Using New York State DEPARTMENT OF LABOR Web-Services

(An approach for external organizations to consume DEPARTMENT OF LABOR Web-Services)

Prepared by

Enterprise Planning and Architecture Standards (EPAS)

Revision History:

Revision No	Revised Date	Author	Description
V1.0	09/26/2008	Vijai Singh	First Draft
V1.1	10/2/2008	Mike Bloss	Formatting
V1.2	07/12/2011	Vijai Singh	Modified the WSRR Section

Table of Contents

Introduction:

Web services provide a language-neutral, environment-neutral programming model that accelerates application integration inside and outside the enterprise and provides an easier-to-use architecture for integrating information. The New York State Department of Labor (Department of Labor) has chosen an open standard based approach to Service Oriented Architecture. The Department of Labor infrastructure is built on IBM's WebSphere technology. DataPower is used as Web Services Gateway for external agencies. Internally, services will be deployed on Websphere Process Server (WPS), Websphere Message Broker (WMB) and Websphere Application Server (WAS). The Department of Labor services will be registered in the Websphere Service Registry and Repository (WSRR). WSRR will also be used for the end point lookup from DataPower, WPS or WMB. However, the Department of Labor will not share the actual endpoint of the services running on WPS, WMB or WAS. Instead the service endpoints will be wrapped and exposed from the DataPower Layer.

The Following guiding principles are the foundation of our service consumption approach:

1. The Subscriber (consumer) needs to have access to the provider's WSDL (Web Services Description Language) file and XSD's to consume services.

2. When the (Department of Labor) is the service provider, it will provide the WSDL and XSD of the service to the outside consumers.

3. From consumption of services perspective, the service consumer can have any technology at their end and this technology can be different from the Department of Labor's technology. The service consumer can generate the service client on any open-systems technology such as .NET, Java, C,C++ etc.

4. The Request and Response Message will be in the form of SOAP Request and SOAP Response. SOAP Request/Response is an XML message that uses its header segment for the security and its body segment for the service request or service response.

5. The Department of Labor WSDL's are WS-I (Web services Interoperability) basic profile 1.0 compliant and the DEPARTMENT OF LABOR uses SOAP 1.1 and WSDL 1.1.

6. The Department of Labor approach for the service delivery is top-down, meaning it is creating the WSDL and XSD's first.

Purpose:

The ability to share Department of Labor created services with other agencies or private entities is one of the driving forces behind the Department of Labor's switch to a Service Oriented Architecture (SOA). Now that Department of Labor is creating services that could potentially be consumed by external entities the question arises, *how will the services work for consumers with infrastructures different than that of the* Department of Labor*?* This white paper will explain how other infrastructures can consume Department of Labor services and how Department of Labor would expect to consume services offered by other agencies or business entities.

The Department of Labor as a service provider and external agencies as service consumers:

External agencies (service consumers) will access the Department of Labor services through a Web Service Gateway. Using alias addresses, the Web Services Gateway will map WSDL-defined web services within Department of Labor's protected environment to external clients. The Gateway thus acts as the proxy. Internal services implemented on the Websphere Process Server (WPS), Websphere Application Server (WAS) or Websphere Message Broker (WMB) will be imported into the Gateway and will be available as proxy services to the external clients as shown in the Figure 1.
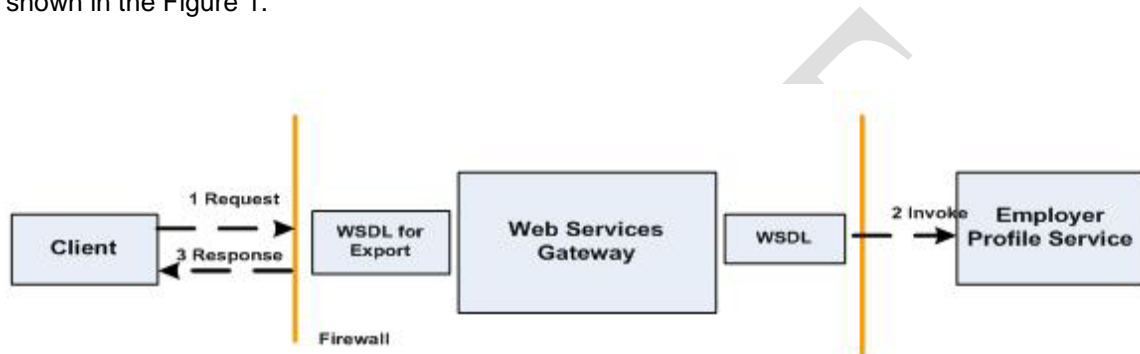

Figure 1: Department of Labor as a service provider and external agencies as service consumers

External agencies (service consumers) and the Department of Labor as a service provider can use JMS, SOAP/JMS, or SOAP/HTTP. Protocol conversion, like SOAP/HTTP to SOAP/JMS, occurs inside the Enterprise Service Bus (ESB.) External agencies (service consumers) using one protocol can invoke Department of Labor services that are exposed using a different protocol as shown in the Figure 2.
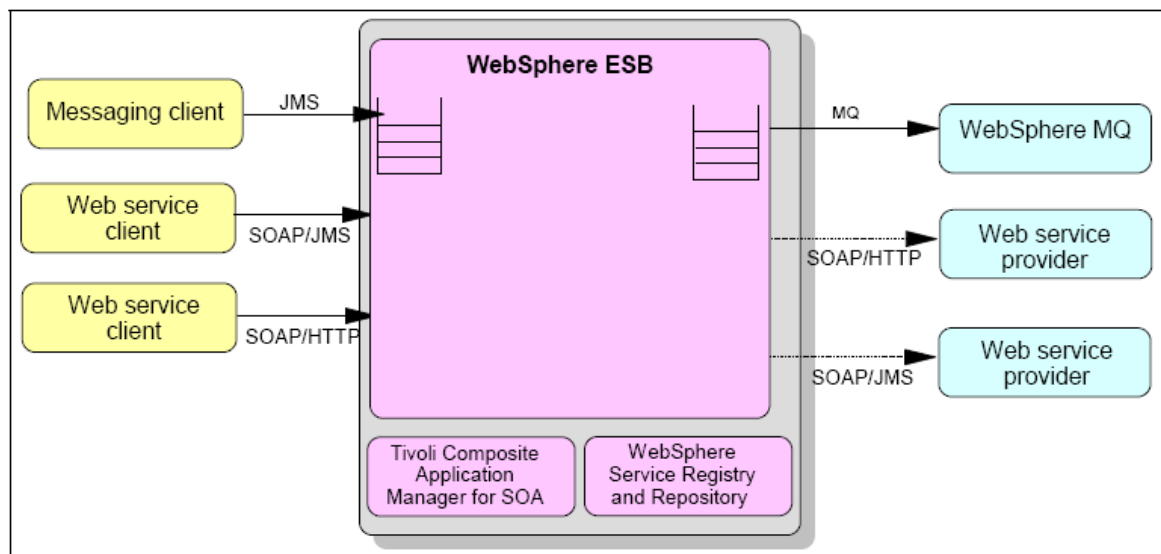

Figure 2: External agencies consumption of Department of Labor Services

Scenario handling the client request from the consumers:

Assume that the Department of Labor is providing an employer profile service that is deployed inside the firewall of the Department of Labor enterprise and the Department of Labor wants to share the service with external agencies and customers.

Followings are the steps to accomplish this.

Step 1. The Department of Labor will create the WSDL document.

Create the WSDL document that describes and invokes the DEPARTMENT OF LABOR employer profile service as shown in the sample below. As you can see, the SOAP address location is pointing to **myhost** that is inside the firewall.

```
<!-- *************************  Ports ***************************** -->

 <!-- ******** AddEmployer *********** -->
 <wsdl:portType name="Employer">
  <wsdl:operation name="AddEmployer">
        <wsdl:input message="emp:AddEmployerRequest" name="AddEmployerRequest"/>
        <wsdl:output message="emp:AddEmployerResponse"
name="AddEmployerResponse"/>
  </wsdl:operation>

  <!-- ******* ModifyEmployer ******** -->
   <wsdl:operation name="ModifyEmployer">
        <wsdl:input message="emp:ModifyEmployerRequest"
name="ModifyEmployerRequest"/>
        <wsdl:output message="emp:ModifyEmployerResponse"
name="ModifyEmployerResponse"/>
  </wsdl:operation>

  <!-- ******* GetEmployer ******** -->
   <wsdl:operation name="GetEmployer">
        <wsdl:input message="emp:GetEmployerRequest" name="GetEmployerRequest"/>
        <wsdl:output message="emp:GetEmployerResponse"
name="GetEmployerResponse"/>
  </wsdl:operation>

  <!-- ******* CheckEmployer ******** -->
   <wsdl:operation name="CheckEmployer">
        <wsdl:input message="emp:CheckEmployerRequest"
name="CheckEmployerRequest"/>
        <wsdl:output message="emp:CheckEmployerResponse"
name="CheckEmployerResponse"/>
  </wsdl:operation>

 </wsdl:portType>

 <!-- ***************************  Bindings ***************************** -->

 <wsdl:binding name="EmployerSOAP" type="emp:Employer">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```xml
  <!-- ******** AddEmployer *********** -->
  <wsdl:operation name="AddEmployer">
   <soap:operation soapAction="AddEmployer"/>
   <wsdl:input name="AddEmployerRequest">
    <soap:body parts="RequestValue " use="literal"/>
   </wsdl:input>
   <wsdl:output name="AddEmployerResponse">
    <soap:body parts="ResponseValue" use="literal"/>
   </wsdl:output>
  </wsdl:operation>

  <!-- ******* ModifyEmployer ******** -->
  <wsdl:operation name="ModifyEmployer">
   <soap:operation soapAction="ModifyEmployer"/>
   <wsdl:input name="ModifyEmployerRequest">
    <soap:body parts="RequestValue " use="literal"/>
   </wsdl:input>
   <wsdl:output name="ModifyEmployerResponse">
    <soap:body parts="ResponseValue " use="literal"/>
   </wsdl:output>
  </wsdl:operation>

  <!-- ******* ModifyEmployer ******** -->
  <wsdl:operation name="GetEmployer">
   <soap:operation soapAction="GetEmployer"/>
   <wsdl:input name="GetEmployerRequest">
    <soap:body parts="RequestValue " use="literal"/>
   </wsdl:input>
   <wsdl:output name="GetEmployerResponse">
    <soap:body parts="ResponseValue " use="literal"/>
   </wsdl:output>
  </wsdl:operation>

          <!-- ******* CheckEmployer ******** -->
  <wsdl:operation name="CheckEmployer">
   <soap:operation soapAction="CheckEmployer"/>
   <wsdl:input name="CheckEmployerRequest">
    <soap:body parts="RequestValue " use="literal"/>
   </wsdl:input>
   <wsdl:output name="CheckEmployerResponse">
    <soap:body parts="ResponseValue " use="literal"/>
   </wsdl:output>
  </wsdl:operation>

 </wsdl:binding>

<!-- ***************************  Services ****************************** -->

 <wsdl:service name="Employer">
  <wsdl:port binding="emp:EmployerSOAP" name="EmployerSOAP">
   <soap:address location="http://myhost:8080/soap/servlet/rpcrouter"/>
  </wsdl:port>
 </wsdl:service>

</wsdl:definitions>
```

Step2. Import the WSDL document into the Gateway

This simply requires supplying a name for the service to be hosted at the Gateway, selecting the SOAP/HTTP transport channel, and specifying the location of the Employer profile WSDL file. The Gateway will generate a new WSDL file that can be shared with your partners. The main difference between these two files is that the Gateway generated file will have Gateway as the service end-point, and the bindings and the port Type are separated into an interface WSDL file. Figure 2 shows the Gateway generated file with the appropriate changes.

```
<definitions targetNamespace=...>
  <service name=" Employer ">
    <port name=" EmployerSOAPBindingPort"
        binding="interface: Employer SOAPBinding">
      <soap:address location="http://gatewayhost:80/wsgwsoap1/soaprpcrouter"/>
    </port>
  </service>
</definitions>
```

Figure 3: Gateway Employer profile service implementation WSDL

Step 3. Share the WSDL document to requestors outside the firewall

This can be done by requesting the Gateway to publish the service to WSRR, in which case service requestors may obtain it by using WSRR lookups.

Step 4. Creating the web service clients at the service consumer end.
The next steps need to be performed by the consumers of Department of Labor services. Following are some examples to create the web service client in different technologies.

Creating the .NET Client:

The WSDL file is the key for a service consumer (client) to consume web services. It describes the particular web service interface. Service consumer (client) programs use a proxy to interact with the web services. It is the proxy that shields all SOAP messaging from client programs.

IBM tooling supports both "rpc" and "document" WSDL binding styles. However, the Microsoft WSDL tool in the .NET Framework SDK doesn't accept this WSDL "rpc" style; it only accepts the SOAP "document" binding style with the "literal" default encoding.

Since the Department of Labor standard is to use the "document" style in WSDL, the Department of Labor has no issue with external agencies/consumers using .NET technology as a web services client.

```
          <wsdl:output name="addPetitionToEmployerClaimantResponse"
            message="psr:AddPetitionToEmployerClaimantResponse" />
        </wsdl:operation>
    </wsdl:portType>
    <!-- ******************************** Bindings ******************************** -->
  - <wsdl:binding name="PetitionServiceSoapBinding" type="psr:Petition">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
      <!-- ******** addPetition *********** -->
    - <wsdl:operation name="addPetition">
        <soap:operation soapAction="" />
      - <wsdl:input name="addPetitionRequest">
          <soap:body use="literal" parts="RequestValue" />
        </wsdl:input>
      - <wsdl:output name="addPetitionResponse">
          <soap:body use="literal" parts="ResponseValue" />
        </wsdl:output>
      </wsdl:operation>
      <!-- ******** modifyPetition *********** -->
    - <wsdl:operation name="modifyPetition">
        <soap:operation soapAction="" />
      - <wsdl:input name="modifyPetitionRequest">
          <soap:body use="literal" parts="RequestValue" />
        </wsdl:input>
      - <wsdl:output name="modifyPetitionResponse">
          <soap:body use="literal" parts="ResponseValue" />
        </wsdl:output>
      </wsdl:operation>
      <!-- ******** getPetitionList *********** -->
    - <wsdl:operation name="getPetitionList">
        <soap:operation soapAction="" />
      + <wsdl:input name="getPetitionListRequest">
      - <wsdl:output name="getPetitionListResponse">
          <soap:body use="literal" parts="ResponseValue" />
        </wsdl:output>
      </wsdl:operation>
```

Figure 4: A Sample of a Department of Labor WSDL file with the SOAP "document" binding style that could be shared with external agencies/consumers

Steps to create the .NET Client for the service-

1. Generate a C# WSDL proxy from the given WSDL file:

The .NET service consumers (clients) needing to consume the Department of Labor services will need to create a web service client that will be deployed on their own proxy server. .NET SDK simplifies the process of creating web service clients by providing the Web Services Description Language (wsdl.exe) utility. This utility generates proxy source code for an existing web service.

The wsdl.exe utility takes a WSDL file as input. This C# proxy source file represents the proxy class web service that clients can compile against. It contains a proxy class that derives from the System.Web.Services.Protocols.SoapHttpClientProtocol class.

2. **Compile the C# proxy class as a DLL Library**

3. Compile the C# source file into a dynamic link library (DLL) and then add a reference to this DLL to any project you want to create. **Create a Visual C# - ASP.NET Web Service project**

Create a new Visual C# ASP.NET Web Service project. Create a Reference to the dll library.

4. **Develop the Client.cs class file**

5. **Build the Project Files**

Build the files that make up the project.

6. **Run the client**

Creating the Java Client to access the services:

Any Java based IDE can generate the Web Service client proxy from the WSDL file. The Department of Labor can use WebSphere Integration Developer (WID), Rational Application Developer (RAD) or Rational Software Architect (RSA) to generate the client proxy.

Steps to generate the Web service Java client proxy

1. To open the Web Service Client wizard, right-click the **WSDL** document in the Project Explorer view and select **Web Services => Generate Client**.
2. In the Web Services dialog, shown in Figure 5, verify that the **Service definition** field points to the **WSDL** and that **Java Proxy** is selected in the **Client type** field, then click **Next** to proceed.
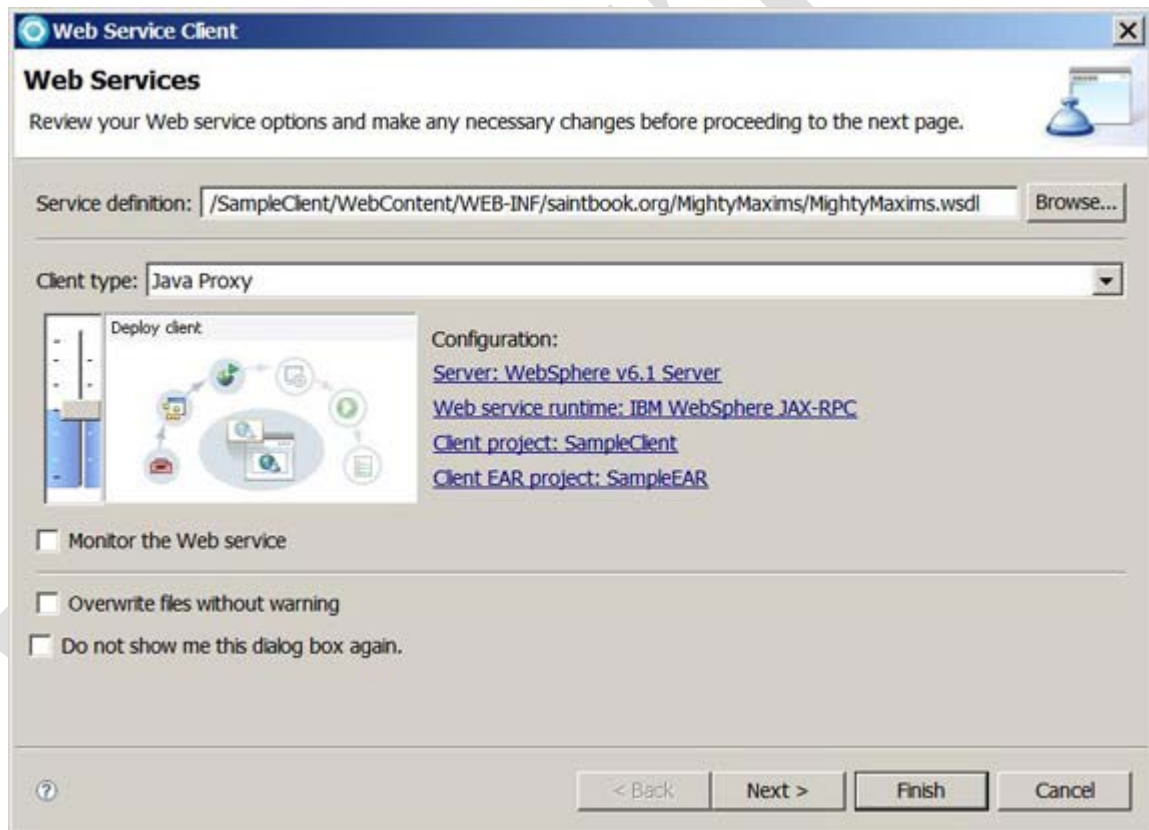
Figure 5: Web Service Client dialog

3. In the **Web Service Proxy Page** dialog, shown in Figure 6, verify that the **Output folder** field points to **/SampleClient/src** and that **no security** is selected in the **Security configuration** field, then click **Finish**.
4. Click **Yes to All** when prompted to enable automatic file overwriting of the Web deployment descriptor /SampleClient/WebContent/WEB-INF/web.xml.

5. **Web Service Proxy Page**
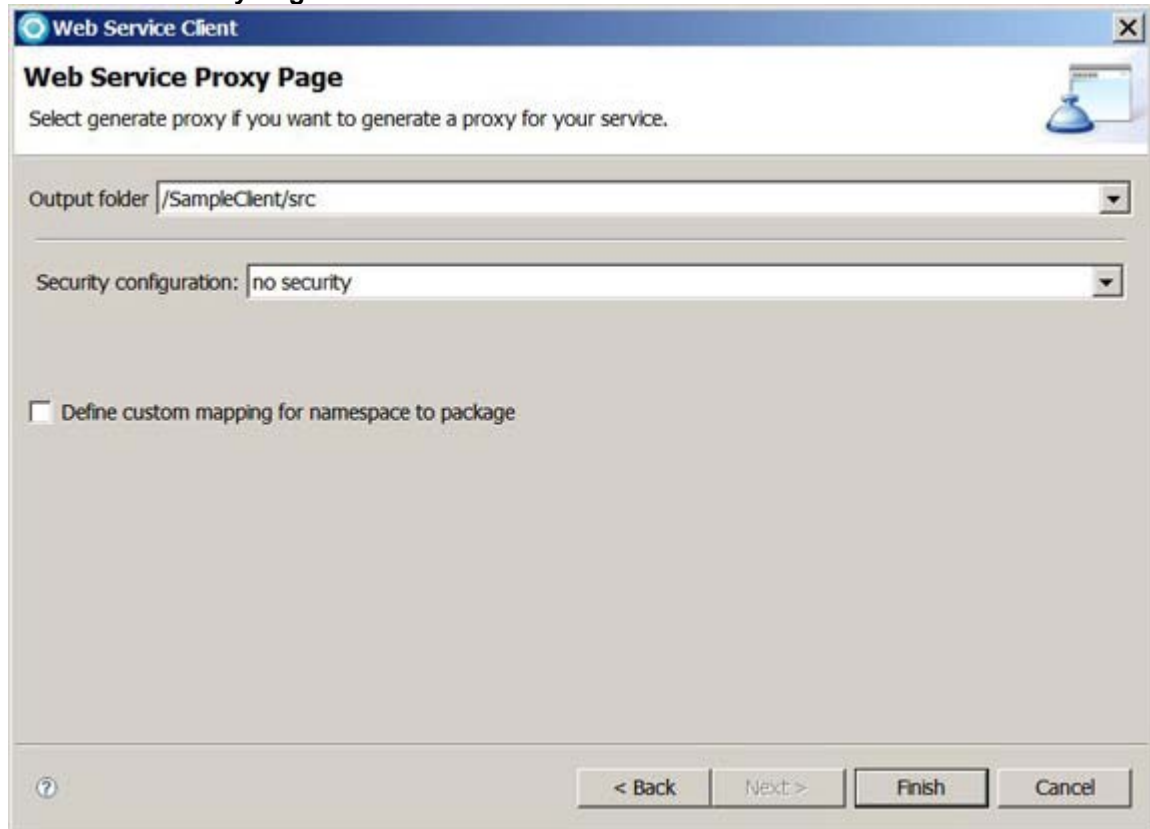
Figure 6: Web Service Proxy Page

6. To verify that the proxy class was generated successfully, navigate to
   SampleClient/JavaResources/Proxy.java in the Project Explorer view, as shown in
   Figure 7.
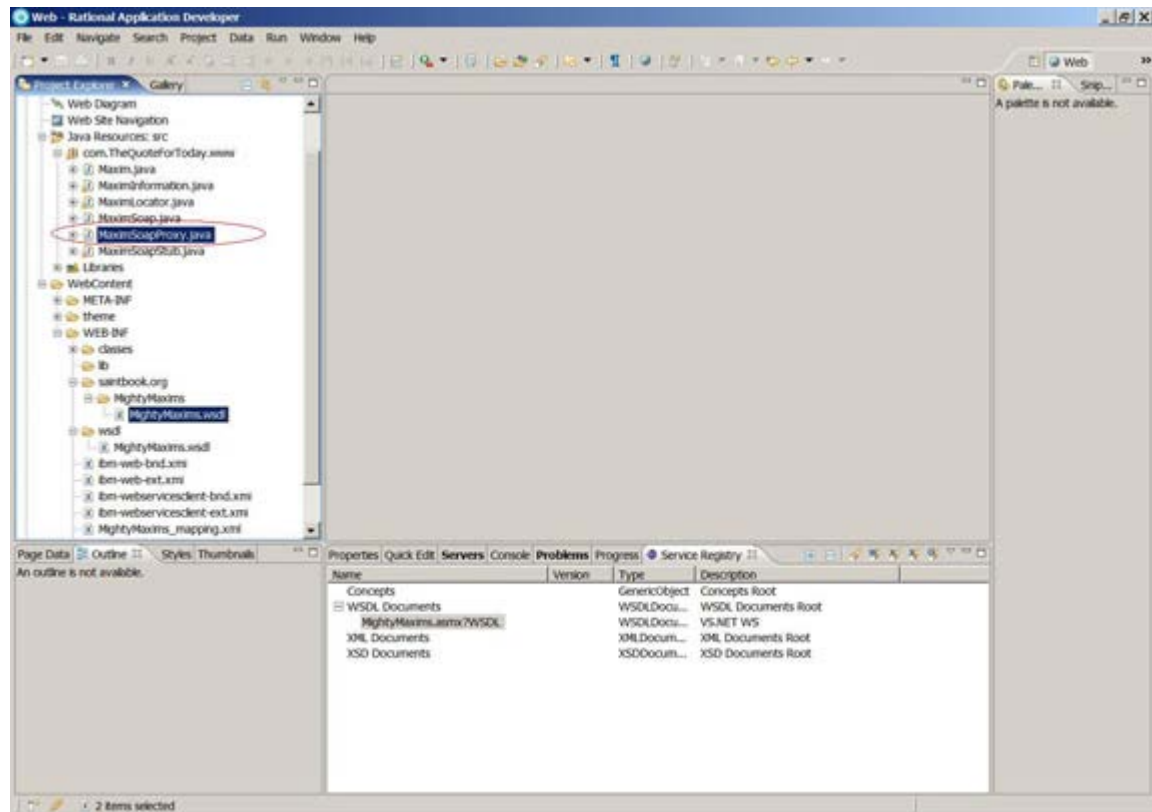
Figure 7: Java bean proxy

These are just examples to create the Clients in .NET and java technology. Similarly, the Client
can be generated in any technology and platform. Java Client classes can also be generated by
other Java integrated development environments.

Step 5. Service requestors send a SOAP request to the Gateway

The service requestors will send the SOAP request to the Gateway, which will invoke the service
inside the firewall as shown in Figure 8. The first layer of DataPower will handle the
authentication/authorization while the routing of the services to the WPS, WMB or WAS will be
done by DataPower inside the second layer. The actual endpoint of the service might be
anywhere on WPS, WMB or WAS.
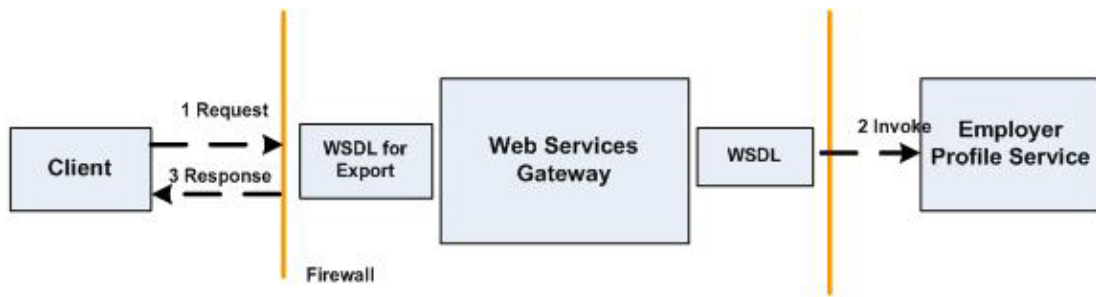
Figure 8: Inbound request through the Gateway

The Department of Labor as a service consumer and external agencies as service providers:

The Department of Labor will import the WSDL document from the service provider outside the firewall into the Gateway, which generates a new WSDL document that will be used by internal service requestors. Figure 9 explains the flow of outbound requests.
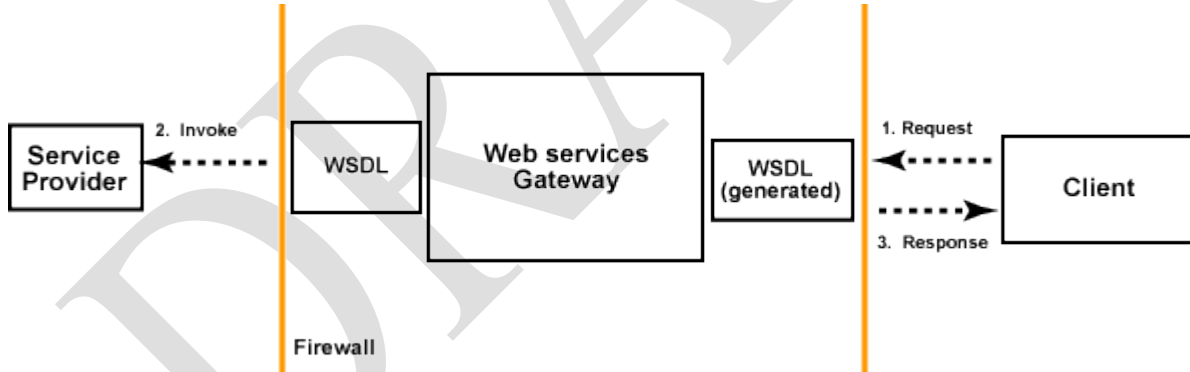
Figure 9: Outbound request through the Gateway

The Department of Labor will generate the Java client (as explained earlier) from the WSDL file (provided by other agencies) to consume services.

Service Governance with WSRR (Web Sphere Service Registry and Repository)

(1) External requestors will use the reference endpoint (not the actual) of the service through WSRR.

(2) The Department of Labor has already published the list of public services to the outside world without giving the service endpoint to the viewer with "Limited Reader" perspective. Potential consumers can have a look of the services published by the Department of Labor.

Here is the CIO share page link to go to the WSRR

http://www.labor.ny.gov/cioshares/soa.shtm

Please click the "WSRR Guest Login" to see the list of services published by the Department of Labor.

Figure 10: WSRR Guest Login

(3) The viewer (with "Limited Reader" role in the WSRR) (potential subscriber) can observe the service capability and details through WSRR. The viewer can be anyone who wants to access these published services. The viewer won't be able to see the service endpoint information without the registration process.

(4)   If viewers are interested in consuming any service developed by the Department of Labor, they can send an Email to the Department of Labor.

By Email:

NYSDOLServiceAdmin@labor.ny.gov

Followings are the future tasks Department of Labor has planned based on the usage pattern by different agencies and other potential users.

- The viewer (potential subscriber) will register with the WSRR system through the registration process, which runs on the process server. It is a long running process that involves human tasks. Requests for registration will be submitted to the Department of Labor service administrator, and he or she will then either approve or deny these requests.
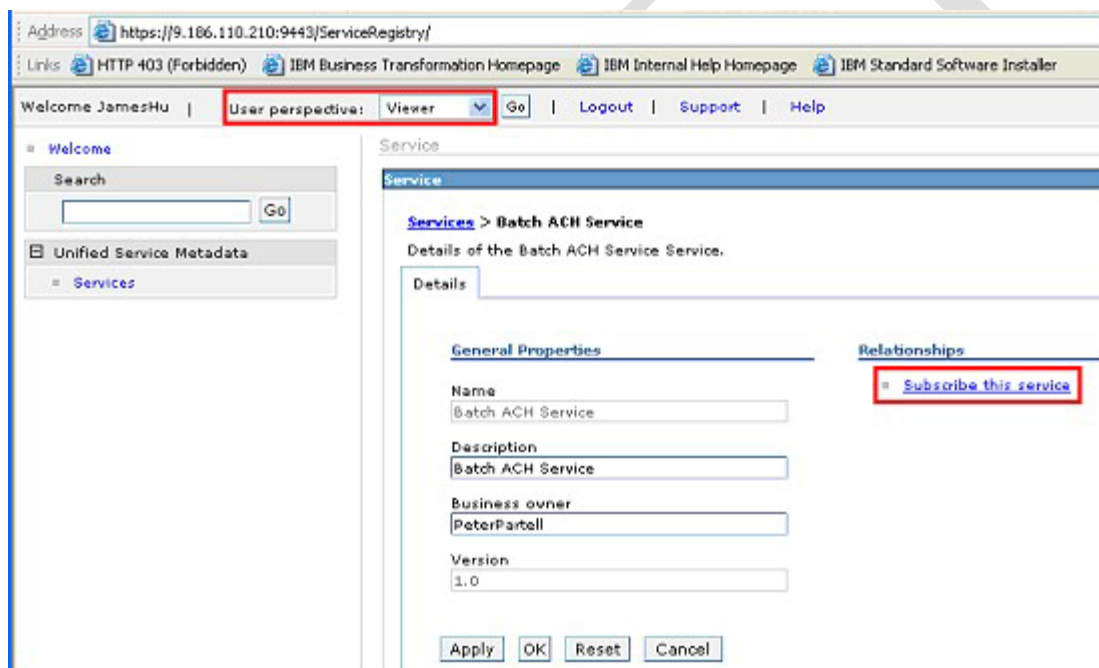

Figure 11: WSRR Guest Login

- After approving the viewer's request, the Department of Labor service administrator must be concerned with the following steps.
- Selecting the right **end point of the service** (there might be several end points for the same service) based on the cost, availability, and various other factors defined in the contract.
- Setting the authentication and authorization access (userid, password etc to LDAP and WSRR)
- EMAIL notification to the viewer (through notification plug-in) and so on
- Once the viewer is registered with Department of Labor, his/her user perspective will change from the "viewer" to the "**SUBSCRIBER**". Now, the subscriber can see the endpoint lookup, published WSDL and policies etc through WSRR. If there is any change

in the capability of the existing service, all registered subscribers of the existing service can be notified by email using WSRR.

- WSRR will also have the actual service end point references registered in it. DataPower, WPS, and WMB can refer to this service end point through mediation. The actual service end points will not be directly exposed to the outside service clients.
- WSRR will be used to manage XSD, WSDL and service policies etc. during various phases of the service life cycle.

- Department of Labor's long term vision is to share the infrastructure with other agencies. Now, different agencies will be the "service producers" NOT just "the consumers". We want to share the same WSRR infrastructure created for the Department of Labor with other agencies. Department of Labor is going to do the following tasks:

  - Create a separate perspective and profile with its own logo, look and feel (based on the agency standard)
  - Only those services which are developed by the specific agency will be shown here.