Chapter 7

Practical Usage

7.1 Introduction

In previous chapter it was discussed about the implementation process of the system in detail. The SOAP optimization tool implemented in this project will be distributed as a JAVA jar library. In this chapter it will discuss about how this tool can be plugged into an existing web-service application. This chapter will use the method specified in JAX-WS (Java API for XML Web Services) as an example. But any method can be used to plug the tool in to web-service.

7.2 How to write the plug-in

Most of the agent applications require the application to be start with the boot class of the agent framework. This tool has built such that it will allow the system to start in usual way and the agent framework will be started within the application when the tool is initialized. It does not require to change the main class of the application to run this tool and the web-service and the client application can be run as usual and this tool can be used as a plug-in.

7.2.1 Server side plug-in

For intercept the SOAP message that passing through web service and the web service client JAX-WS (Java API for XML Web Services) SOAP message handler interface is used [30]. The processing for this tool has to be done in SOAP message level. Therefore it is required to create the message handler in SOAPMessageContext level by implementing the SOAPHandler interface.

Then on service side handler needs to be added to handler chain by using the handler_chani.xml. On the SOAP message handler first it is required to initialize the MADKit kernel. It can be done calling the initEnvironment method of SOAPOptKernel class provided in the distribution binary. SOAPOptKernel is a singleton class and therefore it can be used the SOAPOptKernel.getInstance() to access the instance of it.

The message optimization process has to be done inside the handleMessage method of the SOAPHandler class. A static method is provided in SOAP optimizer tool to handle the optimization of sending message. The method is in SOAPUtils class and the method is handleOutboundMessage. It has to pass the SOAPMessageContext as the parameter to this method and the alteration to the SOAP message will be done inside that method.

7.2.2 Client side plug-in

In client side it is not possible to add an interceptor by using the handler chains xml. Instead of that in client side, it is required to add the message handler specifically to the handler chain list of the handler resolver of the client proxy. In the binary distribution of the SOAP Message Optimizer system there is a readymade client side message handler built in. The name of the class is ClientMessageHandler and it is possible to add this class to the handler chain of the stub generated on JAX-WS web service client. This has to be done inside the JAVA coding of the client module. The kernel initialization and message handling will be done inside this ClientMessageHandler class.

7.3 Summary

This chapter has discussed how to plug this SOAP optimizer tool into existing web-services. It has also discussed in details on creating plug-ins in both server side and client side for JAX-WS platform. In next chapter it will be discussed about the evaluation of the implemented SOAP message optimizer prototype.