SOAP Transport

The increase in interest in web services, and SOAP as a medium for their use, has prompted the inclusion of a SOAP transport in the Zeus platform. It is used only for messages between platforms, not within a platform, and so is introduced to the local platform through the Agent Communication Channel(ACC). This is registered as a SOAP service, using the Apache Axis libraries and SOAP server, and then intercepts all messages using the SOAP transport to or from the local platform. The communication service allows a single string as a parameter, which will be passed on to the target agent as the content of the message. Currently, this means the message should be written in FIPA ACL in order to be understood by the agent, but obviously the transport is content independent, so no change will be needed if different content languages are added later.

Utilising

In order to use the SOAP transport, you need two Zeus platforms that operate the SOAP service between which the message will be sent. The address of an agent on the remote platform should be added to the contacts file for each ACC, identifying the transport to use as SOAP (it is explained later how to achieve this). This means that when a message is sent to that name, the local agent will be directed to send the message to the ACC, which will pass it to the remote ACC using SOAP, which will then pass it to its local endpoint.

Setting Up

There are three locations where configuration changes need to be made to enable the SOAP transport. First, the class `zeus.actors.intrays-.FIPA_SOAP_MESSAGE` needs to be registered as a service with your Axis server. The name it is registered under should be all lowercase. The URL of this service, excluding the hostname, should then be put into the Zeus properties file *.zeus.prp* under the key `soap.server.servlet`, with the port number under the key `soap.server.port`, and the URN under `soap.server.urn`. The default URN value is "urn:agent:soap". Finally, the remote agents need to be placed into the contacts file. Their addresses consist of three elements, separated by double colons. The first is the transport name, in this case "SOAP". Next is the URL to communicate with, which should be the URL of the remote service, for example "192.168.0.12:8080/-axis/services/zeusinterface". Finally, the URN of the service is added, with the same default as before.

Now, whenever a message is sent from an agent using the name of one of the SOAP contacts, it will be passed between the ACCs using SOAP.

Service Descriptions

Service descriptions are an important part of the automation of the semantic web, and DAML-S is the most common language for producing these descriptions. In order to encourage the spread of published service descriptions, the Zeus toolkit has added the feature of automatically generating parts of the service descriptions for the agents that are currently running. No additional action is required from the user of the toolkit, bar ensuring that the resulting files are publsihed, enabling them to be useful.

Zeus makes use of three of the elements of the DAML-S collection of files. The process model defines the types involved in the operations provided by the service, and declares the service. This must be written by the user, and placed in the appropriate published location. The service profile is generated by Zeus, and defines the particular service, describing the intial state and resultant state of the service, and the inputs and outputs that it uses. Combined with the process model, this fully describes a type of service. Finally, service instance declarations are generated to link service types to actual providers of those services.

The automatically generated elements are retrieved from each agent by examining their task definitions, and returning some kind of description to the Facilitator agent, which will then write them out to a filesystem location. Currently, both the format of the files and their locations are according to the Agentcities agreed format, but the code was designed with modularity in mind, so additional renderers could easily be added for other types of description.

The only option that needs to be set by the user is where the files should be written out, which is set through th e`http_root` property in the *.zeus.prp* file. Files will then be placed in either the `classes/`*`serviceName`* directory for process models and service profiles, or the `instance/`*`serviceName_instanceName`* directory for service declarations. These should be visible to the web under a "services" directory from the root of the web server.

DAML Ontologies

In a further step towards the semantic web goals of machine interpretability, the Zeus toolkit is able to produce DAML+OIL ontologies in addition to the former Zeus ontologies. The ontologies are written out as two files, a DAML file that embodies the class structure, and a schema file that defines restricted data types. There is limited support for reading in files of these types that are not created by the Zeus toolkit, but the results may be unpredictable. Importing ontologies is in the early stages of development.