

# **ESB Message Transformation Guide**

**4.2**

# **JBoss Enterprise SOA Platform**



**ISBN:**

**Publication date: February, 2008**

The SOA Platform edition of the JBoss ESB Message Transformation Guide

---

# ESB Message Transformation Guide: JBoss Enterprise SOA Platform

Copyright © 2008 Red Hat, Inc.

Copyright © 2008 Red Hat, Inc.. This material may only be distributed subject to the terms and conditions set forth in the Creative Commons Attribution-Noncommercial-Share Alike 3.0 Unported License (which is presently available at <http://creativecommons.org/licenses/by-nc-sa/3.0/>).

Red Hat and the Red Hat "Shadow Man" logo are registered trademarks of Red Hat, Inc. in the United States and other countries.

All other trademarks referenced herein are the property of their respective owners.

The GPG fingerprint of the security@redhat.com key is:

CA 20 86 86 2B D6 9D FC 65 F6 EC C4 21 91 80 CD DB 42 A6 0E

1801 Varsity Drive  
Raleigh, NC 27606-2072  
USA  
Phone: +1 919 754 3700  
Phone: 888 733 4281  
Fax: +1 919 754 3701  
PO Box 13588  
Research Triangle Park, NC 27709  
USA

---



---

Preface .....	vii
1. Document Conventions .....	vii
2. We Need Feedback .....	viii
1. Introduction .....	1
1. Overview .....	1
2. Smooks .....	3
1. Introduction .....	3
2. Samples and Tutorials .....	3
3. SmooksTransformer Configuration .....	3
4. Profile Based Transformation .....	4
3. XSL and Binary Format Transformations .....	9
1. XSL Transformations .....	9
2. Binary Format Translations .....	9

---

---

## Preface

# 1. Document Conventions

Certain words in this manual are represented in different fonts, styles, and weights. This highlighting indicates that the word is part of a specific category. The categories include the following:

*Courier font*

Courier font represents `commands, file names and paths, and prompts.`

When shown as below, it indicates computer output:

```
Desktop      about.html    logs          paulwesterberg.png
Mail         backupfiles   mail          reports
```

**Courier font**

Bold Courier font represents text that you are to type, such as: `service jonas start`

If you have to run a command as root, the root prompt (`#`) precedes the command:

```
# gconftool-2
```

*italic Courier font*

Italic Courier font represents a variable, such as an installation directory:

```
install_dir/bin/
```

**font**

Bold font represents **application programs** and **text found on a graphical interface**.

When shown like this: **OK**, it indicates a button on a graphical application interface.

Additionally, the manual uses different strategies to draw your attention to pieces of information. In order of how critical the information is to you, these items are marked as follows:



### Note

A note is typically information that you need to understand the behavior of the system.



### Tip

A tip is typically an alternative way of performing a task.



### Important

Important information is necessary, but possibly unexpected, such as a configuration change that will not persist after a reboot.



### Caution

A caution indicates an act that would violate your support agreement, such as recompiling the kernel.



### Warning

A warning indicates potential data loss, as may happen when tuning hardware for maximum performance.

## 2. We Need Feedback

If you find a typographical error in the *SOA ESB Message Transformation Guide*, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: <http://jira.jboss.com/jira/> against the Documentation component of the *SOA Platform* project.

When submitting a bug report, be sure to mention the manual's identifier:

ESB\_MTG

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.



# Introduction

## 1. Overview

JBoss ESB supports message data transformation through a number of mechanisms:

1. **Smooks:** *Milyn Smooks* [<http://milyn.codehaus.org/smooks>] is a Fragment based Data Transformation and Analysis tool (XML and non-XML). It can also be thought of as a management tool that allows you manage transformations across your entire message set using techniques such as message profiling. It supports transformation logic implementation through raw Java, XSLT, FreeMarker, Groovy and more.
2. **XSLT:** JBoss ESB supports message transformation through the standard XSLT usage model.
3. **ActionProcessor Data Transformation:** Transformations involving binary data formats are most easily performed through implementation of the `org.jboss.soa.esb.actions.ActionProcessor` Java interface. The `org.jboss.soa.esb.actions` package (in the “Listeners” module) has a number of out-of-the-box `ActionProcessor` based transformers.



# Smooks

## 1. Introduction

Message Transformation on JBossESB is supported by the `SmooksTransformer` component. This is an ESB Action component that allows the Smooks Data Transformation/Processing Framework to be plugged into an ESB Action Processing Pipeline.

A wide range of source (XML, CSV, EDI etc) and target (XML, Java, CSV, EDI etc) data formats are supported by the `SmooksTransformer` component. A wide range of Transformation Technologies are also supported, all within a single framework. See [Smooks](http://milyn.codehaus.org/smooks) [http://milyn.codehaus.org/smooks] for more details.

## 2. Samples and Tutorials

1. A number of Transformation Quickstart samples accompany the JBossESB distribution. Check out the "transform\_\*" Quickstarts.
2. A number of tutorials are available online on the [Smooks](http://milyn.codehaus.org/smooks) [http://milyn.codehaus.org/smooks] website. Any of these samples can be easily ported to JBossESB.

## 3. SmooksTransformer Configuration

The basic configuration of this action simply takes a `resource-config` property that references a Smooks configuration file. The `resource-config` property value is any valid URI based resource, as defined by the `URIResourceLocator` class.

```
<action name="transform"
        class="org.jboss.soa.esb.actions.converters.SmooksTransformer">
  <property name="resource-config" value="/smooks-config.xml" />
</action>
```

### Input/Output Configuration.

By default, this actions gets its input from (and sets it's output on) the "Default Message Body Location" (i.e. `Message.getBody().add(Body.DEFAULT_LOCATION...)` and `Message.getBody().get(Body.DEFAULT_LOCATION)`). These can be overridden by specifying `input-location` and/or `output-location` configuration properties on the action.

### Java Output Configuration.

As stated above, this action supports source (XML, CSV, EDI etc) to Java object transformation/binding. See the `Transform_XML2POJO*` quickstarts for examples of this and also check out the [Smooks Tutorials](http://milyn.codehaus.org/Tutorials) [http://milyn.codehaus.org/Tutorials].

The constructed Java object model(s) can be used as part of a [model driven transform](http://milyn.codehaus.org/Model+Driven+Transformation) [http://milyn.codehaus.org/Model+Driven+Transformation], or can simply be used by other ESB action instances that follow the `SmooksTransformer` in an action pipeline.

Such Java object graphs are available to subsequent pipeline action instances because they are attached to the ESB Message output by this action and input to the following action(s). They are bound to the `Message` instance `Body` (`Message.getBody().add(...)`) under a key based directly on the objects `beanId` (as defined in the [Smooks Javabean config](http://milyn.codehaus.org/javadoc/v1.0/smooks-cartridges/javabean/org/milyn/javabean/BeanPopulator.html) [http://milyn.codehaus.org/javadoc/v1.0/smooks-cartridges/javabean/org/milyn/javabean/BeanPopulator.html]). This means that the objects are available through the ESB Message Body by performing `Body.get(beanId)` calls.

The full Java object Map can also be made available on the output message by adding a `java-output-location` property, for example:

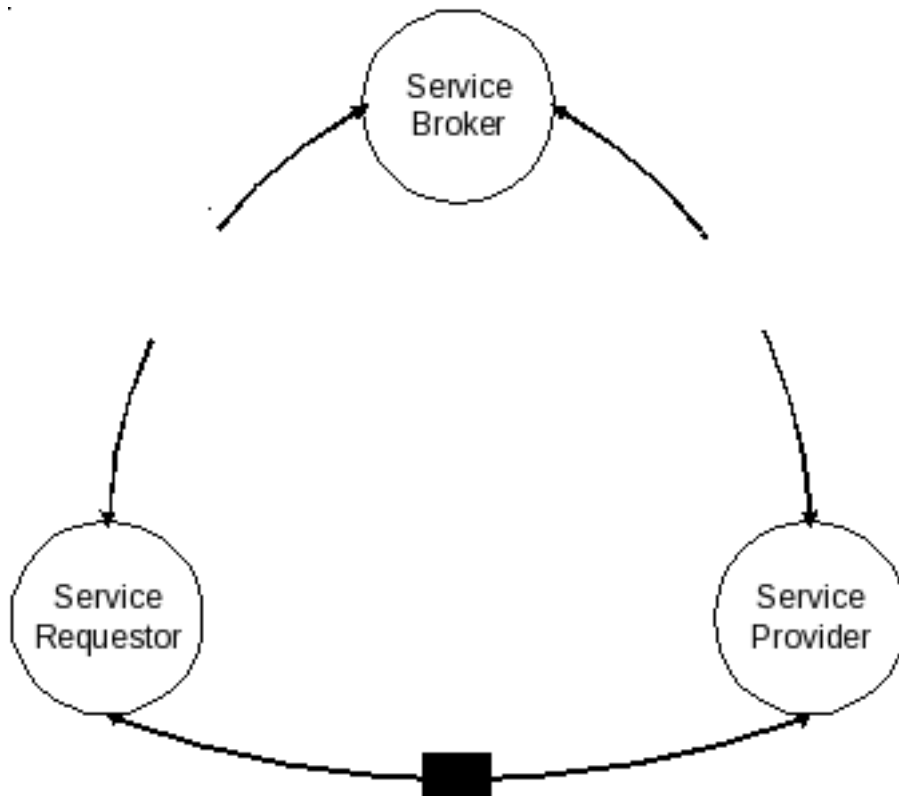
```
<action name="transform"
  class="org.jboss.soa.esb.actions.converters.SmooksTransformer">
  <property name="resource-config" value="/smooks-config.xml" />
  <property name="java-output-location"
value="order-message-objects-map" />
</action>
```

Or, shorthand for binding the map to the "Default Message Body Location":

```
<action name="transform"
  class="org.jboss.soa.esb.actions.converters.SmooksTransformer">
  <property name="resource-config" value="/smooks-config.xml" />
  <property name="java-output-location" value="$default" />
</action>
```

## 4. Profile Based Transformation

In the following example we have messages being exchanged between 3 different Sources and 1 Target. The ESB needs to transform the messages supplied by each of the 3 Sources (supplied in different formats) into Java Objects before supplying them to the "Target" Service. Basically, we've 3 possible transformations for messages being exchanged to the Target. We'll call these transformations "source1", "source2" and "source3" respectfully.



From an ESB configuration perspective, we have a single Service configuration for the "Target" Service. The configuration might look something like the following.

```
<service category="ServiceCat" name="TargetService"
  description="Target Service">
  <listeners>
    <jms-listener name="Gateway-Listener"
      busidref="quickstartGwChannel" is-gateway="true"/>
    <jms-listener name="Service-Listener"
      busidref="quickstartEsbChannel"/>
  </listeners>
  <actions>
    <!--
      A SmooksTransform action to transform the source message into
      the Target Java Object(s)
    -->
    <action name="transform"
      class="org.jboss.soa.esb.actions.converters.SmooksTransformer">
      <property name="resource-config" value="/smooks-config.xml"/>
    </action>

    <!-- An action to process the Java Object(s) -->
    <action name="process" class="com.acme.JavaProcessor" />

  </actions>
</service>
```

As can be seen by the `SmooksTransformer` configuration, we only define a single

transformation configuration file called `/smooks-config.xml`. We need to define 3 different transformation, one for each source. This is done using a Smooks Message Profiling.

Basically, we define the 3 separate transformations in 3 separate Smooks configuration files (`from_source1.xml`, `from_source2.xml` and `from_source3.xml`) and `<import>` them into the top-level `smooks-config.xml`. In each of these configuration files we specify the `default-target-profile` for that configuration set. For example:

```
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd"
  default-target-profile="from:source1">
  <!--
    Source1 to Target Java message transformation
    resource configurations...
  -->
</smooks-resource-list>
```

The top-level `smooks-config.xml` looks as follows:

```
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">
  <import file="classpath:/from_source1.xml" />
  <import file="classpath:/from_source2.xml" />
  <import file="classpath:/from_source3.xml" />
</smooks-resource-list>
```

So what this does (effectively) is to load a single `SmooksTransformer` instance with 3 different transformations, each transformation defined under it's own unique "profile" name. There are actually more uses for message profiling than this, but this view of profiling works fine for this example.

In order for the `SmooksTransformer` to know which of the 3 transformations needs to be applied on a given message, the message (ESB Message) needs to have it's `from` property set before the message flows into the `SmooksTransformer`. This can be done anywhere that makes sense - at the Source itself (`source1`, `source2` etc), or in a content based action that precedes the `SmooksTransformer`. See details of the `transform_XML2POJO2` quickstart (below).

JBossESB also supports profiles additional to the "from" profile, namely "from-type", "to" and "to-type". These can be used in combination, leading to more intricate exchange based transforms - n possible inputs to n possible outputs, sharing profile sets etc. See the "profiling" tutorial on the [Smooks website](http://milyn.codehaus.org/Tutorials) [http://milyn.codehaus.org/Tutorials].

### Transform\_XML2POJO2

The basic scenario outlined above is implemented as a quickstart within the JBossESB distribution. The quickstart is named `transform_XML2POJO2`. In this quickstart there are 2 message sources. It utilises a Groovy script on the action pipeline to detect and set the `from` profile for the incoming message.

- `jboss-esb.xml`: JBossESB Configuration File.

- `smooks-config.xml`: Top Level Transformation Configuration.
- `from-dvdstore.xml`: DVD Store message Transformations Configuration - imported into top-level `smooks-config.xml` (notice the profile configuration?). Designed to transform a DVD Store message into Java Objects.
- `from-petstore.xml`: Pet Store message Transformations Configuration - imported into top-level `smooks-config.xml` (notice the profile configuration?). Designed to transform a Pet Store message into Java Objects (the same Java object model).
- `check-origin.groovy`: Simple Groovy script for checking the origin of the message based on it's content.





# XSL and Binary Format Transformations

## 1. XSL Transformations

XSL Transformations are supported through Smooks. Support for XSLT is available by creating a custom `org.jboss.soa.esb.actions.ActionProcessor` implementation.

A future version of JBoss ESB will include native support for XSLT.

## 2. Binary Format Translations

Binary Transformations are supported through custom implementation of the `org.jboss.soa.esb.actions.ActionProcessor` interface.

JBossESB ships with a number of out-of-the-box binary transformers, for example:

`org.jboss.soa.esb.actions.ObjectToXStream` and  
`org.jboss.soa.esb.actions.ObjectToCSVString`.

