

Oracle Service Bus

Situation

A service oriented architecture must be flexible for changing interfaces, transport protocols and server locations - service clients have to be decoupled from their implementation. Therefore a *configuration driven*, stateless and throughput optimized solution is sought. Once such a central hub is in place, it can implement security and provide monitoring data of the service usage.

WebLogic administrators are sometimes asked to take care of an Oracle Service Bus domain. Now, although Oracle Service Bus is based on the same principles as WebLogic (such as domains, managed servers, node manager, etc.), there are some important differences.

Oracle Service Bus involves a learning curve - not only because of the new technology but also because there is an overwhelming amount of marketing and technical documents.

Oracle Service Bus Primer

This recipe explains the usage and peculiarities of Oracle Service Bus in an easy yet exact way. This recipe is based on an introduction text published in my Middleware and Cloud Computing book - in case it has a familiar ring.

Product History and Evolution

Oracle Service Bus is based on the former BEA service bus (known as AquaLogic Service Bus or ALSB during BEA times). When Oracle acquired BEA, Oracle already had its own service bus which was renamed Oracle Enterprise Service Bus (OESB).

The remainder of this recipe deals with Oracle Service Bus (OSB).

Positioning

Oracle Service Bus is technically independent of the Oracle SOA Suite (but included within the SOA Suite license). Therefore, it is the better choice if you are looking for an external, stand-alone service virtualization. JCA adapters from Oracle SOA Suite can be deployed with OSB since the JCA framework is implemented by WebLogic server.

Oracle Service Bus implements the layer that separates service implementation from service orchestration. Service Bus itself acts as a *service virtualization layer*.

Usage Scenario

Imagine you are running a service which is implemented as a web service and there are several dozens of Java based clients throughout the country accessing this web service.

To improve the system's performance, one day you will have to migrate the server to a different server. Simultaneously, business likes to update a particular service from version 1.0 to version 2.0 with new functionality and apply some changes in the interface.

Now updating all the clients can be a challenging task.

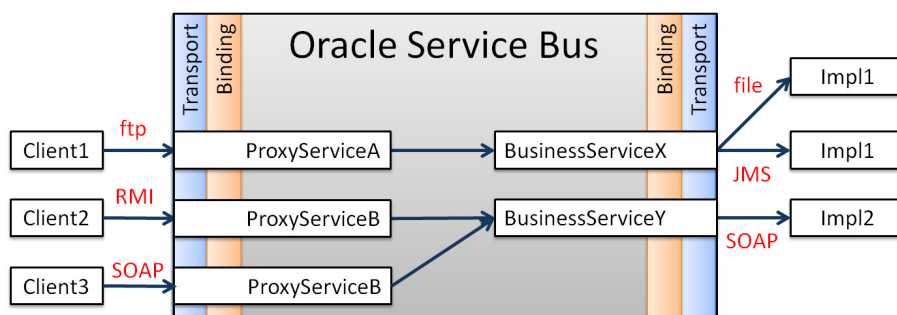
With the SOAP based web services, the location of the service is in the <port> section of the WSDL. The <types> section defines the data types of the interface using XML schema tags.

When using the destination from the WSDL, most clients will be pointing to the former location of the server and using an old interface version.

With Oracle Service Bus you can easily adapt things to accommodate such changes, it acts like a "logical switch". With an SOA in place, the clients will talk to the service bus instead of directly talking to the service implementation.

Proxy and Business Services

The service bus uses a proxy service as an endpoint for incoming requests. A transport layer abstracts the proxy from the transport protocol. So, the functionality of the proxy service is independent of the incoming transport protocol:



Transformations, validation etc. happen within the proxy service.

After the transport layer, the message passes through a binding layer, which transforms the message into a canonical, logical representation. The message parts, such as payload or header, will be accessible via variables such as \$body or \$header.

The outgoing endpoint in Oracle Service Bus is called business service. The naming is unfortunately somewhat misleading, since the business service is not actually the service implementation itself: the business service is just a piece of configuration within Oracle Service Bus (but separate from a proxy service), which defines the interface and location of the service implementation.

Location Transparency

The proxy service defines which business service the message is routed. When you change the routing to another business service, another service implementation will be called. This is an easy but elegant way to deal with the issue of location transparency, as per our example when the address or the port of the server changes.

Content Based Routing

Actually, there is much more that can be done in a proxy service. Inside a proxy service, you can define a message flow to include if-then decisions, loops, alerts, Java call-outs and message transformations. Referring back to our example, you can check a message for the content of a version element in the header or an XML <version> in the message itself. Based on this expression you can decide to route the business service representing the implementation v1.0 or v2.0. This is how you configure a “logical switch” for our example. The technical name for it is ‘content-based routing’.

Compared to, let’s say, a message selector in JMS, the beauty of OSB is that content based routing will work with *all* transport protocols: it doesn’t matter if the proxy service is SOAP based, file, JMS or email. Content based routing can be based on header or payload data.

Message Transformation and System Integration

Now, imagine you need to fork the incoming web service call to another system using a different protocol. You therefore have to change the incoming SOAP requests using an XQuery or XSLT transformation in the proxy service of the service bus, to match the format of the new system.

Protocols








Oracle Service Bus supports a number of different transport protocols. Apart from the list of standard mechanisms, like HTTP, email, file, ftp, JMS, EJB 3 etc., there are a number of Oracle product-specific protocols supported:

- ✓ Local transport for proxy services call other proxy services directly
- ✓ SB transports to invoke another service bus proxy using RMI
- ✓ SOA-DIRECT protocol for SOA Suite interaction
- ✓ DSP to interact with Oracle Data Services Platform
- ✓ JPD for calling Oracle WebLogic Integration Java Process Definition
- ✓ MQ transport to interact with IBM WebSphere MQ
- ✓ Tuxedo transport with transactional integrity

Architecture

Oracle Service Bus technically consists of a number of deployments running on WebLogic server. You can see these deployments if you open the WebLogic admin console and look under deployments. The figure below is just a small part of the total listing.

Deployments

<div>Install Update Delete Start Stop</div> <div>Showing 1 to 50 of 50 Previous Next</div>					
<input type="checkbox"/>	Name	State	Health	Type	Deployment Order
<input type="checkbox"/>	 ALDSP Transport Provider	Active	OK	Web Application	161
<input type="checkbox"/>	 ALSB Cluster Singleton Marker Application	Active	OK	Enterprise Application	80
<input type="checkbox"/>	 ALSB Coherence Cache Provider	Active	OK	Enterprise Application	93
<input type="checkbox"/>	 ALSB Domain Singleton Marker Application	Active	OK	Enterprise Application	85
<input type="checkbox"/>	 ALSB Framework Starter Application	Active	OK	Enterprise Application	90
<input type="checkbox"/>	 ALSB Logging	Active	OK	Enterprise Application	440
<input type="checkbox"/>	 ALSB Publish	Active	OK	Enterprise Application	430

Coherence

OSB 11g comes with a Coherence based result cache. To use it, define a key and a cache timeout for the response of a business service. When calling the service with the same key and within the timeout period, the result is delivered much faster from Coherence.

Operation

Installation

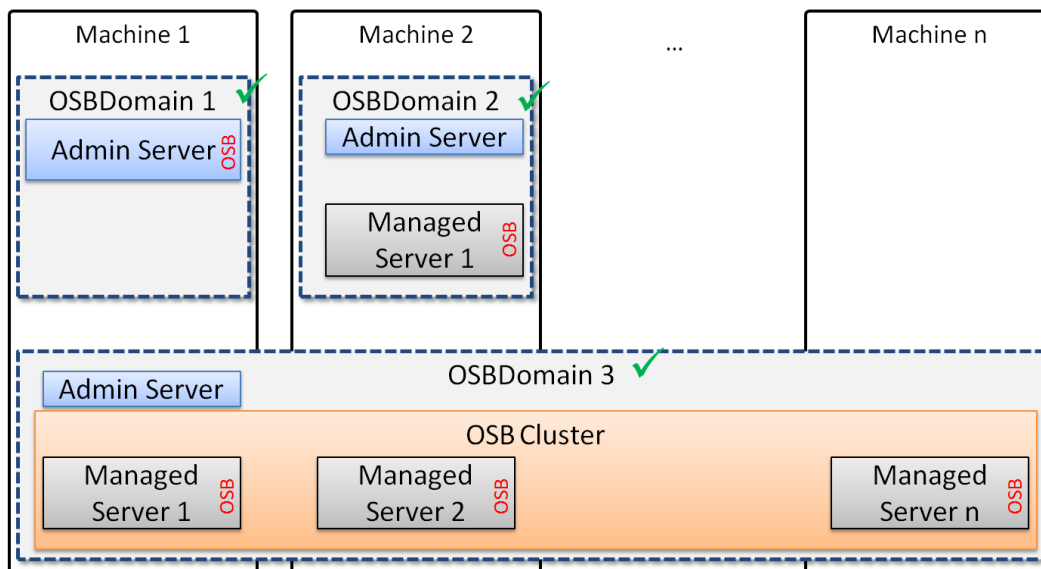
The newest version, OSB 11g, is packaged without WebLogic application server and uses the Oracle Unified Installer.

The core installation steps are as follows:

- ✓ Install WebLogic
- ✓ Install a supported database (if report action or OWSM is used)
- ✓ Install Oracle Service Bus 11g
- ✓ Run the Repository Creation Utility (if OWSM is used for policy management)
- ✓ Create OSB domain

OSB Domains

OSB builds on WebLogic servers, but there are important restrictions when setting up an OSB domain. OSBDomain 1 to 3 in the diagram below are all possible architectures. Note that the OSB console is always running on the WebLogic admin server which is not illustrated below.

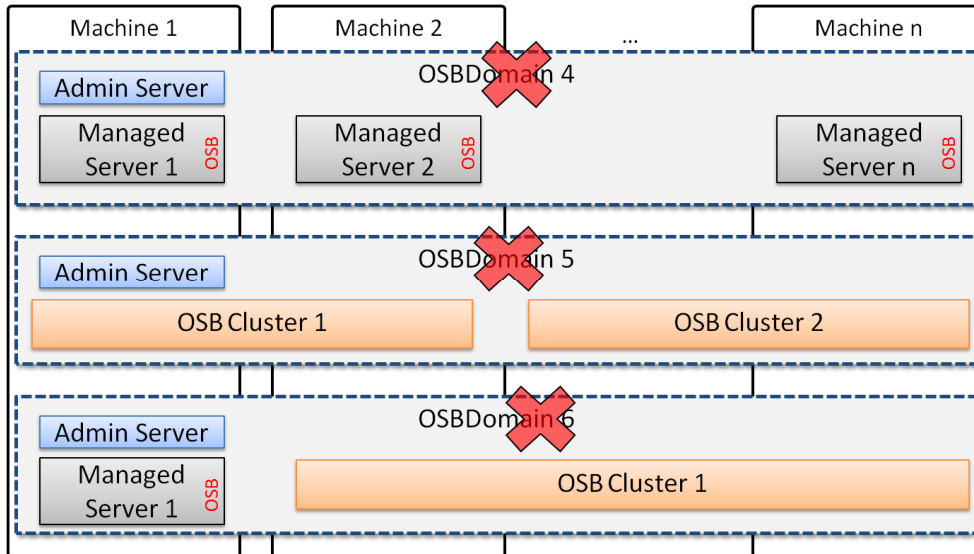


The rules are as follows:

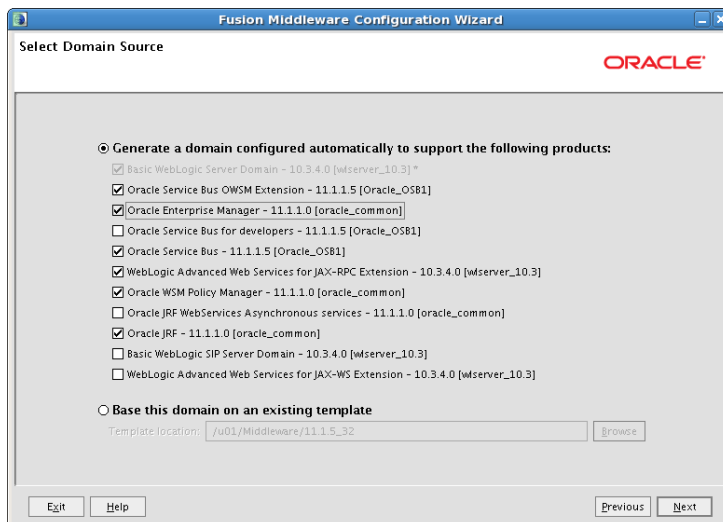
- ✓ If there is more than one managed server, you have to put the managed servers into a cluster. OSB uses uniformly distributed JMS destinations, which only work in a cluster.
- ✓ You can only have one OSB cluster within a domain. This limitation does not exist for a WebLogic only domain.
- ✓ When expanding the OSB cluster, you have to manually add JMS server, JMS system modules and a number of distributed destinations to the newly added server.

- ✓ On the other hand, you cannot have two managed servers running OSB within a domain without a cluster, or 2 OSB clusters within one domain, or an OSB cluster and a separate managed server running OSB.

So the following domain configurations, (all easily possible with WebLogic only), are illegal with OSB.



An OSB domain can be easily configured using the domain wizard. Once OSB is installed it can create OSB domains including support for Oracle Enterprise Manager and Oracle Web Service Manager.



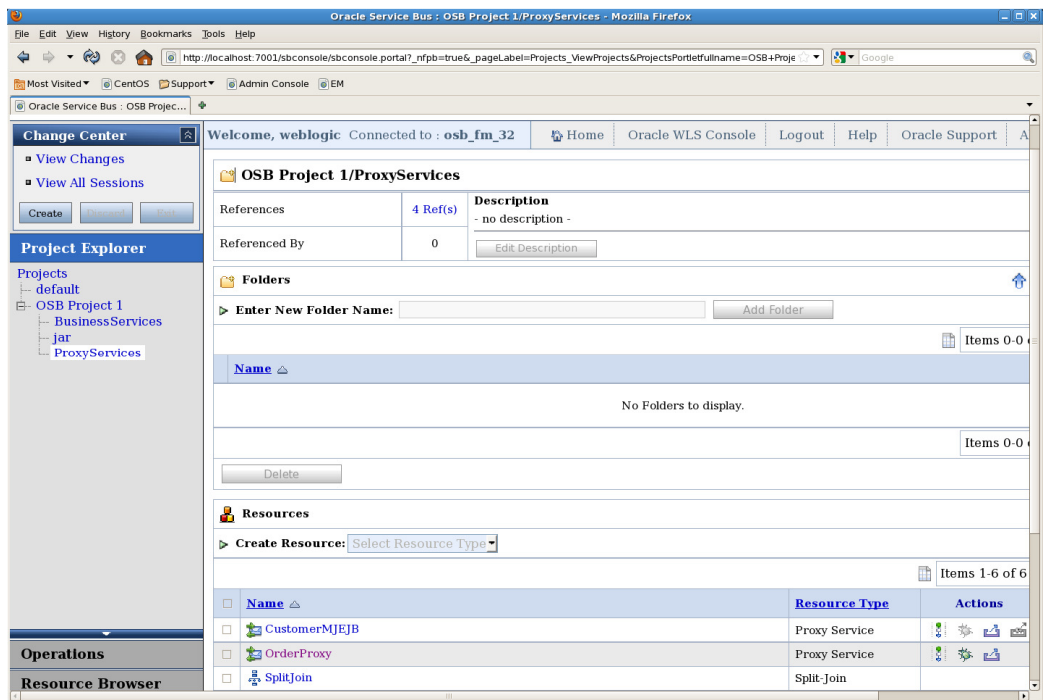
Administration

Within *any* domain you can always access the WebLogic admin console. Typical WebLogic configurations such as defining listen ports, clusters etc. is done with the WebLogic admin console for OSB.

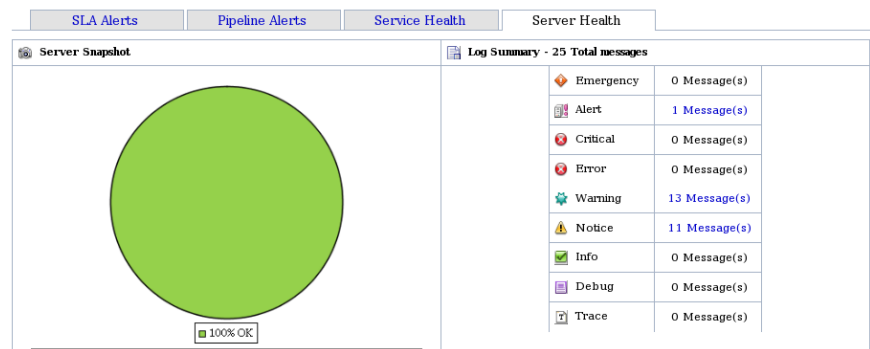
For the OSB specifics, there is a web based Service Bus console with the /sbconsole context root. If your admin server is using the default settings, you can access Service Bus console with the following URL:

http://localhost:7001/sbconsole

As with the WebLogic admin console, the service bus console is only available on the admin server.

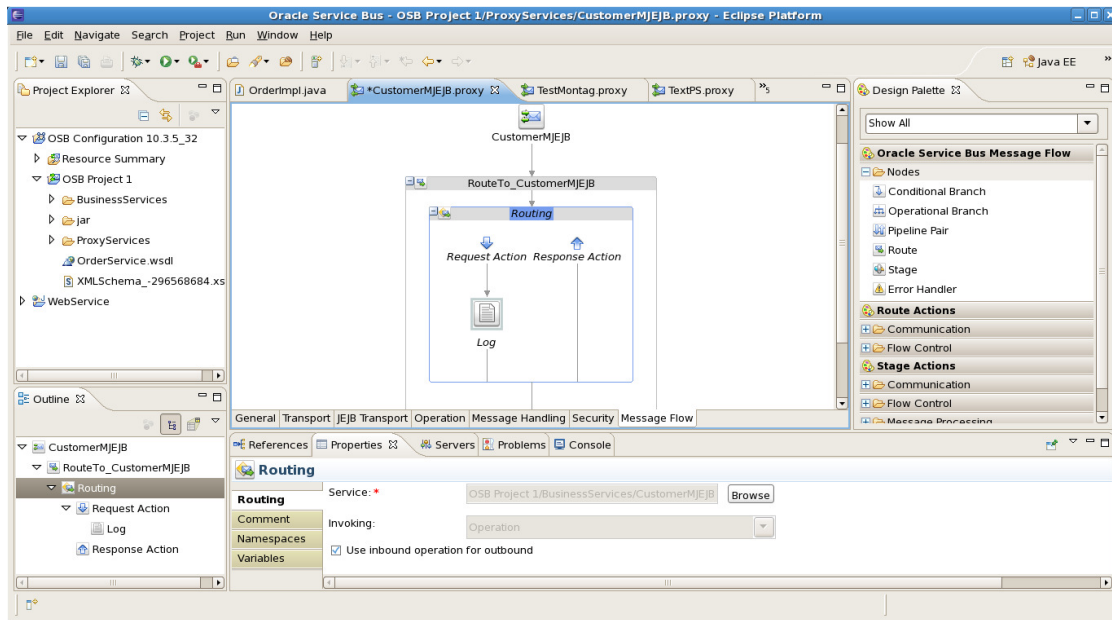


The service bus also contains a dashboard for server and service health and invocation statistics on an operation level as well as execution times for every single action when processing messages in a proxy.



Development

The main part of Service Bus development is the configuration of the message flow within a proxy service. For Oracle Service Bus 11g this can be comfortably done with an IDE based on Eclipse, together with the Eclipse OSB plugin of OSB.



Alternatively, all development can be done from the web-based console.

Discussion

Oracle Service Bus is a stable, well established and fascinating product (no, I am not getting paid for this sentence). The core part of it, such as proxy services with their message flow, is still the same as when the product was launched. Since OSB operates at a very low, technical layer, its possibilities are almost unlimited.

There are however some rough edges, most of them regarding its integration with Oracle Fusion Middleware 11g, in particular SOA Suite:

- ✓ OSB 11g development environment is Eclipse based, however the SOA Suite and BPM development environment is JDeveloper based.
- ✓ To deploy an SOA Suite Adapter you need both JDeveloper and Eclipse.
- ✓ Modeling asynchronous request-reply interaction through service bus is rather difficult and needs better tool support. (Some day there will be one IDE for both...)
- ✓ OSB supports different XQuery and XSLT functions than SOA Suite.
- ✓ Extending and OSB cluster is too complex. You have to add JMS components to the new instance manually.

Answered

Q: What does "OSB is stateless" mean?

A: OSB typically doesn't store any state on behalf of the client. It is not designed for that. Apart from a special Split-Join pattern, which was introduced rather late, there is also no action in OSB to wait for any event.

Q: What does "OSB is configuration driven" mean?

A: An EJB project or an SOA Suite project has to be deployed. Service bus only requires configuration changes that can be done at runtime with the web based Service Bus console.

Q: Isn't changing Service Bus configuration at runtime with hundreds of requests per second a configuration management nightmare?

A: It could be, honestly. Yet having the technical ability to do so is not bad per se.

Q: Why does OSB need a database if it is stateless?

A: It doesn't. Only the report action will be written to a database, therefore OSB has typically been installed with a database.

The current version of OSB also integrates with Oracle Web Service Manager (OWSM) to manage policies. OWSM requires a meta-data repository, which is another reason to have a database.

Q: Couldn't I just build a service bus using Oracle SOA Suite?

A: In theory you could do that. In practical terms, a service bus is configuration driven and optimized for a huge number of stateless interactions.

Directions

Consider the learning curve before starting an OSB project.

Don't believe you will understand OSB because you have SOA Suite knowledge. OSB has a different origin and a different purpose.

When to use

Definitely consider using Oracle Service Bus when implementing an SOA. Remember that the OSB license is included with Oracle SOA Suite.

OSB has been successfully used to virtualize web service requests (years before SOA Suite was available). There are plenty of use cases even without deploying the full SOA Suite.

More?

Links

Oracle Service Bus Documentation

http://docs.oracle.com/cd/E23943_01/soa.htm#osb

Books

There is a good OSB book available explaining the most common development tasks with step-by-step instructions and screenshots:

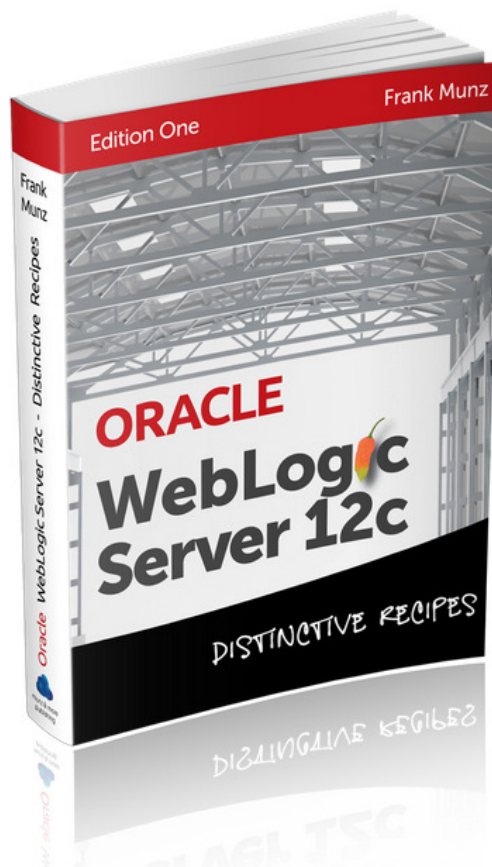
<http://www.packtpub.com/oracle-service-bus-11g-development-cookbook/book>

The SOA Suite Developer Cookbook is covering the integration of OSB with SOA Suite 11g:

<http://www.amazon.de/Oracle-SOA-Suite-Developers-Cookbook/dp/1849683883>

Oracle WebLogic Server 12c - Distinctive Recipes (Architecture, Development and Administration)

Homepage: <http://wls12book.munzandmore.com>
Amazon: <http://amazon.com/dp/0980798019>
Webcast channel: <http://youtube.com/WeblogicBook> (more than 40 free webcasts!)
Book on Facebook: <http://facebook.com/WebLogicBook>



Frank's details:

Twitter: @frankmunz

Blog: <http://www.munzandmore.com/blog>