Technology Assessment

- Low Level Design
- Integration Development Practices
- Hands-on Programming

Exposing Services on WSO2

Time: 60 minutes

Part 1: Exposing a back-end SOAP service as a WSO2 proxy

Create a SOAP proxy named 'CalculatorTestService' which would consume the back-end public SOAP service: http://www.dneonline.com/calculator.asmx?wsdl

- This service should accept the standard BASIC authentication ('base64' encoding) with the following username / password:

  username: wso2_test_usr

  password: P@ssw@rd123
- On receiving the user request, log the following statement in 'CalculatorTestService' insequence: "*CalculatorTestService - insequence called*"
- After that, extract the username and password and verify if username = wso2_test_usr and password = P@ssw@rd123.
- If the above test is not successful, send an error HTTP status code 401 (unauthorized) back to the calling user.
- If the above test is successful (i.e. username and password are valid), then send the request to the back-end service.
- In outsequence, log the statement: "CalculatorTestService - outsequence called". Also, log the full payload. Send this payload to the calling user.

Required Activities

- Submit the code required to implement the CalculatorTestService
- Explain any assumptions or design decisions followed

Part 2: Exposing a back-end SOAP service as a WSO2 REST API

Create a REST service named 'CalculatorTestAPI' which would consume the back-end public SOAP service: http://www.dneonline.com/calculator.asmx?wsdl

- There are 4 operations in the back-end service ('add', 'divide', 'multiply', 'subtract'). Corresponding to each of these operations, a corresponding REST operation has to be exposed. Name the rest resources the same as back-end SOAP operations (i.e. 'add', 'divide', 'multiply', 'subtract')
- The REST service uses JSON requests / responses (i.e. content type should be 'application/json')
- The back-end XML payload request and responses should be converted to JSON request/responses for the REST service.
- No need to use the basic security in this service.

In the insequence:

When the request is received by the REST service, first log the following:

a) full JSON received. Note that received JSON is to be logged and NOT the corresponding XML.
b) the name of the API in the following format (you can use concat function): 'Invoking insequence of <name_of_api>'
c) the complete URL invoked by the client.

This service would only accept POST requests. If the client sends a GET request, send a JSON response as below, and status code as '405'.

```
{
        "status_code": "GSB_001",
        "status": "HTTP_METHOD_NOT_ALLOWED",
        "description": "The requested resource does not support http method other than 'POST'."
}
```

In the outsequence:

Filter for the response status code returned from the back-end service (you can use 'filter' mediator). If the status code is 200, consider this as a success, and return to the user the corresponding JSON of the back-end XML response.

If the status code is other than 200, consider it as a failure and return the following response: HTTP Status Code: 500
Response payload:

```
{
        "status_code": "GSB_002",
        "status": "ERROR",
        "description": "An error has occured. Please contact GSB Support Team for further details"
}
```

Fault sequence:

Create a separate fault sequence, which would capture the following details about the error:

- URL

- Method
- Error Code
- Error Message

Test all scenarios of the service in POST MAN or ARC. Take snapshots to be shared later.

Keep the wso2carbon.log of this testing on desktop, so we can check the logs and any errors.

Required Activities

- Submit the code required to implement the CalculatorTestAPI
- Explain any assumptions or design decisions followed