A close-up of a logo

Description automatically generated

**Best Practices for Creating, Reading, Updating, and Deleting Change Requests with Chaining Process in Postman**

**Objective:**  
To ensure efficient, accurate, and secure interactions with ServiceNow APIs through Postman by following best practices.

**Summary of Learning:**

* Understanding API endpoints and HTTP methods for CRUD operations on change requests in ServiceNow.
* Utilizing Postman for API testing and automation.
* Implementing best practices to enhance reliability, security, and maintainability of API interactions.

**Best Practices:**

|  |  |
| --- | --- |
| Best Practices | Details |
| Organize Requests and Collections | - **Create Collections:** Group related API requests into collections for better management. **Use Folders:** Use folders within collections to organize requests by functionality (e.g., Create, Read, Update, Delete). |
| Use Environment Variables | - **Environment Setup:** Define environments (e.g., Development, Staging, Production) and use environment variables for instance URLs, credentials, and other settings. **Variable Usage:** Use variables in request URLs, headers, and bodies to avoid hardcoding values. |
| Secure Your Credentials | - **Environment Variables:** Store sensitive information like usernames, passwords, and tokens in environment variables. **Encrypted Storage:** Ensure environment variables are securely stored and encrypted. |
| Properly Format JSON Payloads | - **Validation Tools:** Use JSON validation tools like [JSONLint](https://jsonlint.com/) to ensure payloads are correctly formatted. **Template Payloads:** Create template JSON payloads for different API requests to ensure consistency. |
| Ensure Required Fields Are Included | - **Check Documentation:** Ensure required fields are included in the JSON payload by referring to ServiceNow API documentation. |
| Use Descriptive Fields | - Provide meaningful values for fields like short\_description and description. |
| Error Handling | - Add test scripts in Postman to handle errors and verify successful request creation. Example: javascript pm.test("Status code is 201", function () { pm.response.to.have.status(201); }); |
| Validate Response Data | - Use assertions to ensure the response contains expected data fields and values. Example: javascript pm.test("Response contains sys\_id", function () { pm.expect(pm.response.json().result.sys\_id).to.exist; }); |
| Handle Errors Gracefully | - Ensure error responses (e.g., 404 Not Found) are handled appropriately. Example: javascript pm.test("Status code is 200", function () { pm.response.to.have.status(200); }); |
| Partial Updates | - If the API supports it, use PATCH instead of PUT for partial updates to avoid overwriting existing fields. |
| Confirm Updates | - Add test scripts to confirm the changes were applied successfully. Example: javascript pm.test("Status code is 200", function () { pm.response.to.have.status(200); }); pm.test("Short description is updated", function () { pm.expect(pm.response.json().result.short\_description).to.eql("Updated description"); }); |
| Confirm Deletion | - Ensure the change request is deleted by checking the status code and the response body. Example: javascript pm.test("Status code is 204", function () { pm.response.to.have.status(204); }); |
| Automate Tests | - Use Postman’s test scripts to automate the validation of responses and status codes. Write tests to check for expected values in the response body. Example: javascript pm.test("Response time is less than 200ms", function () { pm.expect(pm.response.responseTime).to.be.below(200); }); |
| Chaining Requests | - Ensure variables set in one request are correctly used in subsequent requests. Validate the entire flow by chaining Create, Read, Update, and Delete operations in a sequence. |
| Document Requests | - Use Postman’s description field to document the purpose and usage of each request. Include details about required headers, body parameters, and expected responses. |
| Use HTTPS | - Always use HTTPS to encrypt API requests and responses to protect sensitive data. |
|  |  |