CLAIM PROCESSING SERVICE SIMULATION DEMO

This demo shows a composite application consisting of 2 SOAP services. It allows demonstrating how to learn and simulate the data and performance of one of SOAP services.

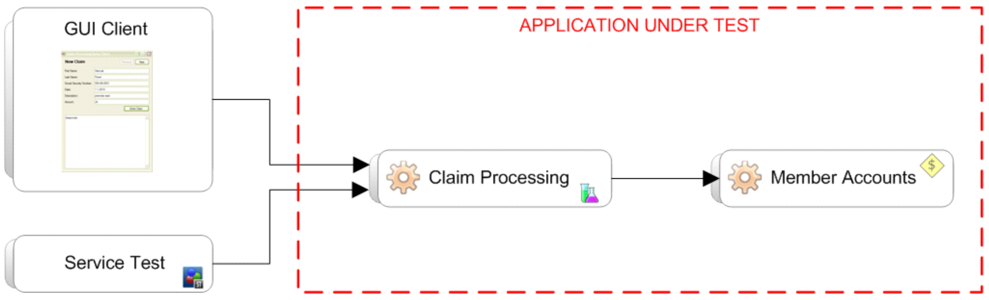

Fig. 1: The Composite Application

The *ClaimProcessing* service can be invoked either by the GUI client or by a Service Test script during the test – see Figure 1. We are going to learn and simulate the *MemberAccounts* service since it is of limited accessibility.

THE USE CASE

The GUI client allows filling in a health insurance claim or selecting one of pre-filled claims.

The claim request contains partial identification of the insured person – some of first name, last name and/or social security number. The client sends an *enterClaim()* request to the *ClaimProcessing* SOAP service.

In order to create a new claim, the *ClaimProcessing* service has to complete the insured person information. It does so by invoking the *memberSearch()* on *MemberAccounts* SOAP service.

The *MemberAccounts* service looks up the person from pre-defined list of members and returns 0 or more member records containing member IDs.

The *ClaimProcessing* service creates a new claim if and only if it was possible to identify the person; it means when there was exactly one record returned by the *MemberAccounts* service, returning fault otherwise. From the predefined claim requests, this does not work for Karel Got (not a member) and a "Poirot" without first name (since there are both Mr. Hercule Poirot and his sister registered).

The *ClaimProcessing* calls the *MemberAccounts* service *getMemberPlan()* to get the member insurance plan containing the claim approval limit – the maximum amount of money to claim without going through further approval process. It also calls the *getMemberDetail()* to get the person contact information and other details to complete the claim.

The *ClaimProcessing* service marks the claim approved if the claimed amount was not higher than the limit in the member plan, then returns the claim identifier.

The client then requests the claim detail from *ClaimProcessing* service and displays the result.

USING REAL SERVICES

Use the `run-real.bat` script to launch the GUI client and both services on their default endpoints:

ClaimProcessing
    http://localhost:8102/ServiceSimulation/Demo/ClaimProcessingService
MemberAccounts
    http://COMPUTERNAME:8101/ServiceSimulation/Demo/MemberAccountsService

LEARNING AND SIMULATING THE MEMBER ACCOUNTS SERVICE

Use the `run-virtualized.bat` script to run the GUI client and both services configuring the *ClaimProcessing* service to use virtual *MemberAccounts* service instead of real one:

ClaimProcessing
    http://localhost:8102/ServiceSimulation/Demo/ClaimProcessingService
MemberAccounts
    http://COMPUTERNAME:7200/MemberAccounts

Open the prepared simulation project and switch the service into learning mode.

If you are running your virtual service in authenticated standalone server, you need to change the server to https://localhost:6085

The communication between ClaimProcessing and MemberAccounts services is being captured – see Figure 2.

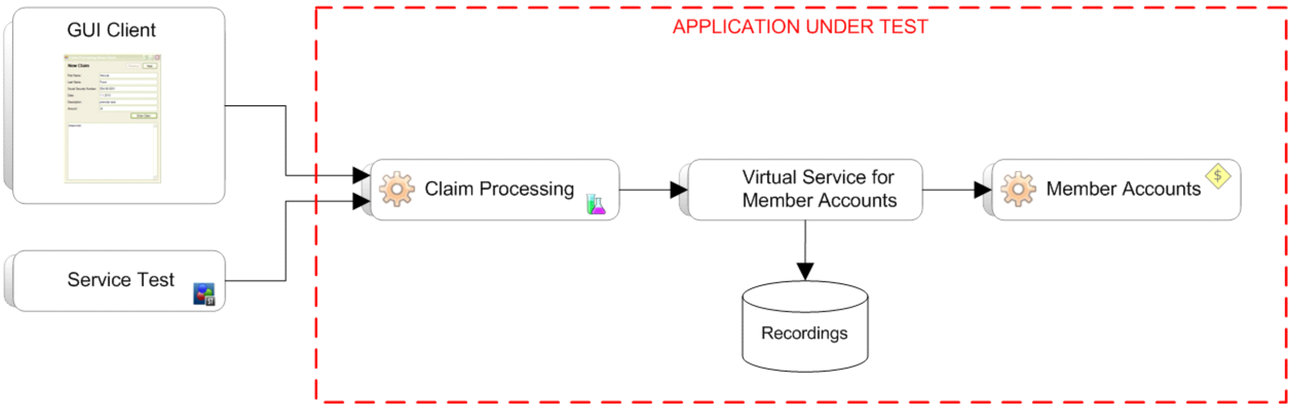

Fig. 2: Recording The Real Member Accounts Service

When you finish the learning, the data and performance simulation models are used to simulate the behavior of *MemberAccounts* service. The real MemberAccounts service can be shut off - See Figure 3. Depending on the selected performance model the simulation response time can be the same or shorter than in case of the real service.

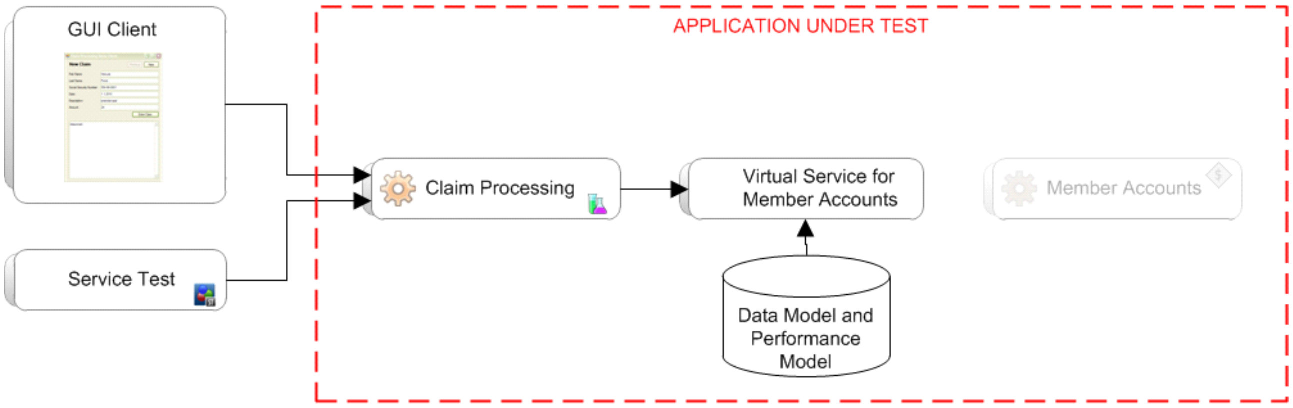

Fig. 3: Simulating The Member Accounts Service

USING THE HTTP AUTHENTICATION

You can use the `run-virtualized-basic.bat` or `run-virtualized-ntlm.bat` script to launch the GUI client and services on virtual endpoints using the HTTP-Basic or NTLM authentication.

You need to have the “hpguest” user account with “hpguest” password created on your computer, or you can optionally change username or password in the `.config` file of Claim Processing Service.

Service Virtualization project is pre-configured with “hpguest” user and password in Virtual Service Credential Store. If you change the credentials in the `.config` file, the Credential Store must be updated as well.

When using HTTP-Basic, you can either use the principle described above for NTLM authentication or take advantage of automatic credential recognition. In that case, credentials are recognized as part of recorded message and after stopping the virtual service, you can follow the link ‘Fix User Store’, which adds the credentials automatically to the credential store.