**MetLife EAD Gateway Framework Model**

The model represents the MetLife EAD Gateway Framework, where we have represented the framework as a F1 Car Race Track and the messages are represented as cars.

* **Technical Descriptions of the Framework:**

MetLife EAD Gateway framework is created on IBM WebSphere Message Broker 8.0. It comprises of some message flows, sub-flows, common procedures and some java components. As part of ESB service provider, our main job is to build services to transform the consumer (Front-End) specific messages into provider(Back-End) specific message formats and vise-versa.

So, depending upon the transformation, we segregated our services in 3 types and details of these services are kept in an XML file which we call as configuration file. The types are:

1. Provider Service or Pass-Through service.

2. Translator Service.

3. Provider Specific Service or Proxy Service.

1. Provider Service or Pass-Through service:

We use this kind of services, where the consumer message format is similar to the provider message format and provider service run on HTTP or SOAP/HTTP or MQ protocol. So, for this kind of services, no message transformation is required. So, ESB only changes the message namespace and hit the provider service. And after getting back the response from provider service, ESB changes back the namespace again and send back to consumer service.

This functionality is already provided in our Gateway. So for creating this kind of ESB services, we just need to create the configuration file giving the provider details, and our service is ready to be exposed to the consumer.

2. Translator Service:

We use this kind of services, where consumer message format is not similar to the provider message format, but both the message formats are in XML and provider service run on HTTP or SOAP/HTTP or MQ protocol. So, for this kind of services, we use XSLTs for doing the transformation as XML to XML transformation is very simple through XSLT. So, in this case, we transform the consumer request message to provider specific xml structure and hit the provider service. And after getting back the response from provider service, we again transform the response message to consumer specific XML structure and send it back to the consumer service.

So, for creating this kind of ESB services, we need to create the XSLT files (request and Response XSLTs) and keep its details in the configuration file. Then we are ready to expose our ESB service.

3. Provider Specific Service or Proxy Service:

We use this kind of services, where the provider message format is any non-XML format or provider service run on any other protocol except HTTP, SOAP/HTTP and MQ. For this kind of services, we create separate message flows and implement the transformation logic there. The message flows are MQ services and the MQ details are kept in configuration file.

The processing level artifacts (configuration files, XXLTs, Provider's WSDLs etc.) in IBM WSRR and once after running a successful transaction we store all the details in our server cache, so that we don't have to go to WSRR every time.

* **Physical Model Representation:**

Now, we have represented this total architecture in our physical model.

If you look at the photos, we will be able to see some blocks whose representations are given below.

Dark Blue Shed Block = Consumer.

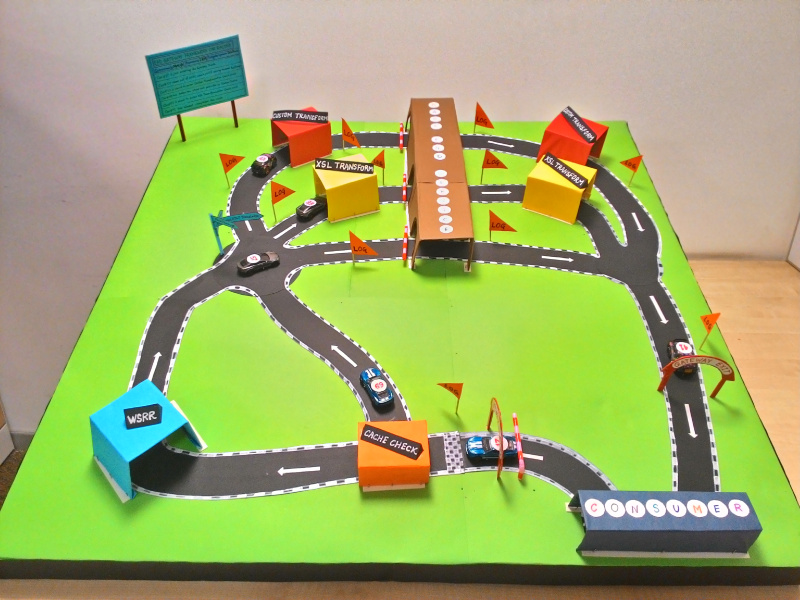
Brown Shed Block = Provider.

Orange Shed Block = for checking the server cache.

Cyan Shed Block = WSRR.

Yellow Shed Blocks = XSLTs.

Red Shed Blocks = Customized Message Flow Services.



Now once a message (represented as cars) comes out of consumer block hits the ESB Gateway. Then, it will check for the server cache and based on the availability of details in cache, it will choose its path, whether to take shortcut or to go via WSRR block. Now in the crossroad point, we are providing 3 routes and based on the details fetched from configuration file, the car will select its route. Then after transformation (in case of Translator service or proxy service) the car will go to the Provider block. After coming out of the provider block the reverse transformation happens and after transformation, the car goes back to the consumer shed.

So, this is the complete representation of the process.