What is a JBI SU and how do I create one?

What is a JBI SU and how do I create one?

The JBI spec includes deployment units for packaging of JBI components and JBI service units (SUs). SU is an abbreviation for a JBI service unit. JBI Components are JBI compliant components that are deployed into ServiceMix and are awaiting a configuration to tell them how to run. A SU is essentially the packaging for a configuration for a JBI component and any of the necessary dependencies for the SU to be deployed. From a packaging perspective, the JBI SU is similar to a JavaEE WAR file if you're familiar with web application development. It's really just an archive to bundle together everything to deploy to the server. So to make use of one of the JBI components, all you really need to do is create a configuration.

To create a configuration, a JBI SU, you will use the Maven archetypes that are provided with ServiceMix. Maven archetypes are used to create a Maven project skeleton to jumpstart project creation via the automation of repetitive tasks by following standard conventions. The result of using a Maven archetype to create a Maven project is a directory structure, a Maven POM file (`pom.xml`) and, depending on the archetype being used, sometimes Java objects and JUnit tests.

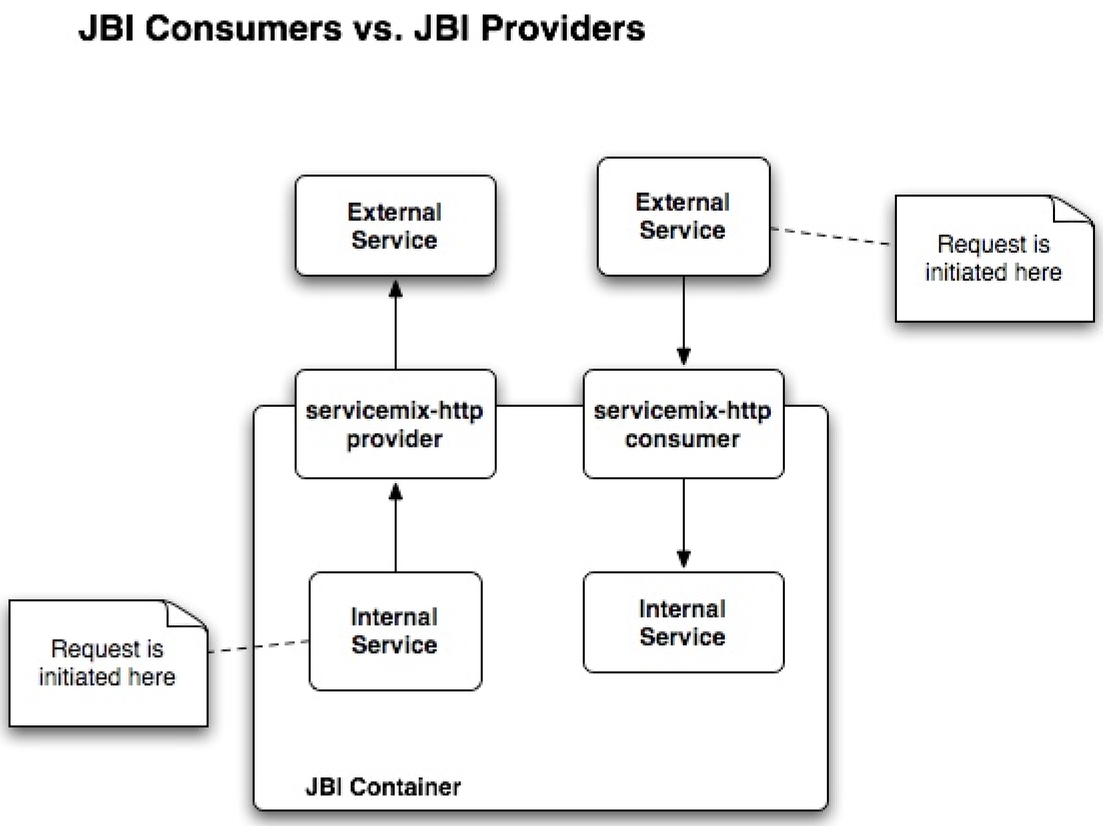
Creating the SU

Here is an example of creating a SU for the servicemix-http component that comes with ServiceMix. The servicemix-http component is a JBI binding component (BC). It is deployed to the ServiceMix container and just sits there awaiting a configuration SU to be deployed to it so it knows how to behave. If you're familiar with JavaEE resource adapters (RARs), the concept of JBI components is very similar to this. The JBI components don't do anything until they're configured, just like a RAR file. In order to tell the JBI component what to do, a configuration for it must be deployed.

What we'll do now is use a Maven archetype to create a Maven project to create a SU specific to the servicemix-http component. However, before creating a SU, we need to know if it's a consumer or a provider.

A Consumer SU or a Provider SU?

JBI components (Service Engines and Binding Components) can act as a service consumer, a service provider or both. The diagram below outlines these concepts using the example of a ServiceMix HTTP Consumer BC and a ServiceMix HTTP Provider BC. Below is a diagram to help explain the difference:



In the diagram above, notice where requests are initiated and the direction of the arrows to denote the flow from the initiator. For the example, we'll be creating a consumer to expose a service already running in the JBI container via HTTP.

Creating a SU Maven Project

For this example, let's suppose that we're creating a consumer SU. A consumer SU contains a configuration that tells the servicemix-http component to expose an endpoint via HTTP for some service that is already deployed to the JBI container. Below is the command to create the consumer SU Maven project for the servicemix-http component, replace the -DarchetypeVersion property with whatever version of ServiceMix that you intend to use:

```
$ [SMX_HOME]/bin/smx-arch su http-consumer \
   -DgroupId=com.mycompany \
   -DartifactId=my-consumer-su
```

The command above utilizes the servicemix-http-consumer-service-unit Maven archetype to create a Maven project named my-consumer-su. This command creates a directory named `my-consumer-su` that contains a Maven project skeleton meaning all the necessary files are in place, you simply need to enter the correct values for the configuration. Below is the directory structure created by this archetype:

```
./pom.xml
./src
./src/main
./src/main/resources
./src/main/resources/xbean.xml
```

The only file that will need to be edited in this case is the `src/main/resources/xbean.xml`. This is the configuration for the JBI component. By default it looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns:http="http://servicemix.apache.org/http/1.0"
       xmlns:replaceMe="http://servicemix.apache.org/replaceMe">

  <http:endpoint service="replaceMe:withYourService"
                 endpoint="soap"
                 role="consumer"
                 locationURI="http://localhost:8192/example/"
                 defaultMep="http://www.w3.org/2004/08/wsdl/in-out"
                 soap="true" />

</beans>
```

Notice the `replaceMe` namespace and the `withYourService` service name. These values need to be replaced to accurately reflect your situation. You can also change the `locationURI` to add a custom port or path. In addition, all of the **Consumer endpoint attributes** listed on the servicemix-http page are available to further customize this configuration.

Once the SU is fully configured, it must be packaged. This is easy to do because the project skeleton provides all the necessary project configuration and use of the Maven JBI plugin for you via the `pom.xml` file. Simply run the following command while sitting in the `my-consumer-su` directory:

```
$ mvn install
```

You will need to have Maven 2.0.4 or higher installed in order to run this command. This packages up the SU and places the build artifact in the target directory.

JBI SU Requirements

The only requirement of a JBI SU is that it contain a JBI deployment descriptor located in `META-INF/jbi.xml`. As mentioned above, the Maven archetypes for SUs creates a `pom.xml` file that includes configuration and use of the Maven JBI plugin. This is a plugin that automatically generates the `META-INF/jbi.xml` file based on other information in the `pom.xml` file. Below is the hierarchy of the contents of the SU archive that is created when running the Maven install goal as noted above:

```
$ jar tvf ./target/my-consumer-su-1.0-SNAPSHOT.jar
     0 Wed Mar 28 20:19:52 MDT 2007 META-INF/
   126 Wed Mar 28 20:19:50 MDT 2007 META-INF/MANIFEST.MF
   292 Wed Mar 28 20:19:50 MDT 2007 META-INF/jbi.xml
  1281 Wed Mar 28 20:19:40 MDT 2007 xbean.xml
     0 Wed Mar 28 20:19:52 MDT 2007 META-INF/maven/
     0 Wed Mar 28 20:19:52 MDT 2007 META-INF/maven/com.mycompany/
     0 Wed Mar 28 20:19:52 MDT 2007 META-INF/maven/com.mycompany/my-consumer-su/
  2613 Wed Mar 28 17:30:46 MDT 2007 META-INF/maven/com.mycompany/my-consumer-su/pom.xml
   119 Wed Mar 28 20:19:50 MDT 2007 META-INF/maven/com.mycompany/my-consumer-su/pom.properties
```

Notice that this SU archive contains a `META-INF/jbi.xml` file. This was created by the Maven JBI plugin. Below is the `jbi.xml` file for the project above:

```
$ cat ./target/my-consumer-su-1.0-SNAPSHOT-installer/META-INF/jbi.xml
<?xml version="1.0" encoding="UTF-8"?>
<jbi xmlns="http://java.sun.com/xml/ns/jbi" version="1.0">
  <services binding-component="false" xmlns:replaceMe="http://servicemix.apache.org/replaceMe">
    <consumes service-name="replaceMe:withYourService" endpoint-name="soap"/>
  </services>
</jbi>
```

This file is a deployment descriptor for the JBI container when the SU is deployed.

Deployment

Once the SU is properly packaged, it must be wrapped in a JBI service assembly (SA) before it can be deployed to the JBI container. JBI SAs are described in What is a JBI SA and how do I create one?.

Additional Information

Have you walked through the Tutorials yet? This is a great place to start if you're new to JBI.