International Journal on Science and Technology (IJSAT)

E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Exploring SAP API Testing with Postman, Swagger, and SOAP UI

Sireesha Perabathini

(Independent Researcher)
Illinois, USA
perabathinisireesha@gmail.com

Abstract

Today's interconnected enterprise environments require efficient system integration to optimize business processes effectively. SAP (Systems, Applications and Products) delivers a powerful collection of enterprise resource planning (ERP) tools which help manage core business operations like procurement, sales, and finance. API integration serves as a crucial element to enable seamless communication between SAP systems and external applications. The paper examines the application of Postman, Swagger, and SOAP UI as API testing tools for SAP APIs by evaluating their functionalities and real-world usage scenarios. Various SAP API types such as OData, SOAP, IDoc, and RFC APIs are examined in this paper which emphasizes the need for API testing to support system reliability. This paper explains how to use tools like Postman, Swagger and SOAP UI for validating SAP API requests and responses along with automating tests and improving team collaboration. Practical examples show that API testing helps enterprise systems function more efficiently by maintaining the integrity of integration points.

Keywords: SAP API, API testing, Postman, Swagger, SOAP UI, integration testing, OData, REST API, SOAP API, enterprise systems, automation testing, API validation, enterprise resource planning (ERP), testing tools, system reliability

I. INTRODUCTION

In today's interconnected enterprise environments, the ability to efficiently integrate systems is crucial. SAP (Systems, Applications, and Products) provides businesses with powerful enterprise resource planning (ERP) tools that manage business processes such as procurement, sales, and finance. Many modern SAP systems offer APIs that allow external systems to communicate with them seamlessly. Testing these APIs is vital to ensure that integration points function as expected and maintain system reliability.

API testing tools such as Postman, Swagger, and SOAP UI are essential for simulating requests, validating responses, and performing integration tests. These tools offer flexibility, ease of use, and a rich feature set for both developers and testers. This paper explores how Postman, Swagger, and SOAP UI can be utilized for SAP API testing, their features, and the steps involved in testing SAP APIs using these tools. Additionally, real-time business use cases will demonstrate how API testing contributes to enterprise operations.

II. OVERVIEW OF SAP API'S

SAP APIs are designed to expose various SAP functionalities as services that can be accessed over the network. These services typically follow the RESTful API architecture, which is based on HTTP requests like GET, POST, PUT, DELETE, etc [2] . In the context of SAP, these APIs can connect with various systems such as SAP S/4HANA, SAP Fiori, SAP Cloud Platform, and others.

The following types of APIs are commonly used in SAP:

- OData APIs: These are the most widely used RESTful APIs in SAP. They allow external applications to perform CRUD (Create, Read, Update, Delete) operations on SAP data models.
- SOAP APIs: These are XML-based APIs that use the Simple Object Access Protocol to communicate.
- IDoc APIs: Intermediate documents (IDocs) are used to transfer data between SAP and other applications.
- RFC (Remote Function Call) APIs: RFC is a proprietary SAP protocol that allows communication with SAP systems.

For testing and debugging these APIs, tools like Postman, Swagger, and SOAP UI are invaluable, offering capabilities for sending requests, receiving responses, and performing tests in different formats.

A. *Postman for SAP API Testing*

Overview of Postman: Postman is a popular API testing tool that enables developers and testers to create, send, and monitor API requests. It supports both RESTful and SOAP-based APIs and is commonly used for testing integrations in various environments.

1) *Key Features of Postman:*

a) Request Building: Postman allows users to create complex HTTP requests by specifying the HTTP method, URL, headers, parameters, and body content [2] [2].
b) Test Automation: Postman provides a scripting environment to write tests and automate the validation of API responses using JavaScript [5].
c) Environment Variables: Variables such as base URLs, tokens, and user credentials can be stored and reused across multiple requests, improving efficiency.
d) Collection and Workspaces: Postman enables users to organize tests into collections and share them with teammates. Workspaces allow teams to collaborate effectively on API testing.
e) Mock Servers: Postman can simulate an API server to test client-side code before the actual backend is available.

2) *Steps for Testing SAP APIs Using Postman*

a) Set Up the Environment: Define environment variables such as SAP system URL, authentication credentials (OAuth tokens, user credentials), and other configurations (such as headers or timeouts).
b) Create API Requests: In Postman, create requests corresponding to the SAP APIs. For example, to test an OData service, you can create a GET request with a URL pointing to the OData endpoint.

c) Authentication: Most SAP APIs require authentication. You can use basic authentication, OAuth 2.0, or API keys [1] depending on the setup. In Postman, authentication settings can be configured in the Authorization tab.
d) Send Requests: Send requests to the SAP system and observe the responses. Postman displays the status code, response time, and body of the response.
e) Monitor API Behavior: Postman offers the ability to create automated tests, schedule them, and integrate them into Continuous Integration/Continuous Deployment (CI/CD) pipelines.

3) Benefits of Postman fir SAP API Testing:

a) Intuitive Interface: The user-friendly graphical interface of Postman makes it easy to build and test API requests, even for non-developers.
b) Collaboration: Postman collections can be shared with teammates, improving collaboration between developers and testers.

B. Swagger for SAP API Testing

Overview of Swagger: Swagger, now known as OpenAPI, is an open-source framework used for designing, building, and documenting RESTful APIs. It provides a comprehensive suite of tools for developing APIs, including the ability to automatically generate client libraries, server stubs, and API documentation [3].

1) Key Features of Swagger

a) API Documentation: Swagger UI automatically generates interactive API documentation, allowing developers and testers to explore and test API endpoints.
b) API Definition: The OpenAPI Specification (OAS) allows the description of API endpoints, request/response formats, authentication methods, and more.
c) Swagger Codegen: It allows for the generation of client libraries and server stubs in various programming languages.

2) Steps for Testing SAP APIs Using Swagger

a) Import or Define API Specification: Define your SAP API using OpenAPI standards or import existing SAP OData or REST API specifications into Swagger Editor.
b) Explore API Endpoints: Swagger UI provides an interactive interface to explore available API endpoints, along with input fields for parameters, headers, and request bodies.
c) Send Requests: You can directly invoke requests from Swagger UI, making it easy to test the API's functionality. It sends the request and shows the response within the same interface.
d) Validate API Response: Swagger UI displays the response status, headers, and body in an easy-to-read format. This is useful for quickly identifying issues and debugging the API.
e) Generate Code: Swagger allows you to generate client code in multiple programming languages. This can be useful when testing the integration with external applications.

3) Benefits of Swagger for SAP API Testing

a) Comprehensive API Documentation: Swagger automatically generates interactive documentation that simplifies understanding API structure and usage.
b) Real-Time Testing: Swagger UI allows testers to directly execute API requests and view results in real time.
c) Standardized API Definition: The OpenAPI Specification ensures that the API design follows a consistent and standardized format, promoting clear communication between teams.

C. *SOAP UI for SAP API Testing*

Overview of SOAP UI: SOAP UI is a widely used API testing tool primarily designed for testing SOAP-based web services but also supports REST APIs [2]. It is particularly valuable when working with older SAP systems or integrating with services that use XML-based SOAP APIs. SOAP UI provides a feature-rich environment for testing both REST and SOAP web services, offering powerful capabilities for functional, security, and performance testing.

1) *Key Features of SOAP UI*

a) Support for SOAP and REST APIs: SOAP UI allows testing of both SOAP and REST APIs, providing versatility for different SAP API types.
b) Security Testing: SOAP UI has advanced features for security testing, such as WS-Security, OAuth, and SSL certificates, which are often necessary for SAP integrations.
c) Data-Driven Testing: With SOAP UI, you can run tests with multiple data sets, making it easy to validate how APIs behave under various scenarios.
d) Assertion and Validation: SOAP UI allows users to add assertions to validate the API response, ensuring that the response matches expected outcomes.

2) *Steps for Testing SAP API's using SOAP UI*

a) Support for SOAP and REST APIs: SOAP UI allows testing of both SOAP and REST APIs, providing versatility for different SAP API types.
b) Security Testing: SOAP UI has advanced features for security testing, such as WS-Security, OAuth, and SSL certificates, which are often necessary for SAP integrations.
c) Data-Driven Testing: With SOAP UI, you can run tests with multiple data sets, making it easy to validate how APIs behave under various scenarios.
d) Assertion and Validation: SOAP UI allows users to add assertions to validate the API response, ensuring that the response matches expected outcomes.
e) Performance Testing: SOAP UI offers tools for load testing and monitoring API performance under stress.

3) *Benefits of SOAP UI for SAP API Testing*

a) Comprehensive Testing Support: SOAP UI is suitable for both functional and performance testing, offering tools for simulating high loads and checking security features [4].
b) Data-Driven Testing: Easily test APIs with various datasets to ensure consistent behavior across different inputs [4].
c) Advanced Security Testing: SOAP UI is designed with robust security testing features, important when dealing with sensitive business data in SAP systems [4].

III. REAL-TIME BUSINESS USESCASES FOR SAP API TESTING

A. *SAP S/4HANA Integration with External E-Commerce Platform:*

1) *Business Scenario:*

A global retailer uses SAP S/4HANA to manage inventory, sales, and orders. To enable e-commerce functionality, an API is exposed to facilitate real-time synchronization between SAP S/4HANA and the retailer's external e-commerce platform. The API manages operations such as updating inventory, submitting orders, and managing shipping data

2) *Testing Tools:*

a) Postman: Used for validating real-time data exchanges and automating regression tests to ensure that integration between the SAP backend and e-commerce front-end runs smoothly.
b) Swagger: Provides interactive API documentation that allows external e-commerce developers to easily explore the available endpoints and understand the parameters for correct API integration.
c) SOAP UI: Ensures the security of data exchange by testing authentication protocols (e.g., OAuth, JWT) and securing the data transmission.

3) Challenges in Testing and Common Defects:

a) Real-time Synchronization: Since the e-commerce platform requires real-time updates, there can be challenges related to latency and data consistency. Any delay in data synchronization between SAP S/4HANA and the e-commerce platform can lead to issues like incorrect inventory counts or delayed order fulfillment.
b) API Rate Limiting: E-commerce platforms often have rate limits on API calls. Handling API rate limiting and ensuring that the SAP API doesn't exceed these limits can be tricky.
c) Latency Issues: Orders or inventory updates taking too long to reflect on the e-commerce platform.
d) Data Mismatch: The wrong inventory or price being displayed on the e-commerce platform due to data synchronization issues.
e) Authentication Failures: Issues in securing API communication, leading to data exposure risks or access problems.

4) Mitigation:

a) Implement load testing and performance testing using tools like JMeter and Load Runner to simulate high traffic scenarios and ensure the system can handle peak loads.
b) Use retry mechanisms or eventual consistency models for handling API rate limiting and network delays.
c) Ensure robust data validation by using Swagger to review and document data mapping rules for APIs to ensure data accuracy between systems.

B. SAP SuccessFactors Integration with HRMS:

1) Business Scenario:

An organization uses SAP SuccessFactors to manage HR processes and integrates it with an external Human Resource Management System (HRMS). The HRMS needs to pull employee data, benefits information, and payroll details through APIs exposed by SAP SuccessFactors.

2) Testing Tools:

a) Postman: Verifies that REST API endpoints for payroll data retrieval are functioning correctly by checking data accuracy and response time.
b) Swagger: Used for understanding and interacting with the HRMS API documentation, allowing HRMS developers to access endpoint details for smooth integration.
c) SOAP UI: Used for security testing, ensuring that encrypted employee data is securely transmitted between SAP SuccessFactors and the HRMS.

3) Challenges in Testing and Common Defects:

a) Sensitive Data: Employee data such as payroll, benefits, and personal information requires strict security measures during transmission. Testing for data encryption and secure transmission (e.g., SSL/TLS) is vital.
b) Compliance with Regulations: Testing must ensure that the integration complies with various data protection regulations like GDPR or HIPAA. Improper handling of sensitive data can lead to legal and reputational consequences.
c) Data Consistency: Ensuring that the data pulled from SuccessFactors is accurate and synchronized with the HRMS system in real-time.
d) Data Corruption: Employee data may be corrupted during transmission due to improperly handled API requests.
e) Insecure Transmission: Lack of proper encryption in API communications, leading to potential data breaches.
f) Data Mapping Errors: Issues with the integration layer, such as mismatches between data fields in SAP SuccessFactors and HRMS, which result in incorrect employee data being transferred.

4) Mitigation:

a) Security Testing using SOAP UI to ensure end-to-end encryption of sensitive data.
b) Utilize Postman's automated tests to validate that data integrity checks are in place, ensuring data consistency between systems.
c) Ensure robust error handling and logging are in place to detect issues like data corruption early, allowing quick remediation.

C. SAP Fiori UI Integration with Third-Party Reporting Tools:

1) Business Scenario:

A company uses SAP Fiori UI for accessing SAP data through modern web interfaces. To enable external reporting tools to access SAP data, an API is exposed that allows real-time access to sales, inventory, and finance reports.

2) Testing Tools:

a) Postman: Automates the testing of API endpoints and ensures that the external reporting tool retrieves the correct and up-to-date data from SAP.
b) Swagger: Provides detailed documentation for external developers to understand how to integrate their reporting tool with SAP Fiori, including data endpoints, query parameters, and response formats.
c) SOAP UI: Tests the performance and reliability of the API when handling large volumes of data.

3) Challenges in Testing and Common Defects:

a) Performance Under Load: Handling large volumes of data requests, especially during peak reporting periods, could cause the SAP Fiori API to become slow or unresponsive. Ensuring that the API performs efficiently when queried by multiple users is a significant challenge.
b) Data Integrity: The external reporting tool needs accurate and up-to-date data for reporting purposes. Any discrepancy in the data retrieved from SAP Fiori could lead to misleading business insights.
c) API Versioning: As SAP Fiori evolves, changes in the API version could impact integration with third-party tools, requiring constant maintenance of the integration layer.

d) Slow API Response: Slow response times, especially when retrieving large datasets, causing delays in generating reports.
e) Data Inconsistencies: The reporting tool may pull incomplete or outdated data from SAP Fiori, leading to incorrect reports.
f) API Compatibility Issues: Changes in the SAP Fiori API over time may break backward compatibility with external reporting tools.

4) Mitigation:

a) Perform load testing using tools like JMeter and LoadRunner or NeoLoad to simulate heavy traffic and ensure the API can handle high request volumes.
b) Use version control in Swagger to manage different versions of the API and ensure that external reporting tools are always compatible with the latest version.
c) Implement data validation rules in Postman to ensure that the data retrieved from SAP Fiori meets the requirements of the external reporting tool, avoiding inconsistencies.

D. Energy and Utilities Industry: SAP IS-U Integration for Meter Data Management:

1) Business Scenario:

In the energy and utilities sector, utilities companies use SAP IS-U (Industry Solution for Utilities) for managing customer data, meter readings, billing, and service requests. Meter data is collected from smart meters and processed by the MDM system. APIs are exposed to synchronize meter data with SAP IS-U and third-party billing systems.

2) Testing Tools:

a) Postman: Validates that meter data, such as usage readings, is correctly transmitted from the smart meters to SAP IS-U for billing processing.
b) Swagger: Provides interactive API documentation for third-party billing systems to integrate with SAP IS-U seamlessly.
c) SOAP UI: Ensures that the data exchange process is secure, testing for vulnerabilities in the API related to encryption and authentication.

3) Challenges in Testing and Common Defects:

a) Real-Time Data Processing: Meter data is collected in real-time, which means that the APIs must process large amounts of incoming data quickly and accurately. Delays in processing could lead to billing errors or incorrect customer service responses.
b) Data Accuracy: Meter readings must be accurate for correct billing. Any discrepancy in data transmission between the smart meters, Meter Data Management system, and SAP IS-U could lead to billing errors.
c) Security and Compliance: Metering data is sensitive and ensuring that the API properly encrypts data and complies with industry regulations (e.g., utility-specific security standards) is essential.
d) Inaccurate Meter Data: Errors in transmission, such as incorrect readings or missing data, could lead to billing discrepancies.
e) Security Vulnerabilities: Insufficient data protection during transmission could expose sensitive customer information.
f) Performance Bottlenecks: The API might not be able to handle high volumes of meter data efficiently, leading to performance issues.

4) *Mitigation:*

a) Perform security testing using SOAP UI to ensure proper data encryption during transmission and compliance with industry standards [4].
b) Use Postman to test real-time data exchanges and validate that meter readings are correctly transmitted and processed in SAP IS-U for accurate billing.
c) Implement load testing to ensure that the system can handle large volumes of meter data in peak periods, avoiding performance bottlenecks.

E. *Use Case: SAP Integration for Public Sector Financial Management:*

1) *Business Scenario:*

A government agency uses SAP to manage public funds, budgeting, and financial reporting. To improve transparency, streamline processes, and ensure timely disbursement of funds, the agency integrates SAP's financial management system (SAP FI) with external government payment platforms and auditing systems via APIs. These APIs facilitate smooth data communication between SAP and external systems to perform tasks including fund allocation, expenditure tracking and reporting for compliance.

2) *Testing Tools:*

a) Postman: Used for automating the testing of API requests for fund disbursements, ensuring the correct processing of budget allocations, and verifying that financial data is accurately sent to external systems.
b) Swagger: Provides an interactive API documentation interface to external developers, such as payment system vendors or auditing firms, to help them integrate their systems with SAP efficiently.
c) SOAP UI: Used for security testing to ensure that sensitive financial data, such as government funds and taxpayer information, is encrypted and securely transmitted

3) *Challenges in Testing and Common Defects:*

a) Security and Compliance: Government data is highly sensitive, and APIs must comply with strict regulations like the General Data Protection Regulation (GDPR) and other country-specific data protection laws. Testing must ensure that data exchanges are encrypted, and that unauthorized access is prevented.
Real-time Data Processing: Financial systems require real-time processing for fund distribution activities. Public dissatisfaction or financial management process breaches may occur due to transaction processing delays or errors.
b) Interoperability with External Systems: The APIs need to function smoothly alongside multiple external systems including payment platforms as well as banking and auditing applications. The differing data structures and communication protocols of systems create additional testing complexity.
c) Data Corruption: During data transmission financial information like fund allocation details may become corrupted because of faulty API handling which results in errors in financial reporting and disbursement.
d) Security Flaws: If authentication or encryption systems are weak they could allow unauthorized access to sensitive government financial data which may lead to fraud or data breaches.

e) Performance Bottlenecks: The API might not perform efficiently during peak financial periods (e.g., end-of-quarter or end-of-year), causing delays in fund disbursement or reports

4) Mitigation:

a) Security Testing: Using SOAP UI, conduct extensive security tests to ensure proper encryption standards (e.g., SSL/TLS) are used during the data exchange and that compliance with security regulations is maintained.

b) Load Testing: Use JMeter/Neoload/LoadRunner to simulate a high volume of financial transactions and data requests to ensure that the APIs can handle large-scale operations without performance degradation.

c) Data Validation: Implement data validation checks using Swagger and Postman to ensure that data is consistently accurate across both SAP and external systems, with automated tests verifying that data fields are correctly mapped between systems.

d) Monitoring and Logging: Set up real-time monitoring to identify any discrepancies or failures during API exchanges, enabling quick remediation of issues before they affect operations.

IV. CONCLUSION

Testing SAP APIs is essential for ensuring system integration and business operation stability. Postman, Swagger, and SOAP UI each offer distinct strengths: Postman excels at automating requests, testing, and team collaboration; Swagger provides comprehensive documentation and testing tools; and SOAP UI supports both SOAP and REST protocols with advanced security and performance testing capabilities. When used together, these tools ensure that SAP APIs function as expected, maintaining security and efficiency in business operations while enabling automated testing workflows, integration point validation, and API security assessments.

REFERENCES

[1] SAP Blogs, “Testing REST APIs in XSA Applications Without UI Layer”,2018[Online].Available: https://community.sap.com/t5/technology-blogs-by-sap/testing-rest-apis-in-xsa-applications-without-ui-layer/ba-p/13355826

[2] Alan J Richardson , "Automating and Testing a REST API: A Case Study in API testing using: Java, REST Assured, Postman, Tracks, cURL and HTTP Proxies," Compendium Developments Ltd (July 4, 2017).

[3] Noah Dietz,, "Fulfilling a RESTful Commitment: Test-Suite Generation for Swagger-based REST APIs," 2018[Online].Available https://digitalcommons.calpoly.edu/cscsp/87

[4] Charitha Kankanamge, “Web Services Testing with SoapUI”, PACKT Publishing, 2012.

[5] SAP Blogs, “Using Postman for testing your Odata development”,2019[Online].Available: https://community.sap.com/t5/technology-blogs-by-members/using-postman-for-testing-your-odata-development/ba-p/13419863