information**

* **ID:** 360ba7bf-4938-4177-9af9-8d5c9fb8fc09
* **Name:** API Challenges by EvilTester
* **Schema:** [v2.1.0 Collection Schema](https://schema.getpostman.com/json/collection/v2.1.0/collection.json)
* **Exporter ID:** 38734170

**Overview:**

This project introduces a robust approach to API testing by combining response validation and automated error reporting in a seamless workflow. After receiving a response, the system intelligently validates it to ensure it meets expected criteria, such as correct status codes and data integrity. For real-time insights, meaningful logs are generated, including status codes and their corresponding meanings, to streamline debugging. In cases of unexpected responses, the project takes proactive action by automatically creating a detailed Jira issue. Each issue includes essential information like the status code, its meaning, and a descriptive summary to guide investigation. This approach not only ensures comprehensive testing but also enhances collaboration and efficiency by making error management faster and more actionable.

**Requests in the Collection**

1. Status of All the Challenges

* **Request Method:** GET
* **URL:** {{EvilTester\_url}}/challenges
* **Headers:** None
* **Description:** Give the status of all the challenges
* **Response:** status of all challenges

**Status code:** 200k

1. Get All the Todos

* **Request Method:** GET
* **URL:** {{EvilTester\_url}}/todos
* **Headers:** None
* **Description: Fetch the all todos**
* **Response:** Give all todos

**Status code:** 200k

1. Get Done Todos

* **Request Method:** GET
* **URL:** {{EvilTester\_url}}/todos?doneStatus=true
* **Headers:** None
* **Description: Fetch the all done todos**
* **Response:** Give all done todos

**Status code:** 200k

1. Get Todos with ID

* **Request Method:** GET
* **URL:** {{EvilTester\_url}}/todos/:id
* **Headers:** None
* **Description: Get todos with id.**

**Ex: id=3**

* **Response:** get todo

**Status code:** 200k

1. Not Found Page

* **Request Method:** GET
* **URL:** {{EvilTester\_url}}/todo
* **Headers:** None
* **Description: Get the page which is not present.**
* **Authentication**

**Type:** Basic Authentication

**Username:** [mahima27kumari@gmail.com](mailto:mahima27kumari@gmail.com)

**Password:** {{JIRA\_TOKEN}}

* **Response:**

**Status code:**404

1. Get Header

* **Request Method:** HEAD
* **URL:** {{EvilTester\_url}}/todos
* **Headers:** None
* **Description:** Give the header of todos.
* **Response:**

**Status code:** 200k

1. Post in Todos

* **Request Method:** POST
* **URL:** {{EvilTester\_url}}/todos
* **Headers:** X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430
* **Body:**

{

"title": "new API creation",

"doneStatus": false,

"description": ""

}

* **Description: Post todos**
* **Response:**

**Status code**: 201

1. Create a New Challenger Session

* **Request Method:** POST
* **URL:** {{EvilTester\_url}}/challenger
* **Headers:** X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430
* **Description: Create new challenger session.**
* **Response:**

**Status code**: 200k

1. /todos (400)

* **Request Method:** POST
* **URL:** {{EvilTester\_url}}/todos
* **Headers:** X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430
* **Body:**

{

"title": "create new todo",

"doneStatus": "bob",

"description": "created via insomnia"

}

* **Description: Post todos**
* **Response:**

**Status code:** 400(bad request)

1. Unauthorized Post

* **Request Method:** POST
* **URL:** {{EvilTester\_url}}/secret/token
* **Headers:** X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430
* **Authentication:**

 **Username:** admin

 **Password:** password

* **Body:** None
* **Description: Post with wrong username and password.**
* **Response:**

**Status code:** 401

1. /todos (404)

* **Request Method:** POST
* **URL:** {{EvilTester\_url}}/todos/784
* **Headers:**

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Body:**

{

"title": "update postman creation"

}

* **Description: Try to post todos with id : 784 which is not present in todos list.**
* **Response:**

**Status code:** 404(Not found)

1. Get JIRA Projects

* **Request Method:** GET
* **URL:** {{JIRA\_URL}}/rest/api/3/project
* **Authentication:**

**Username:** [mahima27kumari@gmail.com](mailto:mahima27kumari@gmail.com)

**Password:** {{JIRA\_TOKEN}}

* **Headers:** None
* **Description: Fetch all jira project.**
* **Response:**

**Status code:** 200k

1. All Issue Types in Project

* **Request Method:** GET
* **URL:** {{JIRA\_URL}}/rest/api/3/issuetype/project?projectId=10004
* **Authentication:**

**Username:** mahima27kumari@gmail.com

**Password:** {{JIRA\_TOKEN}}

* **Headers:** None
* **Query Parameters:**

projectId: 10004

* **Description: Fetch all issue type in project with id 10004.**
* **Response:**

**Status code:** 200k

1. Test Authorization of Jira

* **Request Method:** GET
* **URL**: {{JIRA\_URL}}/rest/api/3/myself
* **Authentication**: Basic Auth

**Username**: mahima27kumari@gmail.com

**Password**: {{JIRA\_TOKEN}}

* **Headers**: None
* **Description**: Verifies the authorization credentials by retrieving information about the authenticated user.
* **Response**: Details of the authenticated Jira user.

1. **Update Todos**

* **Request Method:** put
* **URL**: {{EvilTester\_url}}/todos/9
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Request Body:**

{

"title": "Retro meeting room"

}

* **Description**: Updates the title of the Todo item with ID 9.
* **Response**: Confirmation of the update.

1. Update Todos with No Title

* **Request Method:** put
* **URL**: {{EvilTester\_url}}/todos/9
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Request Body**:

{

"title": ""

}

* **Description**: Updates the title of the Todo item with ID 9 to an empty string
* **Response**: Confirmation of the update.

1. Delete Todo

* **Request Method:** DELETE
* **URL**: {{EvilTester\_url}}/todos/6
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**: Deletes the Todo item with ID 6.
* **Response**: Confirmation of the deletion.

1. OPTIONS Request on /todos

* **Method**: OPTIONS
* **URL**: {{EvilTester\_url}}/todos
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**: Sends an OPTIONS request to determine the HTTP methods supported by the /todos endpoint.
* **Response**: A list of supported HTTP methods.

1. GET /heartbeat

* **Method**: GET
* **URL**: {{EvilTester\_url}}/heartbeat
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**:  
  Fetches the current status of the heartbeat.
* **Response**:

**Status Codes**:

200: Successful retrieval of heartbeat data.

500: Server error.

1. DELETE /heartbeat

* **Method**: DELETE
* **URL**: {{EvilTester\_url}}/heartbeat
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**:  
  Triggers a simulated server error (500) for the heartbeat endpoint.
* **Response**:

**Status Codes**:

204: Heartbeat successfully deleted

500: Server error.

1. PATCH /heartbeat (Simulates 500 Error)

* **Method**: PATCH
* **URL**: {{EvilTester\_url}}/heartbeat
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**:  
  Triggers a simulated server error (500) for the heartbeat endpoint.
* **Response**:

Status Codes:

500: Server error simulated successfully

1. GET /heartbeat (204 Response)

* **Method**: GET
* **URL**: {{EvilTester\_url}}/heartbeat
* **Headers**:

X-Challenger: 85b250f3-d053-49b1-900d-a7278abf9430

* **Description**:  
  Verifies the absence of heartbeat data.
* **Response**:

**Status Codes**:

204: No content, indicating no active heartbeat

500: Server error

**Pre-request and Test Events**

1. **Prerequest Event**

The prerequest event is triggered before the API request is sent.  
**Current Implementation**: No specific actions are defined.  
**Code Snippet**:

{

"listen": "prerequest",

"script": {

"type": "text/javascript",

"exec": [

// Add logic if required before sending the request.

]

}

}

1. **Test Event**

The test event is triggered after receiving the API response.

**Purpose**:

* Validate status codes.
* Log status information.
* Report errors to Jira if the status code is not 200.

1. **Key Scripts**:

* **Validate Status Code**:
* **Status Message Mapping**:

Map response status codes to human-readable messages.

* **Jira Issue Creation**:

Automatically create a Jira ticket for non-200 responses.

Configure headers for Jira authentication.

Define the issue payload with project details, summary, description, and reporter information.

Send a POST request to Jira API to create an issue.

1. **Code Snippet**:

pm.test("Check for non-200 status code", function () {

if (pm.response.code !== 200) {

console.error("Request failed with status code: " + pm.response.code);

}

});

pm.test("Status code is 200", function () {

pm.response.to.have.status(200);

});

// Log Status Code and Status Meaning

let statuscode = pm.response.code;

console.log("Status Code: ", statuscode);

const statusMessage = {

200: "OK",

201: "Created",

204: "No content",

400: "Bad Request",

401: "Unauthorized",

403: "Forbidden",

404: "Not Found",

405: "Method Not Allowed",

500: "Internal Server Error"

};

let statusMeaning = statusMessage[statuscode] || "Unknown Status Code";

console.log("Status Meaning: ", statusMeaning);

// Jira Integration for Non-200 Responses

if (statuscode !== 200) {

let jiraUrl = "{{JIRA\_URL}}/rest/api/3/issue";

let projectKEY = "EAC";

let projectID = 10004;

let reporterID = "712020:848ae3d3-9d2b-4bfe-9b8d-2d7844fdaedf";

let username = "mahima27kumari@gmail.com";

let jiraAuthToken = "{{JIRA\_TOKEN}}";

let basicAuthValue = "Basic " + btoa(username + ':' + jiraAuthToken);

pm.request.headers.add({

key: 'Authorization',

value: basicAuthValue

});

let jiraSummary = `API Error: ${statuscode} - ${statusMeaning}`;

let jiraDescription = `The API returned status code ${statuscode} (${statusMeaning}). Please investigate.`;

// Jira Issue Payload

let jiraPayload = {

fields: {

project: { id: projectID },

summary: jiraSummary,

description: {

type: "doc",

version: 1,

content: [

{

type: "paragraph",

content: [

{ type: "text", text: jiraDescription }

]

}

]

},

issuetype: { id: "10021" },

reporter: { id: reporterID }

}

};

// Create Jira Issue

pm.sendRequest({

url: jiraUrl,

method: "POST",

header: {

"Authorization": basicAuthValue,

"Content-Type": "application/json"

},

body: {

mode: "raw",

raw: JSON.stringify(jiraPayload)

}

}, (err, res) => {

if (err) {

console.error("Error creating Jira issue: ", err);

} else {

console.log("Jira issue created successfully: ", res.json());

}

});

} else {

console.log("Status code is 200. No Jira issue created.");

}

**Variables:**

**API URL:**

* Key: EvilTester\_url
* Value: <https://apichallenges.eviltester.com>

**Jira Base URL**

* Key: JIRA\_URL
* Value: <https://mahima27kumari-1727938070667.atlassian.net>

**Jira Token**

* Key: JIRA\_TOKEN