Dynamic Router

This section explains, through an example scenario, how the Dynamic Router EIP can be implemented using WSO2 ESB. The following topics are covered:

- Introduction to Dynamic Router
- Example scenario
  - Environment set up
  - ESB configuration
  - Simulating the sample scenario
  - How the implementation works

Introduction to Dynamic Router

The Dynamic Router EIP avoids dependence on all possible destinations while maintaining efficiency. It is a router that can self-configure based on special configuration messages from participating destinations. Dynamic Router is available for configuration through a control channel by the receiving parties that can use this control channel.

Example scenario

This example scenario demonstrates a router that takes an incoming request and decides which back-end service to transmit the message to. To make that decision, it uses a property in the message itself, very much like the Content-Based Router. However, it can also cross-check a registry entry to see if a specific endpoint accepts messages with that property. This approach allows you to reconfigure the router when registry entries change.

The diagram below depicts how to simulate the example scenario using WSO2 ESB.
Before digging into implementation details, let's take a look at the relationship between the example scenario and the Dynamic Router EIP by comparing their core components.

<table>
<thead>
<tr>
<th>Dynamic Router EIP (Figure 1)</th>
<th>Dynamic Router Example Scenario (Figure 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Channel</td>
<td>Main sequence</td>
</tr>
<tr>
<td>Dynamic Rule Base</td>
<td>Local Registry</td>
</tr>
<tr>
<td>Dynamic Router</td>
<td>Switch Mediator</td>
</tr>
<tr>
<td>A, B, C</td>
<td>Simple Stock Quote Services</td>
</tr>
</tbody>
</table>

Environment set up

2. Go to `<ESB-HOME>/samples/axis2Server/src/SimpleStockQuoteService` and run `ant` to build and deploy the SimpleStockQuoteService service. For more information see [Deploying sample back-end services](#).
3. Start three sample Axis2 server instances on ports 9000, 9001, and 9002. For instructions, see [Setting Up the ESB Samples - Starting the Axis2 server](#) in the WSO2 ESB documentation.

ESB configuration

Start the ESB server and log into its management console UI ([https://localhost:9443/carbon](https://localhost:9443/carbon)). In the management console, navigate to the **Main** menu and click **Source View** in the **Service Bus** section. Next, copy and paste the following configuration to the source view.

```xml
<definitions xmlns="http://ws.apache.org/ns/synapse">
  <localEntry key="ConfigA">
    <value>foo</value>
    <description/>
  </localEntry>
</definitions>
```
<localEntry key="ConfigC">
  <value>WSO2</value>
  <description/>
</localEntry>
<localEntry key="ConfigB">
  <value>bar</value>
  <description/>
</localEntry>
<proxy xmlns="http://ws.apache.org/ns/synapse"
  name="DynamicRouterProxy" startOnLoad="true" statistics="disable"
  trace="disable" transports="http,https">
  <target>
    <inSequence>
      <log level="full"/>
      <switch source="get-property('To')">
        <case regex="http://localhost:9000.*">
          <filter xmlns:m0="http://services.samples"
            xpath="get-property('ConfigA') = //m0:getQuote/m0:request/m0:symbol/text()">
            <then>
              <send>
                <endpoint>
                  <address uri="http://localhost:9000/services/SimpleStockQuoteService"/>
                </endpoint>
              </send>
            </then>
            <else>
              <log level="custom">
                <property name="MESSAGE" value="Registry Value Doesn't Matched"/>
              </log>
            </else>
          </filter>
        </case>
        <case regex="http://localhost:9001.*">
          <filter xmlns:m0="http://services.samples"
            xpath="get-property('ConfigB') = //m0:getQuote/m0:request/m0:symbol/text()">
            <then>
              <send>
                <endpoint>
                  <address uri="http://localhost:9001/services/SimpleStockQuoteService"/>
                </endpoint>
              </send>
            </then>
            <else>
              <log level="custom">
                <property name="MESSAGE" value="Registry Value Doesn't Matched"/>
              </log>
            </else>
          </filter>
        </case>
      </switch>
    </inSequence>
  </target>
</proxy>
The above configuration helps you explore the sample scenario.

Simulating the sample scenario

The sample client used to simulate the sample is the **Stock Quote Client**, which can operate in several modes. For more details on this sample client and its operation modes, see **Sample Clients** in the WSO2 ESB documentation.

To execute the sample client, send the following requests from the `<ESB_HOME>/samples/axis2Client` directory:

```
ant stockquote
-Daddurl=http://localhost:9000/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/services/DynamicRouterProxy -Dsymbol=foo

ant stockquote
-Daddurl=http://localhost:9001/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/ -Dsymbol=bar

ant stockquote
-Daddurl=http://localhost:9002/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/ -Dsymbol=WSO2

ant stockquote
-Daddurl=http://localhost:9000/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/ -Dsymbol=bar

ant stockquote
-Daddurl=http://localhost:9001/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/ -Dsymbol=WSO2

ant stockquote
-Daddurl=http://localhost:9002/services/SimpleStockQuoteService
-Dtrpurl=http://localhost:8280/ -Dsymbol=foo
```

You will see that only the first three commands are sent to the back-end services. This is because the symbols passed within those requests are the symbols associated with the particular endpoint service.

You need to send the corresponding port correctly with the symbol to get the response as follows:


If you send a different request you view the following message in the ESB Console:

```
[ESB-Core] INFO - LogMediator MESSAGE = Registry Value Doesn't Matched
```

How the implementation works

Let's investigate the elements of the ESB configuration in detail. The line numbers below are mapped with the ESB
configuration shown above.

- **sequence** [line 70 in the ESB config] - This is the default main sequence invoked when a message is received by the ESB.
- **switch** [line 73 in the ESB config] - The switch inside the mediator of the main sequence. It checks the To header inside the SOAP message to see which endpoint the message is intended for. Based on this, one of the three sequences is invoked: SendServiceA, SendServiceB or SendServiceC.
- **local entry** [line 7 in the ESB config] - This is one of the local entries created as a registry entry to see if a specific endpoint accepts messages with a given symbol. The other two local entries are in line 11 and line 15 in the ESB configuration.
- **sequence** [line 20 in the ESB config] - This is the sequence with the key SendServiceA. This sequence cross checks the relevant local entry to see whether the specific endpoint accepts messages with a defined property. If the particular endpoint supports a requested property, the request message will be passed to that endpoint.

The other sequences starting in lines 36 and 53 in the ESB configuration follow the same pattern as this sequence.