Requests and Responses

Overview

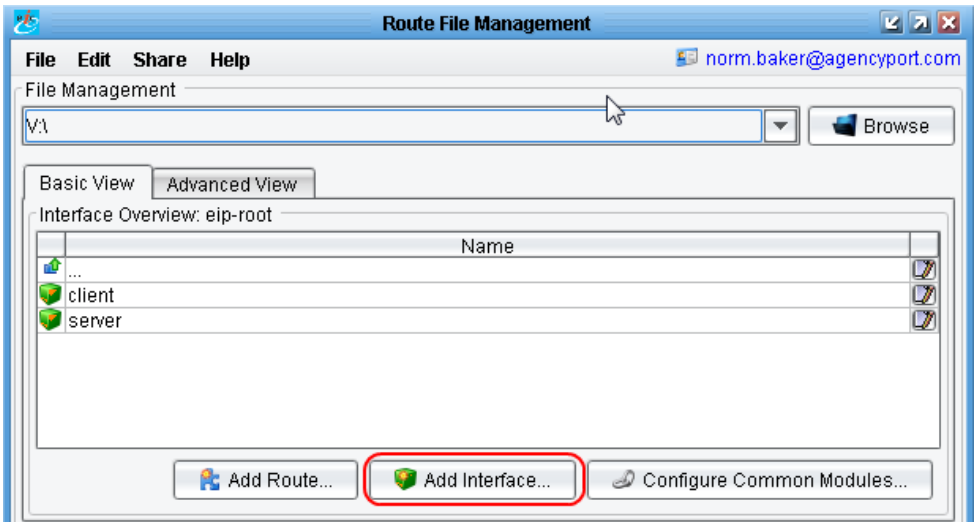
This interface demonstrates how to accomplish a request-response pair. This is a relatively standard mechanism of creating web-services, and is easily achieved in the eiConsole.

This interface has three functioning routes; the request route, the response route, and a sample web-service route. It performs no data transformation.

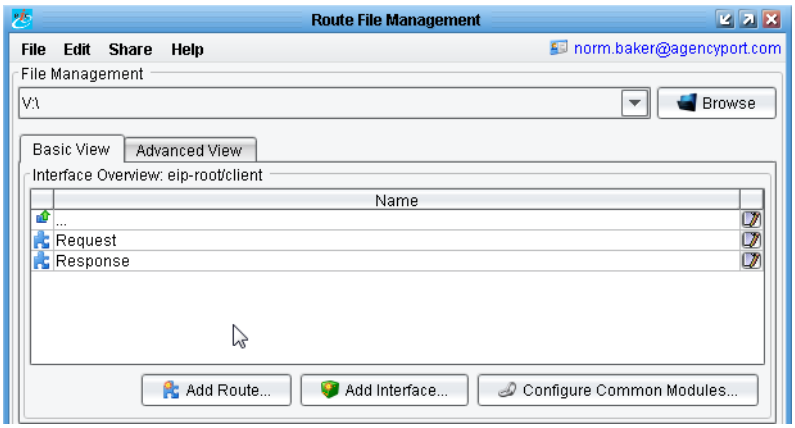
Interface

Create a new Working Directory called “service-root”.

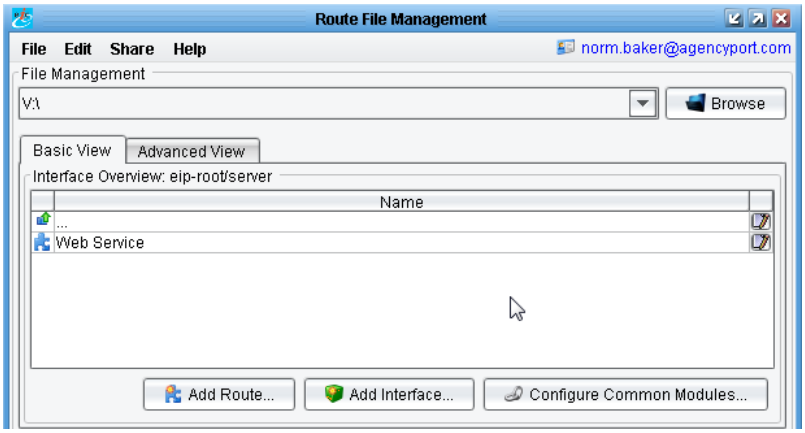
Add two interfaces called “client” and “server”:



We will then create two routes in the “client” interface: “Request” and “Response”:

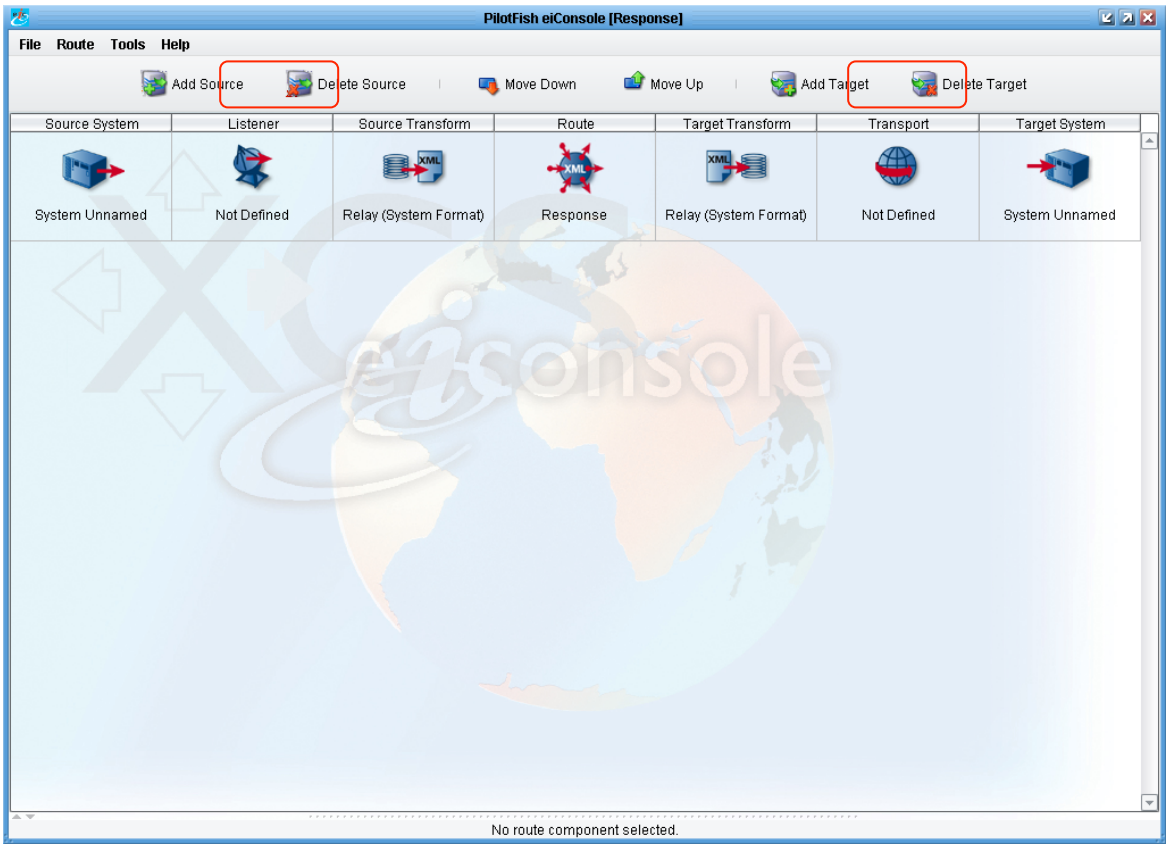


Last, but no means least we will create one route in the "server" interface called "Web Service":

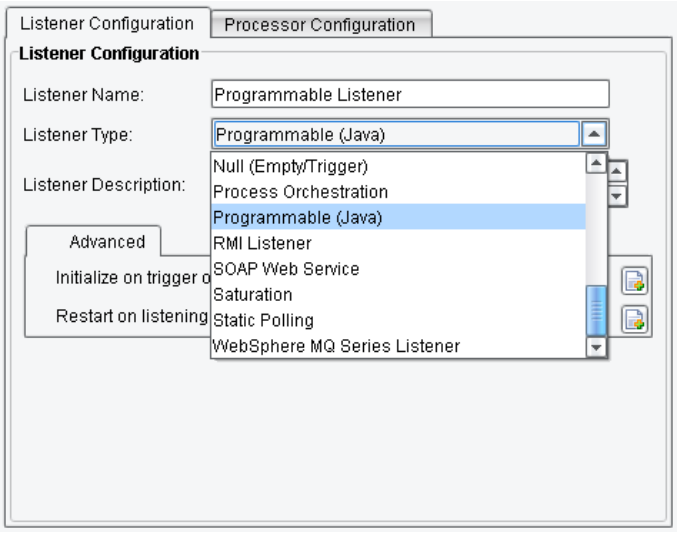


Response Route

We will start back-to-front by configuring the 'Response' route first. Add a new Source and Target:



The response route uses a 'Programmable Listener' that our request route will eventually call, give it a short unique name e.g. "Response Listener":



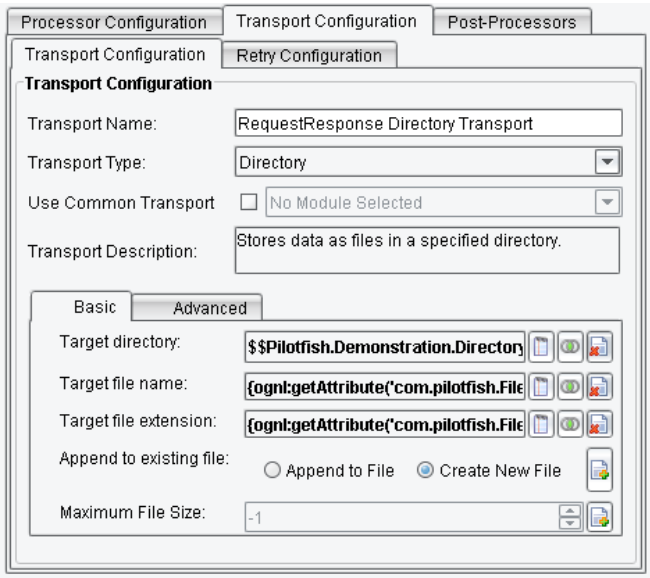
Nothing substantive happens in this route; the data received by the listener is written directly to disk.

Note that the file name and file extension are coming from our request route; transaction attributes persist across programmable listeners, meaning that here, we are using the file name and extension from the original file our request route picked up.

Add a 'Directory Transport' and use the following two OGNL expressions for file name and extension:

getAttribute('com.pilotfish.FileName')

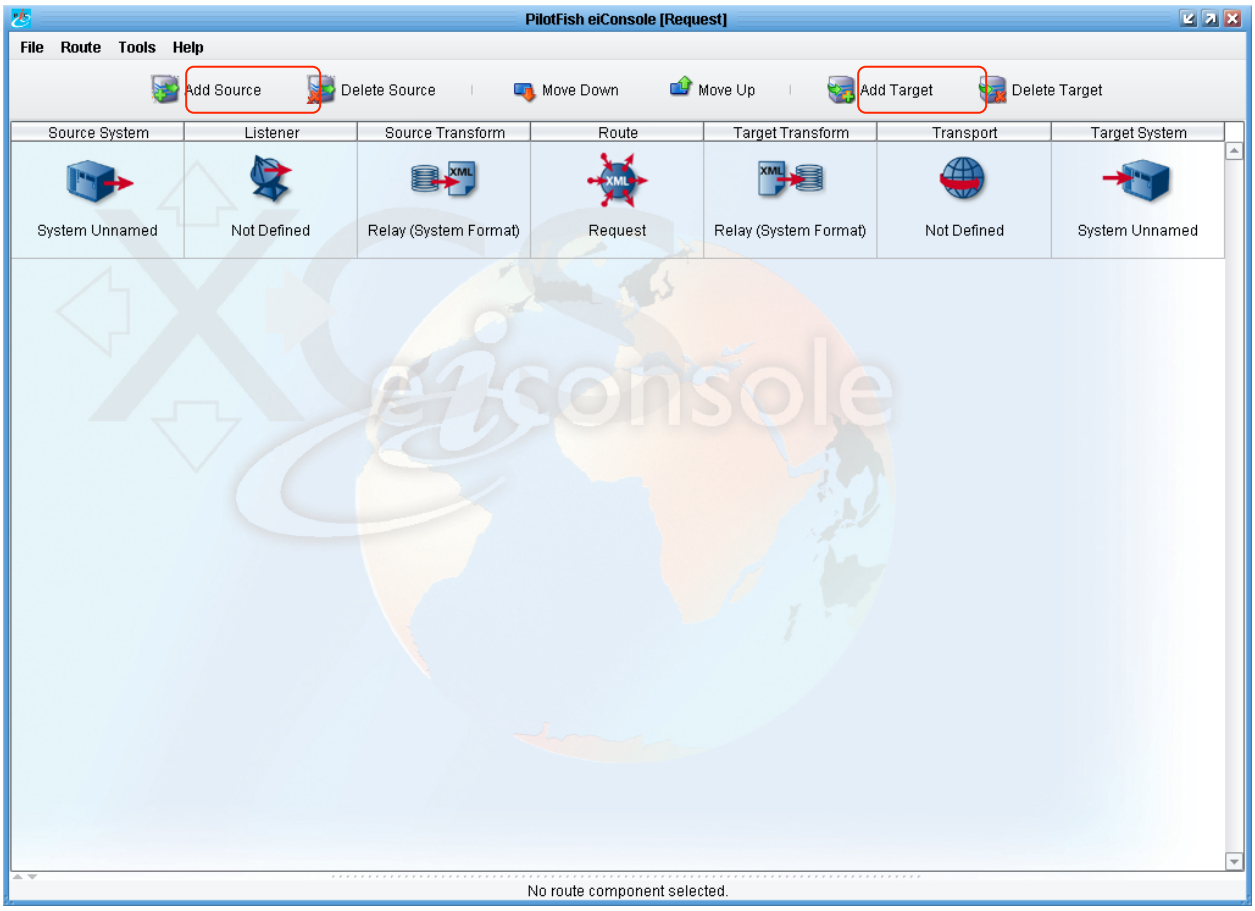
getAttribute('com.pilotfish.FileExtension')



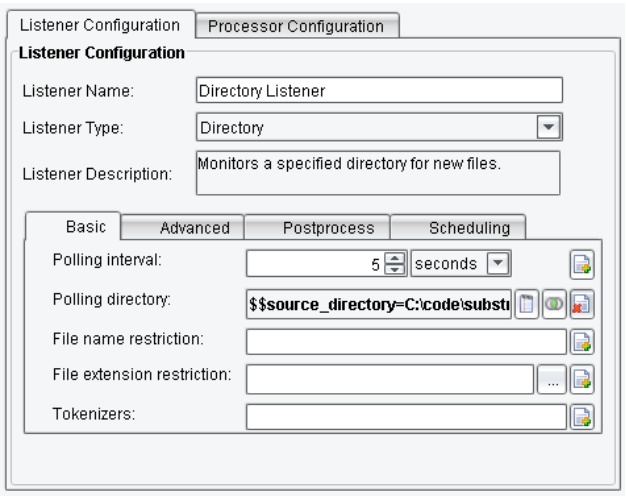
Request Route

The request route will poll a directory and pull in the sample file "people.xml" and post it to the web-service.

Start by opening the 'Request' route and adding a new Source and Target:

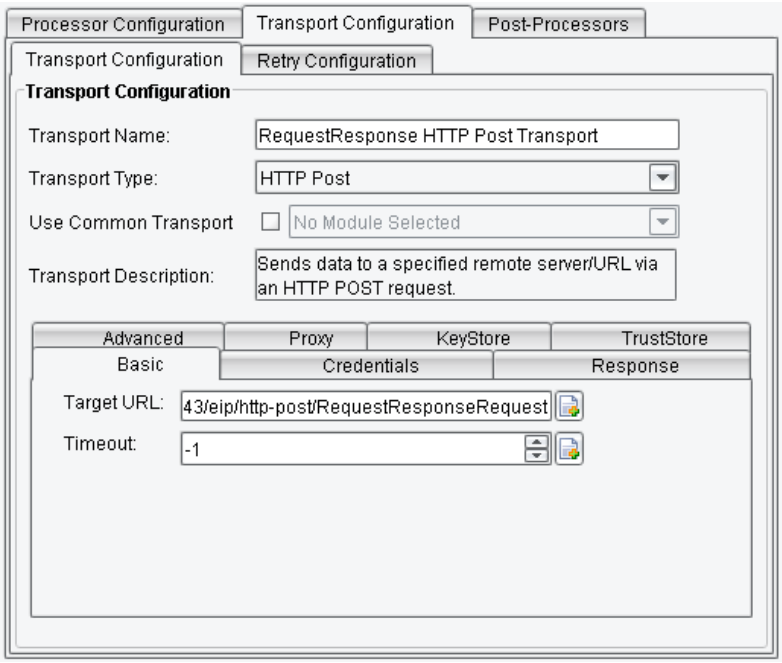


The listener will poll the same input directory used previously at an interval of 5 seconds:

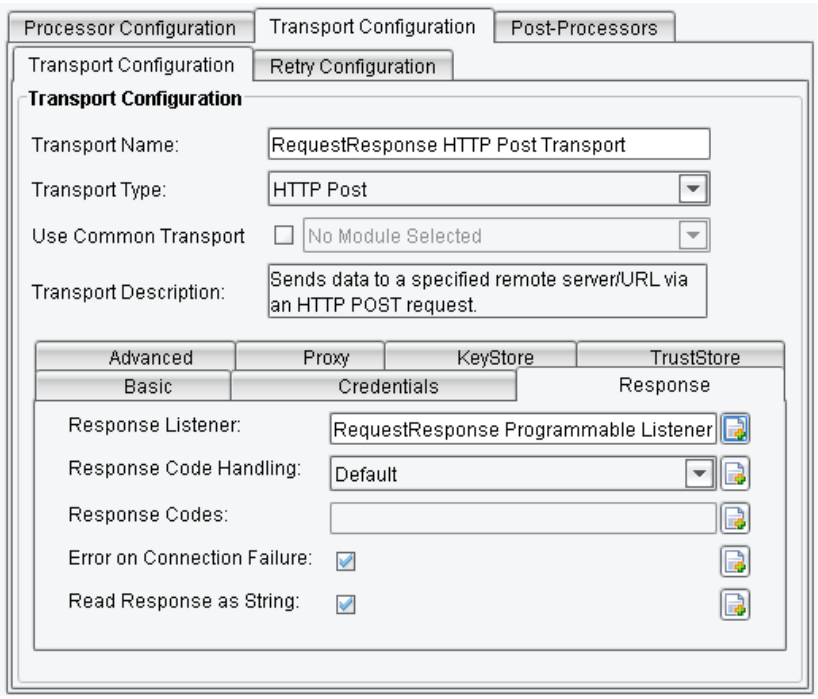


The transport has two import configuration values; the first is the Target URL. This determines the web-service to be hit. Here, we're anticipating to target our web-service route running in emulator mode so we'll configure the URL to:

http://localhost:8443/eip/http-post/webservice



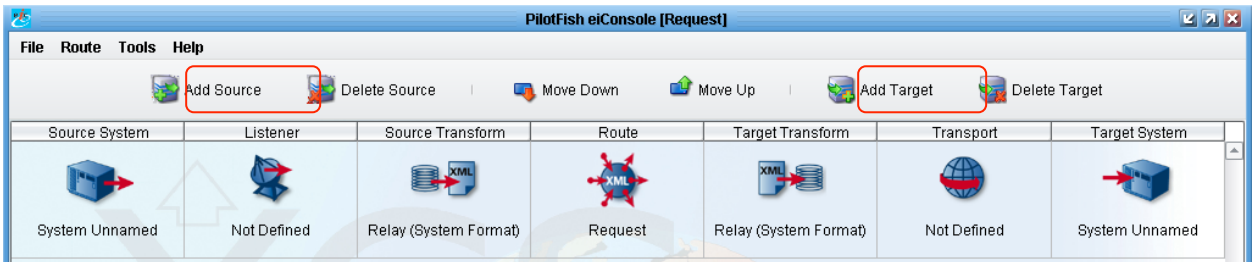
The second important configuration is the 'Response Listener'; if left blank, the HTTP request will be performed asynchronously. If a response listener – a "Programmable Listener" – is provided, the request will be performed synchronously, that is, expecting data to be returned. The HTTP response will be used to invoke the response listener created above:



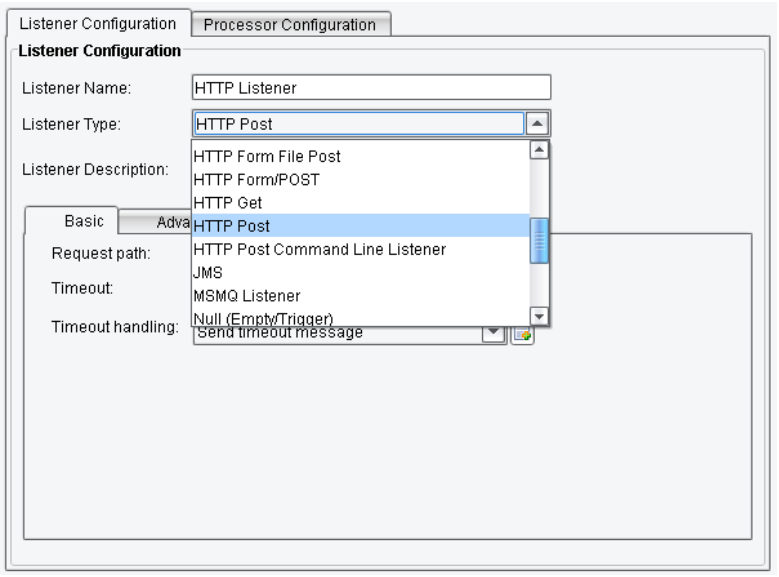
Web-Service Route

The web-service route is very simple; it consists of an HTTP listener and a synchronous response.

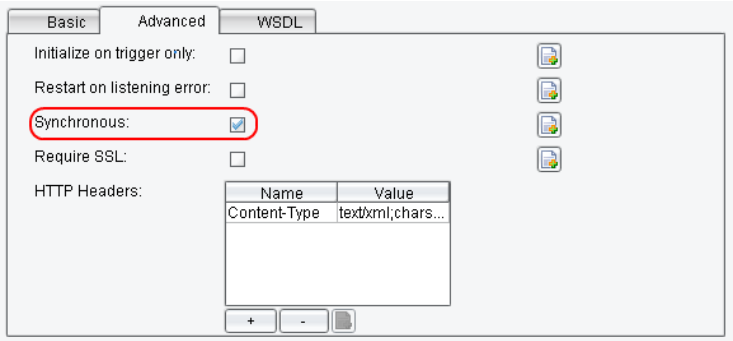
Start by opening the 'Web-Service' route and adding a new Source and Target:



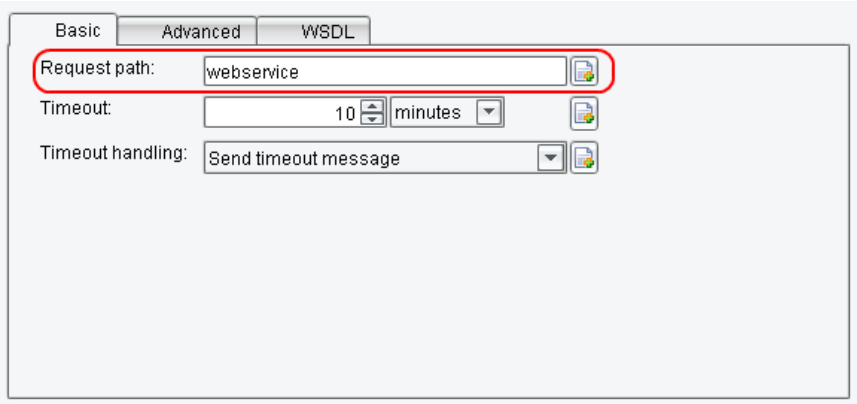
There are a number of different listeners that support synchronous response; we'll use the 'HTTP Post Listener':



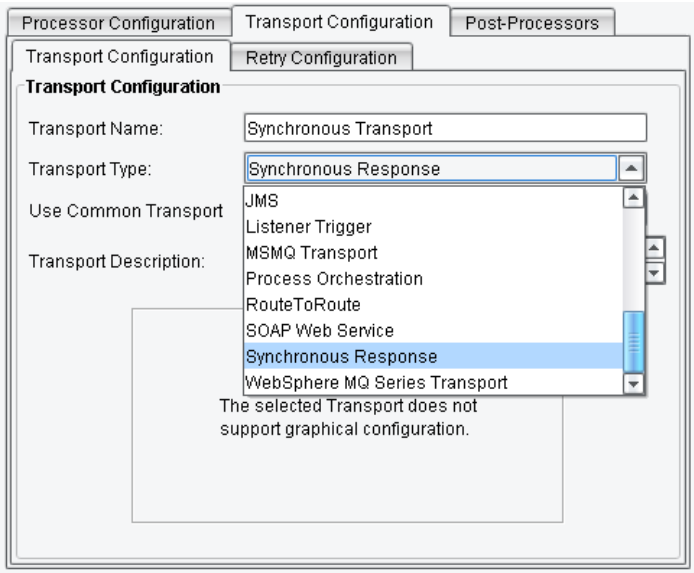
This listener is made synchronous if and only if the "Synchronous" configuration option is selected under the "Advanced" tab:



Give the listener a 'Request path' of "webservice":



The transport should be configured to be a 'Synchronous Response Transport':



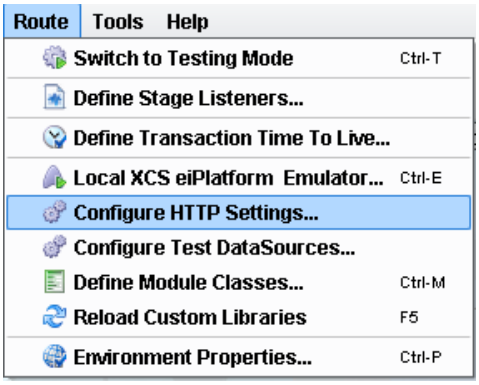
The Synchronous Response transport is contextual; it will figure out how to respond based on the incoming transaction.

This concludes this setup of an HTTP request-response pair.

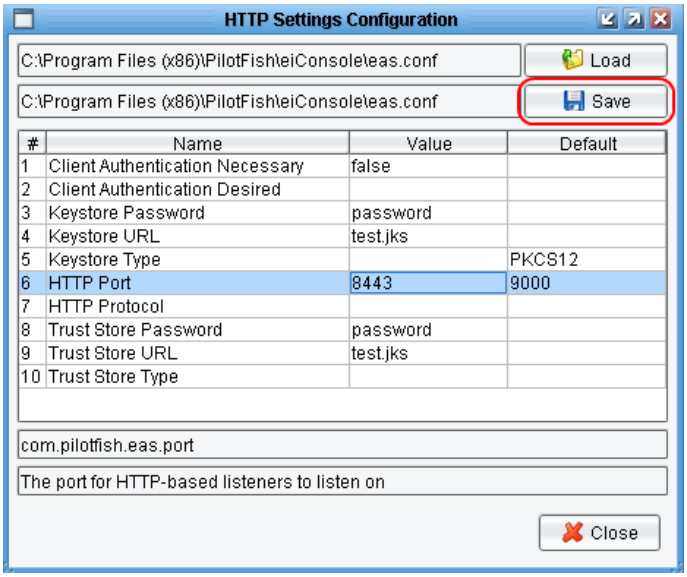
Platform Emulator

We will now deploy our new interface using the XCS Platform Emulator.

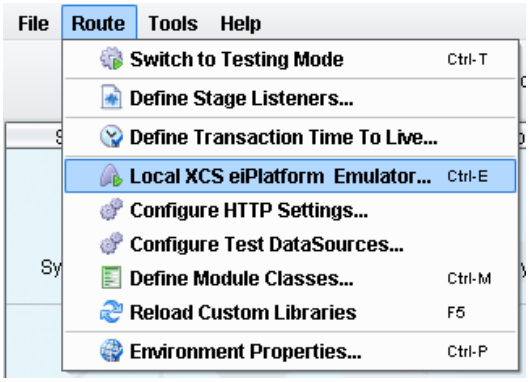
Open the web-service route and click 'Configure HTTP Settings…':



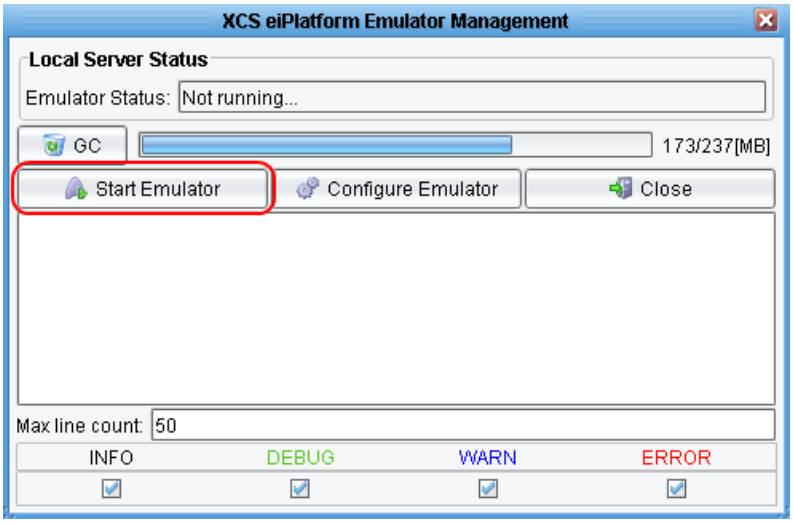
This will launch the window below then you can update the port to: 8443, remember to hit 'Save' after making your changes.



You are now ready to launch the Emulator Manager:



From the pop-up you can control the emulator. We'll start it and note the port it's running on:



We can check if the service is available by opening the following URL in a browser:

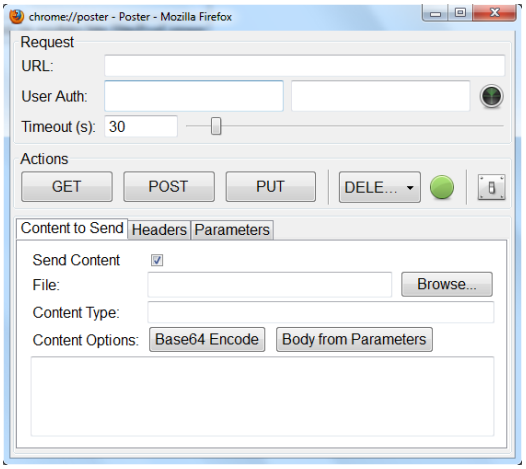
http://localhost:8443/eip/http-post/webservice

We should see a registered listener with a path of "/webservice":

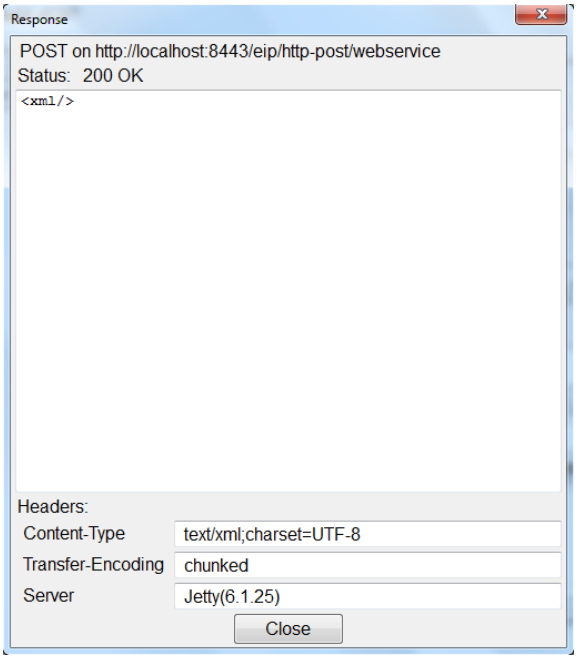
PilotFish XCS eiPlatform HttpPostServlet Version: EIP9.12R1 Build: r6267
Registered listeners:

Path	Name	Class
/webservice	Request Response.Web Service.HL7 LLP Listener	com.pilotfish.eip.modules.http.HttpPostListener

We can now test our web-service by using a simple FireFox application called Poster:



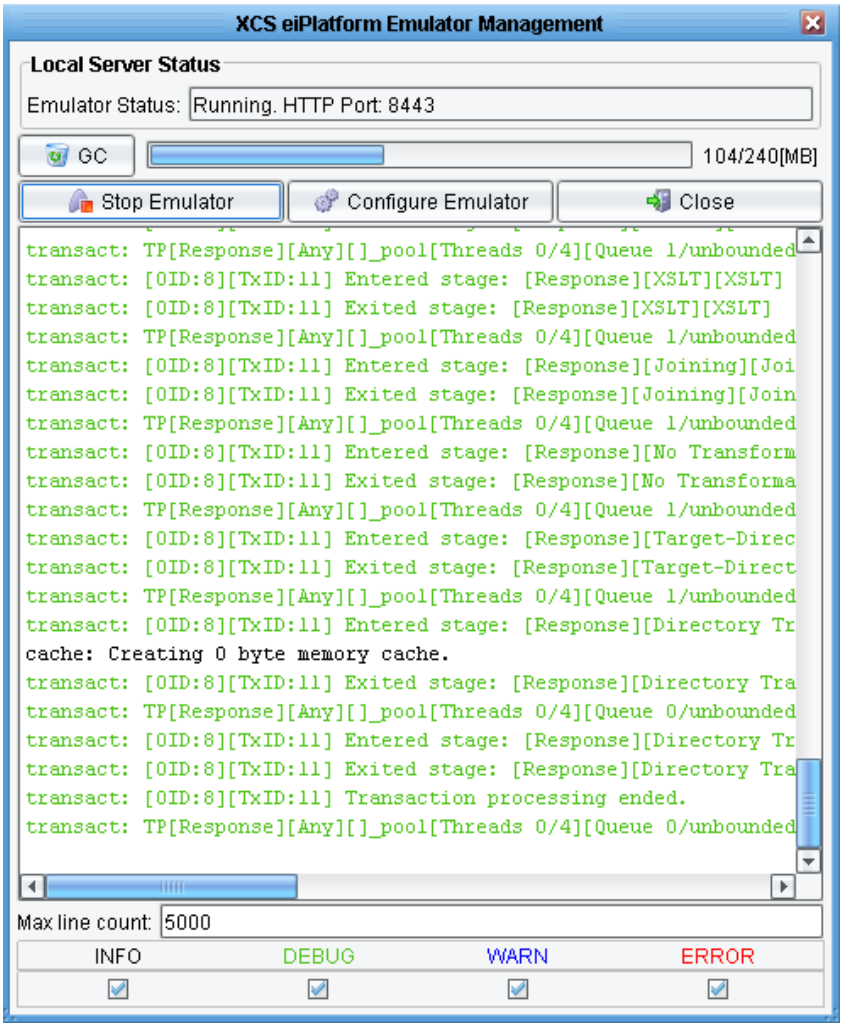
We should get an echo response to our POST request:



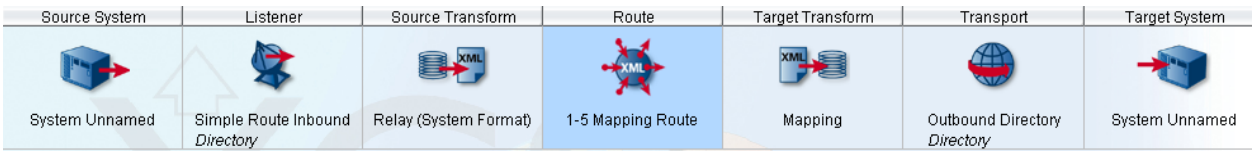
Last thing to test is that all routes are working correctly. For this we will drop the "person.xml" file into the listener's input directory.

If all goes well we should see a file with the same name appear in the transport's output directory.

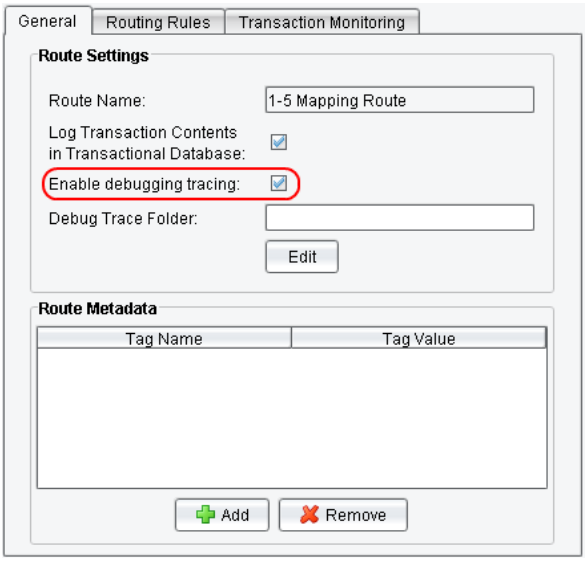
You can also see the logs rolling in the Emulator Manager as the transaction flows through each of the routes and stages (you may need to increase the Max Line Count):



To debug a transaction via the Emulator you need to 'enable debugging tracing' on each route. To do this stop the Emulator and close the Manager. Select the 'Routing' stage:

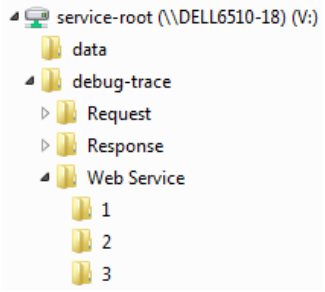


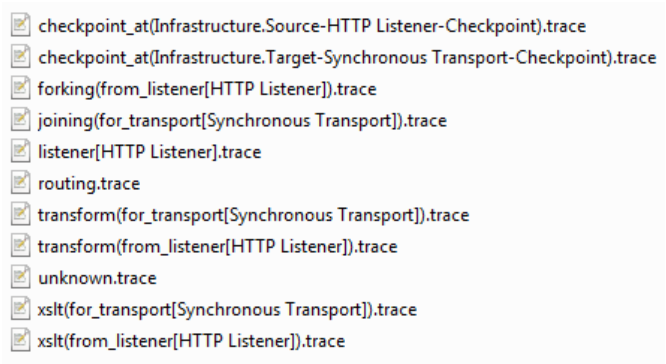
Then select the 'enable debugging tracing' option:



If you do not specify a directory it will automatically generate a 'debug-trace' folder in your working directory.

The next time your run the Emulator and fire a transaction through it will show up in the logs:





The 'trace' files can be opened in a text editor.