RESTful SOA in the real world

Sastry Malladi

Distinguished Architect.

eBay, Inc.

Agenda

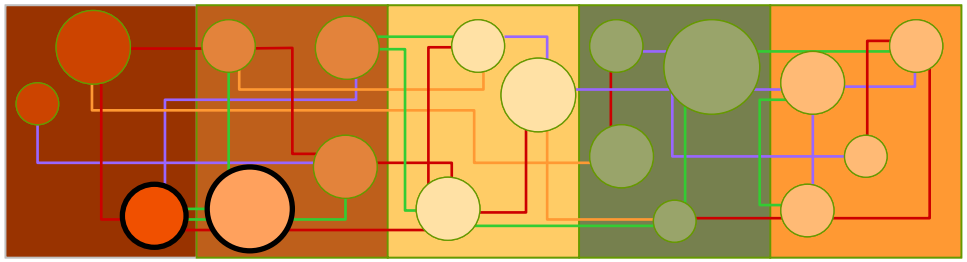
- Putting SOA and REST in perspective
- Case study : RESTful SOA at eBay
- Patterns for REST URL mapping of SOA services
- Demo
- Summary

Putting SOA and REST in perspective

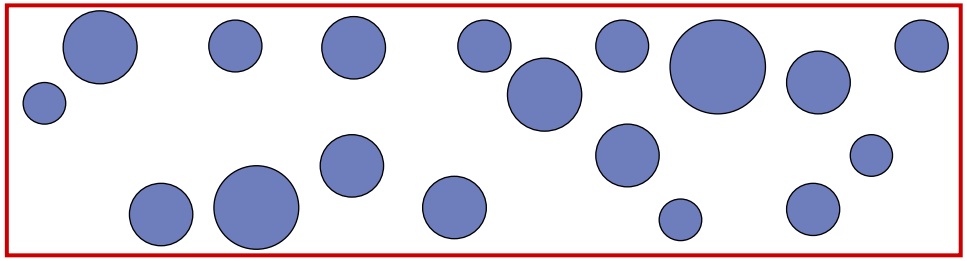
- SOA is an architectural style and SOA services can be accessed in multiple ways
- SOA services can be accessed either via WS-* style or via REST style
  - WS-* style here just refers to SOAP and bare minimum WS-* stuff that is required
- They aren't mutually exclusive
- They both have their respective use cases
  - A thick programmatic client with lots of auto-generated tooling
  - A browser based or thin HTTP client
- It is not desirable to implement the same business logic twice – once for WS style access and once for REST style access
- Note that we are not talking about Service Orientation Vs Resource Orientation - Topic is about giving RESTful access to SOA Services
- So how do you build such SOA services ?

SOA

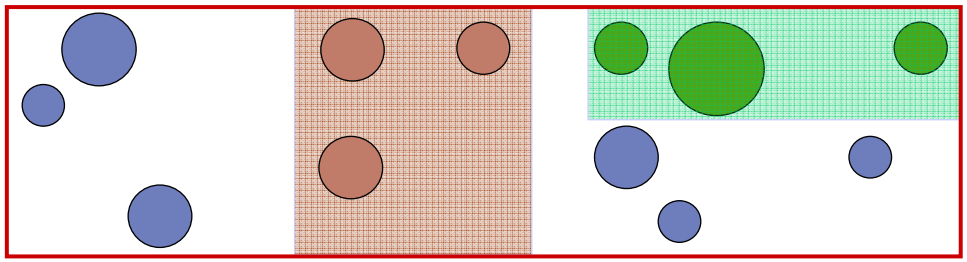
SOA is an *Architecture evolution*, not a *Technology revolution*



SOA is an architecture to move from brittle, hardwired, application silos that inhibit change…



… to shared, reusable, business and application services…



… which eliminates application redundancy and complexity, and enable Business Agility, Innovation and Operational Excellence.

RESTful SOA

REST

- Resource oriented
- Resources are uniformly represented through a URI (name and a location)
- Interactions with the resource are stateless
- Maps to HTTP GET, POST. PUT and DELETE verbs on the resource.
- Different resource representations : XML, RSS, Atom, JSON, ..
- Security : At the transport level, not message level (e.g OAuth for authorization)

RESTFul SOA

- Interacting and manipulating resources backed by a SOA service, typically through a mapping layer
- It is not direct resource manipulation, but resource manipulation through SOA service operations
- As such, if the service interface is not appropriately modeled, accessing through REST style isn't going to be pretty
- Different output data formats : XML, RSS, Atom, JSON, ..
- Security : At the transport level (e.g. OAuth for authorization)

Numerous industry perspectives on REST

- How should a RESTful service be described ?
  - Just text documentation - consumable by humans only (i.e., not tools)
  - WADL - (Web Application Description Language) – How many description languages does the consumer need to use for the same service ?
  - Use WSDL itself - HTTP bindings in WSDL and use appropriate tooling to generate code.
- Real world industry trends
  - Same service accessed by many protocols, data formats, styles (browser, programs)
  - Reduced investments (development costs - productivity, better performance and scalability)
  - Enterprises typically have existing services, everything is not re-built from ground up – Need a way to leverage that.
  - Don't necessarily care about religious arguments about what is REST and what is not. “Just give me the data I want in the format I want using a standard protocol”
- WADL Vs WSDL
  - Request/responses are both described in schema
  - WADL is resource centric, WSDL is service centric
  - Security etc, is not covered in WADL, but on the other hand, WSDL is more complex

WADL

```
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

 <resources base="http://www.somecompany.com/mySearchService/V1/">
   <resource path="itemSearch">
      <method name="GET" id="search">
         <request>
            <param name="keyword" type="xsd:string"
               style="query" required="true"/>
          …
         </request>

         <response status="200">
           <representation mediaType="application/json"
                element="tns:ResultSet"/>
         </response>
         ….
      </method>
    </resource>
  </resources>
```

WSDL 2.0 HTTP binding

```
<description …
  <types ../>
  <interface … />
  <binding name=“mySearchServiceHttpBinding"
     interface="tns:mySearchServiceInterface"
     type=http://www.w3.org/ns/wsdl/http whttp:methodDefault="GET">
     <operation ref="tns:searchOperation"
           whttp:location=“itemSearch/
           whttp:method=GET"
           whttp:inputSerialization=“XML”
           whttp:outputSerialization=“JSON”/>
   </binding>
   <service … />
</description >
```

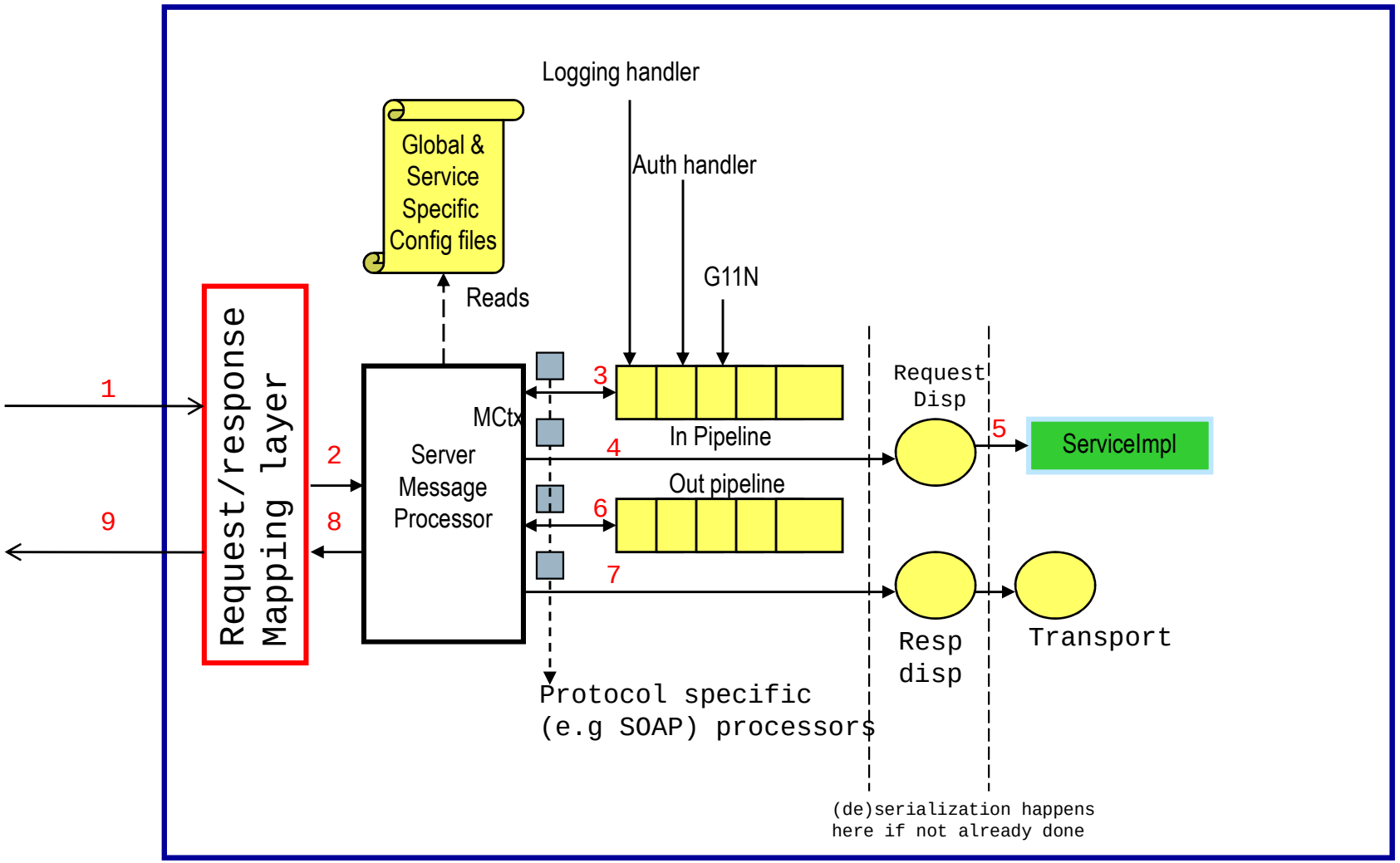
Security - Typical scenarios

- Browser
  - For anyone registering, issue a Access Key and Access Secret.
  - When connecting to a REST URL in the browser, specify two query parameters.
    - the access key, and two a signature – which is calculated using Access Secret of the message.
    - On the server side, the Access Secret corresponding to Access Key is retrieved, the signature is calculated and compared
- Application
  - While invoking REST URL specify a “redirect URL” query parameter
  - On the server backend, user is redirected to a sign-in page, and upon successful login, redirect back to the user specified redirect-URL passing in a “verification string” and a security server URL
  - The application then invokes a security server URL passing the verification string, and get back an OAuth access token.
  - Then simply make subsequent REST calls with the OAuth access token in query parameters.

Case Study : Restful SOA @eBay

- Built a highly optimized SOA framework (Service Container) that
  - Allows description of the service using WSDL
    - SOAP as well as Http/REST bindings
  - Implement the service (business logic) once
  - Generate code for programmatic access via SOAP or HTTP/REST
  - Generate REST URL mapping for direct browser access
  - Out of the box support for JSON, NV, XML, RSS, ATOM
  - Low latency and overhead (total overhead under 5ms)
  - Local binding (deployment time option)
  - Integrated and built-in monitoring
  - Policy based resource modeling and protection (Authn, AuthZ, RL)
  - Service and consumer decoupling via ESB
  - Integrated tooling - Developer and operational

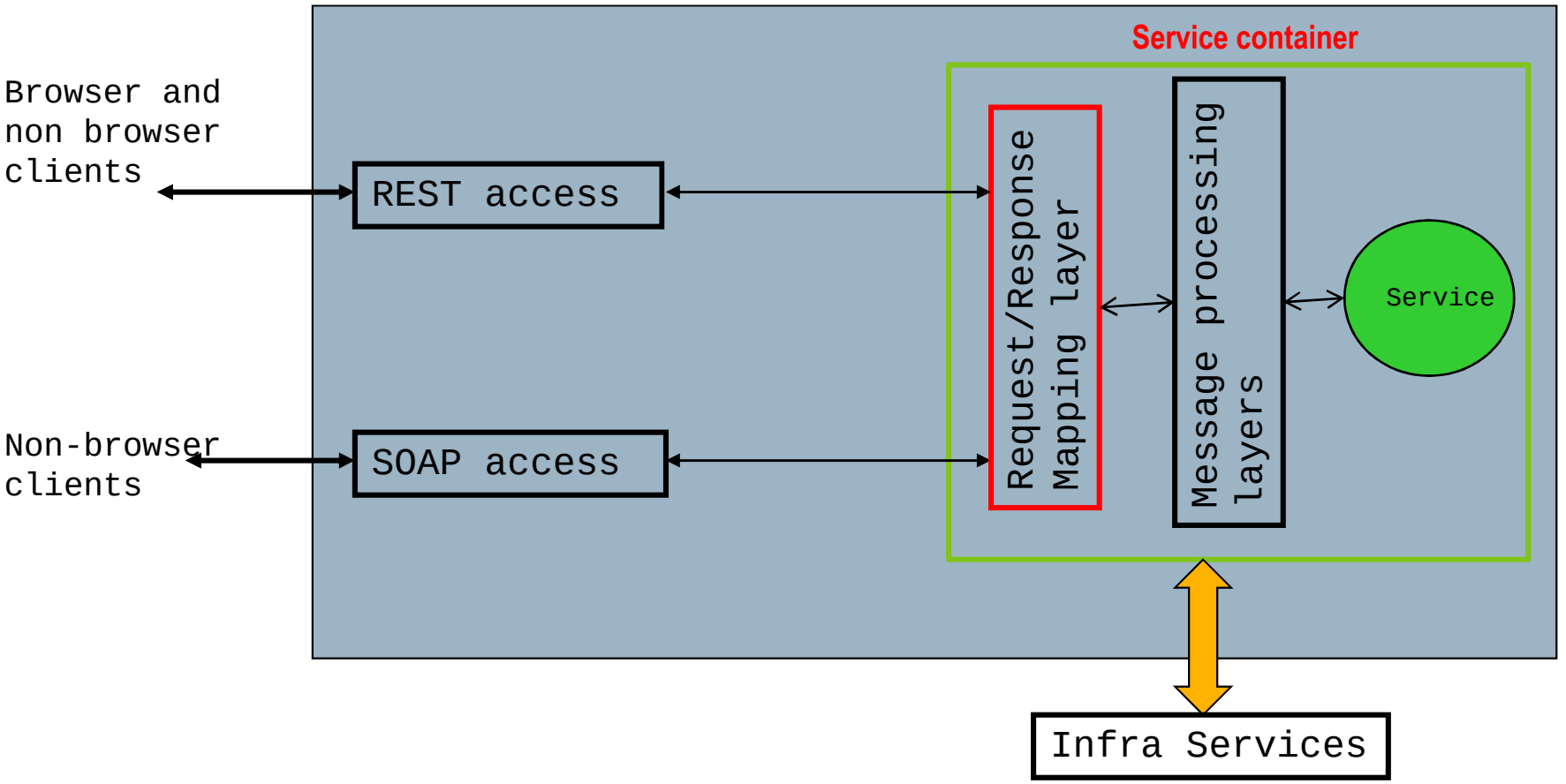
Pipeline architecture – Service Container



Patterns for REST URL mapping

- Mapping natively at Service Container level with the combination of WSDL HTTP bindings
  - Config options for request parameters and headers
- Mapping at a layer in front of the Service Container layer (e.g ESB)
  - For both request mapping and response transformations
- Combination of the above two
  - Basic mapping at Service Container layer
  - Additional mappings at ESB tier, including output transformations (Atom, RSS, ..)
- Through Atom Adaptor services
- Considerations
  - Rate Limiting (Traffic control and throttling)
  - Security (authentication)
  - Monitoring
  - Resource versioning

Mapping layer at Service Container level

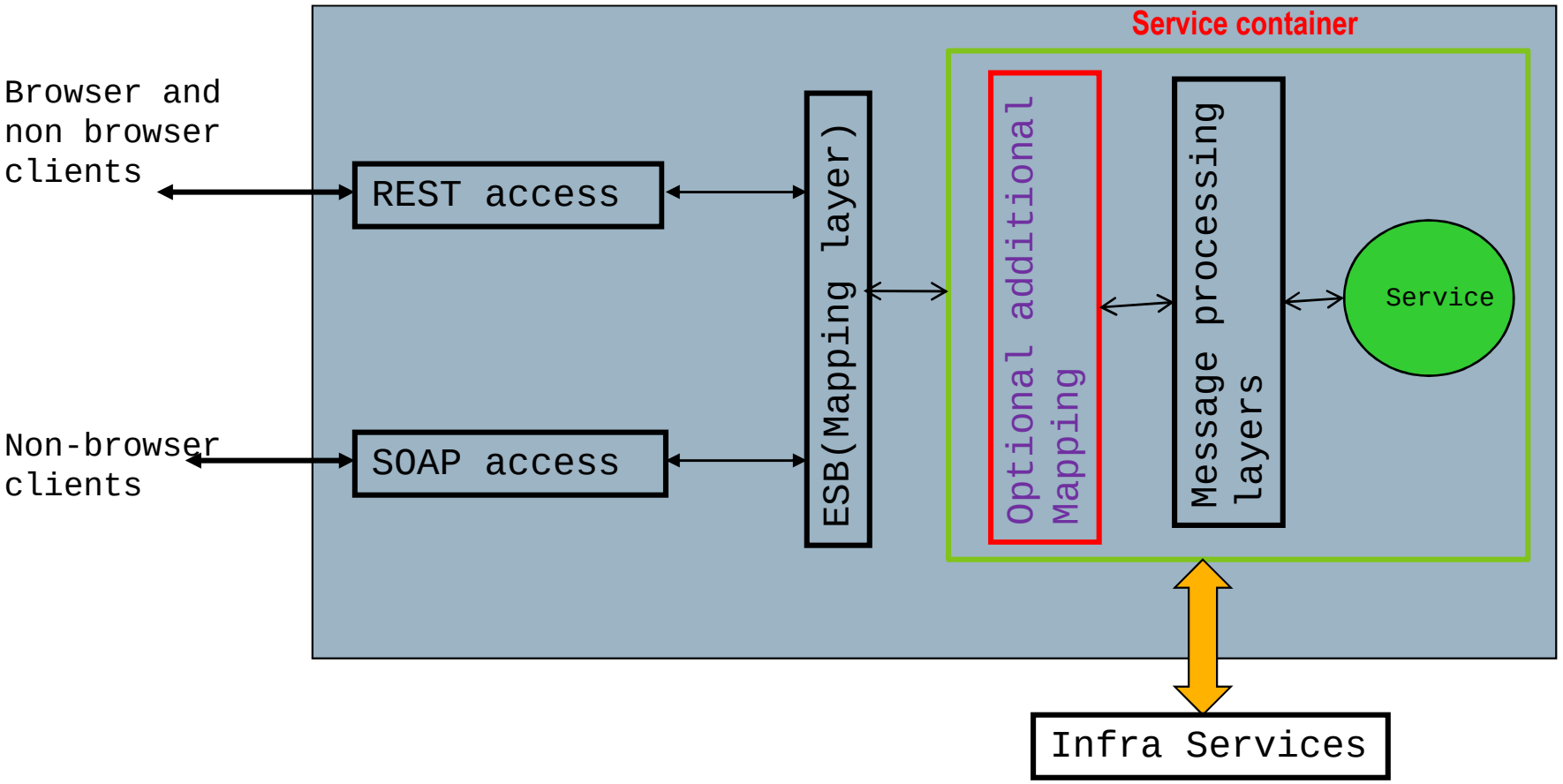


Mapping at Service container level : Service config file snippet

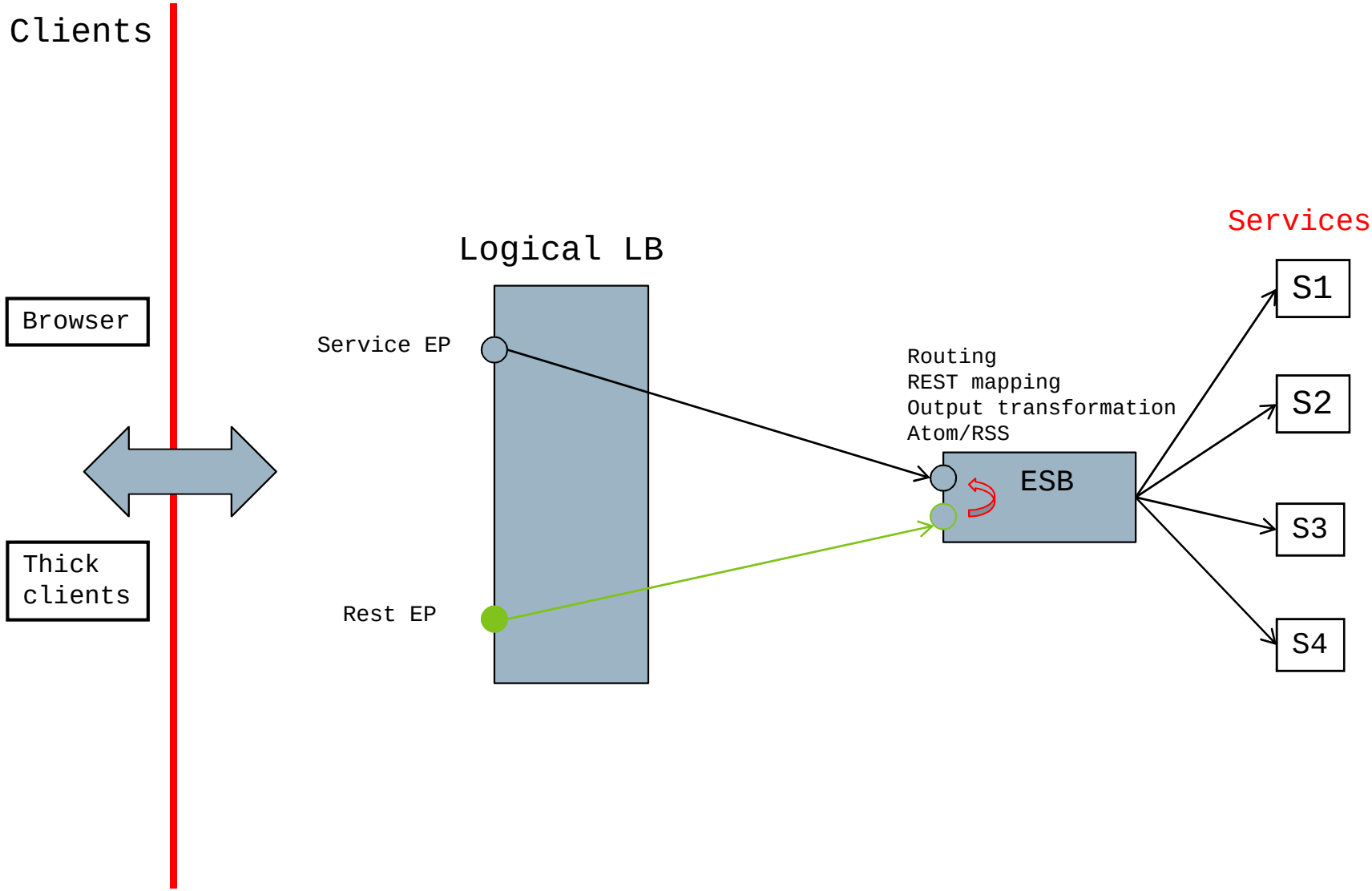
```
<provider-options>

   <header-mapping-options>
      <option name="X-EBAY-SOA-OPERATION-NAME">path[2]</option>
      <option name="X-EBAY-SOA-RESPONSE-DATA-FORMAT">query[format]</option>
   </header-mapping-options>

   <operation-mapping-options>
       <operation name="getCatalog">browse</operation>
       <operation name="updateCatalog">update</operation
   </operation-mapping-options>

   <request-params-mapping>
       <operation name="getCatalog">
            <option name="catalogID">path[3]</option>
       </operation>
   </request-params-mapping>

</provider-options>
```

- http://host:port/CatalogService/**browse/books**
- http://host:port/CatalogService/**browse/books?format=json**

Mapping at a layer in front of Service Container (e.g. ESB)



ESB tier


Clients
Browser
Thick clients
Logical LB
Service EP
Rest EP
Routing
REST mapping
Output transformation
Atom/RSS
ESB
Services
S1
S2
S3
S4

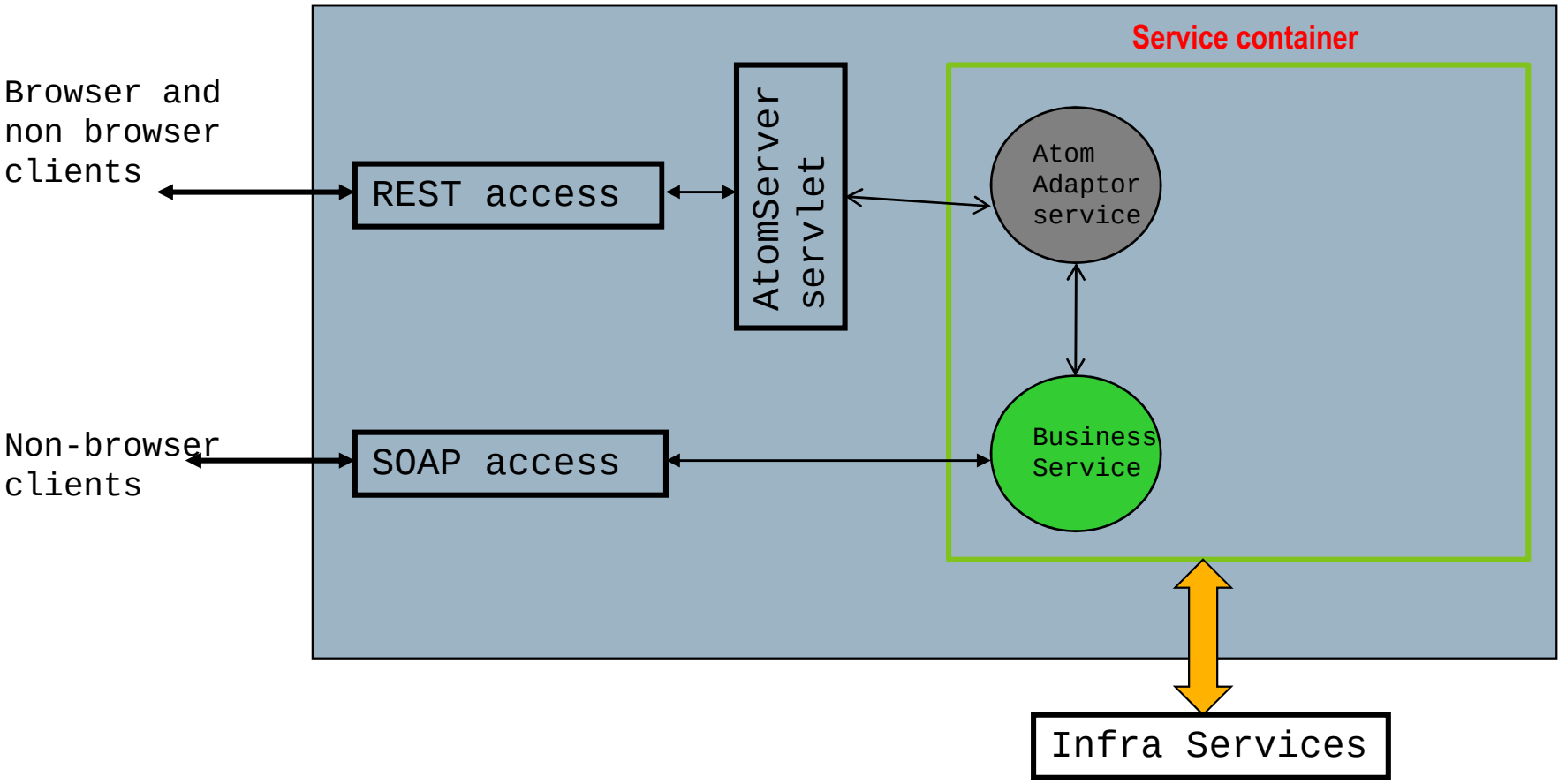
Mapping at an ESB tier : Configuration file snippet

- No WSDL Request/Response structure knowledge at ESB tier
- Mapping is dynamic and context sensitive (i.e, not a static 1:1 mapping
- Reserved path elements and parameters (e.g Version)

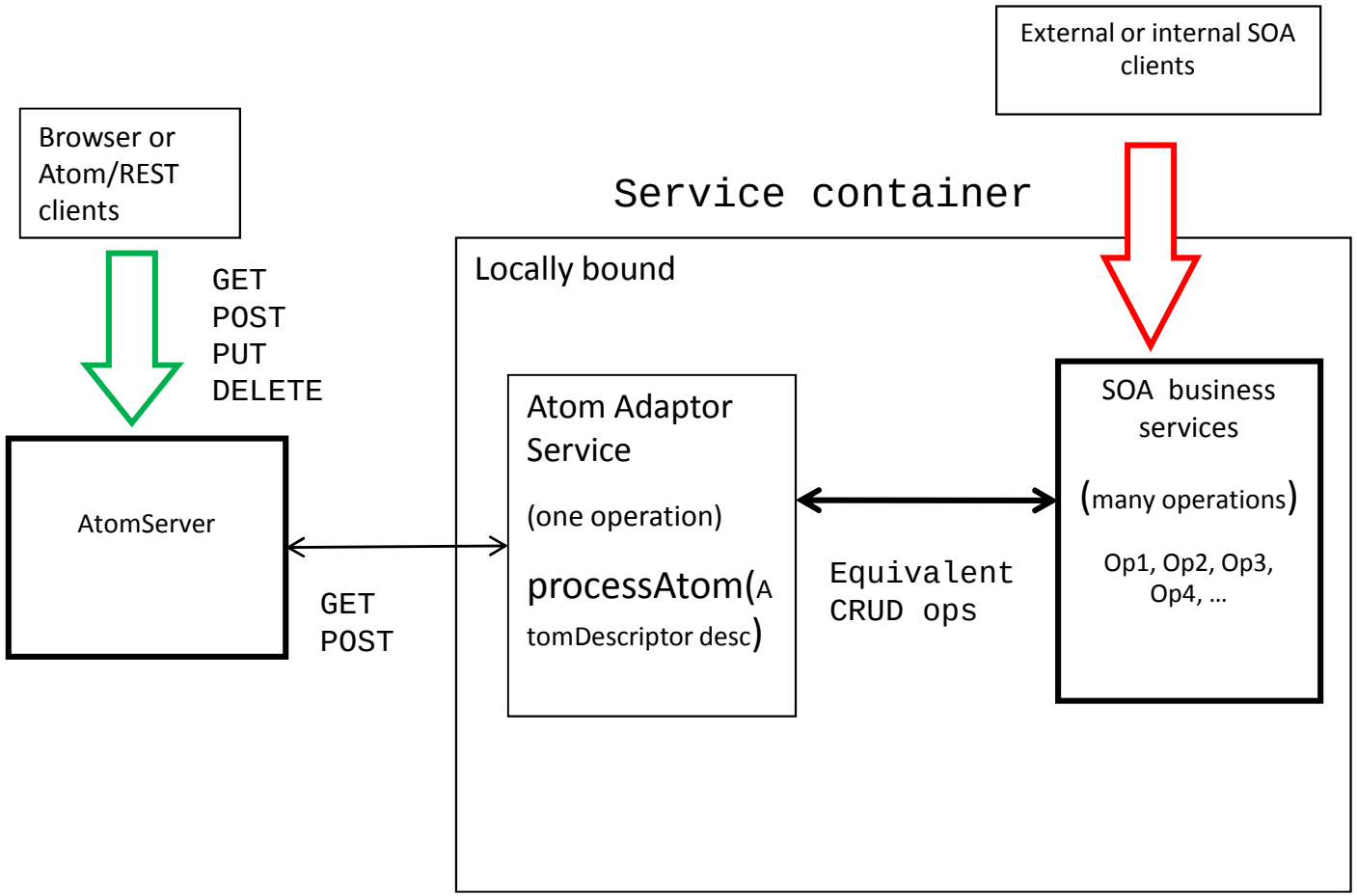
```
<mapping xmlns = "http://www.ebay.com/soa/">
   <url-mapping url = “catalogsvc/browse/">
      <request-params-mapping service = “CatalogService” >
          <operation name=“getCatalog" request=“getCatalogRequest" >
             <option name=“catalogID”>path[2] </option>
             <option name=”Version” alias=“Ver” style=“query”
                      default=”V1”
                      optional=”true” >path[3] </option>
          </operation>
   </url-mapping>
</mapping>
```

- http://host:port/CatalogService/**browse/books**
- http://host:port/CatalogService/**browse/books?Ver=V2**
- http://host:port/CatalogService/**browse/books/V2**
- http://host:port/CatalogService/**browse/books?format=json**

Mapping through an Atom Adaptor



Mapping through Atom Adaptor services – Details



Demo

- Basic SOA service creation – CatalogService – getCatalog, updateCatalog
- Client creation
- Invoke in local mode and remote binding mode, basically programmatic SOAP
- Invoke in browser - change data formats
  - Define header path mapping for JSON, NV
  - demo GET/POST

Summary

- SOA is an Architectural style and principles and doesn't conflict or contradict REST approach – They are complimentary
- Restful access to SOA service is about giving Resource oriented access to the data behind the SOA service, and is not necessarily about changing service orientation to resource orientation
- There are multiple approaches to describing REST access to a service, but the approach that eBay followed is a combination of using the WSDL HTTP bindings and a URL mapping layer
- There are multiple patterns for REST URL mapping to SOA services and typically a combination of those patterns is always used
- It is desirable to implement a business service once, but give both SOAP and REST access to the same service.
- If the SOA service in question doesn't have a proper design and modeling of the interface, just defining the REST URL mappings to make it look like resource isn't going to be pretty !