ITINERARY RESOLVER EXPLANATION

<https://docs.microsoft.com/es-es/biztalk/esb-toolkit/how-the-itinerary-on-ramp-sample-works>

Following the list of service invocation steps in the itinerary is the section containing details of the resolvers (represented by connection strings) that allow the Itinerary service to locate or provide resolution information for each service defined in the itinerary.

|  |
| --- |
| <ResolverGroups xmlns="">  <Resolvers serviceId="Microsoft.Practices.ESB.Services.Transform0"><![CDATA[BRE:\\Policy=ResolveMap;Version=1.0;UseMsg=False;]]></Resolvers>  <Resolvers serviceId="Microsoft.Practices.ESB.Services.Routing1"><![CDATA[STATIC:\\TransportType=FILE;TransportLocation=C:\Projects\Microsoft.Practices.ESB\Source\Samples\DynamicResolution\Test\Filedrop\OUt\%MessageID%.xml;Action=;EndpointConfig=;JaxRpcResponse=False;MessageExchangePattern=;TargetNamespace=;TransformType=;]]><![CDATA[UDDI3:\\ServerUrl=http://localhost/uddi;SearchQualifiers=andAllKeys;CategorySearch=;BindingKey=uddi:esb:orderfileservicev3.1;]]></Resolvers>  <Resolvers serviceId="ProcessAndRespond2" />  </ResolverGroups>  </Itinerary> |

The following are the three steps defined in the itinerary preceding configuration:

|  |
| --- |
| 1. Execute the Microsoft.Practices.ESB.Services.Transform orchestration to transform the message with the ResolverMap policy using BizTalk Business Rules Engine (BRE). 2. Execute the Microsoft.Practices.ESB.Services.Routing orchestration to route the transformed message to multiple locations using the routing Microsoft.Practices.ESB.Services.Routing1. The **<ResolverGroups>** section contains a **<Resolvers>** element with this identifier, which defines the connection strings. 3. Execute the ProcessAndRespond orchestration provided with this sample. The implementation of this orchestration sends as the response a copy of the request message back to the Itinerary Test Client.   With the completion of each service, the service advances the itinerary and promotes the next service defined in the itinerary to be the current service instance, with its state set to **Pending**. |

The following is the sequence of events that occur after the test client application submits the message:

* The **OnRamp.Itinerary** receive port **receives** the **message**.
* The **ItineraryReceiveXml** pipeline **extracts** **the itinerary** from the SOAP header, validates and pre-processes it, writes the itinerary as a message context property into the inbound message, and **publishes** the message to the BizTalk Message Box database.
* A subscription for the **Microsoft.Practices.ESB.Services.Transform** service orchestration **triggers** invocation of this orchestration. **The orchestration first retrieves the current itinerary step** by passing the current message as a parameter, as shown in the following code.

|  |
| --- |
| itineraryStep = itinerary.Itinerary.GetItineraryStep(InboundMessage); |

The **ItineraryStep** object **contains** all the **information** about the **current** **service** instance for execution, as well as any resolvers associated with it.

The **Resolver** object is retrieved from the **ItineraryStep** instance and the ESB Resolver Framework is used to resolve the full name of the transformation map, as shown in the following code.

|  |
| --- |
| **resolverDictionary** = Microsoft.Practices.ESB.Resolver.ResolverMgr.**Resolve**(InboundMessage, resolver);  // Set the transform type.  transformType = resolverDictionary.Item("Resolver.TransformType"); |

The Microsoft BizTalk ESB Toolkit Resolver and Adapter Framework **accomplishes** this by loading the appropriate resolver from the cache (in this example, the BizTalk Business Rules Engine resolver), which invokes the **ResolverMap** policy and populates the **ResolverDictionary** object.

After the orchestration completes, the code calls the **AdvanceItinerary** method, as shown in the following code.

|  |
| --- |
| // Call the Itinerary helper to advance to the next step.  itinerary.Itinerary.Advance(OutboundMessage, itineraryStep.ItineraryStep); |

This advances the current itinerary by updating its properties and promoting the next service defined in the itinerary as the one to execute next. The method copies the itinerary into the outbound message, which the service publishes back into the Message Box database through a direct-bound send port.

A subscription for the Microsoft.Practices.ESB.Services.Delivery service orchestration triggers invocation of this orchestration. This orchestration follows a similar process to the first one, obtaining the current Itinerary step. However, this orchestration iterates through a collection of resolvers returned by the ItineraryStep instance. For each resolver in the collection, the delivery orchestration uses the Microsoft BizTalk ESB Toolkit Resolver and Adapter Framework to resolve the transport locations and promote them as context properties within the outgoing message, as shown in the following code.

|  |
| --- |
| // Move to retrieve the first resolver.  **resolver** = resolvers.Current;  // Pass the resolver configuration to the Resolver Manager  // for resolution.  **resolverDictionary** = Microsoft.Practices.ESB.Resolver.ResolverMgr.Resolve(InboundMessage, resolver);  // Set the transport properties.  transportLocation = resolverDictionary.Item("Resolver.TransportLocation");  transportType = resolverDictionary.Item("Resolver.TransportType");  // Call the Adapter Manager to set all necessary properties.  Microsoft.Practices.ESB.Adapter.AdapterMgr.**SetEndpoint**(resolverDictionary, DeliveryMessage);  // Set the delivery port address and type.  DeliveryPort(Microsoft.XLANGs.BaseTypes.**Address**) = transportLocation;  DeliveryPort(Microsoft.XLANGs.BaseTypes.**TransportType**) = transportType; |

|  |
| --- |
|  |

A subscription for the ProcessAndRespond orchestration triggers invocation of this orchestration because of a match of the message context properties defined for the filter expression properties.

|  |
| --- |
| (Microsoft.Practices.ESB.Itinerary.Schemas.ServiceName == :"ProcessAndRespond")  && Microsoft.Practices.ESB.Itinerary.Schemas.ServiceState == "Pending")  && (Microsoft.Practices.ESB.Itinerary.Schemas.ServiceType == "Orchestration") |

The ProcessAndRespond orchestration advances the itinerary and sends the original request message back to the on-ramp service to the Itinerary Test Client application as the response.