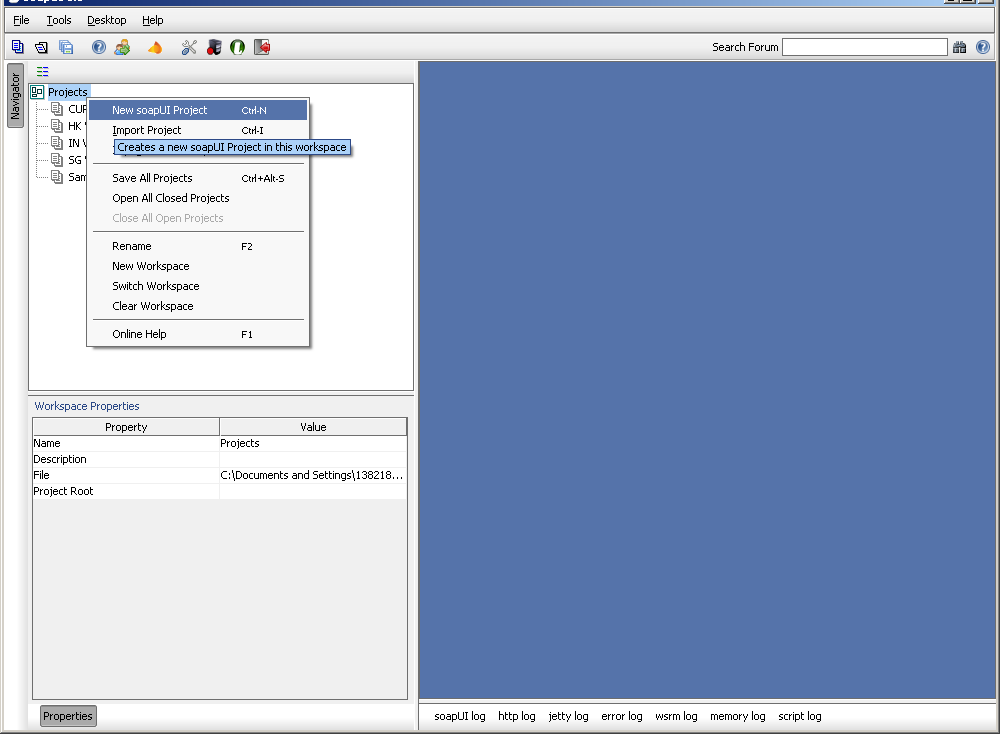
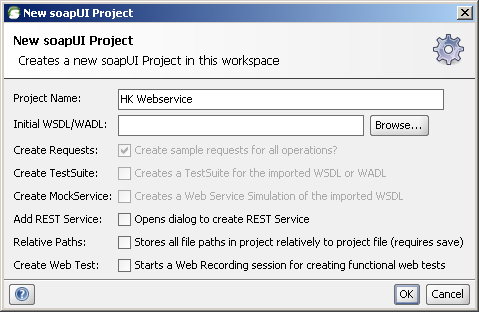
Sample Steps:

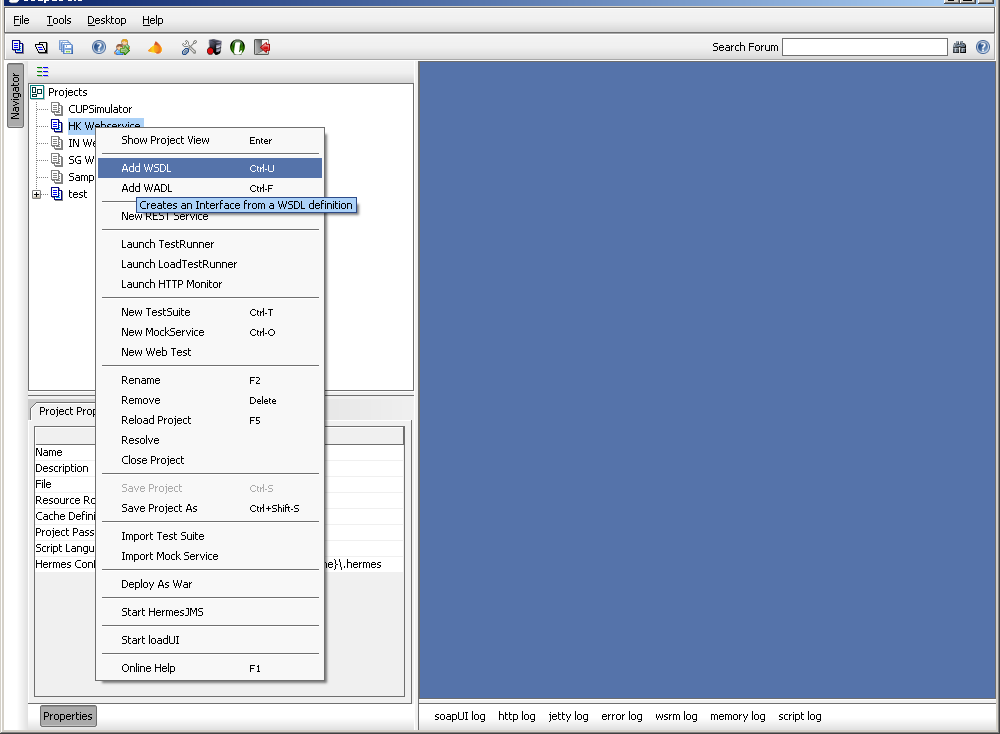
# Create new project.



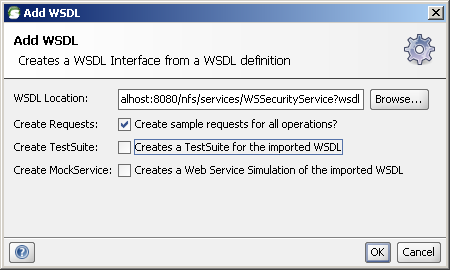


# Add WSDL

**Before add WSDL, please change the section of wsdd configuration file.  
 <globalConfiguration>  
 <parameter name="axis.disableServiceList" value="false"/>  
 </globalConfiguration>**



<http://localhost:8080/nfs/services/WSSecurityService?wsdl>

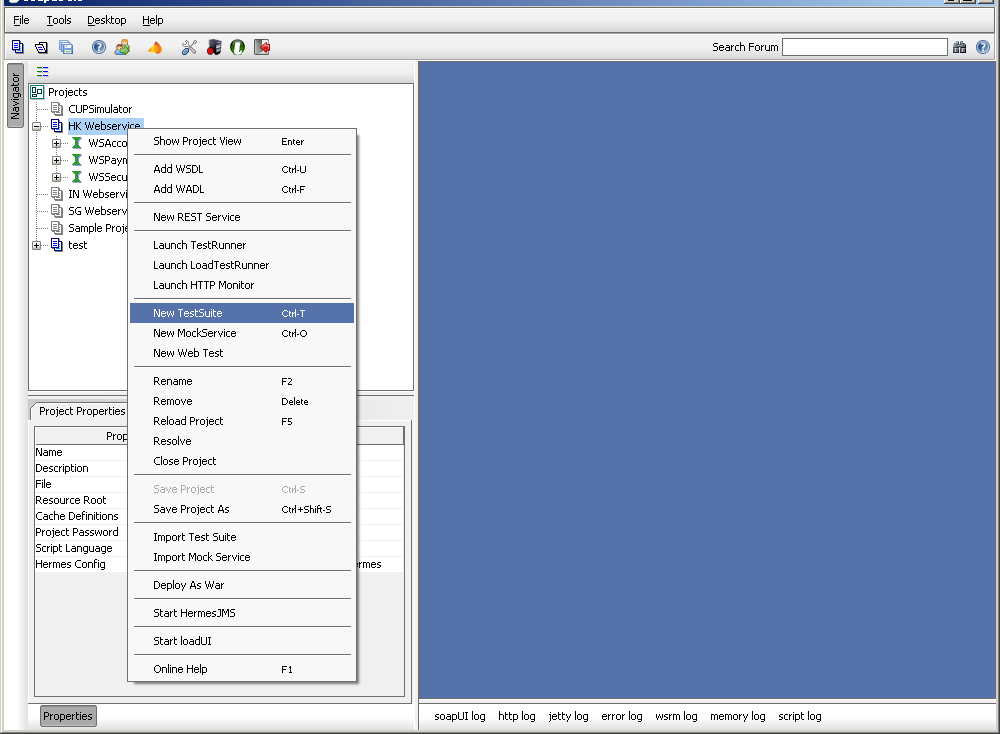


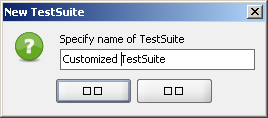
# Do the same for below 2 URL

<http://localhost:8080/nfs/services/WSAccountService?wsdl>

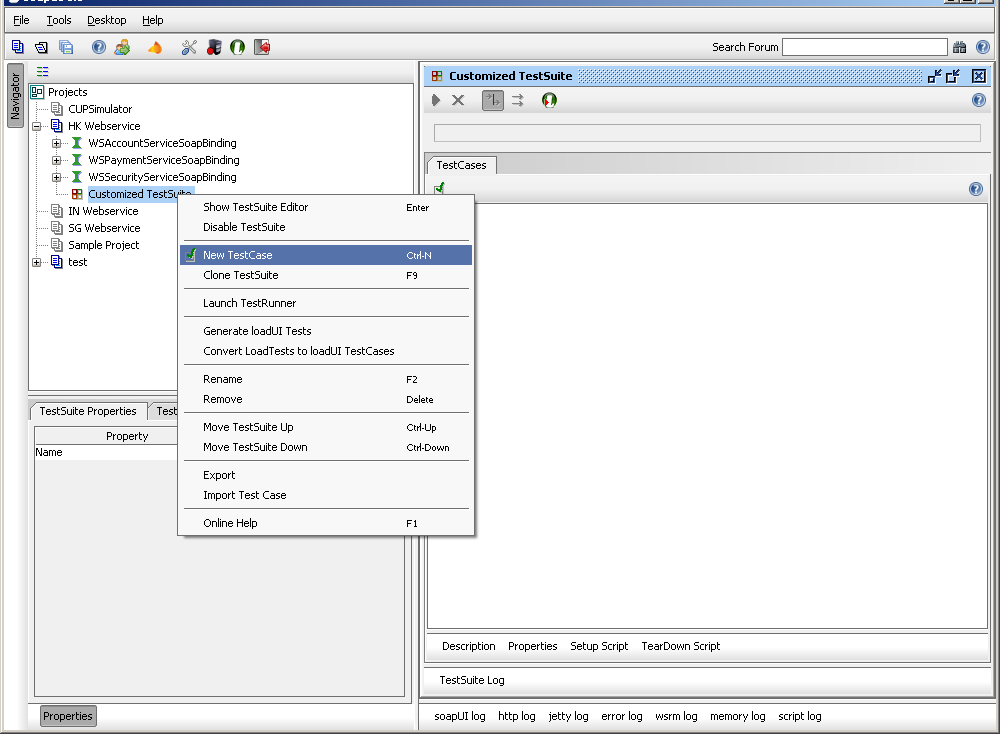
<http://localhost:8080/nfs/services/WSPaymentService?wsdl>

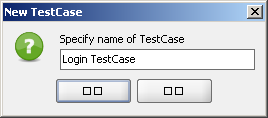
# Create TestSuite



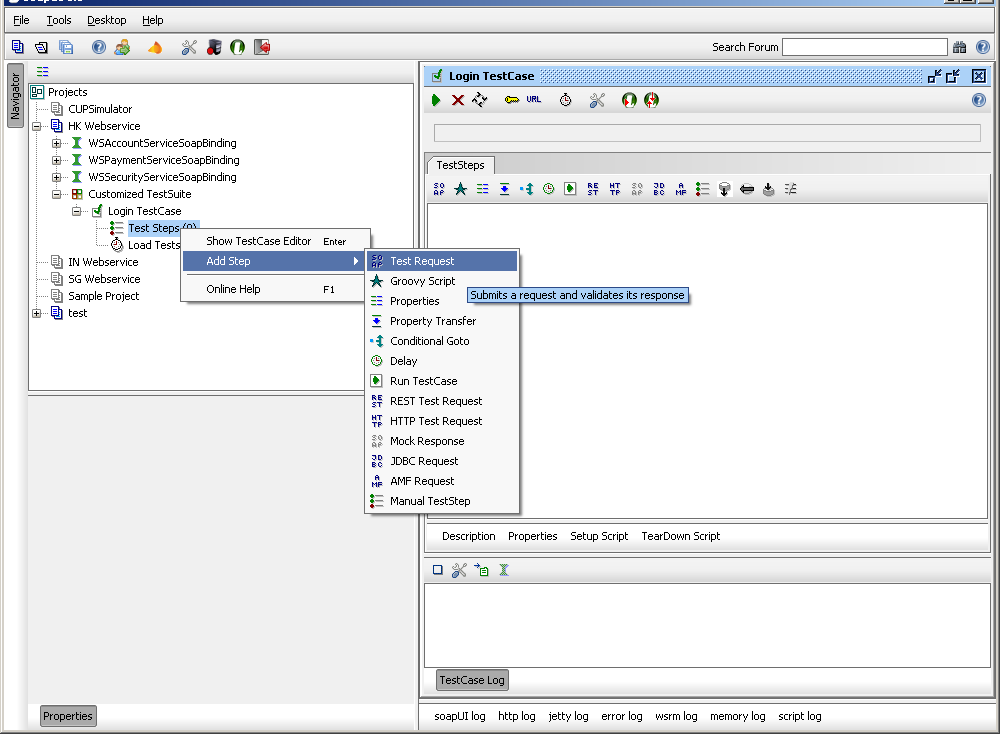


# Add test case for login

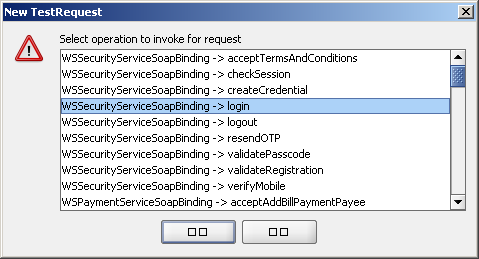




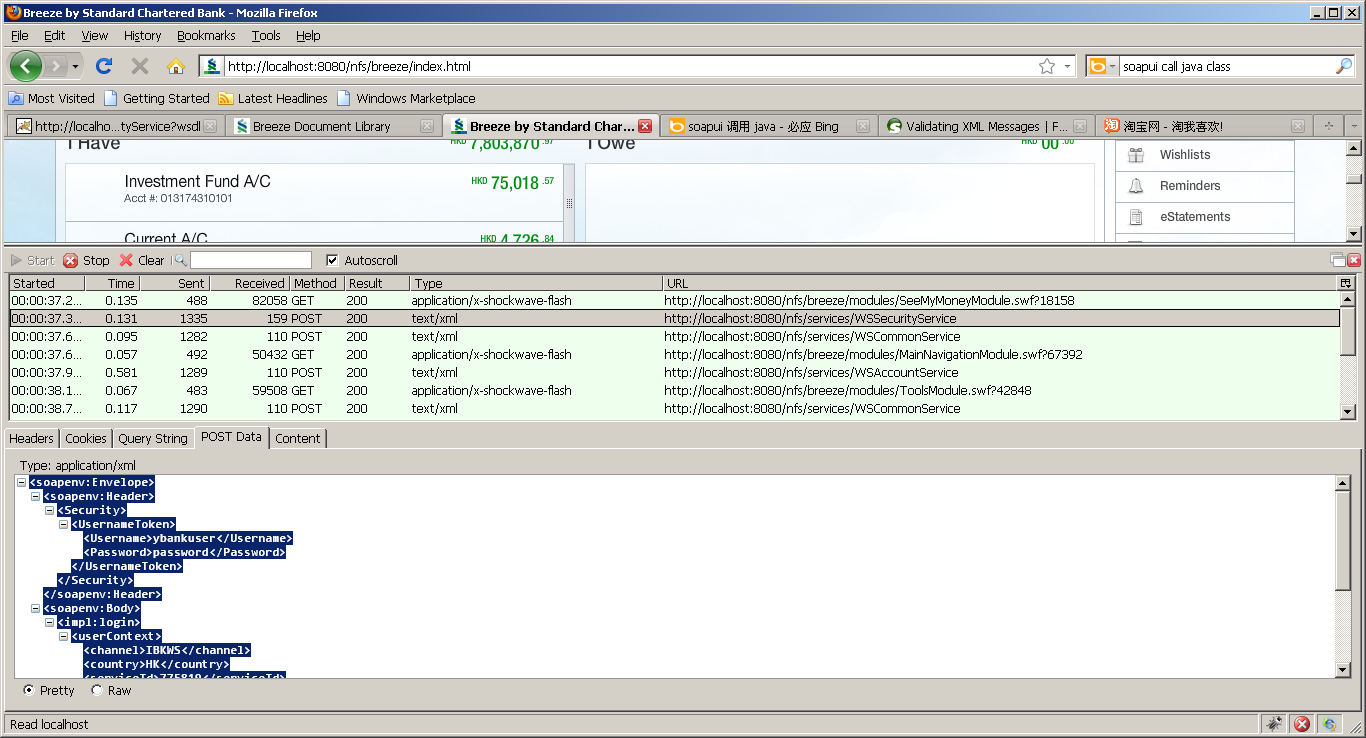
# Add step to invoke webservice



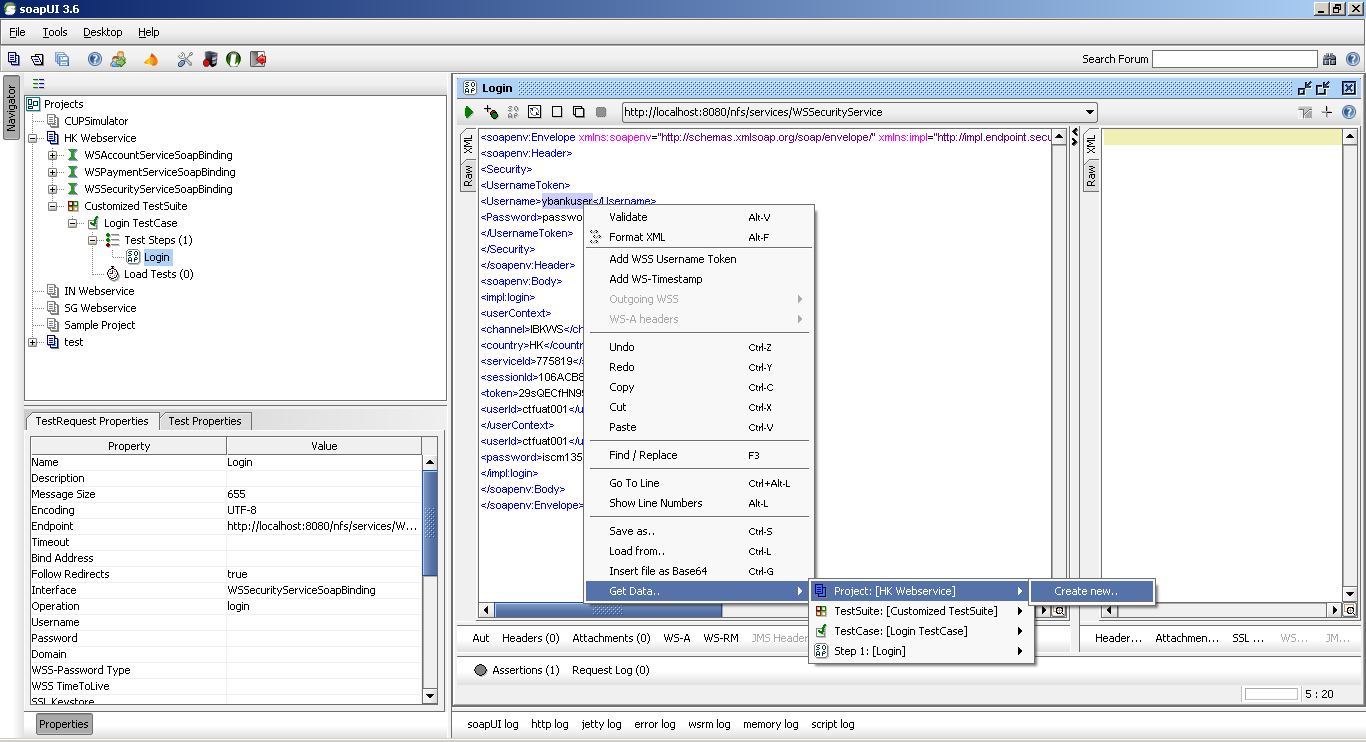


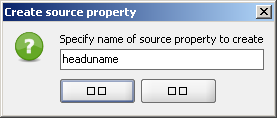


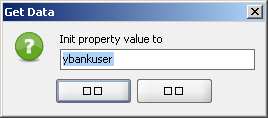
# Copy the request content from HTTP Fox



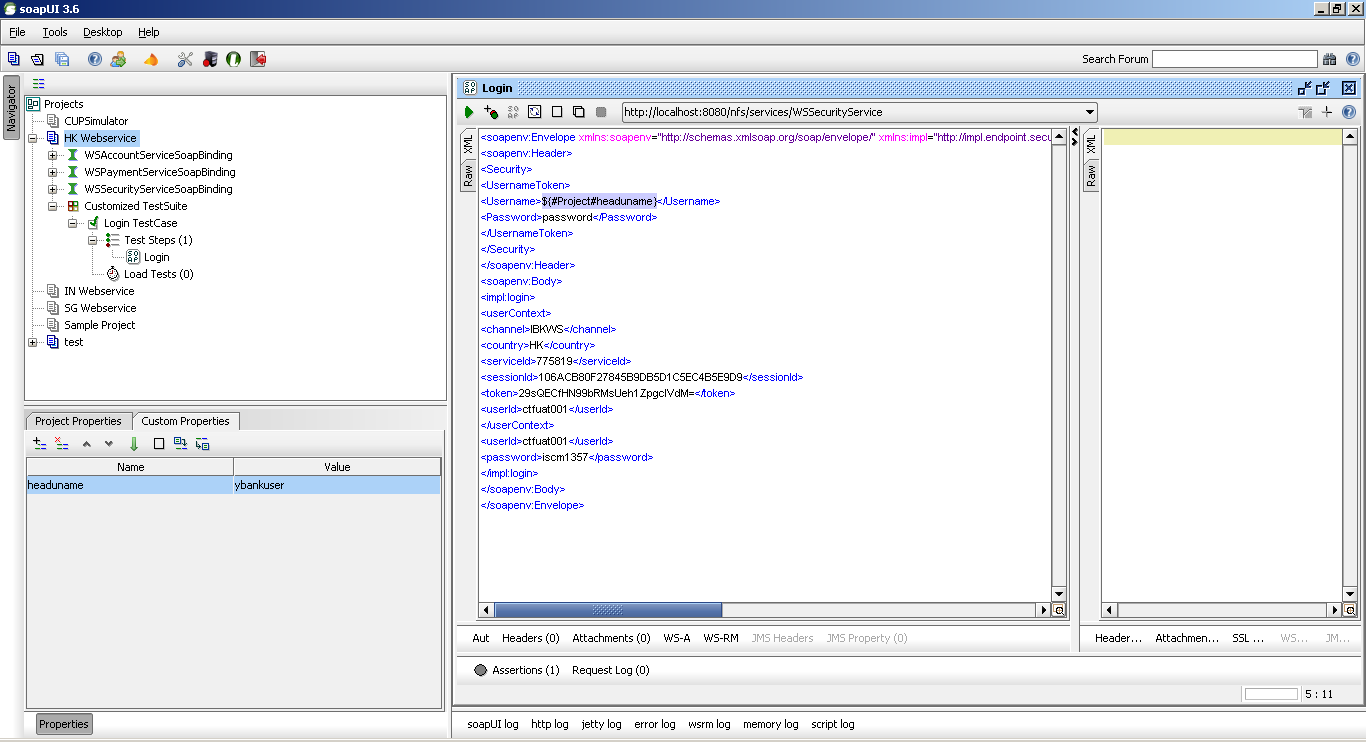
# Replace the field values with parameter





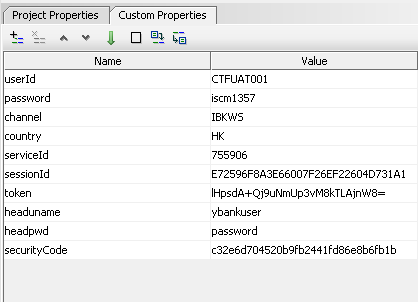


# The parameter can be found in the “customer properties”

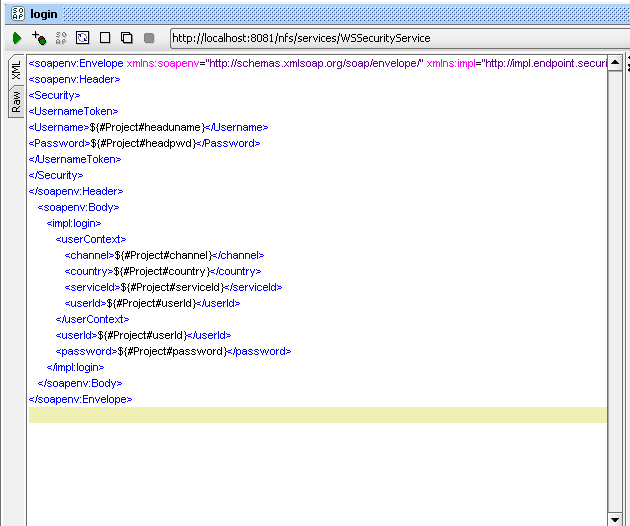


# Repeat the steps to replace all the field values

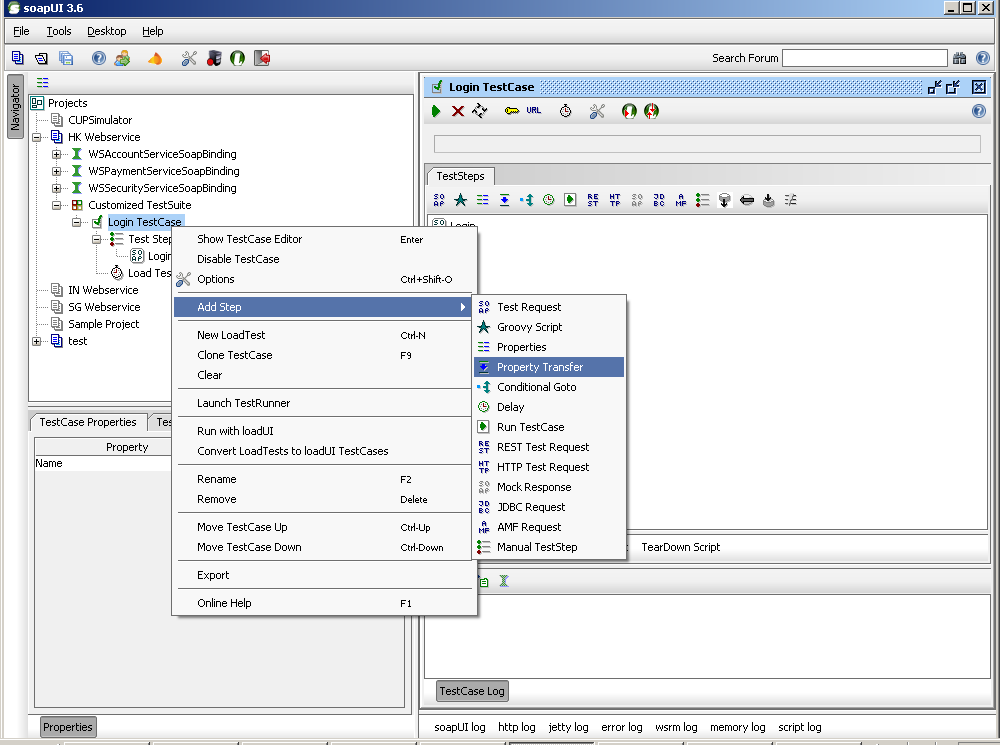
The “customer properties” will look like below

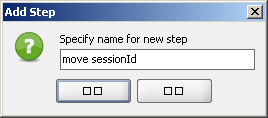


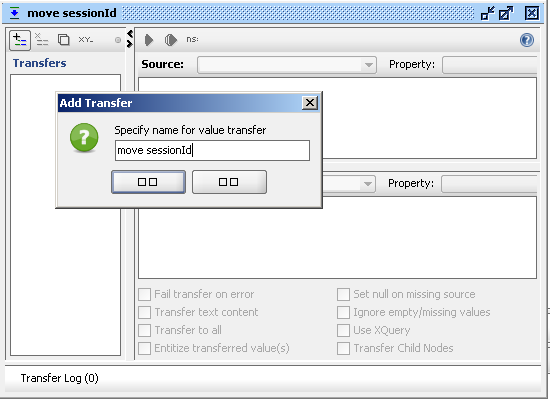
The request will look like below



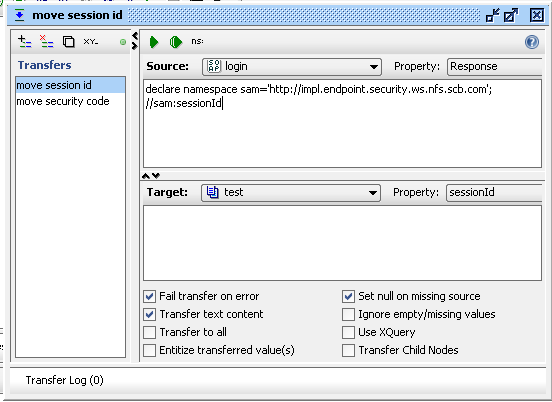
# Add test step to copy response data to parameters, so that can be used in other steps



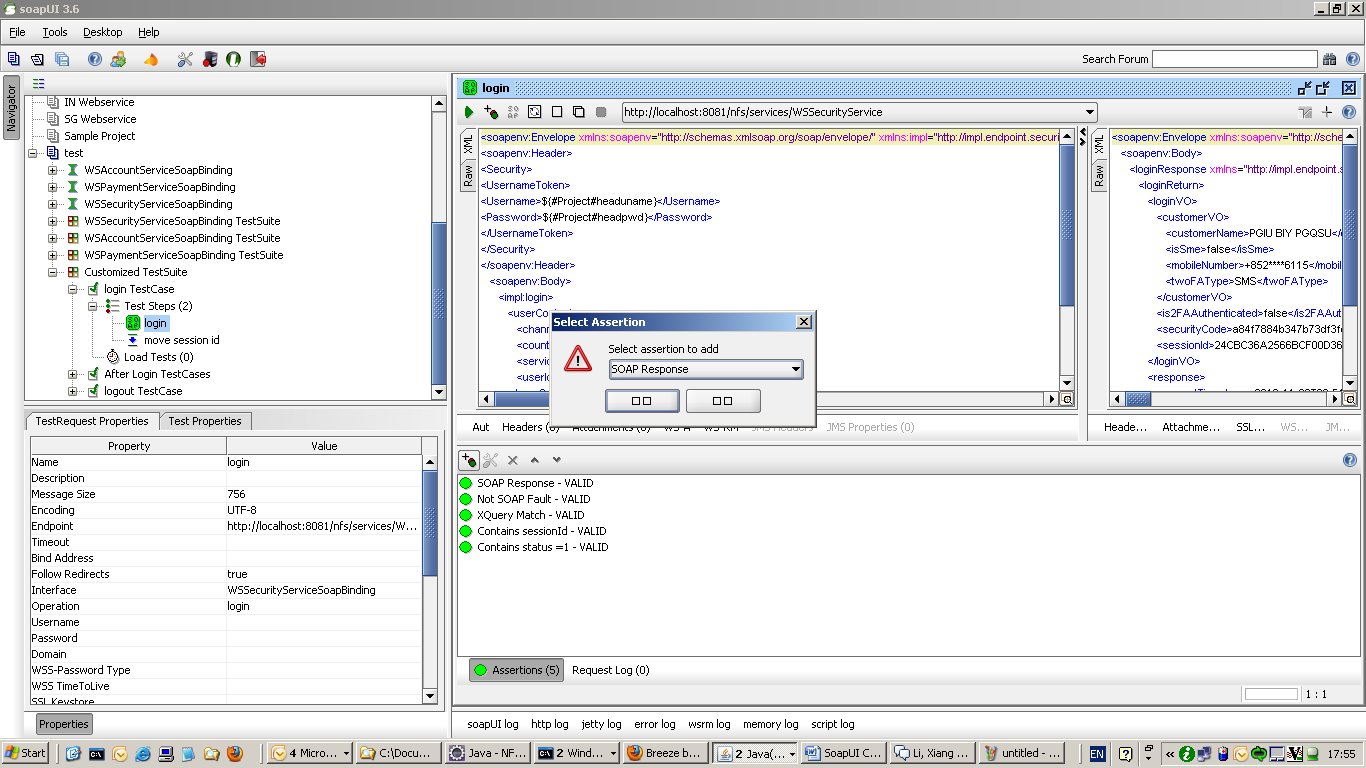




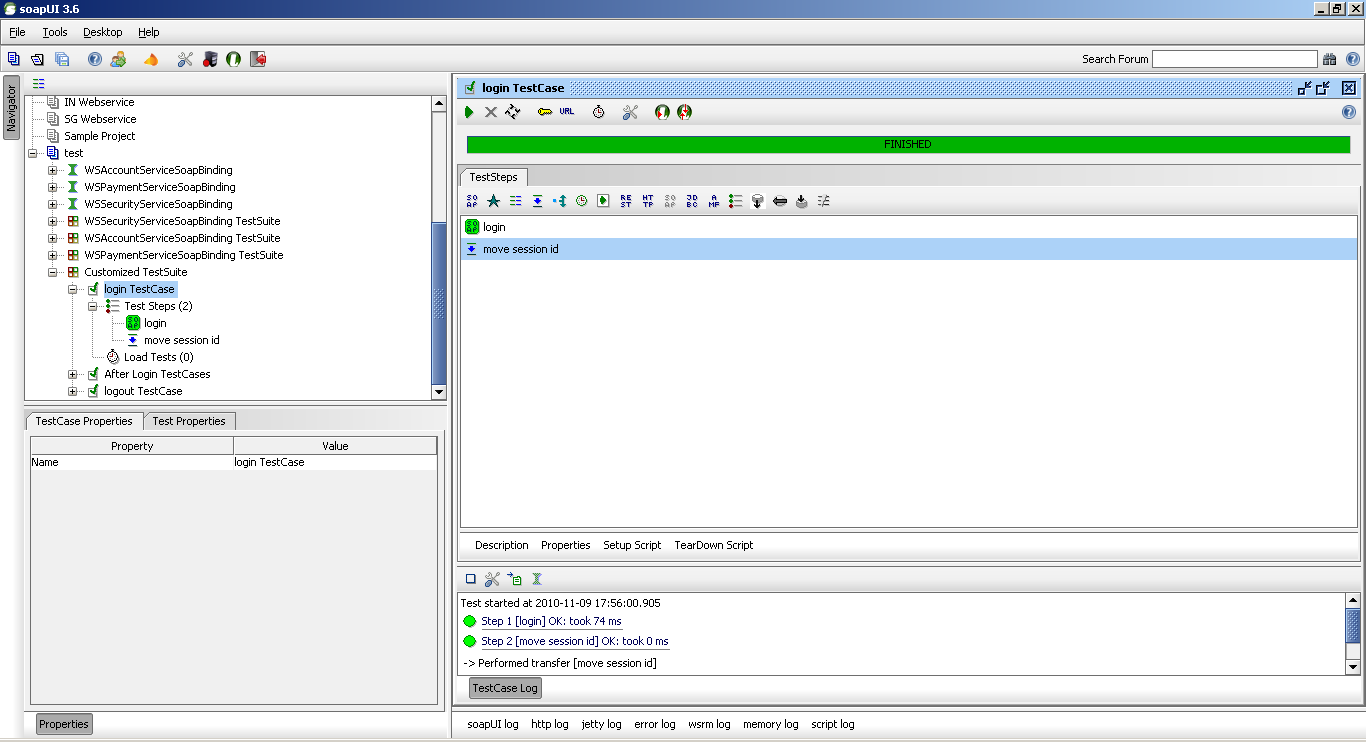
Copy the sessionId value from login response to the property “sessionId”



# Add assertion for the service call



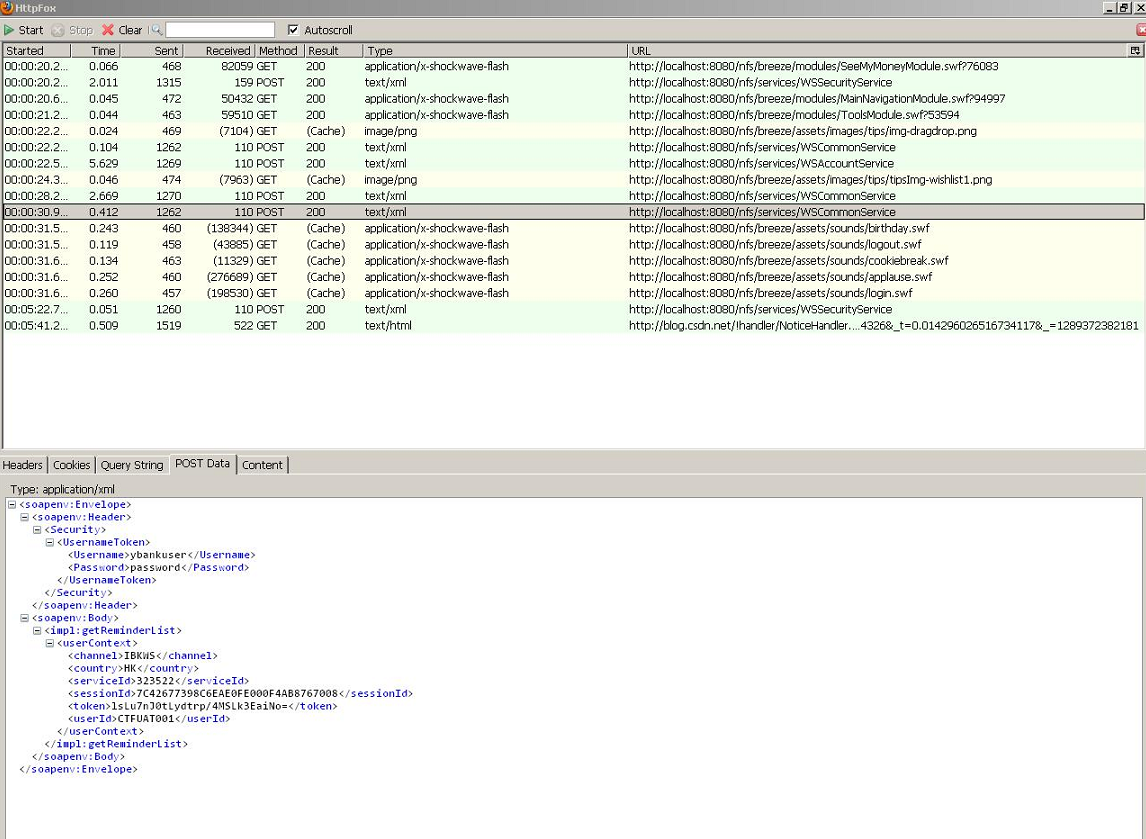
# Run the test case



Exercise:

# Add test cases for “get Product Summary / third party transfer/ logout”

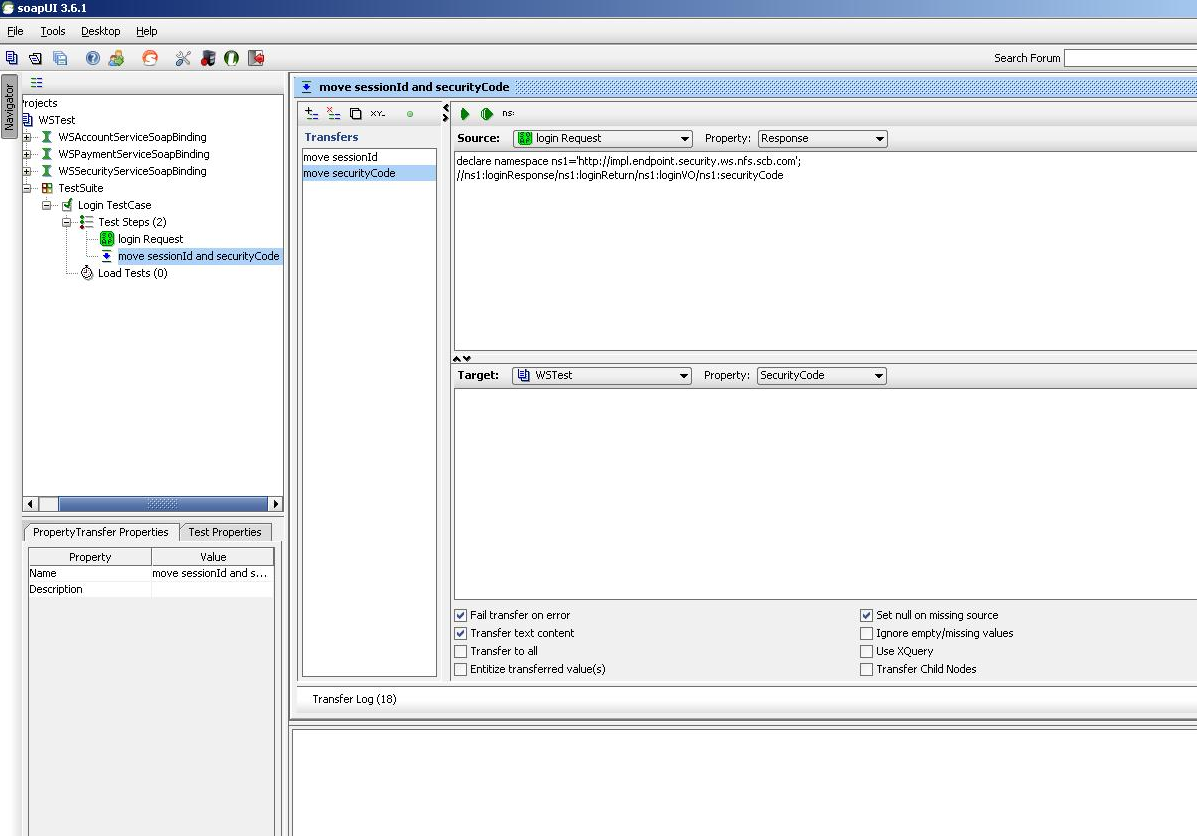
We can get the web service soap construct via HttpFox and use it when mock the soap message request in SoapUI.



**Login**

Above has the detail steps for how to create it, and here only descript how to transfer pre-request’s response value to the next action’s request, the simple process is: get the response’s value 🡪 set into customize property 🡪 get the value from customize property , following descript how to get sessionId and securityToken from login response:

Declare the naming space and then query the property value (Xpath pattern)



After login, we need calculate the token value according to sessionId userId serviced and the securityCode which we can get it from login response, and then set it to the customize property, we will use it in the UserContext for all the other operations’ authentication. Following is the code of generating the token:

import java.security.MessageDigest;

import org.apache.commons.codec.binary.Base64;

import org.apache.commons.lang.StringUtils;

public class GenerateSecurityHash {

public static final String ENC\_UTF8 = "UTF-8";

public static final String ENC\_SHA1 = "SHA-1";

public static void main (String[] args) throws Exception {

String code = hashContextToken(

"ADF9FEDA27D212EEE3FF67BF457B3C12", // sessionId

"CTFUAT001", // userId

"323522", // serviceId

"5f8370f318032e54c75bbf3127c0350b"); // securityCode

System.out.println(code);

}

private static String hashContextToken(String sessionId, String userId, String serviceId, String securityCode) throws Exception {

StringBuffer sb = new StringBuffer();

sb.append("HK"); // country code

sb.append("|").append(sessionId);

sb.append("|").append(userId);

sb.append("|").append(serviceId);

sb.append("|").append(securityCode);

return sha1Base64(sb.toString());

}

private static String sha1Base64(String clearTxt) throws Exception {

if (StringUtils.isNotEmpty(clearTxt)) {

MessageDigest msgDigest = MessageDigest.getInstance(ENC\_SHA1);

byte[] hashedBytes = msgDigest.digest(clearTxt.getBytes(ENC\_UTF8));

return new String(Base64.encodeBase64(hashedBytes), ENC\_UTF8);

} else {

return clearTxt;

}

}

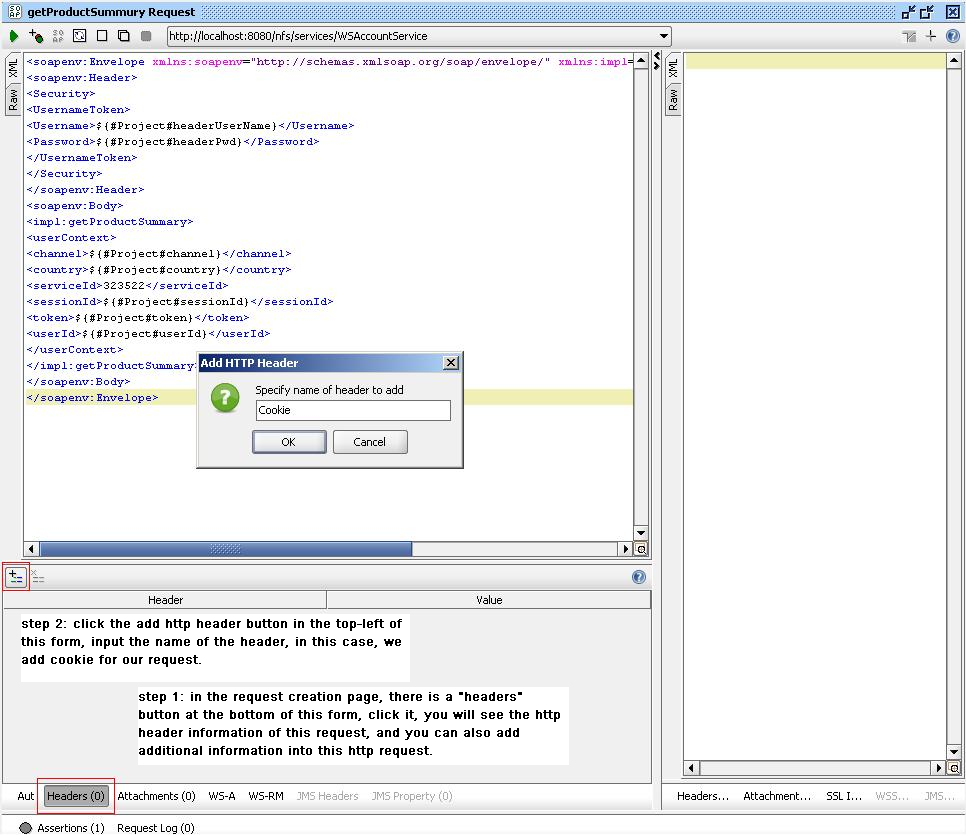
}

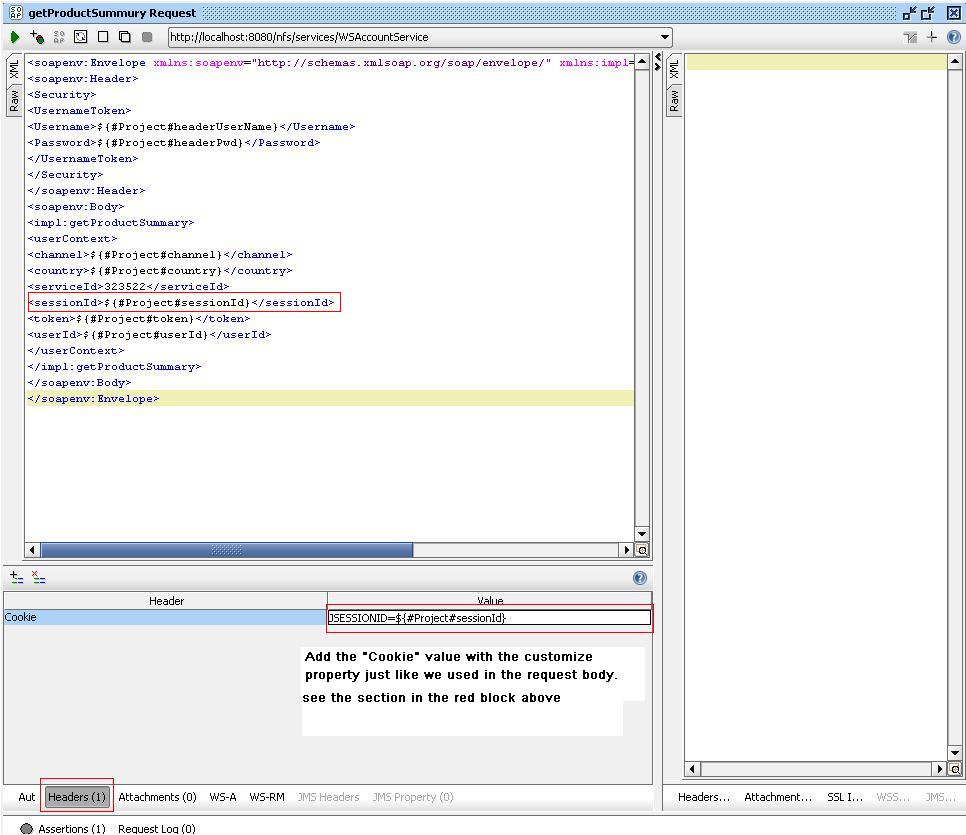
**GetProductSummary**:

The steps of creating the ws request for getProductSummary is the same as the login process above. Then replace the request value with the customize properties, additional, we need **set the cookie in the http header(it is necessary for all actions except login action)**, and then run this case, please see the following picture.

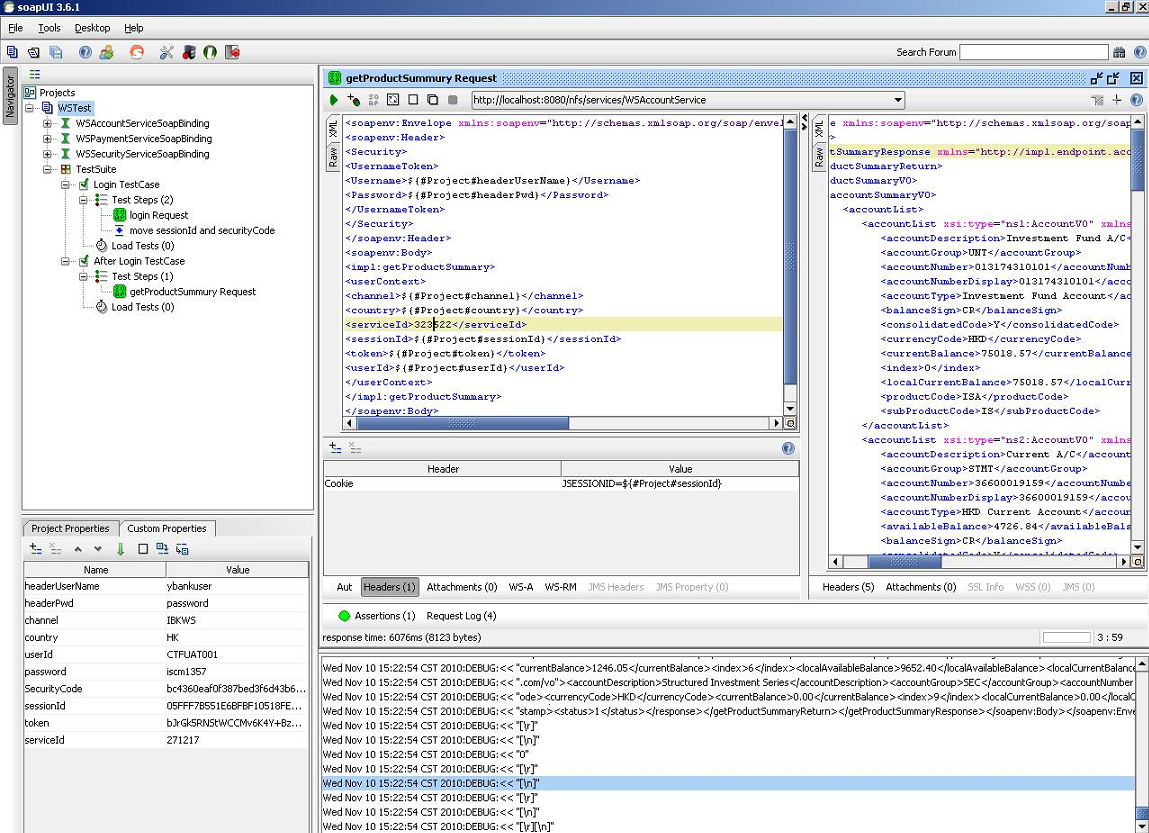
Steps of setting cookie:

Please see the inline comments in the following two pictures for the detail steps, and the key information is figured out with **red** block, there is also some detail comments with them.





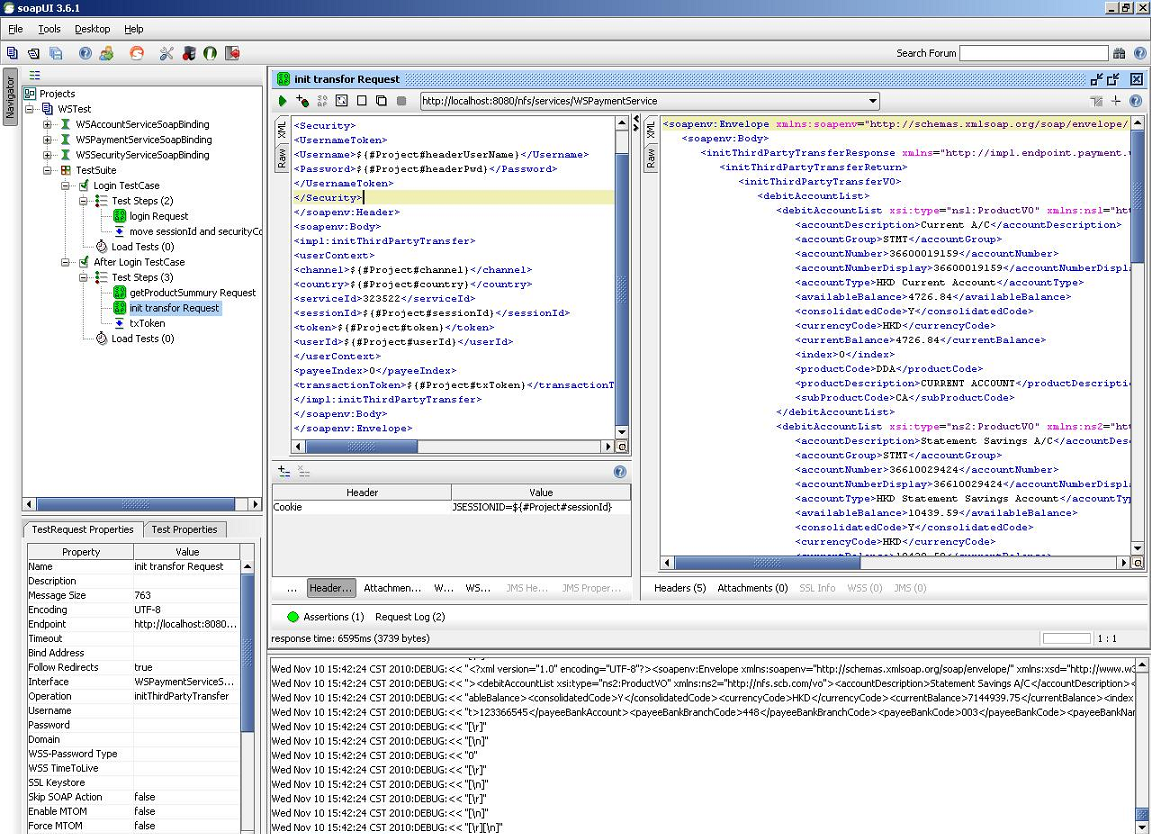
Following is the result of this web service call:



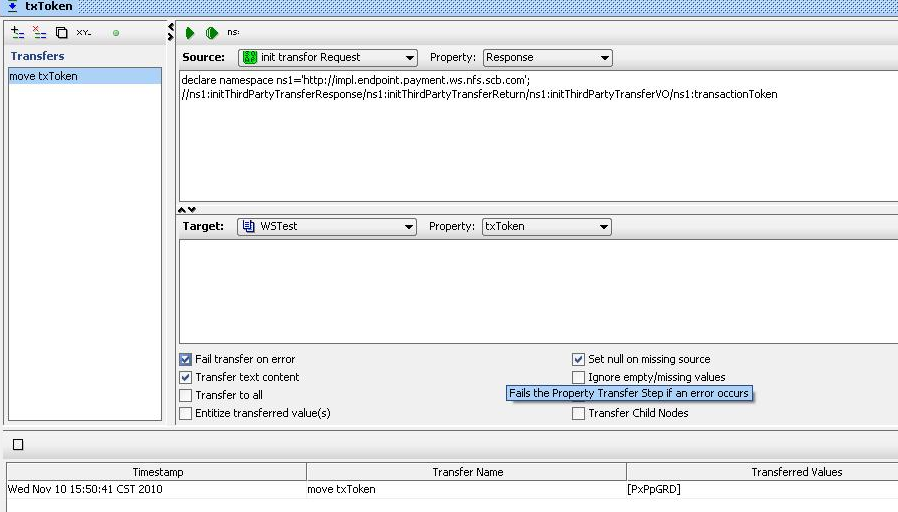
**Third party transfer**

initThirdPartyTransfer

Construct the soap request by httpFox(what’s this), set the cookie as the above steps, and then run the test case.

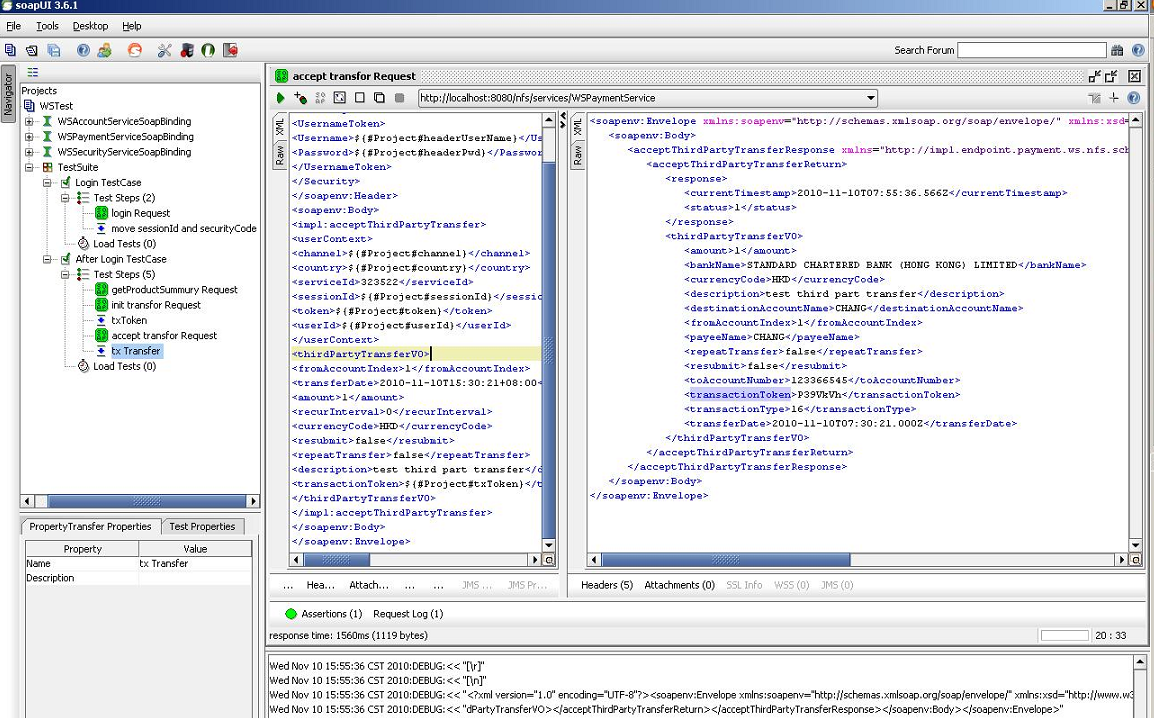


Move the transaction token value to the customize property from init response:

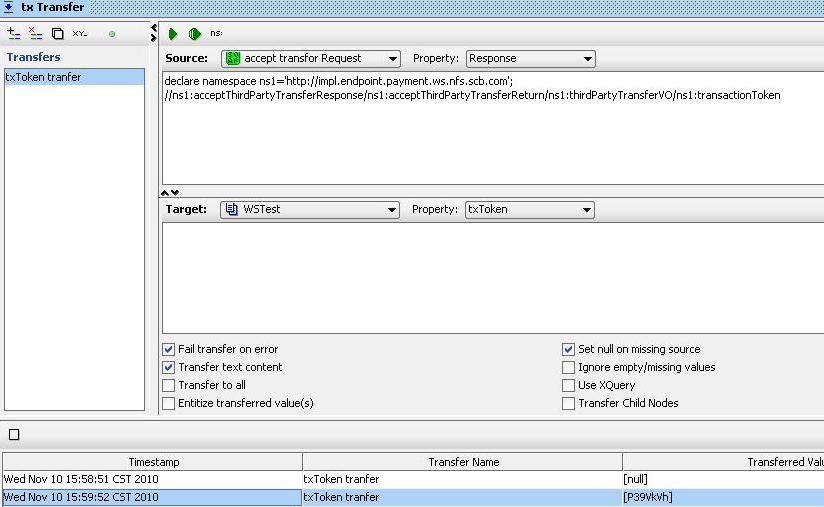


acceptThirdPartyTransfer

Construct the soap request by httpFox, set the cookie as the above steps, and then run the test case.

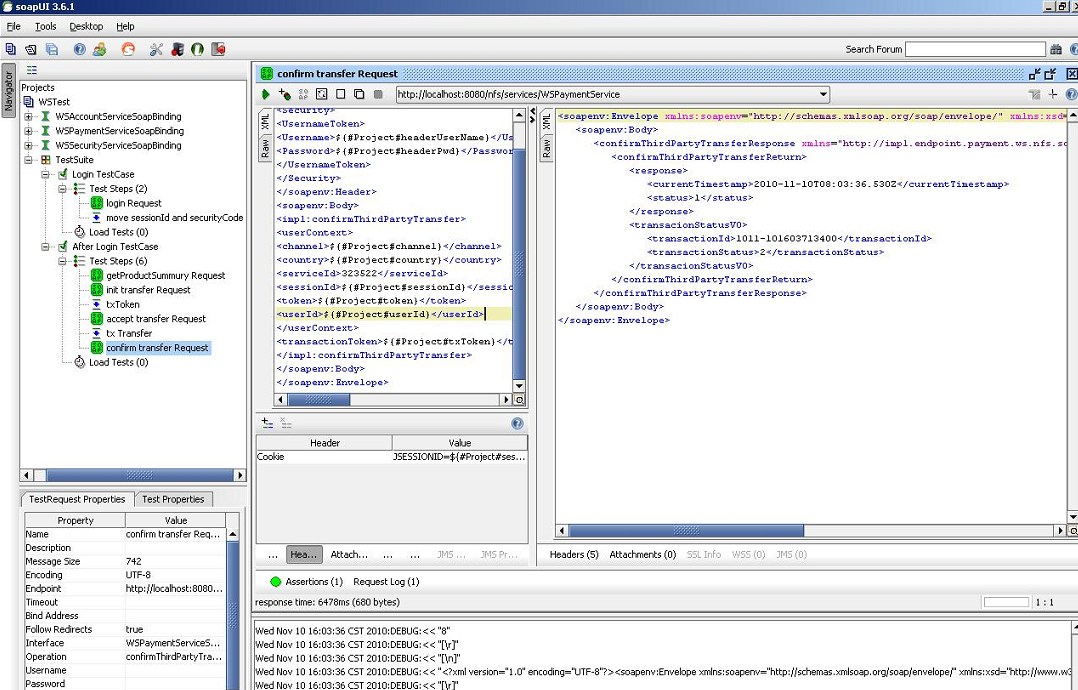


Move the transaction token value to the customize property from accept response:



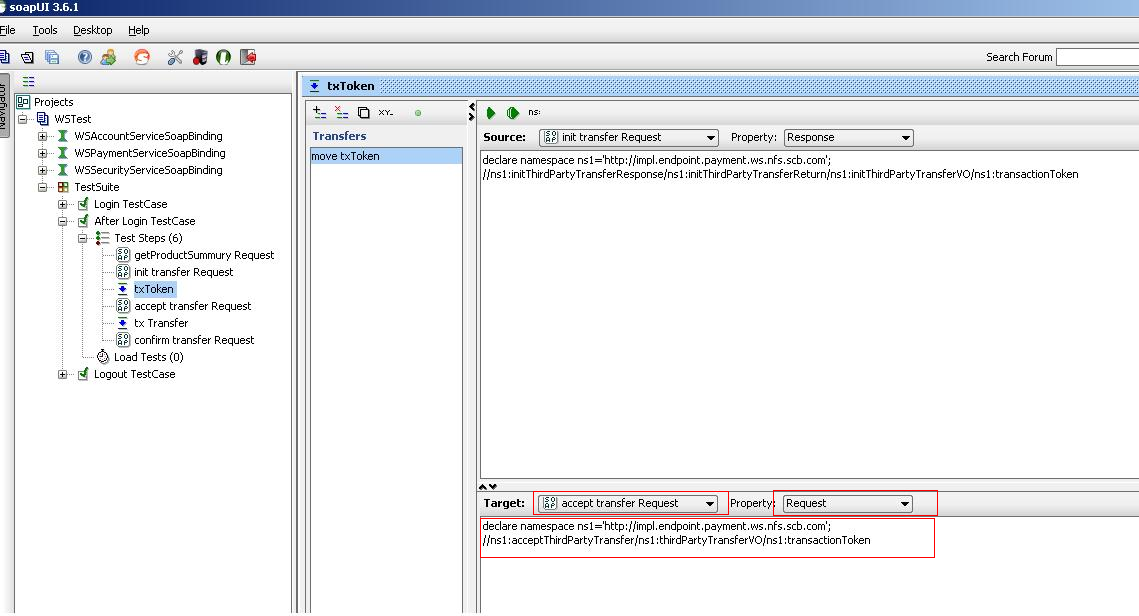
confirmThirdPartyTransfer

Construct the soap request by httpFox, set the cookie as the above steps, and then run the test case.



Another way of transferring parameter between requests, no need set customize property, following is an example:

The scenario is transfer the transaction token of init response to the accept request. The steps are same as the other transfer property, just change the **target property**, select the target request and set the XPath where need the response value, following is an example:



**Logout**(it is the same as the above process)

