

# SOAPProcessor

---

JBoss Webservices SOAP Processor.

This action supports invocation of a JBossWS hosted webservice endpoint through any JBossESB hosted listener. This means the ESB can be used to expose Webservice endpoints for Services that don't already expose a Webservice endpoint. You can do this by writing a thin Service Wrapper Webservice (e.g. a JSR 181 implementation) that wraps calls to the target Service (that doesn't have a Webservice endpoint), exposing that Service via endpoints (listeners) running on the ESB. This also means that these Services are invocable over any transport channel supported by the ESB (http, ftp, jms etc).

## "ESB Message Aware" Webservice Endpoints

Note that Webservice endpoints exposed via this action have direct access to the current JBossESB Message instance used to invoke this action's `process(org.jboss.soa.esb.message.Message)` method. It can access the current Message instance via the `getMessage()` method and can change the Message instance via the `setMessage(org.jboss.soa.esb.message.Message)` method. This means that Webservice endpoints exposed via this action are "ESB Message Aware".

## Webservice Endpoint Deployment

Any JBossWS Webservice endpoint can be exposed via ESB listeners using this action. That includes endpoints that are deployed from inside (i.e. the Webservice .war is bundled inside the .esb) and outside (e.g. standalone Webservice .war deployments, Webservice .war deployments bundled inside a .ear) a .esb deployment.

## WSDLs

WSDLs for Webservices exposed via JBossESB are available through the "Contracts" application (deployed with the ESB components). This application can be accessed through

your JBoss App/ESB Server at "<http://<host:port>/contract>". This application lists URLs that can be used by your Webservice Client (e.g. soapUI) for accessing a Service's WSDL, enabling WSDL based invocation of the Service **through an ESB Endpoint**.

## JAXB Introductions

The native JBossWS SOAP stack uses JAXB to bind to and from SOAP. This typically means that an unannotated typeset could not be used to build a JSR 181 endpoint on JBossWS. To overcome this we use a JBossESB and JBossWS feature called "JAXB Introductions". This feature allows you define an XML configuration to "Introduce" the JAXB Annotations.

For details on how to configure JBossWS 2.0.x to use [JAXBIntroductions](#), see [JAXBIntroductionsOnJBossWS](#). The one exception to the instructions found there is that the jboss-jaxb-intros.jar file is already provided as part of the JBossESB distribution, so there's no need to build this lib. It can be found in the "extras/jaxbintros" directory in the JBossESB distribution.

## Action Configuration

The `<action ... ></action>` configuration for this action is very straightforward. The action requires only one mandatory property value, which is the "jbossws-endpoint" property. This property names the JBossWS endpoint that the SOAPProcessor is exposing (invoking).

```
<action name="ShippingProcessor" class="org.jboss.soa.esb.actions.soap.SOAPProcessor">
  <property name="jbossws-endpoint" value="ABI_Shipping"></property>
  <property name="rewrite-endpoint-url" value="true/false"></property>
</action>
```

The optional "rewrite-endpoint-url" property is there to support load balancing on HTTP endpoints, in which case the Webservice endpoint container will have been configured to set the HTTP(S) endpoint address in the WSDL to that of the Load Balancer. The "rewrite-endpoint-url" property can be used to turn off HTTP endpoint address rewriting in situations such as this. It has no effect for non-HTTP protocols.

## Quickstarts

A number of quickstarts that demonstrate how to use this action are available in the JBossESB distribution (samples/quickstarts). See the "webservice\_bpel", "webservice\_mtom", "webservice\_producer", "webservice\_wsaddressing" and "webservice\_wssecurity" quickstarts.