# REST Assured Logging and Debugging

In **Rest Assured**, logging and debugging are important for understanding how your requests are being sent and what responses you're getting — especially when you're troubleshooting failed tests or verifying behavior.

Here's a detailed breakdown of **logging and debugging in Rest Assured**, including examples:

---

## 🔍 1. Enabling Logging in Rest Assured

Rest Assured provides built-in methods to log the following:

- Request details (`log().all()`)
- Response details (`then().log().all()`)
- Specific parts (headers, body, status, etc.)

---

### ✅ Log Request Details

```
java

import static io.restassured.RestAssured.*;

public class LogRequestExample {
    public static void main(String[] args) {
        given()
            .log().all()  // Log request URI, headers, parameters, etc.
            .when()
            .get("https://jsonplaceholder.typicode.com/posts/1")
            .then()
            .statusCode(200);
    }
}
```

---

### ✅ Log Response Details

```
java

import static io.restassured.RestAssured.*;

public class LogResponseExample {
    public static void main(String[] args) {
        when()
            .get("https://jsonplaceholder.typicode.com/posts/1")
        .then()
            .log().all()  // Log full response (headers + body + status)
            .statusCode(200);
    }
}
```

---

## 🛠️ 2. Logging on Failure Only

Useful for cleaner test output — logs only when a test fails.

```
java


given()
    .log().ifValidationFails()  // Log request only if validation fails
.when()
    .get("https://jsonplaceholder.typicode.com/posts/1")
.then()
    .log().ifValidationFails()  // Log response only if validation fails
    .statusCode(404); // This will fail and log everything
```

## 🎯 3. Logging Specific Parts

- **Request body only**: `.log().body()`
- **Request headers only**: `.log().headers()`
- **Response body only**: `.log().body()`
- **Status line**: `.log().status()`

```
java


given()
    .log().headers()
    .log().body()
.when()
    .get("https://jsonplaceholder.typicode.com/posts/1")
.then()
    .log().status();
```

## 🛠️ 4. Debugging with Filters

Filters allow you to intercept and log/customize requests and responses.

### Example: Using `RequestLoggingFilter` and `ResponseLoggingFilter`

```
java


import io.restassured.filter.log.RequestLoggingFilter;
import io.restassured.filter.log.ResponseLoggingFilter;

public class FiltersExample {
    public static void main(String[] args) {
        given()
            .filter(new RequestLoggingFilter())   // Logs request
            .filter(new ResponseLoggingFilter())  // Logs response
        .when()
            .get("https://jsonplaceholder.typicode.com/posts/1")
        .then()
            .statusCode(200);
    }
}
```

## 5. Writing Logs to a File

You can write logs to a file using **`PrintStream`**:

```
java


import java.io.PrintStream;
import java.io.FileNotFoundException;

public class LogToFileExample {
    public static void main(String[] args) throws FileNotFoundException {
        PrintStream logFile = new PrintStream("restassured_log.txt");

        given()
            .filter(new RequestLoggingFilter(logFile))
            .filter(new ResponseLoggingFilter(logFile))
        .when()
            .get("https://jsonplaceholder.typicode.com/posts/1")
        .then()
            .statusCode(200);
    }
}
```

## ✅ Best Practices for Logging in Test Suites

| Scenario | Recommended Logging |
|---|---|
| Development/debugging | **`.log().all()`** |
| CI/CD pipelines | **`.log().ifValidationFails()`** |
| Production test reports | Log to files using **`PrintStream`** |
| Custom logging needs | Use **`Filters`** |

Let me know if you'd like a mini project or assignment based on Rest Assured logging and debugging.