# SoapUI Testing & Debugging Cheatsheet

A comprehensive guide to testing and debugging web services using SoapUI, covering essential features, configurations, and best practices.

## Getting Started with SoapUI

### Installation and Setup

**1. Download SoapUI:**

- Visit the official website: https://www.soapui.org/downloads/soapui.html
- Choose the appropriate version for your operating system.

**2. Installation:**

- Follow the installation wizard instructions.
- For Windows, run the executable file.
- For macOS, drag the application to the Applications folder.
- For Linux, extract the archive and run the `soapui.sh` script.

**3. Launch SoapUI:**

- Open the SoapUI application after installation.

**4. Create a New Project:**

- Go to `File > New SoapUI Project`.
- Enter the project name and the WSDL URL of the web service you want to test.

**5. Import WSDL:**

- SoapUI will automatically import the WSDL and create test suites and test cases based on the WSDL definitions.

### SoapUI Interface Overview

| | |
|---|---|
| **Navigator Panel:** | Displays the project structure including test suites, test cases, and requests. |
| **Editor Area:** | Used to view and edit requests, responses, and configurations. |
| **Properties Panel:** | Displays properties and settings for selected items in the Navigator panel. |
| **Log Panel:** | Shows logs and events during test execution. |

### Basic Configuration

**1. Setting Endpoints:**

- Modify the endpoint URL in the request editor to point to the correct service URL.

**2. Adding Authentication:**

- Configure authentication settings (e.g., Basic, WS-Security) in the request properties.

**3. Configuring Request Headers:**

- Add or modify HTTP headers in the request editor.

## Creating and Running Tests

### Creating Test Suites and Test Cases

**1. Create a Test Suite:**

- Right-click on the project in the Navigator panel and select `New TestSuite`.
- Enter a name for the test suite.

**2. Create a Test Case:**

- Right-click on the test suite and select `New TestCase`.
- Enter a name for the test case.

**3. Add Test Steps:**

- Right-click on the test case and select `Add Step`.
- Choose the type of test step (e.g., SOAP Request, REST Request, Groovy Script).

**4. Configure Test Steps:**

- Configure the properties and settings for each test step, such as the request body, endpoint URL, and assertions.

### Types of Assertions

| | |
|---|---|
| **SOAP Assertion:** | Verifies the SOAP envelope structure and content. |
| **XPath Assertion:** | Validates specific elements or attributes in the XML response using XPath expressions. |
| **JSONPath Assertion:** | Validates specific elements or attributes in the JSON response using JSONPath expressions. |
| **String Match Assertion:** | Checks if the response contains a specific string. |
| **Response Time Assertion:** | Verifies that the response time is within an acceptable range. |
| **Schema Compliance Assertion:** | Validates that the response is compliant with the defined schema. |

### Running Tests

**1. Run a Test Case:**

- Right-click on the test case in the Navigator panel and select `Run`.

**2. Run a Test Suite:**

- Right-click on the test suite and select `Run TestSuite`.

**3. View Results:**

- Check the `Log Panel` and the `Assertion Results` tab to see the test results.

**4. Analyze Failures:**

- Investigate failed assertions and errors in the logs to identify the cause of the failure.

**5. Rerun Tests:**

- Fix any issues and rerun the tests to ensure they pass.

## Advanced Testing Techniques

### Data-Driven Testing

**1. Create a Data Source:**

- Add a data source test step (e.g., `Excel`, `JDBC`, `File`) to the test case.

**2. Configure Data Source:**

- Configure the data source properties, such as the file path, database connection details, or query.

**3. Use Property Transfer:**

- Use the `Property Transfer` test step to transfer data from the data source to the request.

**4. Loop Through Data:**

- Add a `Loop` test step to iterate through the data source rows.

**5. Execute Requests:**

- Configure the request test step to use the transferred data from the data source.

### Mock Services

| | |
|---|---|
| **Create Mock Service:** | Right-click on the project and select `New MockService`. |
| **Add Mock Operations:** | Add mock operations to the mock service, corresponding to the operations in the WSDL. |
| **Configure Responses:** | Define mock responses for each operation, including the response body, headers, and status code. |
| **Start Mock Service:** | Start the mock service to simulate the behavior of the actual service. |
| **Test Against Mock:** | Configure your tests to point to the mock service URL instead of the real service URL. |

### Scripting with Groovy

**1. Add a Groovy Script Step:**

- Add a `Groovy Script` test step to the test case.

**2. Write Groovy Code:**

- Use Groovy to perform custom logic, such as manipulating request or response data, performing calculations, or interacting with external systems.

**3. Access SoapUI Objects:**

- Use the `context` object to access SoapUI objects, such as test case properties, request and response data, and project settings.

**Example: Get Request Content**

```
def request =
context.expand('${YourRequestName#Reques
t}')
log.info request
```

**Example: Set Property Value**

```
context.setProperty('YourPropertyName',
'YourValue')
```

## Debugging and Troubleshooting

### Debugging Techniques

**1. Use the Log Panel:**

- Monitor the Log Panel for error messages, warnings, and debug information.

**2. Add Log Statements:**

- Add `log.info()` statements in Groovy scripts to print values and trace execution flow.

**3. Inspect Request and Response:**

- Examine the request and response XML or JSON to identify any discrepancies or errors.

**4. Use Breakpoints:**

- Set breakpoints in Groovy scripts to pause execution and inspect variables.

**5. Validate Assertions:**

- Review failed assertions to understand the cause of the failure.

### Common Issues and Solutions

| | |
|---|---|
| **Issue: Invalid Endpoint URL** | Solution: Verify that the endpoint URL is correct and accessible. |
| **Issue: Authentication Failure** | Solution: Check the authentication settings and credentials. |
| **Issue: Schema Validation Error** | Solution: Ensure that the request and response XML or JSON are compliant with the schema. |
| **Issue: Timeout Errors** | Solution: Increase the timeout settings in the request properties. |
| **Issue: Data Type Mismatch** | Solution: Check the data types of the request and response elements. |
| **Issue: Missing Dependencies** | Solution: Ensure that all required libraries and dependencies are included in the project. |

### Error Handling

**1. Implement Error Handling in Groovy:**

- Use `try...catch` blocks to handle exceptions in Groovy scripts.

**2. Check for Null Values:**

- Check for null values before accessing object properties to prevent `NullPointerException`.

**3. Use Assertions for Validation:**

- Use assertions to validate the expected behavior and values.

**Example: Error Handling in Groovy**

```
try {
 def result = 10 / 0
} catch (Exception e) {
 log.error "Error: " + e.getMessage()
}
```